



Kostenloses eBook

LERNEN

R Language

Free unaffiliated eBook created from
Stack Overflow contributors.

#r

Inhaltsverzeichnis

Über	1
Kapitel 1: Erste Schritte mit R Language	2
Bemerkungen	2
Bearbeiten von R-Dokumenten im Stapelüberlauf	2
Einige Merkmale von R, die Einwanderer aus anderen Sprachen ungewöhnlich finden	2
Examples	2
R installieren	2
Nur Windows:	2
Für Windows	2
Für OSX / macOS	3
Alternative 1	3
Alternative 2	3
Für Debian, Ubuntu und Derivate	3
Für Red Hat und Fedora	4
Für Archlinux	4
Hallo Welt!	4
Hilfe bekommen	4
Interaktiver Modus und R-Skripte	5
Der interaktive Modus	5
Verwendung von R als Taschenrechner	5
Die erste Handlung	6
R-Skripte	8
Kapitel 2: * wende die Familie der Funktionen (Funktionale) an	9
Bemerkungen	9
Mitglieder der *apply Familie	9
Examples	9
Verwenden Sie anonyme Funktionen mit apply	10
Massendatei wird geladen	11
Mehrere `data.frames` kombinieren (`lapply`, `mapply`)	12

Eingebaute Funktionale verwenden	13
Integrierte Funktionen: lapply (), sapply () und mapply ()	13
lapply ()	13
sapply ()	13
mapply ()	14
Verwendung benutzerdefinierter Funktionen	14
Benutzerdefinierte Funktionen	14
Kapitel 3: .Rprofile	16
Bemerkungen	16
Examples	16
.Rprofile - der erste ausgeführte Code	16
Einrichten Ihres R Home-Verzeichnisses	16
Optionen für die Seitengröße einstellen	16
Legen Sie den Standardhilfetyp fest	16
Legen Sie eine Site-Bibliothek fest	16
Stellen Sie einen CRAN-Spiegel ein	17
Festlegen des Speicherorts Ihrer Bibliothek	17
Benutzerdefinierte Verknüpfungen oder Funktionen	17
Laden Sie die nützlichsten Pakete vor	17
Siehe auch	17
.Rprofile Beispiel	18
Anlaufen	18
Optionen	18
Benutzerdefinierte Funktionen	18
Kapitel 4: Aggregation von Datenrahmen	19
Einführung	19
Examples	19
Aggregation mit der Basis R	19
Aggregation mit Dplyr	20
Aggregation mit data.table	21
Kapitel 5: Aktualisieren von R und der Paketbibliothek	23

Examples.....	23
Unter Windows.....	23
Kapitel 6: Analysieren Sie Tweets mit R.....	25
Einführung.....	25
Examples.....	25
Tweets herunterladen.....	25
R Bibliotheken.....	25
Holen Sie sich Text von Tweets.....	26
Kapitel 7: Ändern von Strings durch Ersetzen.....	27
Einführung.....	27
Examples.....	27
Zeichnen Sie Zeichenfolgen mithilfe von Erfassungsgruppen neu an.....	27
Beseitigen Sie doppelte aufeinanderfolgende Elemente.....	27
Kapitel 8: ANOVA.....	29
Examples.....	29
Grundlegende Verwendung von aov ().....	29
Grundlegende Verwendung von Anova ().....	30
Kapitel 9: Arima-Modelle.....	31
Bemerkungen.....	31
Examples.....	31
Modellierung eines AR1-Prozesses mit Arima.....	31
Kapitel 10: Ausdruck: parse + eval.....	40
Bemerkungen.....	40
Examples.....	40
Code im String-Format ausführen.....	40
Kapitel 11: Balkendiagramm.....	41
Einführung.....	41
Examples.....	41
Barplot () - Funktion.....	41
Kapitel 12: Basisplot.....	49
Parameter.....	49

Bemerkungen.....	49
Examples.....	49
Grundlegende Darstellung.....	49
Matplot.....	52
Histogramme.....	58
Plots kombinieren.....	60
par().....	60
layout().....	61
Dichteplot.....	62
Empirische kumulative Verteilungsfunktion.....	64
Erste Schritte mit R_Plots.....	65
Kapitel 13: Benutzereingabe abrufen.....	67
Syntax.....	67
Examples.....	67
Benutzereingabe in R.....	67
Kapitel 14: Beschleunigen Sie Code, der stark zu vektorisieren ist.....	68
Examples.....	68
Beschleunigen Sie mit Rcpp die zähste Vektorisierung für Loops.....	68
Beschleunigung für das Schleifen von Bytes durch Kompilieren.....	69
Kapitel 15: Bewährte Methoden für die Vektorisierung von R-Codes.....	71
Examples.....	71
Durch Zeilenoperationen.....	71
Kapitel 16: Bibliographie in RMD.....	75
Parameter.....	75
Bemerkungen.....	75
Examples.....	76
Angabe einer Bibliographie und Zitierautoren.....	76
Inline-Referenzen.....	78
Zitierstile.....	78
Kapitel 17: Box-Plot.....	81
Syntax.....	81
Parameter.....	81

Examples.....	81
Erstellen Sie ein Box-and-Whisker-Diagramm mit <code>boxplot () {graphics}</code>	82
Einfacher Boxplot (Sepal.Length).....	82
Boxplot der Kelchlänge nach Arten gruppiert.....	82
Ordnung bringen.....	83
Gruppennamen ändern.....	84
Kleine Verbesserungen.....	85
Farbe.....	85
Nähe der Box.....	86
Sehen Sie sich die Zusammenfassungen an, auf denen die Boxplots basieren. <code>plot=FALSE</code>.....	86
Zusätzliche Boxplot-Style-Parameter.....	87
Box.....	87
Median.....	87
Schnurrhaar.....	87
Klammer.....	87
Ausreißer.....	88
Beispiel.....	88
Kapitel 18: Caret.....	90
Einführung.....	90
Examples.....	90
Vorverarbeitung.....	90
Kapitel 19: Code-Profiling.....	92
Examples.....	92
Systemzeit.....	92
<code>proc.time ()</code>	92
Linienprofilierung.....	93
Mikrobankmark.....	94
Benchmarking mit Mikrobankmark.....	95
Kapitel 20: Daten reinigen.....	97
Einführung.....	97
Examples.....	97

Fehlende Daten aus einem Vektor entfernen.....	97
Unvollständige Zeilen entfernen.....	97
Kapitel 21: Daten zwischen langen und breiten Formularen umformen.....	99
Einführung.....	99
Bemerkungen.....	99
Hilfreiche Pakete.....	99
Examples.....	99
Die Umformfunktion.....	99
Lang bis breit.....	100
Weit bis lang.....	100
Daten umformen.....	101
Basis R.....	101
Das Tidyr-Paket.....	102
Das Paket data.table.....	102
Kapitel 22: Datenerfassung.....	103
Einführung.....	103
Examples.....	103
Eingebaute Datensätze.....	103
Beispiel.....	103
Datensätze in Paketen.....	104
Gapminder.....	104
Perspektiven für die Weltbevölkerung 2015 - Bevölkerungsabteilung der Vereinten Nationen...104	104
Pakete für den Zugriff auf offene Datenbanken.....	104
Eurostat.....	104
Pakete für den Zugriff auf eingeschränkte Daten.....	106
Datenbank der menschlichen Mortalität.....	106
Kapitel 23: Datenrahmen.....	110
Syntax.....	110
Examples.....	110
Erstellen Sie ein leeres data.frame.....	110
Zeilen und Spalten aus einem Datenrahmen subsetting.....	111

Syntax für Zeilen und Spalten zuzugreifen: [, [[und \$	111
Wie eine Matrix: data[rows, columns]	112
Mit numerischen Indizes	112
Mit Spaltennamen (und Zeilenamen)	112
Zeilen und Spalten zusammen	113
Eine Warnung zu den Abmessungen:	113
Wie eine Liste	113
Mit Klammern data[columns]	114
Mit doppelten Klammern data[[one_column]]	114
Mit \$ auf Spalten zugreifen	114
Nachteile von \$ für den Zugriff auf Spalten	114
Erweiterte Indexierung: Negative und logische Indizes	115
Negative Indizes lassen Elemente aus	115
Logische Vektoren zeigen bestimmte Elemente an, die beibehalten werden sollen	115
Komfortfunktionen zum Bearbeiten von data.frames	116
Teilmenge	116
verwandeln	116
mit und innerhalb	117
Einführung	117
Konvertieren Sie mit do.call in einer Liste gespeicherte Daten in einen einzelnen Datenrah	118
Konvertieren Sie alle Spalten eines data.frame in eine Zeichenklasse	119
Zeilen nach Spaltenwerten unterteilen	120
Kapitel 24: Datentabelle	121
Einführung	121
Syntax	121
Bemerkungen	122
Installation und Unterstützung	122
Paket laden	123
Examples	123
Eine Datentabelle anlegen	123
Bauen	123

Einlesen	124
Ändern Sie einen Datenrahmen	124
Erzwingen Sie ein Objekt in data.table	124
Spalten hinzufügen und ändern	125
Ganze Spalten bearbeiten	125
Untermengen von Spalten bearbeiten	125
Spaltenattribute bearbeiten	126
Spezielle Symbole in data.table	126
.SD	126
.Scols	127
.N	128
Code schreiben, der mit data.frame und data.table kompatibel ist	128
Unterschiede in der Subsetting-Syntax	128
Strategien zur Aufrechterhaltung der Kompatibilität mit data.frame und data.table	129
Schlüssel in data.table setzen	130
Kapitel 25: Datum und Uhrzeit	133
Einführung	133
Bemerkungen	133
Klassen	133
Datum / Uhrzeit-Format auswählen	133
Spezialisierte Pakete	134
Examples	134
Aktuelles Datum und Uhrzeit	134
Gehe zum Ende des Monats	135
Zum ersten Tag des Monats gehen	135
Verschieben Sie ein Datum um mehrere Monate konsistent	135
Kapitel 26: Datum-Zeit-Klassen (POSIXct und POSIXlt)	137
Einführung	137
Bemerkungen	137
Fallstricke	137

verwandte Themen	137
Spezialisierte Pakete	137
Examples	137
Datum-Uhrzeit-Objekte formatieren und drucken	137
Analysieren von Zeichenfolgen in Datum-Uhrzeit-Objekten	138
Anmerkungen	138
Fehlende Elemente	138
Zeitzonen	139
Datum-Uhrzeit-Arithmetik	139
Kapitel 27: Debuggen	140
Examples	140
Browser verwenden	140
Debug verwenden	141
Kapitel 28: Die Charakterklasse	142
Einführung	142
Bemerkungen	142
verwandte Themen	142
Examples	142
Zwang	142
Kapitel 29: Die Date-Klasse	143
Bemerkungen	143
verwandte Themen	143
Durcheinandergebrannte Notizen	143
Weitere Notizen	143
Examples	143
Datumsangaben	143
Termine	144
Zeichenketten in Datumsobjekte analysieren	146
Kapitel 30: Die logische Klasse	147
Einführung	147
Bemerkungen	147

Stenografie	147
Examples	147
Logische Operatoren	147
Zwang	148
Interpretation von NAs	148
Kapitel 31: dplyr	149
Bemerkungen	149
Examples	149
dplyr's einzelne Tabellenverben	149
Syntax-Gemeinsamkeiten	149
Filter	150
ordnen	151
wählen	152
mutieren	153
zusammenfassen	154
gruppieren nach	154
Alles zusammenfügen	155
fassen Sie mehrere Spalten zusammen	156
Teilmengebeobachtung (Zeilen)	158
dplyr::filter() - Wählen Sie eine Teilmenge von Zeilen in einem dplyr::filter() , die logi	158
dplyr::distinct() - Doppelte Zeilen entfernen:	158
Aggregation mit dem Operator %>% (Pipe)	159
Beispiele für NSE- und String-Variablen in dplyr	160
Kapitel 32: Durchführen eines Permutationstests	162
Examples	162
Eine ziemlich allgemeine Funktion	162
Kapitel 33: E / A für das Binärformat von R	165
Examples	165
Rds- und RData-Dateien (Rda)	165
Umgebung	165
Kapitel 34: E / A für Datenbanktabellen	167

Bemerkungen.....	167
Spezialisierte Pakete.....	167
Examples.....	167
Daten aus MySQL-Datenbanken lesen.....	167
Allgemeines.....	167
Grenzen verwenden.....	167
Daten aus MongoDB-Datenbanken lesen.....	167
Kapitel 35: E / A für Fremdtabellen (Excel, SAS, SPSS, Stata).....	169
Examples.....	169
Daten mit rio importieren.....	169
Excel-Dateien importieren.....	169
Lesen von Excel-Dateien mit dem xlsx-Paket.....	170
Lesen von Excel-Dateien mit dem XLconnect-Paket.....	170
Lesen von Excel-Dateien mit dem Paket openxlsx.....	171
Lesen von Excel-Dateien mit dem Paket readxl.....	171
Lesen von Excel-Dateien mit dem RODBC-Paket.....	172
Lesen von Excel-Dateien mit dem gdata-Paket.....	173
Lesen und Schreiben von Stata-, SPSS- und SAS-Dateien.....	173
Importieren oder Exportieren einer Federdatei.....	175
Kapitel 36: E / A für geographische Daten (Shapefiles usw.).....	176
Einführung.....	176
Examples.....	176
Importieren und Exportieren von Shapefiles.....	176
Kapitel 37: E / A für Rasterbilder.....	177
Einführung.....	177
Examples.....	177
Laden Sie ein Multilayer-Raster.....	177
Kapitel 38: Einführung in geographische Karten.....	179
Einführung.....	179
Examples.....	179
Grundlegende Kartenerstellung mit map () aus den Paketkarten.....	179

50 Landeskarten und erweiterte Choroplethen mit Google Viz.....	183
Interaktive Kartenpläne.....	184
Dynamische HTML-Karten mit Broschüre erstellen.....	186
Dynamische Prospektkarten in glänzenden Anwendungen.....	188
Kapitel 39: Eingabe und Ausgabe.....	191
Bemerkungen.....	191
Examples.....	191
Datenrahmen lesen und schreiben.....	191
Schreiben.....	191
lesen.....	191
Weitere Ressourcen.....	192
Kapitel 40: Erinnerung an Beispiele.....	193
Einführung.....	193
Examples.....	193
Datentypen.....	193
Vektoren.....	193
Matrizen.....	193
Datenrahmen.....	193
Listen.....	193
Umgebungen.....	194
Plotten (mit Plot).....	194
Häufig verwendete Funktionen.....	194
Kapitel 41: Erstellen von Berichten mit RMarkdown.....	196
Examples.....	196
Tabellen drucken.....	196
Einschließlich LaTeX Preamble-Befehle.....	198
Einschließlich Bibliographien.....	199
Grundlegende R-Markdown-Belegstruktur.....	200
R-Markdown-Codeabschnitte.....	200
Beispiel für ein R-Markdown-Dokument.....	200
Konvertierung von R-Markdown in andere Formate.....	201

Kapitel 42: Extrahieren und Auflisten von Dateien in komprimierten Archiven	203
Examples	203
Extrahieren von Dateien aus einem ZIP-Archiv	203
Dateien in einem ZIP-Archiv auflisten	203
Dateien in einem .tar-Archiv auflisten	203
Extrahieren von Dateien aus einem .tar-Archiv	203
Extrahieren Sie alle ZIP-Archive in einem Verzeichnis	204
Kapitel 43: Faktoren	205
Syntax	205
Bemerkungen	205
Die Ganzzahl der Ebene zuordnen	206
Moderne Nutzung von Faktoren	206
Examples	207
Grundlegende Faktorenbildung	207
Faktorstufen mit einer Liste konsolidieren	208
Ebenen über factor konsolidieren (factor_approach)	209
Ebenen mit ifelse konsolidieren (ifelse_approach)	209
list_approach mit einer Liste list_approach (list_approach)	210
Benchmarking für jeden Ansatz	210
Faktoren	210
Faktoren ändern und neu ordnen	212
Wiederherstellungsfaktoren von Null	217
Problem	217
Lösung	218
Kapitel 44: Farbschemata für Grafiken	219
Examples	219
viridis - druck- und farbbblindfreundliche Paletten	219
RColorBrewer	222
Eine praktische Funktion, um einen Farbvektor zu erkennen	224
Farbraum - Klicken und Ziehen der Benutzeroberfläche für Farben	225
grundlegende R-Farbfunktionen	226
Farbbblindfreundliche Paletten	227

Kapitel 45: Fehlende Werte	230
Einführung	230
Bemerkungen	230
Examples	230
Untersuchen der fehlenden Daten	230
Lesen und Schreiben von Daten mit NA-Werten	230
Verwendung von NAs verschiedener Klassen	231
TRUE / FALSE und / oder NA	231
Fehlende Werte auslassen oder ersetzen	232
Fehlende Werte neu codieren	232
Fehlende Werte entfernen	233
Fehlende Werte aus den Berechnungen ausschließen	233
Kapitel 46: Fehlertoleranter / belastbarer Code	234
Parameter	234
Bemerkungen	234
tryCatch	234
Auswirkungen der Auswahl bestimmter Rückgabewerte der Handlerfunktionen	234
Warnmeldung "Unerwünscht"	235
Examples	235
TryCatch () verwenden	235
Funktionsdefinition mit tryCatch	235
Dinge testen	236
Untersuchung der Ausgabe	237
Kapitel 47: Formel	238
Examples	238
Die Grundlagen der Formel	238
Erstellen Sie lineare, quadratische und zweite Interaktionsbedingungen	239
Kapitel 48: Fourier-Serie und Transformationen	242
Bemerkungen	242
Examples	243
Die Fourierreihe	243

Kapitel 49: Funktionale Programmierung	250
Examples	250
Integrierte Funktionen höherer Ordnung	250
Kapitel 50: Funktionsauswahl in R - Entfernen überflüssiger Funktionen	251
Examples	251
Entfernen von Features mit einer Varianz von null oder nahezu null	251
Entfernen von Features mit hoher NA-Anzahl	251
Entfernen von stark korrelierten Features	251
Kapitel 51: Generalisierte lineare Modelle	253
Examples	253
Logistische Regression auf Titanic-Dataset	253
Kapitel 52: ggplot2	256
Bemerkungen	256
Examples	256
Streudiagramme	256
Mehrere Plots anzeigen	257
Bereiten Sie Ihre Daten für das Plotten vor	261
Fügen Sie dem Plot horizontale und vertikale Linien hinzu	263
Fügen Sie für alle kategorialen Variablen eine gemeinsame horizontale Linie hinzu	263
Fügen Sie für jede kategoriale Variable eine horizontale Linie hinzu	263
Horizontale Linie über gruppierten Balken hinzufügen	263
Vertikale Linie hinzufügen	263
Vertikales und horizontales Balkendiagramm	263
Geigenhandlung	263
Erstellen Sie grundlegende Diagramme mit qplot	263
Kapitel 53: Glänzend	266
Examples	266
App erstellen	266
Eine Datei	266
Zwei Dateien	266
Erstellen Sie die ui.R Datei	266

Erstellen Sie die server.R Datei.....	267
Radio knopf.....	267
Kontrollkästchen Gruppe.....	267
Kästchen auswählen.....	268
Starten Sie eine Shiny-App.....	269
1. Zwei Dateien App.....	269
2. Eine Datei App.....	270
Kontrollieren Sie Widgets.....	270
Debuggen.....	272
Showcase-Modus.....	272
Reactive Log Visualizer.....	272
Kapitel 54: GPU-beschleunigtes Computing.....	274
Bemerkungen.....	274
Examples.....	274
gpuR gpuMatrix-Objekte.....	274
gpuR vclMatrix-Objekte.....	274
Kapitel 55: Hashmaps.....	276
Examples.....	276
Umgebungen als Hash-Maps.....	276
Einführung.....	276
Einfügung.....	276
Schlüsselsuche.....	277
Überprüfen der Hashkarte.....	277
Flexibilität.....	278
Einschränkungen.....	279
Paket: Hash.....	280
Paket: listenv.....	281
Kapitel 56: Heatmap und Heatmap.2.....	282
Examples.....	282
Beispiele aus der offiziellen Dokumentation.....	282
stats :: heatmap.....	282

Beispiel 1 (Grundnutzung).....	282
Beispiel 2 (kein Spalten-Dendrogramm (noch Umordnen)).....	283
Beispiel 3 ("Nichts").....	283
Beispiel 4 (mit reorder ()).....	284
Beispiel 5 (keine Nachbestellung ()).....	285
Beispiel 6 (leicht künstlich mit Farbbalken, ohne zu bestellen).....	286
Beispiel 7 (leicht künstlich mit Farbbalken, mit Bestellung).....	287
Beispiel 8 (Verwenden Sie für variable Cluster eher den Abstand basierend auf cor ()).....	288
Optimierungsparameter in heatmap.2.....	290
Kapitel 57: Hierarchische lineare Modellierung.....	296
Examples.....	296
Grundmodellanpassung.....	296
Kapitel 58: Hierarchisches Clustering mit hclust.....	297
Einführung.....	297
Bemerkungen.....	297
Examples.....	297
Beispiel 1 - Grundlegende Verwendung von hclust, Anzeige des Dendrogramms, Plotcluster.....	297
Beispiel 2 - hclust und Ausreißer.....	301
Kapitel 59: Implementieren Sie das State Machine Pattern mithilfe der S4-Klasse.....	304
Einführung.....	304
Examples.....	304
Zeilen mit State Machine analysieren.....	304
Kapitel 60: JSON.....	318
Examples.....	318
JSON zu / von R-Objekten.....	318
Kapitel 61: Klassen.....	320
Einführung.....	320
Bemerkungen.....	320
Examples.....	320
Vektoren.....	320
Überprüfen Sie die Klassen.....	320

Vektoren und Listen.....	321
Kapitel 62: Kombinatorik.....	323
Examples.....	323
Aufzählen von Kombinationen einer angegebenen Länge.....	323
Ohne Ersatz.....	323
Mit ersatz.....	323
Zählen von Kombinationen einer angegebenen Länge.....	324
Ohne Ersatz.....	324
Mit ersatz.....	324
Kapitel 63: Kontrollieren Sie Flussstrukturen.....	325
Bemerkungen.....	325
Struktur von for-Schleifen optimieren.....	325
Vektorisieren für Loops.....	326
Examples.....	327
Grundlegend für die Schleifenkonstruktion.....	327
Optimale Konstruktion einer for-Schleife.....	327
Schlecht für Schleifen optimiert.....	328
Gut für die Schleife optimiert.....	328
vapply Funktion.....	328
colMeans Funktion.....	328
Effizienzvergleich.....	328
Die anderen Looping-Konstrukte: während und wiederholen.....	329
Die while Schleife.....	329
Die repeat.....	330
Mehr zur break.....	331
Kapitel 64: Lauflängencodierung.....	333
Bemerkungen.....	333
Erweiterungen.....	333
Examples.....	333
Lauflängencodierung mit "rle".....	333
Identifizierung und Gruppierung nach Läufen in Basis R.....	334

Identifizieren und Gruppieren nach Läufen in data.table.....	334
Laufgrößencodierung zum Komprimieren und Dekomprimieren von Vektoren.....	335
Kapitel 65: Lesen und Schreiben von Tabellendaten in Klartextdateien (CSV, TSV usw.).....	337
Syntax.....	337
Parameter.....	337
Bemerkungen.....	338
Examples.....	338
.Csv-Dateien importieren.....	338
Import mit Basis R.....	338
Anmerkungen.....	338
Importieren mit Paketen.....	339
Importieren mit data.table.....	339
Anmerkungen.....	340
.Tsv-Dateien als Matrizen importieren (Basis-R).....	340
.Csv-Dateien exportieren.....	341
Export mit Basis R.....	341
Exportieren mit Paketen.....	341
Importieren Sie mehrere CSV-Dateien.....	341
Dateien mit fester Breite importieren.....	342
Import mit Basis R.....	342
Importieren mit readr.....	342
Kapitel 66: Lineare Modelle (Regression).....	344
Syntax.....	344
Parameter.....	344
Examples.....	345
Lineare Regression im mtcars-Dataset.....	345
Die Regression plotten (Basis).....	346
Gewichtung.....	348
Prüfung auf Nichtlinearität mit polynomialer Regression.....	350
Qualitätsprüfung.....	353
Verwenden der Funktion "Vorhersagen".....	354

Kapitel 67: Listen	356
Examples	356
Schnelle Einführung in die Listen	356
Einführung in die Listen	358
Gründe für die Verwendung von Listen	359
Konvertieren Sie eine Liste in einen Vektor, wobei leere Listenelemente leer bleiben	360
Serialisierung: Verwenden von Listen zur Weitergabe von Informationen	361
Kapitel 68: Lösen von ODEs in R	363
Syntax	363
Parameter	363
Bemerkungen	363
Examples	363
Das Lorenz-Modell	363
Lotka-Volterra oder: Beute gegen Raubtier	365
ODEs in kompilierten Sprachen - Definition in R	366
ODEs in kompilierten Sprachen - Definition in C	367
ODEs in kompilierten Sprachen - Definition in Fortran	368
ODEs in kompilierten Sprachen - ein Benchmark-Test	370
Kapitel 69: Lubridat	372
Syntax	372
Bemerkungen	372
Examples	372
Analysieren von Datumsangaben und Datumsangaben von Zeichenfolgen mit Lubridat	373
Termine	373
Datumsangaben	373
Dienstprogrammfunktionen	373
Parser-Funktionen	374
Analysieren von Datum und Uhrzeit in Lubridate	375
Datum und Uhrzeit in Lubridate bearbeiten	375
Momente	375
Intervalle, Dauern und Perioden	376
Rundungsdaten	377

Unterschied zwischen Zeitraum und Dauer	378
Zeitzone	379
Kapitel 70: Maschinelles lernen	380
Examples	380
Ein zufälliges Gesamtstrukturmodell erstellen	380
Kapitel 71: Matrizen	381
Einführung	381
Examples	381
Matrizen erstellen	381
Kapitel 72: Meta: Dokumentationsrichtlinien	383
Bemerkungen	383
Examples	383
Gute Beispiele machen	383
Stil	383
Eingabeaufforderungen	383
Konsolenausgabe	383
Zuordnung	384
Code-Kommentare	384
Abschnitte	384
Kapitel 73: Musterabgleich und Ersatz	385
Einführung	385
Syntax	385
Bemerkungen	385
Unterschiede zu anderen Sprachen	385
Spezialisierte Pakete	385
Examples	385
Substitutionen machen	385
Übereinstimmungen finden	386
Gibt es eine Übereinstimmung?	386
Orte abgleichen	386
Übereinstimmende Werte	386

Einzelheiten	387
Zusammenfassung der Spiele	387
Single und Global Match	387
Finden Sie Übereinstimmungen in großen Datensätzen	389
Kapitel 74: Netzwerkanalyse mit dem igraph-Paket	390
Examples	390
Einfache gerichtete und nicht gerichtete Netzwerkgrafik	390
Kapitel 75: Nichtstandardmäßige Bewertung und Standardbewertung	392
Einführung	392
Examples	392
Beispiele mit Standard-Dplyr-Verben	392
Kapitel 76: Numerische Klassen und Speichermodi	394
Examples	394
Numerisch	394
Kapitel 77: Objektorientierte Programmierung in R	396
Einführung	396
Examples	396
S3	396
Kapitel 78: Operationen einstellen	398
Bemerkungen	398
Examples	398
Legen Sie Operatoren für Vektorpaare fest	398
Sets vergleichen	398
Sets kombinieren	398
Legen Sie die Mitgliedschaft für Vektoren fest	399
Kartesische oder "Kreuz" -Produkte von Vektoren	399
Funktionen auf Kombinationen anwenden	400
Erstellen Sie eindeutige Elemente, und legen Sie doppelte Elemente aus einem Vektor ab	400
Messsatzüberdeckungen / Venn-Diagramme für Vektoren	401
Kapitel 79: ordentlich	402
Examples	402

Tbl_df erstellen.....	402
Tidyverse: eine Übersicht.....	402
Was ist tidyverse ?.....	402
Wie benutze ich es?.....	403
Was sind das für Pakete?.....	403
Kapitel 80: Pakete installieren.....	405
Syntax.....	405
Parameter.....	405
Bemerkungen.....	405
Verwandte Dokumente.....	405
Examples.....	405
Laden Sie Pakete aus Repositorys herunter und installieren Sie sie.....	405
Verwendung von CRAN.....	405
Verwendung von Bioconductor.....	406
Paket aus lokaler Quelle installieren.....	407
Installiere Pakete von GitHub.....	407
Verwenden eines CLI-Paketmanagers - grundlegende Pacman-Verwendung.....	409
Installieren Sie die lokale Entwicklungsversion eines Pakets.....	409
Kapitel 81: Pakete mit devtools erstellen.....	411
Einführung.....	411
Bemerkungen.....	411
Examples.....	411
Pakete erstellen und verteilen.....	411
Erstellung der Dokumentation.....	411
Bau des Paketskeletts.....	412
Edition der Paketeigenschaften.....	412
1. Beschreibung des Pakets.....	412
2. Optionale Ordner.....	413
Fertigstellung und Build.....	413
Verteilung Ihres Pakets.....	413

Durch Github.....	413
Durch CRAN.....	413
Vignetten erstellen.....	414
Bedarf.....	414
Vignetten-Erstellung.....	414
Kapitel 82: Pakete prüfen.....	415
Einführung.....	415
Bemerkungen.....	415
Examples.....	415
Paketinformationen anzeigen.....	415
Zeigen Sie die integrierten Datensätze des Pakets an.....	415
Listen Sie die exportierten Funktionen eines Pakets auf.....	415
Paketversion anzeigen.....	415
Geladene Pakete in der aktuellen Sitzung anzeigen.....	416
Kapitel 83: Parallelverarbeitung.....	417
Bemerkungen.....	417
Examples.....	417
Parallelverarbeitung mit foreach-Paket.....	417
Parallelverarbeitung mit Parallelpaket.....	418
Zufallszahlengenerierung.....	419
mcpipelineDo.....	420
Beispiel.....	420
Andere Beispiele.....	420
Kapitel 84: Pivot und Unpivot mit data.table.....	422
Syntax.....	422
Parameter.....	422
Bemerkungen.....	422
Examples.....	422
Tabellendaten mit data.table - I.....	422
Tabellendaten mit data.table - II pivotieren und aufheben.....	424
Kapitel 85: R in LaTeX mit Knitr.....	426

Syntax.....	426
Parameter.....	426
Bemerkungen.....	426
Examples.....	427
R in Latex mit Knitr und Code Externalisierung.....	427
R in Latex mit Knitr und Inline Code Chunks.....	428
R in LaTeX mit Knitr und Internal Code Chunks.....	428
Kapitel 86: R Markdown-Notizbücher (von RStudio).....	429
Einführung.....	429
Examples.....	429
Ein Notizbuch erstellen.....	429
Brocken einfügen.....	430
Chunk-Code ausführen.....	431
Code in Chunks aufteilen.....	431
Ausführungsfortschritt.....	432
Mehrere Chunks ausführen.....	433
Vorschau Ausgabe.....	434
Speichern und teilen.....	435
Kapitel 87: Randomisierung.....	436
Einführung.....	436
Bemerkungen.....	436
Examples.....	436
Zufällige Draws und Permutationen.....	436
Zufällige Permutation.....	436
Zieht ohne Ersatz.....	437
Zieht mit Ersatz.....	437
Ändern der Draw-Wahrscheinlichkeiten.....	438
Samen setzen.....	439
Kapitel 88: Raster- und Bildanalyse.....	441
Einführung.....	441
Examples.....	441

Berechnung der GLCM-Textur.....	441
Mathematische Morphologien.....	443
Kapitel 89: räumliche Analyse.....	446
Examples.....	446
Erstellen Sie räumliche Punkte aus dem XY-Datensatz.....	446
Importieren einer Formdatei (.shp).....	447
rgdal.....	447
Raster.....	447
tmap.....	448
Kapitel 90: Rcpp.....	449
Examples.....	449
Inline-Code kompilieren.....	449
Rcpp-Attribute.....	449
Rcpp mit Plugins erweitern.....	451
Zusätzliche Build-Abhängigkeiten angeben.....	451
Kapitel 91: Rechenzeichen.....	452
Bemerkungen.....	452
Examples.....	452
Reichweite und Zusatz.....	452
Addition und Subtraktion.....	453
Kapitel 92: Recycling.....	456
Bemerkungen.....	456
Examples.....	456
Recycling-Verwendung beim Subsetting.....	456
Kapitel 93: Reguläre Ausdrücke (Regex).....	458
Einführung.....	458
Bemerkungen.....	458
Charakterklassen.....	458
Quantifizierer.....	458
Anfangs- und Endzeilenindikatoren.....	458
Unterschiede zu anderen Sprachen.....	458

Zusätzliche Ressourcen	459
Examples	459
Whitespace beseitigen	459
Leerzeichen trimmen	459
Führenden Whitespace entfernen	459
Nachlaufende Whitespace entfernen	460
Alle Leerzeichen entfernen	460
Bestätigen Sie ein Datum im Format "JJJJMMTT"	460
Validieren Sie die Postabkürzungen der US-Bundesstaaten	461
Bestätigen Sie die US-Telefonnummern	462
Escape-Zeichen in Regex-Mustern	462
Unterschiede zwischen Perl- und POSIX-Regex	463
Vorausschau / Rückblick	463
Kapitel 94: Reguläre Ausdrücke in R	464
Einführung	464
Examples	464
Verwenden Sie `grep`, um eine Zeichenfolge in einem Zeichenvektor zu finden	464
Kapitel 95: Reproduzierbares R	466
Einführung	466
Bemerkungen	466
Verweise	466
Examples	466
Reproduzierbarkeit der Daten	466
dput() und dget()	466
Paket-Reproduzierbarkeit	467
Kapitel 96: RESTful R Services	468
Einführung	468
Examples	468
opencpu Apps	468
Kapitel 97: RMarkdown und Knitr-Präsentation	469
Syntax	469

Parameter.....	469
Bemerkungen.....	469
Unteroptionsparameter:.....	469
Examples.....	472
Rstudio Beispiel.....	472
Hinzufügen einer Fußzeile zu einer loslides-Präsentation.....	473
Kapitel 98: RODBC.....	476
Examples.....	476
Verbindung zu Excel-Dateien über RODBC.....	476
SQL Server-Verwaltungsdatenbankverbindung zum Abrufen einzelner Tabellen.....	476
Verbindung zu relationalen Datenbanken.....	476
Kapitel 99: Rohoperatoren (%>% und andere).....	477
Einführung.....	477
Syntax.....	477
Parameter.....	477
Bemerkungen.....	477
Pakete, die %>%.....	477
Dokumentation finden.....	478
Hotkeys.....	478
Überlegungen zur Leistung.....	478
Examples.....	478
Grundlegende Verwendung und Verkettung.....	478
Funktionsabläufe.....	479
Zuordnung mit% <>%.....	480
Inhalte mit% \$% verfügbar machen.....	481
Verwendung der Pipe mit Dplyr und ggplot2.....	481
Nebeneffekte mit% T>% erstellen.....	482
Kapitel 100: roxygen2.....	484
Parameter.....	484
Examples.....	484
Dokumentation eines Pakets mit roxygen2.....	484
Schreiben mit roxygen2.....	484

Erstellung der Dokumentation	485
Kapitel 101: R-Version wird aktualisiert	486
Einführung	486
Examples	486
Installation von der R-Website	486
Aktualisierung von R aus mit dem installr-Paket	486
Entscheidung über die alten Pakete	487
Pakete aktualisieren	490
Überprüfen Sie die R-Version	491
Kapitel 102: Schreibfunktionen in R	492
Examples	492
Benannte Funktionen	492
Anonyme Funktionen	493
RStudio-Codeausschnitte	493
Spaltennamen als Argument einer Funktion übergeben	494
Kapitel 103: Selbstbeobachtung	496
Examples	496
Funktionen zum Lernen von Variablen	496
Kapitel 104: Spaltenweiser Betrieb	498
Examples	498
Summe jeder Spalte	498
Kapitel 105: Spark-API (SparkR)	500
Bemerkungen	500
Examples	500
Spark-Kontext einrichten	500
Spark-Kontext in R einrichten	500
Spark-Cluster abrufen	500
Daten zwischenspeichern	500
Erstellen Sie RDDs (Resilient Distributed Datasets)	501
Vom Datenrahmen:	501
Von csv:	501

Kapitel 106: Split-Funktion	503
Examples	503
Grundlegende Verwendung von Split	503
Verwenden von Split im Split-Apply-Combine-Paradigma	505
Kapitel 107: sqldf	507
Examples	507
Grundlegende Anwendungsbeispiele	507
Kapitel 108: Standardisieren Sie Analysen, indem Sie eigenständige R-Skripts schreiben	510
Einführung	510
Bemerkungen	510
Examples	510
Die grundlegende Struktur des Standalone-R-Programms und wie es aufgerufen wird	510
Das erste eigenständige R-Skript	510
Eigenständiges R-Skript vorbereiten	511
Linux / Mac	511
Windows	511
Verwenden von littler zum Ausführen von R-Skripten	512
Kleiner installieren	512
Von R:	512
Verwenden von apt-get (Debian, Ubuntu):	512
Verwenden von littler mit Standard-.r-Skripts	512
Littler für Shebanged- Skripte verwenden	513
Kapitel 109: String-Manipulation mit dem Stringi-Paket	514
Bemerkungen	514
Examples	514
Zählmuster im String	514
Zeichenketten duplizieren	515
Vektoren einfügen	515
Text nach einem festen Muster aufteilen	515
Kapitel 110: Strsplit-Funktion	517
Syntax	517

Examples.....	517
Einführung.....	517
Kapitel 111: Subsetting.....	519
Einführung.....	519
Bemerkungen.....	519
Examples.....	520
Atomare Vektoren.....	520
Listen.....	522
Matrizen.....	523
Auswahl einzelner Matrixeinträge anhand ihrer Positionen.....	524
Datenrahmen.....	525
Andere Objekte.....	526
Vektor-Indizierung.....	527
Elementweise Matrixoperationen.....	528
Einige Funktionen, die mit Matrizen verwendet werden.....	528
Kapitel 112: Text-Mining.....	530
Examples.....	530
Daten auskratzen, um N-Gramm-Word-Clouds zu erstellen.....	530
Kapitel 113: Überlebensanalyse.....	534
Examples.....	534
Random Forest Survival-Analyse mit randomForestSRC.....	534
Einführung - Grundlegende Anpassung und Darstellung von parametrischen Überlebensmodellen	535
Kaplan Meier schätzt die Überlebenskurven und Risikotabellen mit Überlebenden.....	536
Kapitel 114: Umfang der Variablen.....	539
Bemerkungen.....	539
Examples.....	539
Umgebungen und Funktionen.....	539
Unterfunktionen.....	540
Globale Zuordnung.....	540
Explizite Zuweisung von Umgebungen und Variablen.....	541
Funktion beenden.....	542
Pakete und Maskierung.....	542

Kapitel 115: Umformen mit Tidy	543
Einführung	543
Examples	543
Umformung vom langen in das breite Format mit Spread ()	543
Umformen von Breit zu Langformat mit Gather ()	544
h21	544
Kapitel 116: Variablen	545
Examples	545
Variablen, Datenstrukturen und grundlegende Operationen	545
Arten von Datenstrukturen	546
Gemeinsame Operationen und einige Warnhinweise	546
Beispielobjekte	547
Einige Vektoroperationen	547
Einige Vektoroperationen Warnungen!	547
Einige Matrixoperationen Warnung!	547
"Private" Variablen	548
Kapitel 117: Vektoren erstellen	549
Examples	549
Zahlenfolge	549
seq ()	549
Vektoren	550
Benannte Vektoren erstellen	552
Erweitern eines Vektors mit der Funktion rep ()	553
Vektoren von eingebauten Konstanten: Folgen von Buchstaben und Monatsnamen	554
Kapitel 118: Verarbeitung natürlicher Sprache	556
Einführung	556
Examples	556
Erstellen Sie eine Termfrequenzmatrix	556
Kapitel 119: Veröffentlichung	558
Einführung	558
Bemerkungen	558

Examples.....	558
Tabellen formatieren.....	558
Drucken in einfachem Text.....	558
Drucken von Tabellen mit Trennzeichen.....	558
Weitere Ressourcen.....	559
Ganze Dokumente formatieren.....	559
Weitere Ressourcen.....	559
Kapitel 120: Verteilungsfunktionen.....	560
Einführung.....	560
Bemerkungen.....	560
Examples.....	560
Normalverteilung.....	560
Binomialverteilung.....	561
Kapitel 121: Verwenden von Texreg zum Exportieren von Modellen in Papierform.....	565
Einführung.....	565
Bemerkungen.....	565
Links.....	565
Examples.....	565
Lineare Regressionsergebnisse drucken.....	565
Kapitel 122: Verwendung der Rohrzuordnung in Ihrem eigenen Paket.....	567
Einführung.....	567
Examples.....	567
Pipe in eine Utility-Funktionsdatei einfügen.....	567
Kapitel 123: Wahrscheinlichkeitsverteilungen mit R.....	568
Examples.....	568
PDF und PMF für verschiedene Distributionen in R.....	568
Kapitel 124: Web Scraping und Parsing.....	569
Bemerkungen.....	569
Rechtmäßigkeit.....	569
Examples.....	569
Grundschaßen mit Rvest.....	569

Verwendung von rvest, wenn ein Login erforderlich ist.....	570
Kapitel 125: Web-Crawling in R.....	572
Examples.....	572
Standard-Kratzverfahren mit dem RCurl-Paket.....	572
Kapitel 126: xgboost.....	573
Examples.....	573
Cross Validation und Tuning mit xgboost.....	573
Kapitel 127: Zeichenfolgen lesen und schreiben.....	576
Bemerkungen.....	576
Examples.....	576
Zeichenketten drucken und anzeigen.....	576
Lesen oder Schreiben einer Dateiverbindung.....	578
Erfassen Sie die Ausgabe des Betriebssystembefehls.....	579
Funktionen, die einen Zeichenvektor zurückgeben.....	579
Funktionen, die einen Datenrahmen zurückgeben.....	579
Kapitel 128: Zeitreihen und Prognosen.....	581
Bemerkungen.....	581
Examples.....	581
Explorative Datenanalyse mit Zeitreihendaten.....	581
Ts-Objekt erstellen.....	582
Kapitel 129: Zufallsforstalgorithmus.....	584
Einführung.....	584
Examples.....	584
Grundlegende Beispiele - Klassifizierung und Regression.....	584
Kapitel 130: Zufallszahlengenerator.....	586
Examples.....	586
Zufällige Permutationen.....	586
Reproduzierbarkeit des Zufallszahlengenerators.....	586
Zufallszahlen mit verschiedenen Dichtefunktionen erzeugen.....	587
Gleichmäßige Verteilung zwischen 0 und 10.....	587
Normalverteilung mit 0 Mittelwert und Standardabweichung von 1.....	587

Binomialverteilung mit 10 Versuchen und einer Erfolgswahrscheinlichkeit von 0,5.....	587
Geometrische Verteilung mit 0,2 Erfolgswahrscheinlichkeit.....	587
Hypergeometrische Verteilung mit 3 weißen Kugeln, 10 schwarzen Kugeln und 5 Unentschieden.....	588
Negative Binomialverteilung mit 10 Versuchen und einer Erfolgswahrscheinlichkeit von 0,8.....	588
Poissonverteilung mit Mittelwert und Varianz (Lambda) von 2.....	588
Exponentialverteilung mit der Rate von 1,5.....	588
Logistische Verteilung mit 0 Standort und Skala von 1.....	588
Chi-Quadrat- Verteilung mit 15 Freiheitsgraden.....	588
Beta- Verteilung mit den Formparametern $a = 1$ und $b = 0,5$	588
Gamma- Verteilung mit Formparameter 3 und Skala = 0,5.....	589
Cauchy- Verteilung mit 0 Standort und Skala von 1.....	589
Log-Normalverteilung mit 0-Mittelwert und Standardabweichung von 1 (auf Log-Skala).....	589
Weibullverteilung mit Formparameter 0,5 und Skala von 1.....	589
Wilcoxon- Verteilung mit 10 Beobachtungen in der ersten Probe und 20 in der zweiten.....	589
Multinomialverteilung mit 5 Objekt und 3 Boxen mit den angegebenen Wahrscheinlichkeiten.....	589
Kapitel 131: Zwang	590
Einführung.....	590
Examples.....	590
Impliziter Zwang.....	590
Credits	591



You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [r-language](#)

It is an unofficial and free R Language ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official R Language.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit R Language

Bemerkungen

Bearbeiten von R-Dokumenten im Stapelüberlauf

In den [Dokumentationsrichtlinien](#) finden Sie allgemeine Regeln zum Erstellen von Dokumentation.

Einige Merkmale von R, die Einwanderer aus anderen Sprachen ungewöhnlich finden

- Im Gegensatz zu anderen Sprachen benötigen Variablen in R keine Typdeklaration.
- Dieselben Variablen können bei Bedarf unterschiedliche Datentypen zu unterschiedlichen Zeitpunkten zugewiesen werden.
- Die Indizierung atomarer Vektoren und Listen beginnt bei 1 und nicht bei 0.
- R- `arrays` (und der Spezialfall von Matrizen) haben ein `dim` Attribut, das sie von den "atomaren Vektoren" von R unterscheidet, die keine Attribute haben.
- Mit einer Liste in R können Sie verschiedene Objekte unter einem Namen (d. H. Dem Namen der Liste) geordnet zusammenstellen. Diese Objekte können **Matrizen** , **Vektoren** , **Datenrahmen** , **sogar andere Listen** usw. sein. Es ist nicht einmal erforderlich, dass diese Objekte in irgendeiner Weise miteinander in Beziehung stehen.
- [Recycling](#)
- [Fehlende Werte](#)

Examples

R installieren

Sie möchten [RStudio](#) nach der Installation von R installieren. RStudio ist eine Entwicklungsumgebung für R, die viele Programmieraufgaben vereinfacht.

Nur Windows:

[Visual Studio](#) (ab Version 2015 Update 3) enthält jetzt eine Entwicklungsumgebung für R namens [R Tools](#) , die einen Live-Interpreter, IntelliSense und ein Debugging-Modul enthält. Wenn Sie diese Methode wählen, müssen Sie R nicht wie im folgenden Abschnitt beschrieben installieren.

Für Windows

1. Gehen Sie zur [CRAN](#)- Website, klicken Sie auf R für Windows herunterladen und laden Sie die neueste Version von R herunter.
2. Klicken Sie mit der rechten Maustaste auf die Installationsdatei, und führen Sie den Administrator als Administrator aus.
3. Wählen Sie die Betriebssystemsprache für die Installation.
4. Befolgen Sie die Anweisungen zur Installation.

Für OSX / macOS

Alternative 1

(0. Stellen Sie sicher, dass [XQuartz](#) installiert ist.)

1. Gehen Sie zur [CRAN](#)- Website und laden Sie die neueste Version von R herunter.
2. Öffnen Sie das Disk-Image und führen Sie das Installationsprogramm aus.
3. Befolgen Sie die Anweisungen zur Installation.

Dadurch werden sowohl R als auch R-MacGUI installiert. Die GUI wird in den Ordner / Applications / als R.app gestellt. Dort kann sie entweder doppelt angeklickt oder in den Doc gezogen werden. Wenn eine neue Version veröffentlicht wird, wird R.app durch die (Neu-) Installation überschrieben, frühere Hauptversionen von R werden jedoch beibehalten. Der eigentliche R-Code befindet sich im Verzeichnis /Library/Frameworks/R.Framework/Versions/. Die Verwendung von R innerhalb von RStudio ist ebenfalls möglich und würde denselben R-Code mit einer anderen GUI verwenden.

Alternative 2

1. Installieren Sie homebrew (den fehlenden Paketmanager für macOS), indem Sie den Anweisungen auf <https://brew.sh/> folgen.
2. `brew install R`

Diejenigen, die sich für die zweite Methode entscheiden, sollten sich dessen bewusst sein, dass der Betreuer der Mac-Gabel davon abrät und nicht auf Fragen zu Problemen in der R-SIG-Mac-Mailingliste antwortet.

Für Debian, Ubuntu und Derivate

Sie können die Version von R, die Ihrer Distribution entspricht, über `apt-get` . Diese Version wird jedoch häufig weit hinter der aktuellsten Version von CRAN liegen. Sie können CRAN zu Ihrer Liste der erkannten "Quellen" hinzufügen.

```
sudo apt-get install r-base
```

Sie können eine neuere Version direkt von CRAN erhalten, indem Sie CRAN zu Ihrer Quellenliste hinzufügen. Folgen Sie den [Anweisungen](#) von CRAN für weitere Details. Beachten Sie insbesondere die Notwendigkeit, dies auch auszuführen, damit Sie `install.packages()` . Linux-Pakete werden normalerweise als Quelldateien verteilt und müssen kompiliert werden:

```
sudo apt-get install r-base-dev
```

Für Red Hat und Fedora

```
sudo dnf install R
```

Für Archlinux

R ist direkt im `Extra` Paket-Repo verfügbar.

```
sudo pacman -S r
```

Weitere Informationen zur Verwendung von R unter Archlinux finden Sie auf der [ArchWiki R-Seite](#) .

Hallo Welt!

```
"Hello World!"
```

Lesen Sie auch [die ausführliche Beschreibung, wie, wann, ob und warum eine Zeichenfolge gedruckt werden soll](#) .

Hilfe bekommen

Sie können die Funktion `help()` oder `?` Verwenden `?` um auf Dokumentationen zuzugreifen und in R. nach Hilfe zu suchen. Für noch allgemeinere Suchen können Sie `help.search()` oder `??` .

```
#For help on the help function of R
help()

#For help on the paste function
help(paste)      #OR
help("paste")    #OR
?paste           #OR
?"paste"
```

Besuchen Sie <https://www.r-project.org/help.html> für weitere Informationen

Der interaktive Modus

Die grundlegendste Art, R zu verwenden, ist der *interaktive* Modus. Sie geben Befehle ein und erhalten sofort das Ergebnis von R.

Verwendung von R als Taschenrechner

Starten Sie R , indem Sie `R` an der Eingabeaufforderung des Betriebssystems oder durch die Ausführung `RGui` unter Windows. Unten sehen Sie einen Screenshot einer interaktiven R-Sitzung unter Linux:

```
user:~$ R

R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

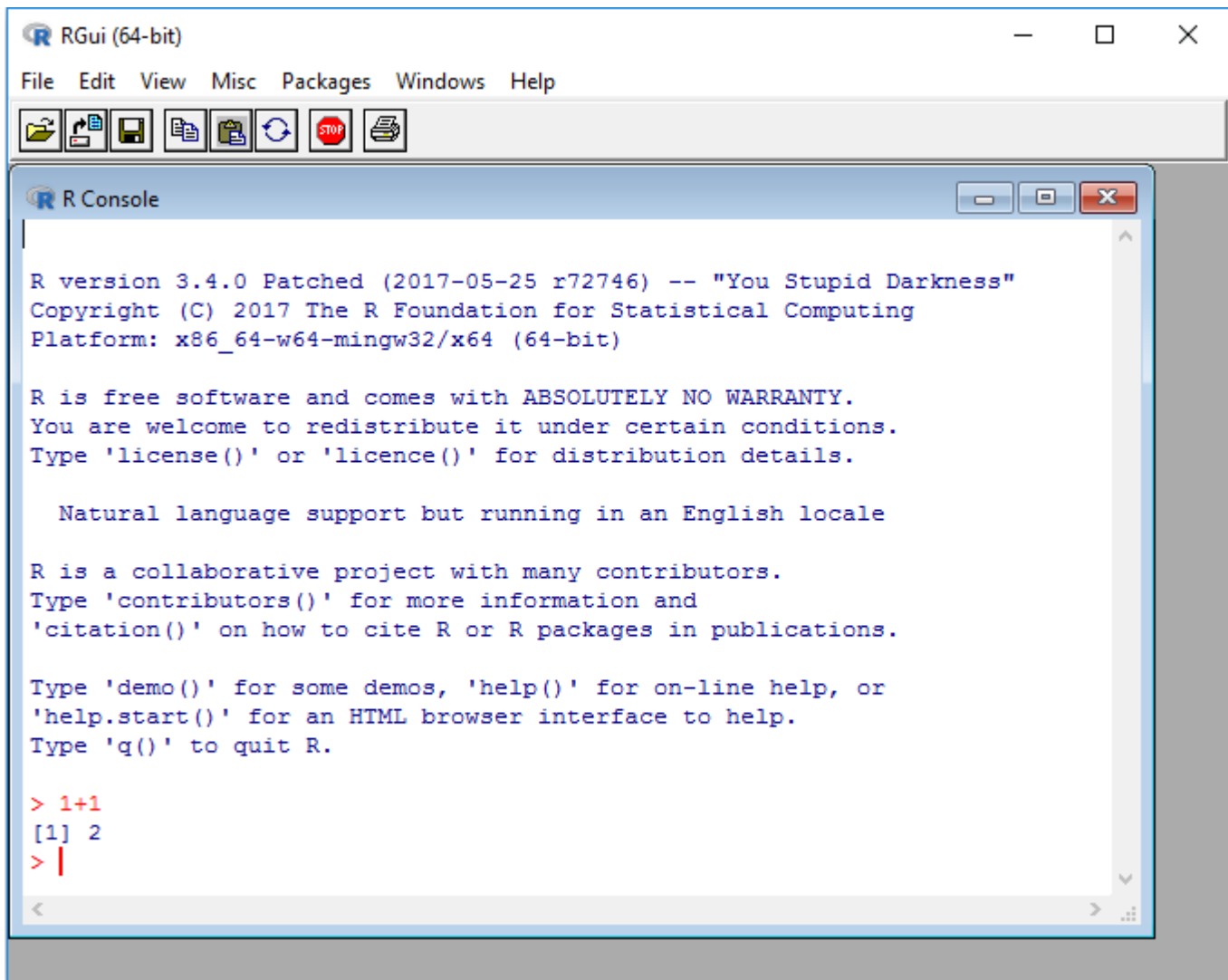
R ist freie Software und kommt OHNE JEGLICHE GARANTIE.
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.
Tippen Sie 'license()' or 'licence()' für Details dazu.

R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.
Tippen Sie 'contributors()' für mehr Information und 'citation()',
um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.

Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.

> 1+1
[1] 2
> █
```

Dies ist RGui unter Windows, die grundlegendste Arbeitsumgebung für R unter Windows:



Nach dem Zeichen `>` können Ausdrücke eingegeben werden. Wenn ein Ausdruck eingegeben wurde, wird das Ergebnis mit R angezeigt. In der Abbildung oben wird R als Taschenrechner verwendet: Typ

```
1+1
```

das Ergebnis sofort sehen, `2`. Die führende `[1]` gibt an, dass R einen Vektor liefert. In diesem Fall enthält der Vektor nur eine Zahl (2).

Die erste Handlung

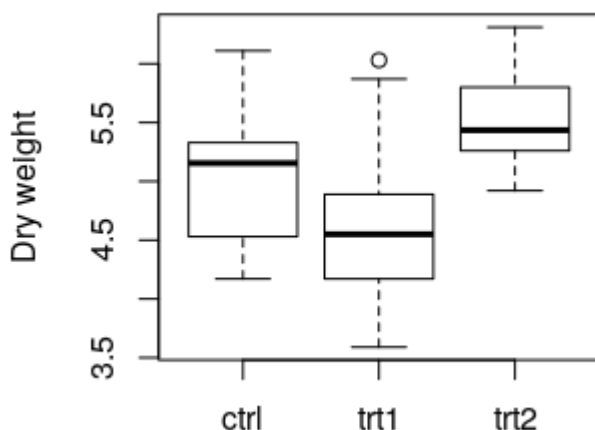
Mit R können Diagramme erstellt werden. Im folgenden Beispiel wird der Datensatz `PlantGrowth`, der zusammen mit R als Beispieldatensatz `PlantGrowth` wird

Geben Sie in die Eingabeaufforderung R folgende Zeilen ein, die nicht mit `##`. Zeilen, die mit `##` sollen das Ergebnis dokumentieren, das R zurückgibt.

```
data(PlantGrowth)
str(PlantGrowth)
## 'data.frame':    30 obs. of  2 variables:
## $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
```

```
## $ group : Factor w/ 3 levels "ctrl","trt1",...: 1 1 1 1 1 1 1 1 1 1 ...
anova(lm(weight ~ group, data = PlantGrowth))
## Analysis of Variance Table
##
## Response: weight
##          Df Sum Sq Mean Sq F value    Pr(>F)
## group      2  3.7663   1.8832   4.8461 0.01591 *
## Residuals 27 10.4921   0.3886
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
boxplot(weight ~ group, data = PlantGrowth, ylab = "Dry weight")
```

Das folgende Diagramm wird erstellt:



`data(PlantGrowth)` lädt den Beispieldatensatz `PlantGrowth` handelt es sich um Aufzeichnungen von Trockenmassen von Pflanzen, die zwei unterschiedlichen Behandlungsbedingungen oder keiner Behandlung unterzogen wurden (Kontrollgruppe). Der Datensatz wird unter dem Namen `PlantGrowth` zur Verfügung `PlantGrowth`. Ein solcher Name wird auch als [Variable bezeichnet](#).

Um Ihre eigenen Daten zu laden, können die folgenden zwei Dokumentationsseiten hilfreich sein:

- [Lesen und Schreiben von Tabellendaten in Klartextdateien \(CSV, TSV usw.\)](#)
- [E / A für Fremdtabellen \(Excel, SAS, SPSS, Stata\)](#)

`str(PlantGrowth)` zeigt Informationen über den Datensatz an, der geladen wurde. Die Ausgabe gibt an, dass `PlantGrowth` ein `PlantGrowth` ist. `data.frame` ist der Name von R für eine Tabelle. Der `data.frame` aus zwei Spalten und 30 Zeilen. In diesem Fall entspricht jede Zeile einer Anlage. Details der beiden Spalten werden in den Zeilen angezeigt, die mit `$`: Die erste Spalte heißt `weight` und enthält Zahlen (`num`, das Trockengewicht der jeweiligen Pflanze). Die zweite Kolonne, `group`, enthält die Behandlung, der die Pflanze unterzogen wurde. Dies sind kategoriale Daten, die in R `factor`. [Weitere Informationen zu Datenrahmen](#).

Um die Trockenmassen der drei verschiedenen Gruppen zu vergleichen, wird eine Einweg-ANOVA mit `anova(lm(...)).weight ~ group` bedeutet „die Werte der Spalte vergleichen `weight`, die durch die Werte der Spalte Gruppierung `group`“. In R. wird dies als [Formel bezeichnet](#). `data = ...` gibt den Namen der Tabelle an, in der die Daten gefunden werden können.

Das Ergebnis zeigt unter anderem, dass zwischen einigen der drei Gruppen ein signifikanter Unterschied besteht (Spalte `Pr(>F)`), `p = 0.01591`. Post-hoc-Tests müssen wie der Tukey-Test

durchgeführt werden, um zu bestimmen, welche Gruppenmittel signifikant voneinander abweichen.

`boxplot(...)` erstellt eine Boxdarstellung der Daten. woher die zu zeichnenden Werte kommen. `weight ~ group` bedeutet: "Zeichnen Sie die Werte der Spaltengewichtung *gegen* die Werte der Spaltengruppe `group`". `ylab = ...` gibt die Beschriftung der y-Achse an. Weitere Informationen: [Basisaufzeichnung](#)

Geben Sie `q()` oder `Strg-D` ein, um die R-Sitzung zu beenden.

R-Skripte

Um Ihre Recherche zu dokumentieren, ist es günstig, die Befehle, die Sie zur Berechnung verwenden, in einer Datei zu speichern. Für diesen Effekt können Sie **R-Skripte** erstellen. Ein R-Skript ist eine einfache Textdatei, die R-Befehle enthält.

Erstellen Sie eine Textdatei mit dem Namen `plants.R` und füllen Sie sie mit dem folgenden Text aus. Einige Befehle sind aus dem obigen Codeblock bekannt:

```
data(PlantGrowth)

anova(lm(weight ~ group, data = PlantGrowth))

png("plant_boxplot.png", width = 400, height = 300)
boxplot(weight ~ group, data = PlantGrowth, ylab = "Dry weight")
dev.off()
```

Führen Sie das Skript aus, indem Sie es in Ihr Terminal eingeben (Das Terminal Ihres Betriebssystems, **keine** interaktive R-Sitzung wie im vorherigen Abschnitt!)

```
R --no-save <plant.R >plant_result.txt
```

Die Datei `plant_result.txt` enthält die Ergebnisse Ihrer Berechnung, als hätten Sie sie in die interaktive Eingabeaufforderung R eingegeben. Dadurch werden Ihre Berechnungen dokumentiert.

Die neuen Befehle `png` und `dev.off` werden zum Speichern des Boxplots auf der Festplatte verwendet. Die beiden Befehle müssen den Zeichenbefehl einschließen, wie im obigen Beispiel gezeigt. `png("FILENAME", width = ..., height = ...)` öffnet eine neue PNG-Datei mit dem angegebenen Dateinamen, der Breite und Höhe in Pixel. `dev.off()` beendet das Plotten und speichert den Plot auf der Festplatte. Es wird keine Ausgabe gespeichert, bis `dev.off()` aufgerufen wird.

Erste Schritte mit R Language online lesen: <https://riptutorial.com/de/r/topic/360/erste-schritte-mit-r-language>

Kapitel 2: * wende die Familie der Funktionen (Funktionale) an

Bemerkungen

Eine Funktion in der `*apply` Familie ist eine Abstraktion einer `for` Schleife. Gegenüber den `for` Loops-Funktionen haben `*apply` Funktionen folgende Vorteile:

1. Erfordert weniger Code zum Schreiben.
2. Hat keinen Iterationszähler.
3. Verwendet temporäre Variablen nicht zum Speichern von Zwischenergebnissen.

`for` Schleifen sind jedoch allgemeiner und geben uns mehr Kontrolle, sodass komplexe Berechnungen durchgeführt werden können, die mit `*apply` Funktionen nicht immer trivial sind.

Die Beziehung zwischen `for` Schleifen und `*apply` Funktionen wird in der [Dokumentation für for Schleifen](#) erläutert.

Mitglieder der `*apply` Familie

Die `*apply` Funktionsfamilie enthält mehrere Varianten desselben Prinzips, die sich hauptsächlich in der Art der ausgegebenen Ausgabe unterscheiden.

Funktion	Eingang	Ausgabe
<code>apply</code>	<code>matrix</code> , <code>data.frame</code> oder <code>array</code>	Vektor oder Matrix (abhängig von der Länge jedes zurückgegebenen Elements)
<code>sapply</code>	Vektor oder <code>list</code>	Vektor oder Matrix (abhängig von der Länge jedes zurückgegebenen Elements)
<code>lapply</code>	Vektor oder <code>list</code>	<code>list</code>
<code>vapply</code>	Vektor oder `Liste	Vektor oder Matrix (abhängig von der Länge jedes zurückgegebenen Elements) der benutzerdefinierten Klasse
<code>mapply</code>	mehrere Vektoren, <code>lists</code> oder eine Kombination	<code>list</code>

Siehe "Beispiele", um zu sehen, wie jede dieser Funktionen verwendet wird.

Examples

Verwenden Sie anonyme Funktionen mit apply

`apply` wird verwendet, um eine Funktion (möglicherweise eine anonyme) über die Ränder eines Arrays oder einer Matrix auszuwerten.

Verwenden Sie das `iris` Dataset, um diese Idee zu veranschaulichen. Der `iris` Datensatz hat Messungen von 150 Blumen von 3 Arten. Mal sehen, wie dieser Datensatz aufgebaut ist:

```
> head(iris)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5         1.4         0.2   setosa
2          4.9         3.0         1.4         0.2   setosa
3          4.7         3.2         1.3         0.2   setosa
4          4.6         3.1         1.5         0.2   setosa
5          5.0         3.6         1.4         0.2   setosa
6          5.4         3.9         1.7         0.4   setosa
```

Stellen Sie sich vor, Sie möchten den Mittelwert *jeder* dieser Variablen kennen. Eine Möglichkeit, dieses Problem zu lösen, könnte die Verwendung einer `for` Schleife sein, aber R-Programmierer bevorzugen oft die Verwendung von "`apply`" (Gründe, siehe Anmerkungen):

```
> apply(iris[1:4], 2, mean)

Sepal.Length Sepal.Width Petal.Length Petal.Width
  5.843333    3.057333    3.758000    1.199333
```

- Im ersten Parameter setzen wir die Untermenge `iris`, dass sie nur die ersten 4 Spalten enthält, da der `mean` nur für numerische Daten gilt.
- Der zweite Parameterwert von `2` gibt an, dass wir nur an den Spalten arbeiten möchten (der zweite Index des Arrays $r \times c$). `1` würde die Zeile bedeuten.

Auf die gleiche Weise können wir aussagekräftigere Werte berechnen:

```
# standard deviation
apply(iris[1:4], 2, sd)
# variance
apply(iris[1:4], 2, var)
```

Einschränkung : R hat einige eingebaute Funktionen, die besser zur Berechnung von Spalten- und Reihensummen und `colMeans` `rowMeans` sind: `colMeans` und `rowMeans`.

Nun machen wir eine andere und sinnvollere Aufgabe: Berechnen Sie den Mittelwert *nur* für die Werte, die größer als `0.5`. Dafür werden wir unsere eigene erstellen `mean` Funktion.

```
> our.mean.function <- function(x) { mean(x[x > 0.5]) }
> apply(iris[1:4], 2, our.mean.function)

Sepal.Length Sepal.Width Petal.Length Petal.Width
  5.843333    3.057333    3.758000    1.665347
```

(Beachten Sie den Unterschied im Mittelwert von `Petal.Width`)

Was aber, wenn wir diese Funktion nicht im Rest unseres Codes verwenden möchten? Dann können wir eine anonyme Funktion verwenden und unseren Code folgendermaßen schreiben:

```
apply(iris[1:4], 2, function(x) { mean(x[x > 0.5]) })
```

Wie wir gesehen haben, können wir mit `apply` die gleiche Operation für Spalten oder Zeilen eines Datasets mit nur einer Zeile ausführen.

Einschränkung : Da `apply` je nach Länge der Ergebnisse der angegebenen Funktion sehr unterschiedliche Ausgabearten liefert, ist dies möglicherweise nicht die beste Wahl, wenn Sie nicht interaktiv arbeiten. Einige der anderen `*apply` Familienfunktionen sind etwas vorhersagbarer (siehe Anmerkungen).

Massendatei wird geladen

für eine große Anzahl von Dateien, die möglicherweise in einem ähnlichen Prozess und mit gut strukturierten Dateinamen bearbeitet werden müssen.

Zunächst muss ein Vektor der Dateinamen erstellt werden, auf die zugegriffen werden soll. Dazu gibt es mehrere Optionen:

- Vektor manuell mit `paste0()`

```
files <- paste0("file_", 1:100, ".rds")
```

- Mit `list.files()` mit einem regex Suchbegriff für den Dateityp, erfordert die Kenntnis von regulären Ausdrücken ([regex](#)), wenn andere Dateien gleichen Typs im Verzeichnis befinden.

```
files <- list.files("./", pattern = "\\..rds$", full.names = TRUE)
```

Dabei ist `x` ein Vektor eines Teils des verwendeten Dateinamensformats.

`lapply` gibt jede Antwort als Element einer Liste aus.

`readRDS` ist spezifisch für `.rds` Dateien und ändert sich je nach Anwendung des Prozesses.

```
my_file_list <- lapply(files, readRDS)
```

Dies ist nicht notwendigerweise schneller als eine for-Schleife von `testing`, sondern ermöglicht, dass alle Dateien ein Element einer Liste sind, ohne sie explizit zuzuweisen.

Schließlich müssen wir häufig mehrere Pakete gleichzeitig laden. Mit diesem Trick können Sie `library()` auf alle Bibliotheken anwenden, die wir importieren möchten:

```
lapply(c("jsonlite", "stringr", "igraph"), library, character.only=TRUE)
```

Mehrere `data.frames` kombinieren (`lapply`, `mapply`)

In dieser Übung werden vier lineare Bootstrap-Regressionsmodelle generiert und die Zusammenfassungen dieser Modelle in einem einzigen Datenrahmen kombiniert.

```
library(broom)

#* Create the bootstrap data sets
BootData <- lapply(1:4,
  function(i) mtcars[sample(1:nrow(mtcars),
    size = nrow(mtcars),
    replace = TRUE), ]))

#* Fit the models
Models <- lapply(BootData,
  function(BD) lm(mpg ~ qsec + wt + factor(am),
    data = BD))

#* Tidy the output into a data.frame
Tidied <- lapply(Models,
  tidy)

#* Give each element in the Tidied list a name
Tidied <- setNames(Tidied, paste0("Boot", seq_along(Tidied)))
```

An dieser Stelle können wir zwei Methoden zum Einfügen der Namen in den data.frame verwenden.

```
#* Insert the element name into the summary with `lapply`
#* Requires passing the names attribute to `lapply` and referencing `Tidied` within
#* the applied function.
Described_lapply <-
  lapply(names(Tidied),
    function(nm) cbind(nm, Tidied[[nm]]))

Combined_lapply <- do.call("rbind", Described_lapply)

#* Insert the element name into the summary with `mapply`
#* Allows us to pass the names and the elements as separate arguments.
Described_mapply <-
  mapply(
    function(nm, dframe) cbind(nm, dframe),
    names(Tidied),
    Tidied,
    SIMPLIFY = FALSE)

Combined_mapply <- do.call("rbind", Described_mapply)
```

Wenn Sie ein Fan von `magrittr` Stil-Pipes sind, können Sie die gesamte Aufgabe in einer einzelnen Kette `magrittr` (dies ist jedoch möglicherweise nicht ratsam, wenn Sie Zwischenobjekte wie die Modellobjekte selbst benötigen):

```
library(magrittr)
library(broom)
Combined <- lapply(1:4,
  function(i) mtcars[sample(1:nrow(mtcars),
```

```

                                size = nrow(mtcars),
                                replace = TRUE), ])) %>%
lapply(function(BD) lm( mpg ~ qsec + wt + factor(am), data = BD)) %>%
lapply(tidy) %>%
setNames(paste0("Boot", seq_along(.))) %>%
mapply(function(nm, dframe) cbind(nm, dframe),
        nm = names(.),
        dframe = .,
        SIMPLIFY = FALSE) %>%
do.call("rbind", .)

```

Eingebaute Funktionale verwenden

Integrierte Funktionen: lapply (), sapply () und mapply ()

R verfügt über integrierte Funktionen, von denen die bekannteste wahrscheinlich die anwendbare Funktionsfamilie ist. Hier finden Sie eine Beschreibung einiger der häufigsten Anwendungsfunktionen:

- `lapply()` = nimmt eine Liste als Argument und wendet die angegebene Funktion auf die Liste an.
- `sapply()` = dasselbe wie `lapply()`, versucht jedoch die Ausgabe in einen Vektor oder eine Matrix zu vereinfachen.
 - `vapply()` = eine Variante von `sapply()` in der der Typ des Ausgabeobjekts angegeben werden muss.
- `mapply()` = wie `lapply()`, kann jedoch mehrere Vektoren als Eingabe an die angegebene Funktion übergeben. Kann wie `sapply()` vereinfacht werden.
 - `Map()` ist ein Alias für `mapply()` mit `SIMPLIFY = FALSE`.

lapply ()

`lapply()` kann mit zwei verschiedenen Iterationen verwendet werden:

- `lapply(variable, FUN)`
- `lapply(seq_along(variable), FUN)`

```

# Two ways of finding the mean of x
set.seed(1)
df <- data.frame(x = rnorm(25), y = rnorm(25))
lapply(df, mean)
lapply(seq_along(df), function(x) mean(df[[x]]))

```

sapply ()

`sapply()` versucht, seine Ausgabe in einen Vektor oder eine Matrix aufzulösen.

```
# Two examples to show the different outputs of sapply()
sapply(letters, print) ## produces a vector
x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
sapply(x, quantile) ## produces a matrix
```

mapply ()

`mapply()` funktioniert ähnlich wie `lapply()` außer dass es mehrere Vektoren als Eingabe `lapply()` kann (daher das `m` für multivariate).

```
mapply(sum, 1:5, 10:6, 3) # 3 will be "recycled" by mapply
```

Verwendung benutzerdefinierter Funktionen

Benutzerdefinierte Funktionen

Benutzer können ihre eigenen Funktionen in unterschiedlichem Umfang erstellen. Die folgenden Beispiele stammen von [Functionals](#) by Hadley Wickham:

```
randomise <- function(f) f(runif(1e3))

lapply2 <- function(x, f, ...) {
  out <- vector("list", length(x))
  for (i in seq_along(x)) {
    out[[i]] <- f(x[[i]], ...)
  }
  out
}
```

Im ersten Fall akzeptiert `randomise` ein einzelnes Argument `f` und ruft es für eine Stichprobe von Uniform-Zufallsvariablen auf. Um die Äquivalenz zu demonstrieren, rufen wir `set.seed` unten auf:

```
set.seed(123)
randomise(mean)
#[1] 0.4972778

set.seed(123)
mean(runif(1e3))
#[1] 0.4972778

set.seed(123)
randomise(max)
#[1] 0.9994045

set.seed(123)
max(runif(1e3))
#[1] 0.9994045
```

Das zweite Beispiel ist eine Neuimplementierung von `base::lapply`, die mithilfe von Funktionen eine Operation (`f`) auf jedes Element in einer Liste (`x`) anwendet. Mit dem Parameter `...` kann

der Benutzer zusätzliche Argumente an `f`, z. B. die Option `na.rm` in der Funktion `mean`:

```
lapply(list(c(1, 3, 5), c(2, NA, 6)), mean)
# [[1]]
# [1] 3
#
# [[2]]
# [1] NA

lapply2(list(c(1, 3, 5), c(2, NA, 6)), mean)
# [[1]]
# [1] 3
#
# [[2]]
# [1] NA

lapply(list(c(1, 3, 5), c(2, NA, 6)), mean, na.rm = TRUE)
# [[1]]
# [1] 3
#
# [[2]]
# [1] 4

lapply2(list(c(1, 3, 5), c(2, NA, 6)), mean, na.rm = TRUE)
# [[1]]
# [1] 3
#
# [[2]]
# [1] 4
```

* wende die Familie der Funktionen (Funktionale) an online lesen:

<https://riptutorial.com/de/r/topic/3567/--wende-die-familie-der-funktionen--funktionale--an>

Kapitel 3: .Rprofile

Bemerkungen

In der [Efficient R-Programmierung](#) gibt es ein schönes Kapitel

Examples

.Rprofile - der erste ausgeführte Code

`.Rprofile` ist eine Datei mit R-Code, die ausgeführt wird, wenn Sie R aus dem Verzeichnis starten, das die `.Rprofile` Datei enthält. Die ähnlich benannte `Rprofile.site`, die sich im `Rprofile.site` R befindet, wird standardmäßig jedes Mal ausgeführt, wenn Sie R aus einem beliebigen Verzeichnis laden. `Rprofile.site` und in größerem `.Rprofile` kann verwendet werden, um eine R-Sitzung mit persönlichen `.Rprofile` und verschiedenen von Ihnen definierten `.Rprofile` zu initialisieren.

Wichtiger Hinweis: Wenn Sie RStudio verwenden, können `.Rprofile` in jedem RStudio-Projektverzeichnis ein separates `.Rprofile`.

Hier einige Beispiele für Code, den Sie möglicherweise in eine `.Rprofile`-Datei einfügen.

Einrichten Ihres R Home-Verzeichnisses

```
# set R_home
Sys.setenv(R_USER="c:/R_home") # just an example directory
# but don't confuse this with the $R_HOME environment variable.
```

Optionen für die Seitengröße einstellen

```
options(papersize="a4")
options(editor="notepad")
options(pager="internal")
```

Legen Sie den Standardhilfetyp fest

```
options(help_type="html")
```

Legen Sie eine Site-Bibliothek fest


```
.Library.site <- file.path(chartr("\\", "/", R.home()), "site-library")
```

Stellen Sie einen CRAN-Spiegel ein

```
local({r <- getOption("repos")
  r["CRAN"] <- "http://my.local.cran"
  options(repos=r)})
```

Festlegen des Speicherorts Ihrer Bibliothek

Auf diese Weise müssen Sie nicht alle Pakete mit jedem R-Versionsupdate erneut installieren.

```
# library location
.libPaths("c:/R_home/Rpackages/win")
```

Benutzerdefinierte Verknüpfungen oder Funktionen

Manchmal ist es nützlich, eine Abkürzung für einen langen R-Ausdruck zu haben. Ein allgemeines Beispiel für das Festlegen einer aktiven Bindung, um auf das letzte Ergebnis der obersten Ebene des Ausdrucks zuzugreifen, ohne den `.Last.value`.

```
makeActiveBinding(".", function(){.Last.value}, .GlobalEnv)
```

Da `.Rprofile` nur eine R-Datei ist, kann sie beliebigen R-Code enthalten.

Laden Sie die nützlichsten Pakete vor

Dies ist eine schlechte Praxis und sollte generell vermieden werden, da der Code zum Laden von Paketen von den Skripts getrennt wird, in denen diese Pakete tatsächlich verwendet werden.

Siehe auch

In der `help(Startup)` zu den verschiedenen Startskripts und weiteren Aspekten. Insbesondere können auch zwei systemweite `Profile` geladen werden. Die erste, `Rprofile`, kann globale Einstellungen enthalten, die andere Datei `Profile.site` kann lokale Auswahlen enthalten, die der Systemadministrator für alle Benutzer treffen kann. Beide Dateien befinden sich im Verzeichnis `${RHOME}/etc` der R-Installation. Dieses Verzeichnis enthält auch die globalen Dateien `Renviron` und `Renviron.site` die beide mit einer lokalen Datei `~/.Renviron` im `~/.Renviron` des Benutzers ergänzt werden können.

.Rprofile Beispiel

Anlaufen

```
# Load library setwidth on start - to set the width automatically.
.First <- function() {
  library(setwidth)
  # If 256 color terminal - use library colorout.
  if (Sys.getenv("TERM") %in% c("xterm-256color", "screen-256color")) {
    library("colorout")
  }
}
```

Optionen

```
# Select default CRAN mirror for package installation.
options(repos=c(CRAN="https://cran.gis-lab.info/"))

# Print maximum 1000 elements.
options(max.print=1000)

# No scientific notation.
options(scipen=10)

# No graphics in menus.
options(menu.graphics=FALSE)

# Auto-completion for package names.
utils::rc.settings(ipck=TRUE)
```

Benutzerdefinierte Funktionen

```
# Invisible environment to mask defined functions
.env = new.env()

# Quit R without asking to save.
.env$q <- function (save="no", ...) {
  quit(save=save, ...)
}

# Attach the environment to enable functions.
attach(.env, warn.conflicts=FALSE)
```

.Rprofile online lesen: <https://riptutorial.com/de/r/topic/4166/-rprofile>

Kapitel 4: Aggregation von Datenrahmen

Einführung

Aggregation ist eine der häufigsten Anwendungen für R. In R gibt es mehrere Möglichkeiten, dies zu erläutern.

Examples

Aggregation mit der Basis R

Dazu verwenden wir das Funktionsaggregat, das wie folgt verwendet werden kann:

```
aggregate(formula, function, data)
```

Der folgende Code zeigt verschiedene Möglichkeiten zur Verwendung der Aggregatfunktion.

CODE:

```
df = data.frame(group=c("Group 1","Group 1","Group 2","Group 2","Group 2"), subgroup =  
c("A","A","A","A","B"),value = c(2,2.5,1,2,1.5))  
  
# sum, grouping by one column  
aggregate(value~group, FUN=sum, data=df)  
  
# mean, grouping by one column  
aggregate(value~group, FUN=mean, data=df)  
  
# sum, grouping by multiple columns  
aggregate(value~group+subgroup,FUN=sum,data=df)  
  
# custom function, grouping by one column  
# in this example we want the sum of all values larger than 2 per group.  
aggregate(value~group, FUN=function(x) sum(x[x>2]), data=df)
```

AUSGABE:

```
> df = data.frame(group=c("Group 1","Group 1","Group 2","Group 2","Group 2"), subgroup =  
c("A","A","A","A","B"),value = c(2,2.5,1,2,1.5))  
> print(df)  
  group subgroup value  
1 Group 1      A    2.0  
2 Group 1      A    2.5  
3 Group 2      A    1.0  
4 Group 2      A    2.0  
5 Group 2      B    1.5  
>  
> # sum, grouping by one column  
> aggregate(value~group, FUN=sum, data=df)  
  group value  
1 Group 1  4.5
```

```

2 Group 2    4.5
>
> # mean, grouping by one column
> aggregate(value~group, FUN=mean, data=df)
  group value
1 Group 1  2.25
2 Group 2  1.50
>
> # sum, grouping by multiple columns
> aggregate(value~group+subgroup, FUN=sum, data=df)
  group subgroup value
1 Group 1      A    4.5
2 Group 2      A    3.0
3 Group 2      B    1.5
>
> # custom function, grouping by one column
> # in this example we want the sum of all values larger than 2 per group.
> aggregate(value~group, FUN=function(x) sum(x[x>2]), data=df)
  group value
1 Group 1  2.5
2 Group 2  0.0

```

Aggregation mit Dplyr

Das Aggregieren mit Dplyr ist einfach! Sie können dazu die Funktionen `group_by()` und `summary()` verwenden. Einige Beispiele sind unten angegeben.

CODE:

```

# Aggregating with dplyr
library(dplyr)

df = data.frame(group=c("Group 1","Group 1","Group 2","Group 2","Group 2"), subgroup =
c("A","A","A","A","B"), value = c(2,2.5,1,2,1.5))
print(df)

# sum, grouping by one column
df %>% group_by(group) %>% summarize(value = sum(value)) %>% as.data.frame()

# mean, grouping by one column
df %>% group_by(group) %>% summarize(value = mean(value)) %>% as.data.frame()

# sum, grouping by multiple columns
df %>% group_by(group, subgroup) %>% summarize(value = sum(value)) %>% as.data.frame()

# custom function, grouping by one column
# in this example we want the sum of all values larger than 2 per group.
df %>% group_by(group) %>% summarize(value = sum(value[value>2])) %>% as.data.frame()

```

AUSGABE:

```

> library(dplyr)
>
> df = data.frame(group=c("Group 1","Group 1","Group 2","Group 2","Group 2"), subgroup =
c("A","A","A","A","B"), value = c(2,2.5,1,2,1.5))
> print(df)
  group subgroup value

```

```

1 Group 1      A    2.0
2 Group 1      A    2.5
3 Group 2      A    1.0
4 Group 2      A    2.0
5 Group 2      B    1.5
>
> # sum, grouping by one column
> df %>% group_by(group) %>% summarize(value = sum(value)) %>% as.data.frame()
  group value
1 Group 1    4.5
2 Group 2    4.5
>
> # mean, grouping by one column
> df %>% group_by(group) %>% summarize(value = mean(value)) %>% as.data.frame()
  group value
1 Group 1   2.25
2 Group 2   1.50
>
> # sum, grouping by multiple columns
> df %>% group_by(group, subgroup) %>% summarize(value = sum(value)) %>% as.data.frame()
  group subgroup value
1 Group 1      A    4.5
2 Group 2      A    3.0
3 Group 2      B    1.5
>
> # custom function, grouping by one column
> # in this example we want the sum of all values larger than 2 per group.
> df %>% group_by(group) %>% summarize(value = sum(value[value>2])) %>% as.data.frame()
  group value
1 Group 1    2.5
2 Group 2    0.0

```

Aggregation mit data.table

Die Gruppierung mit dem Paket `data.table` erfolgt mit der Syntax `dt[i, j, by]` die laut ausgelesen werden kann als: "*Nimm dt, Teilmengenzeilen mit i, dann berechne j, gruppiert nach*" in der `dt`-Anweisung. Mehrere Berechnungen oder Gruppen sollten in eine Liste aufgenommen werden. Da ein Alias für `list()` `()` ist, können beide austauschbar verwendet werden. In den folgenden Beispielen verwenden wir `()`.

CODE:

```

# Aggregating with data.table
library(data.table)

dt = data.table(group=c("Group 1","Group 1","Group 2","Group 2","Group 2"), subgroup =
c("A","A","A","A","B"), value = c(2,2.5,1,2,1.5))
print(dt)

# sum, grouping by one column
dt[,.(value=sum(value)),group]

# mean, grouping by one column
dt[,.(value=mean(value)),group]

# sum, grouping by multiple columns
dt[,.(value=sum(value)),.(group,subgroup)]

```

```
# custom function, grouping by one column
# in this example we want the sum of all values larger than 2 per group.
dt[,.(value=sum(value[value>2])),group]
```

AUSGABE:

```
> # Aggregating with data.table
> library(data.table)
>
> dt = data.table(group=c("Group 1","Group 1","Group 2","Group 2","Group 2"), subgroup =
c("A","A","A","A","B"),value = c(2,2.5,1,2,1.5))
> print(dt)
   group subgroup value
1: Group 1      A   2.0
2: Group 1      A   2.5
3: Group 2      A   1.0
4: Group 2      A   2.0
5: Group 2      B   1.5
>
> # sum, grouping by one column
> dt[,.(value=sum(value)),group]
   group value
1: Group 1  4.5
2: Group 2  4.5
>
> # mean, grouping by one column
> dt[,.(value=mean(value)),group]
   group value
1: Group 1  2.25
2: Group 2  1.50
>
> # sum, grouping by multiple columns
> dt[,.(value=sum(value)),.(group,subgroup)]
   group subgroup value
1: Group 1      A   4.5
2: Group 2      A   3.0
3: Group 2      B   1.5
>
> # custom function, grouping by one column
> # in this example we want the sum of all values larger than 2 per group.
> dt[,.(value=sum(value[value>2])),group]
   group value
1: Group 1  2.5
2: Group 2  0.0
```

Aggregation von Datenrahmen online lesen: <https://riptutorial.com/de/r/topic/10792/aggregation-von-datenrahmen>

Kapitel 5: Aktualisieren von R und der Paketbibliothek

Examples

Unter Windows

Standardinstallation von R unter Windows gespeicherten Dateien (und damit der Bibliothek) in einem dedizierten Ordner pro R-Version für Programmdateien.

Das bedeutet, dass Sie standardmäßig mit mehreren Versionen von R parallel und somit mit separaten Bibliotheken arbeiten.

Wenn Sie dies nicht wünschen und Sie es vorziehen, immer mit einer einzelnen R-Instanz zu arbeiten, die Sie nicht schrittweise aktualisieren möchten, wird empfohlen, den R-Installationsordner zu ändern. Geben Sie im Assistenten einfach diesen Ordner an (ich verwende persönlich `c:\stats\R`). Bei jedem Upgrade besteht die Möglichkeit, dieses R zu überschreiben. Ob Sie auch (alle) Pakete aktualisieren möchten, ist eine heikle Wahl, da dadurch möglicherweise ein Teil Ihres Codes `tm` (dies erschien mir im Paket `tm`). Du darfst:

- Erstellen Sie zuerst eine Kopie Ihrer gesamten Bibliothek, bevor Sie Pakete aktualisieren
- Pflegen Sie Ihr eigenes Quellpaket-Repository, beispielsweise mit dem Paket `miniCRAN`

Wenn Sie alle Pakete aktualisieren möchten - ohne Überprüfung, können Sie use `packageStatus` wie `packageStatus` :

```
pkgs <- packageStatus() # choose mirror
upgrade(pkgs)
```

Schließlich gibt es ein sehr praktisches Paket, um alle Operationen auszuführen, nämlich `installr`, selbst wenn es eine dedizierte GUI gibt. Wenn Sie `gui` verwenden möchten, müssen Sie `Rgui` verwenden und das Paket nicht in RStudio laden. Die Verwendung des Pakets mit Code ist so einfach wie:

```
install.packages("installr") # install
setInternet2(TRUE) # only for R versions older than 3.3.0
installr::updateR() # updating R.
```

Ich verweise auf die großartige Dokumentation <https://www.r-statistics.com/tag/installr/> und insbesondere auf den schrittweisen Ablauf mit Screenshots unter Windows: <https://www.r-statistics.com/2015/06/a-step-by-step-screenshots-tutorial-for-upgrade-r-on-windows/>

Beachten Sie, dass ich immer noch für ein einzelnes Verzeichnis plädiere, dh. Verweis auf die R-Version im Namen des Installationsordners entfernen.

Aktualisieren von R und der Paketbibliothek online lesen:

<https://riptutorial.com/de/r/topic/4088/aktualisieren-von-r-und-der-paketbibliothek>

Kapitel 6: Analysieren Sie Tweets mit R

Einführung

(Optional) Jedes Thema hat einen Fokus. Erzählen Sie den Lesern, was sie hier finden und informieren Sie zukünftige Mitwirkende darüber, was dazu gehört.

Examples

Tweets herunterladen

Zuerst müssen Sie Tweets herunterladen. Sie müssen Ihren Tweeter-Account einrichten. Viele Informationen dazu finden Sie im Internet. Die folgenden zwei Links waren hilfreich für mein Setup (zuletzt überprüft im Mai 2017)

Ich fand insbesondere die folgenden zwei Links hilfreich (zuletzt überprüft im Mai 2017):

[Link 1](#)

[Link 2](#)

R Bibliotheken

Sie benötigen die folgenden R-Pakete

```
library("devtools")
library("twitter")
library("ROAuth")
```

Angenommen, Sie haben Ihre Schlüssel Sie müssen den folgenden Code ausführen

```
api_key <- XXXXXXXXXXXXXXXXXXXXXXXX
api_secret <- XXXXXXXXXXXXXXXXXXXXXXXX
access_token <- XXXXXXXXXXXXXXXXXXXXXXXX
access_token_secret <- XXXXXXXXXXXXXXXXXXXXXXXX

setup_twitter_oauth(api_key, api_secret)
```

Ändern Sie XXXXXXXXXXXXXXXXXXXXXXXX in Ihre Schlüssel.

Nehmen wir an, wir möchten Tweets auf Kaffee herunterladen. Der folgende Code wird es tun

```
search.string <- "#coffee"
no.of.tweets <- 1000

c_tweets <- searchTwitter(search.string, n=no.of.tweets, lang="en")
```

Sie erhalten 1000 Tweets auf "Kaffee".

Holen Sie sich Text von Tweets

Nun müssen wir auf den Text der Tweets zugreifen. So machen wir es auf diese Weise (wir müssen auch die Tweets von Sonderzeichen entfernen, die wir vorerst nicht benötigen, wie Emoticons mit der Funktion `sapply`.)

```
coffee_tweets = sapply(c_tweets, function(t) t$text())  
  
coffee_tweets <- sapply(coffee_tweets, function(row) iconv(row, "latin1", "ASCII", sub=""))
```

und Sie können Ihre Tweets mit der `head` Funktion überprüfen.

```
head(coffee_tweets)
```

Analysieren Sie Tweets mit R online lesen: <https://riptutorial.com/de/r/topic/10086/analysieren-sie-tweets-mit-r>

Kapitel 7: Ändern von Strings durch Ersetzen

Einführung

`sub` und `gsub` werden verwendet, um Strings anhand von Mustern zu bearbeiten. Weitere [Informationen](#) zu verwandten Funktionen und zu [regulären Ausdrücken finden Sie unter Musterübereinstimmung und -austausch](#).

Examples

Zeichnen Sie Zeichenfolgen mithilfe von Erfassungsgruppen neu an

Wenn Sie die Reihenfolge von Zeichenfolgen ändern möchten, können Sie im `pattern` Klammern verwenden `pattern` um Teile der Zeichenfolge zusammenzufassen. Diese Gruppen können im `replacement` mit fortlaufenden Nummern adressiert werden.

Das folgende Beispiel zeigt, wie Sie einen Vektor mit Namen der Form "Nachname, Vorname" in einen Vektor der Form "Vorname Nachname" umordnen können.

```
library(randomNames)
set.seed(1)

strings <- randomNames(5)
strings
# [1] "Sigg, Zachary"      "Holt, Jake"         "Ortega, Sandra"     "De La Torre,
# [5] "Perkins, Donovan"  Nichole"

sub("^(.+),\\s(.+)$", "\\2 \\1", strings)
# [1] "Zachary Sigg"       "Jake Holt"          "Sandra Ortega"      "Nichole De La Torre"
# [5] "Donovon Perkins"
```

Wenn Sie nur den Nachnamen benötigen, können Sie nur die ersten Klammerpaare ansprechen.

```
sub("^(.+),\\s(.+)", "\\1", strings)
# [1] "Sigg"      "Holt"      "Ortega"    "De La Torre" "Perkins"
```

Beseitigen Sie doppelte aufeinanderfolgende Elemente

Nehmen wir an, wir wollen duplizierte Untersequenzelemente aus einer Zeichenfolge entfernen (es kann mehr als eine sein). Zum Beispiel:

```
2,14,14,14,19
```

und konvertiere es in:

```
2,14,19
```

Mit `gsub` können wir es erreichen:

```
gsub("(\\d+)(,\\1)+", "\\1", "2,14,14,14,19")  
[1] "2,14,19"
```

Es funktioniert auch für mehr als eine andere Wiederholung, zum Beispiel:

```
> gsub("(\\d+)(,\\1)+", "\\1", "2,14,14,14,19,19,20,21")  
[1] "2,14,19,20,21"
```

Lassen Sie uns den regulären Ausdruck erklären:

1. `(\\d+)` : Eine durch `()` begrenzte Gruppe 1, die eine beliebige Ziffer (mindestens eine) findet. Denken Sie daran, dass Sie hier den doppelten Backslash (`\\`) verwenden müssen, da ein Backslash für eine Zeichenvariable ein spezielles Escape-Zeichen für literale Zeichenfolgenbegrenzer (`"` oder `'`) darstellt. `\\d` entspricht: `[0-9]` .
2. `,` : Ein Satzzeichen: `,` (wir können Leerzeichen oder andere Trennzeichen enthalten)
3. `\\1` : Eine mit der Gruppe 1 identische Zeichenfolge, dh die wiederholte Nummer. Wenn dies nicht der Fall ist, stimmt das Muster nicht überein.

Versuchen wir eine ähnliche Situation: Beseitigen Sie aufeinander folgende wiederholte Wörter:

```
one,two,two,three,four,four,five,six
```

Dann ersetzen Sie einfach `\\d` durch `\\w`, wobei `\\w` beliebigen Wortzeichen entspricht, einschließlich Buchstaben, Ziffern oder Unterstrichen. Es ist äquivalent zu `[a-zA-Z0-9_]` :

```
> gsub("(\\w+)(,\\1)+", "\\1", "one,two,two,three,four,four,five,six")  
[1] "one,two,three,four,five,six"  
>
```

Das obige Muster enthält dann als einen Fall doppelte Ziffern.

Ändern von Strings durch Ersetzen online lesen: <https://riptutorial.com/de/r/topic/9219/andern-von-strings-durch-ersetzen>

Kapitel 8: ANOVA

Examples

Grundlegende Verwendung von `aov()`

Die Varianzanalyse (aov) wird verwendet, um zu bestimmen, ob die Mittelwerte von zwei oder mehr Gruppen signifikant voneinander abweichen. Es wird angenommen, dass die Antworten unabhängig voneinander sind. Normal verteilt (innerhalb jeder Gruppe) und die Abweichungen innerhalb der Gruppe werden als gleich angenommen.

Um die Analyse abzuschließen, müssen die Daten im langen Format vorliegen (siehe Thema "[Daten neu gestalten](#)"). `aov()` ist ein Wrapper um die `lm()` Funktion und verwendet die Wilkinson-Rogers- `aov()` $y \sim f$ wobei y die Antwortvariable (unabhängig) und f eine (kategoriale) Faktorvariable ist, die die Gruppenmitgliedschaft darstellt. *Wenn f eine numerische und keine `aov()` Variable ist, `aov()` die Ergebnisse einer linearen Regression im ANOVA-Format, was unerfahrene Benutzer überraschen kann.*

Die Funktion `aov()` verwendet die Summe der Quadrate vom Typ I (sequentiell). Bei dieser Art von Quadratsumme werden alle (Haupt- und Interaktionseffekte) der Reihe nach getestet. Das Ergebnis ist, dass dem ersten getesteten Effekt auch eine gemeinsame Varianz zwischen ihm und anderen Effekten im Modell zugewiesen wird. Damit die Ergebnisse eines solchen Modells zuverlässig sind, sollten die Daten ausgewogen sein (alle Gruppen haben die gleiche Größe).

Wenn die Annahmen für die Summe der Quadrate des Typs I nicht zutreffen, kann die Summe der Quadrate des Typs II oder des Typs III zutreffen. Die Summe der Quadrate des Typs II testet jeden Haupteffekt nach jedem anderen Haupteffekt und kontrolliert somit die überlappende Varianz. Die Summe der Quadrate des Typs II setzt jedoch keine Wechselwirkung zwischen den Haupteffekten voraus.

Schließlich testet Typ III Quadratsumme jeden Haupteffekt nach jedem anderen Haupteffekt *und nach* jeder Interaktion. Dies macht die Summe der Quadrate des Typs III zu einer Notwendigkeit, wenn eine Interaktion vorliegt.

Quadratsummen vom Typ II und Typ III werden in der `Anova()` Funktion implementiert.

Verwenden des `mtcars` Datensatzes als Beispiel.

```
mtCarsAnovaModel <- aov(wt ~ factor(cyl), data=mtcars)
```

So zeigen Sie die Zusammenfassung des ANOVA-Modells an:

```
summary(mtcarsAnovaModel)
```

Man kann auch die Koeffizienten des zugrundeliegenden `lm()` Modells extrahieren:

```
coefficients(mtCarsAnovaModel)
```

Grundlegende Verwendung von Anova ()

Bei einem unsymmetrischen Design und / oder nicht orthogonalen Kontrasten sind die Quadratsummen vom Typ II oder Typ III erforderlich. Die `Anova()` Funktion aus dem `car` - Paket führt diese durch. Bei der Summe der Quadrate vom Typ II wird keine Wechselwirkung zwischen den Haupteffekten angenommen. Wenn Wechselwirkungen angenommen werden, ist die Quadratsumme vom Typ III angemessen.

Die `Anova()` Funktion umgibt die `lm()` Funktion.

Verwenden Sie die `mtcars` -Datensätze als Beispiel, um den Unterschied zwischen Typ II und Typ III zu demonstrieren, wenn eine Interaktion getestet wird.

```
> Anova(lm(wt ~ factor(cyl)*factor(am), data=mtcars), type = 2)
Anova Table (Type II tests)

Response: wt
              Sum Sq Df F value    Pr(>F)
factor(cyl)      7.2278  2 11.5266 0.0002606 ***
factor(am)        3.2845  1 10.4758 0.0032895 **
factor(cyl):factor(am) 0.0668  2  0.1065 0.8993714
Residuals        8.1517 26
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> Anova(lm(wt ~ factor(cyl)*factor(am), data=mtcars), type = 3)
Anova Table (Type III tests)

Response: wt
              Sum Sq Df F value    Pr(>F)
(Intercept)    25.8427  1 82.4254 1.524e-09 ***
factor(cyl)      4.0124  2  6.3988  0.005498 **
factor(am)        1.7389  1  5.5463  0.026346 *
factor(cyl):factor(am) 0.0668  2  0.1065  0.899371
Residuals        8.1517 26
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA online lesen: <https://riptutorial.com/de/r/topic/3610/anova>

Kapitel 9: Arima-Modelle

Bemerkungen

Die `Arima` Funktion im Prognosepaket ist expliziter im Umgang mit Konstanten, was es für einige Benutzer relativ zur `arima` Funktion in der Basis R erleichtern kann.

ARIMA ist ein allgemeiner Rahmen für das Modellieren und Vorhersagen von Zeitreihendaten, wobei (hauptsächlich) die Serie selbst verwendet wird. Der Zweck des Rahmens besteht darin, die kurz- und langfristige Dynamik in einer Reihe zu differenzieren, um die Genauigkeit und Sicherheit der Vorhersagen zu verbessern. Noch poetischer bieten ARIMA-Modelle eine Methode, um zu beschreiben, wie Schocks auf ein System durch die Zeit übertragen werden.

Aus ökonometrischer Sicht sind ARIMA-Elemente notwendig, um die Korrelation der Serien zu korrigieren und die Stationarität sicherzustellen.

Examples

Modellierung eines AR1-Prozesses mit Arima

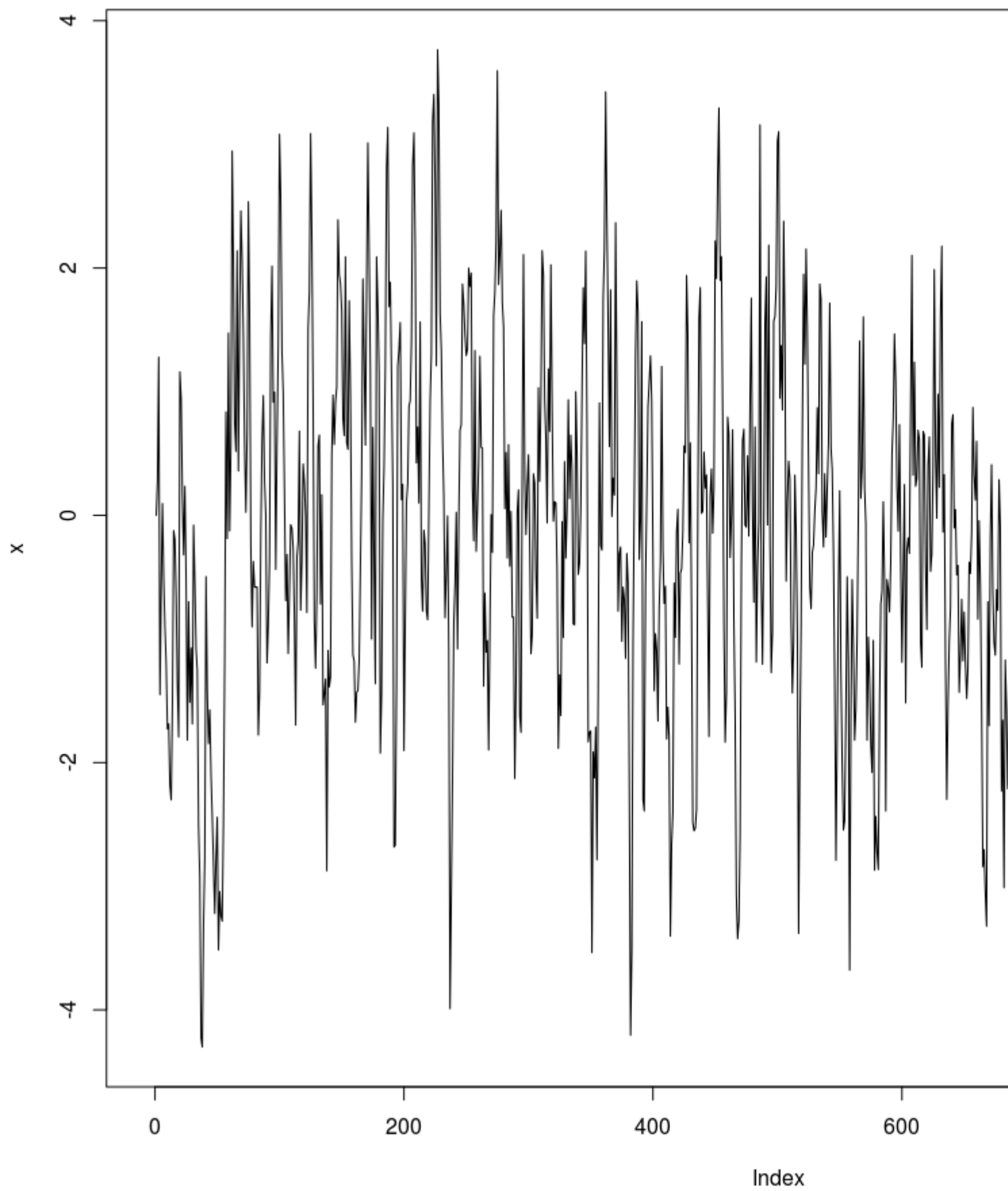
Wir werden den Prozess modellieren

$$x_t = .7x_{t-1} + \epsilon \quad \epsilon \sim N(0, 1)$$

```
#Load the forecast package
library(forecast)

#Generate an AR1 process of length n (from Cowpertwait & Meltcalfe)
# Set up variables
set.seed(1234)
n <- 1000
x <- matrix(0,1000,1)
w <- rnorm(n)

# loop to create x
for (t in 2:n) x[t] <- 0.7 * x[t-1] + w[t]
plot(x,type='l')
```



Wir werden ein Arima-Modell mit autoregressiver Ordnung 1, 0 Differenzierungsgrad und einer MA-Ordnung von 0 anpassen.

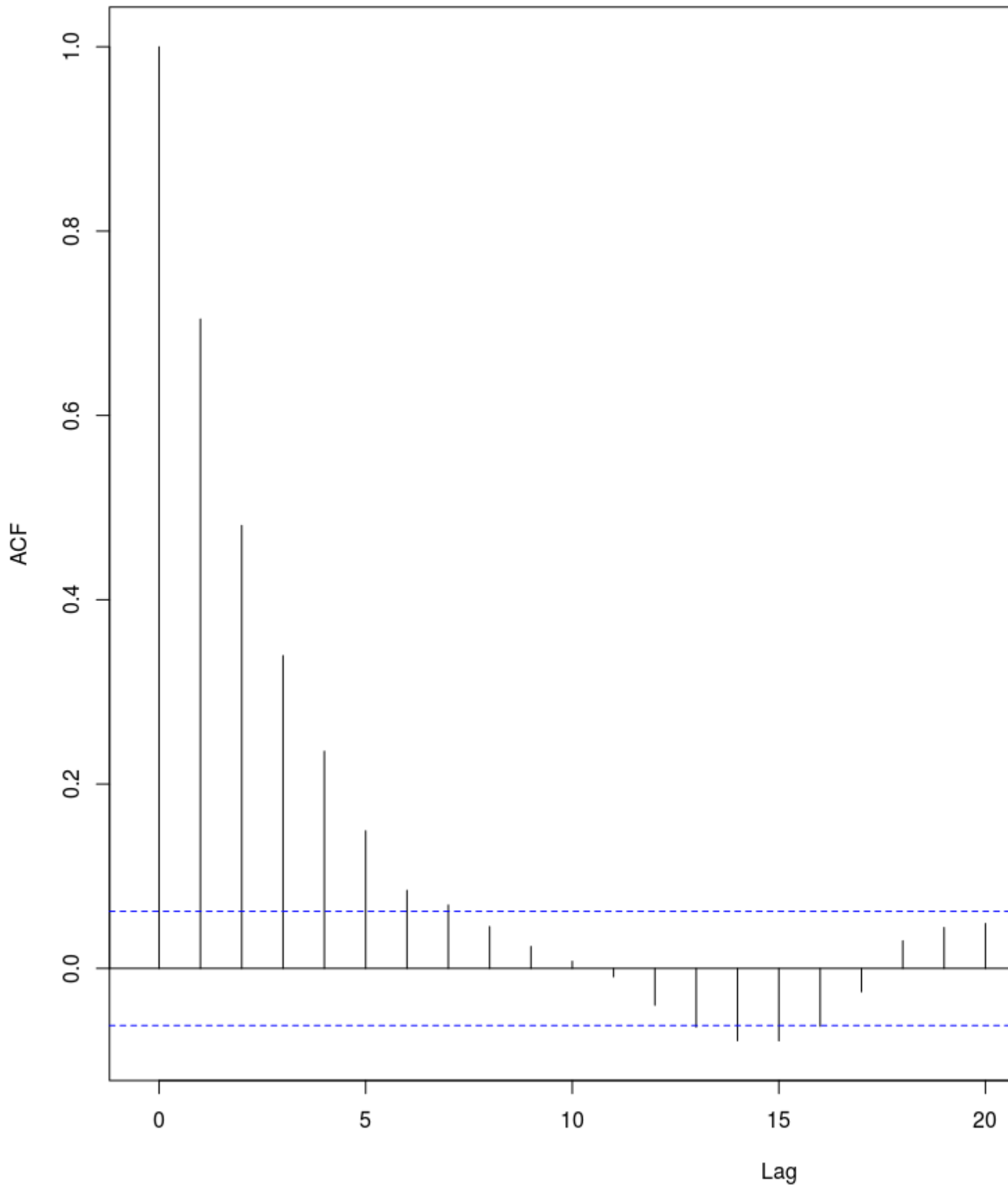
```
#Fit an AR1 model using Arima
fit <- Arima(x, order = c(1, 0, 0))
summary(fit)
# Series: x
# ARIMA(1,0,0) with non-zero mean
#
# Coefficients:
#          ar1  intercept
#          0.7040   -0.0842
# s.e.    0.0224    0.1062
#
# sigma^2 estimated as 0.9923:  log likelihood=-1415.39
# AIC=2836.79   AICc=2836.81   BIC=2851.51
#
# Training set error measures:
#              ME      RMSE      MAE MPE MAPE      MASE      ACF1
# Training set -8.369365e-05 0.9961194 0.7835914 Inf  Inf 0.91488 0.02263595
# Verify that the model captured the true AR parameter
```

Beachten Sie, dass unser Koeffizient dem wahren Wert der generierten Daten nahe kommt

```
fit$coef[1]
#          ar1
# 0.7040085

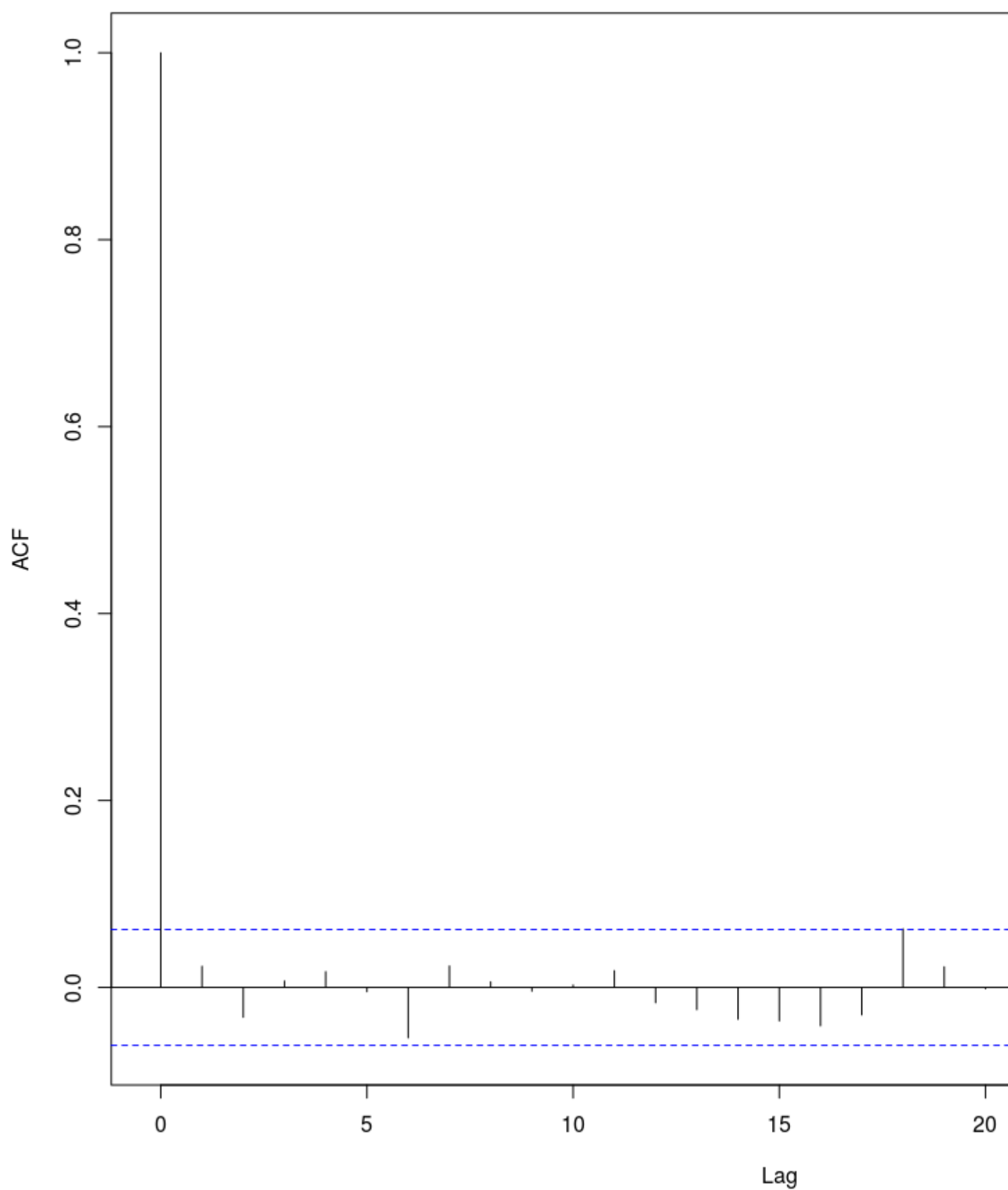
#Verify that the model eliminates the autocorrelation
acf(x)
```

Series 1



```
acf(fit$resid)
```

Series fit\$resid



```

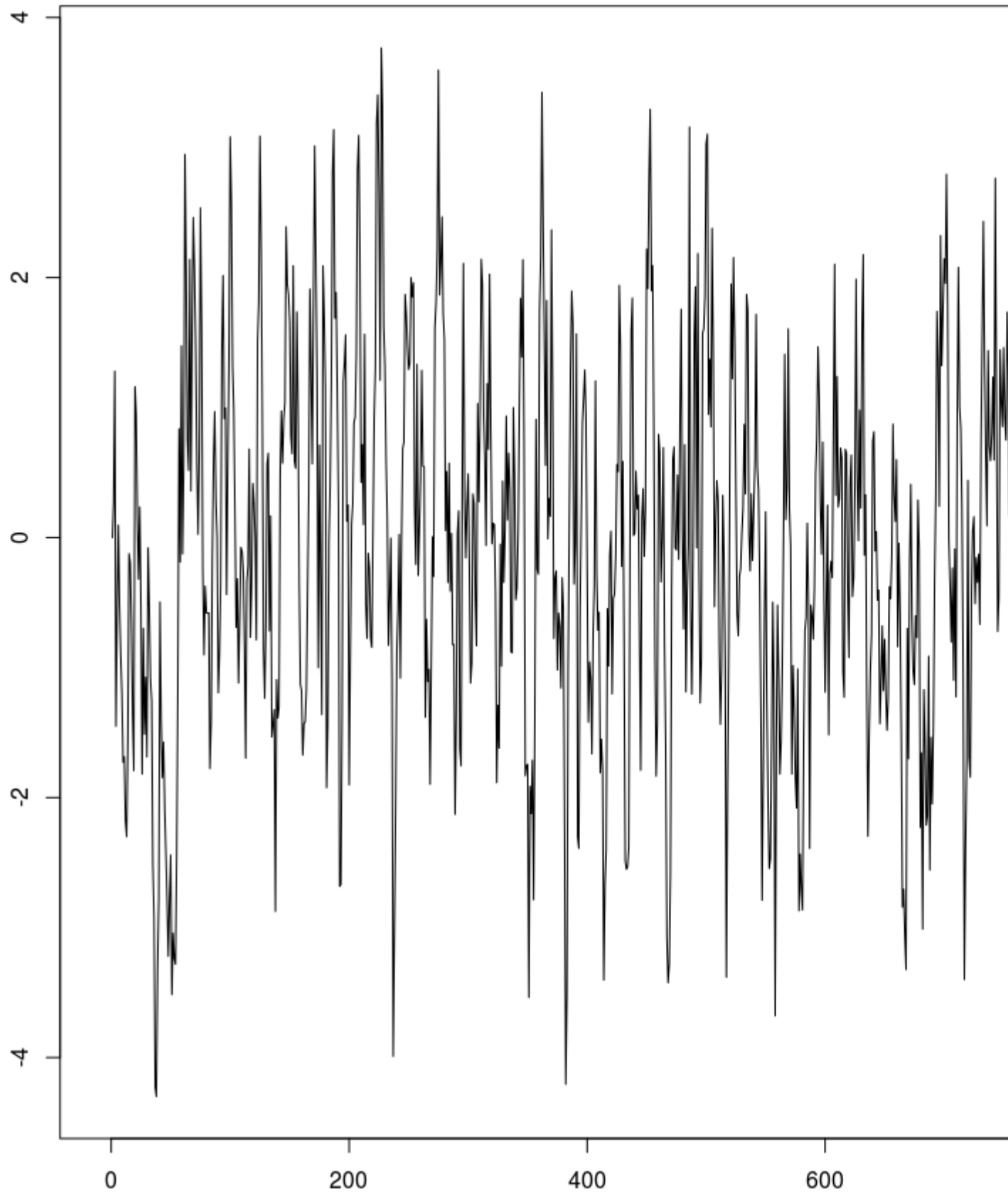
#Forecast 10 periods
fcst <- forecast(fit, h = 100)
fcst
      Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
1001    0.282529070 -0.9940493  1.559107 -1.669829  2.234887
1002    0.173976408 -1.3872262  1.735179 -2.213677  2.561630
1003    0.097554408 -1.5869850  1.782094 -2.478726  2.673835
1004    0.043752667 -1.6986831  1.786188 -2.621073  2.708578
1005    0.005875783 -1.7645535  1.776305 -2.701762  2.713514
...

#Call the point predictions
fcst$mean
# Time Series:
# Start = 1001
# End = 1100
# Frequency = 1
[1] 0.282529070 0.173976408 0.097554408 0.043752667 0.005875783 -0.020789866 -
0.039562711 -0.052778954
[9] -0.062083302
...

#Plot the forecast
plot(fcst)

```

Forecasts from ARIMA(1,0,0) with non-zero



Arima-Modelle online lesen: <https://riptutorial.com/de/r/topic/1725/arima-modelle>

Kapitel 10: Ausdruck: parse + eval

Bemerkungen

Die Funktion `parse` konvertiert Text und Dateien in Ausdrücke.

Die Funktion `eval` wertet Ausdrücke aus.

Examples

Code im String-Format ausführen

In diesem Beispiel möchten wir Code ausführen, der in einem String-Format gespeichert ist.

```
# the string
str <- "1+1"

# A string is not an expression.
is.expression(str)
[1] FALSE

eval(str)
[1] "1+1"

# parse convert string into expressions
parsed.str <- parse(text="1+1")

is.expression(parsed.str)
[1] TRUE

eval(parsed.str)
[1] 2
```

Ausdruck: parse + eval online lesen: <https://riptutorial.com/de/r/topic/5746/ausdruck--parse-plus-eval>

Kapitel 11: Balkendiagramm

Einführung

Der Zweck des Balkendiagramms ist die Anzeige der Frequenzen (oder Proportionen) der Pegel einer Faktorvariablen. Beispielsweise wird ein Balkendiagramm verwendet, um die Häufigkeit (oder Anteile) von Individuen in verschiedenen sozioökonomischen (Faktor-) Gruppen (Stufen hoch, mittel, niedrig) bildlich darzustellen. Eine solche Darstellung hilft dabei, einen visuellen Vergleich zwischen den verschiedenen Faktorstufen zu erstellen.

Examples

Barplot () - Funktion

In der Balkendiagrammdarstellung werden die Faktorstufen auf der X-Achse und die Frequenzen (oder Anteile) der verschiedenen Faktorstufen auf der Y-Achse angezeigt. Für jede Faktorebene wird ein Balken mit einheitlicher Breite konstruiert, wobei die Höhen proportional zur Frequenz (oder dem Anteil) der Faktorstufe sind.

Die Funktion `barplot()` befindet sich im Grafikpaket der `barplot()` von R. Die Funktion `barplot()` muss mindestens ein Argument enthalten. Die R-Hilfe nennt dies `heights`, die entweder Vektor oder Matrix sein müssen. Wenn es ein Vektor ist, sind seine Mitglieder die verschiedenen Faktorebenen.

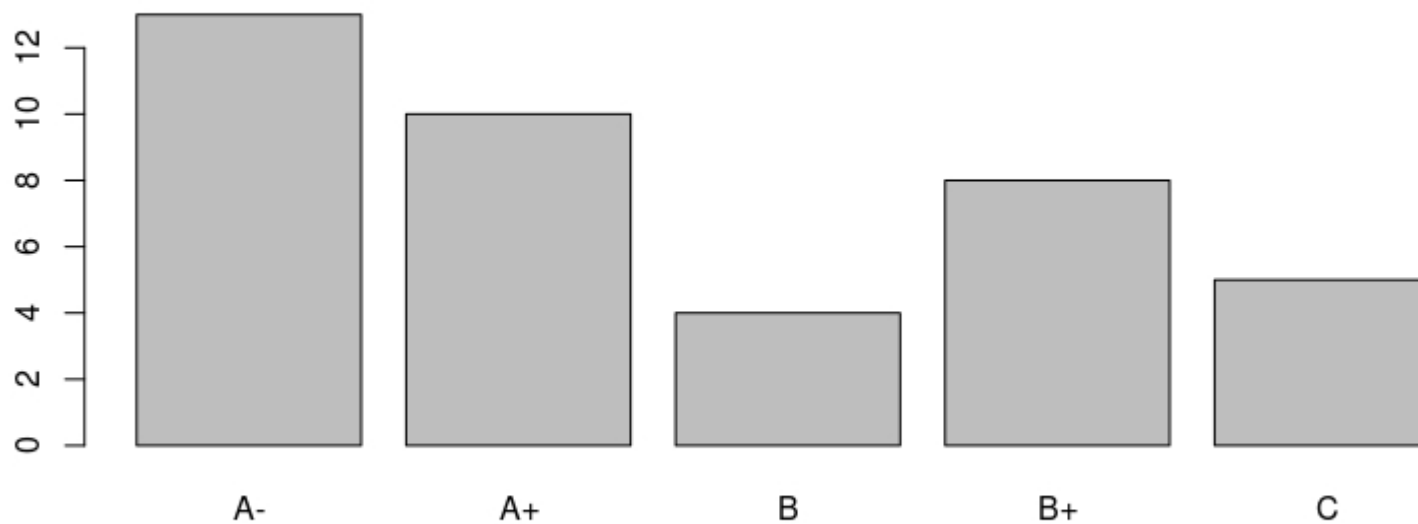
Um `barplot()` zu veranschaulichen, betrachten Sie die folgende Datenvorbereitung:

```
> grades<-c("A+", "A-", "B+", "B", "C")
> Marks<-sample(grades, 40, replace=T, prob=c(.2, .3, .25, .15, .1))
> Marks
[1] "A+" "A-" "B+" "A-" "A+" "B"  "A+" "B+" "A-" "B"  "A+" "A-"
[13] "A-" "B+" "A-" "A-" "A-" "A-" "A+" "A-" "A+" "A+" "C"  "C"
[25] "B"  "C"  "B+" "C"  "B+" "B+" "B+" "A+" "B+" "A-" "A+" "A-"
[37] "A-" "B"  "C"  "A+"
>
```

Ein Balkendiagramm des Markierungsvektors wird von erhalten

```
> barplot(table(Marks), main="Mid-Marks in Algorithms")
```

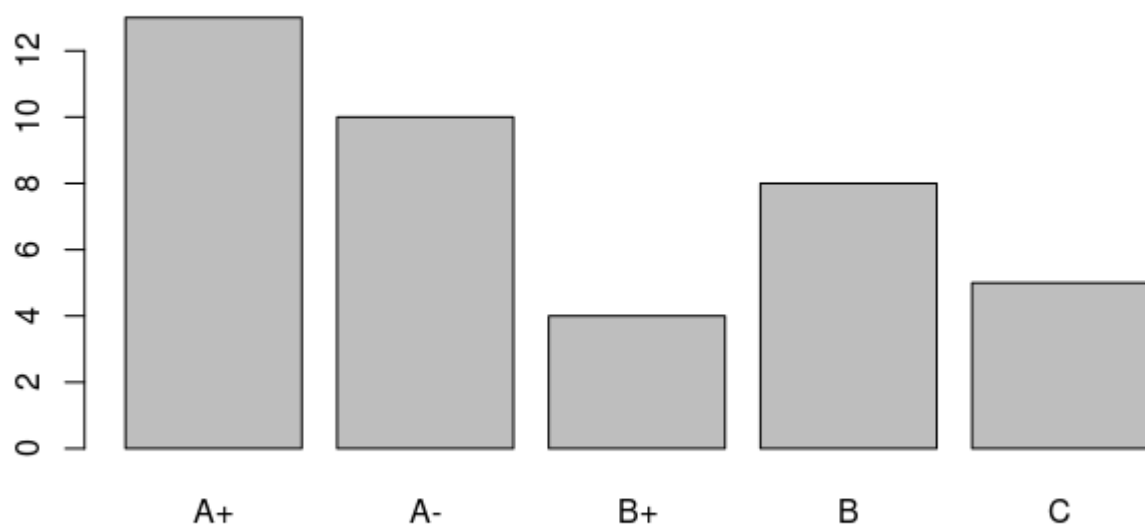
Mid-Marks in Algorithms



Beachten Sie, dass die `Barplot()` - Funktion die Faktorebenen auf der X-Achse in der `lexicographical order` der Ebenen platziert. Mit dem Parameter `names.arg` können die Balken in der Reihenfolge platziert werden, wie sie in dem Vektor *Grade* angegeben ist .

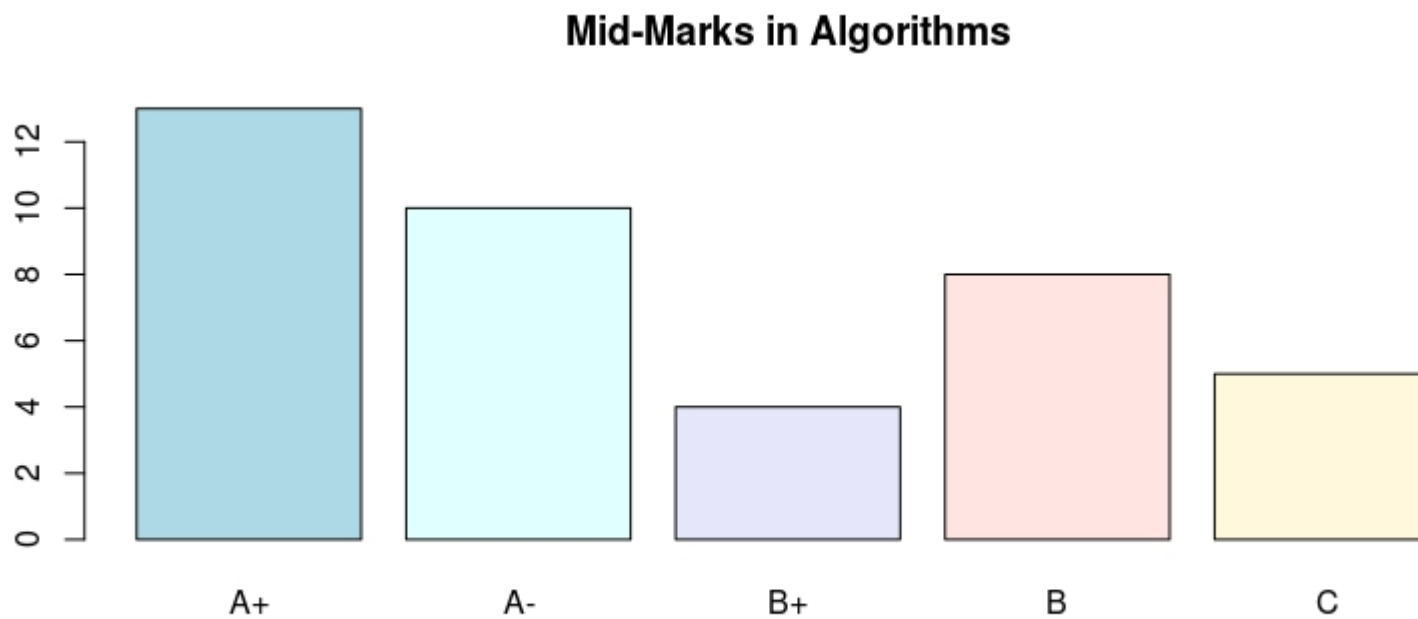
```
# plot to the desired horizontal axis labels  
> barplot(table(Marks),names.arg=grades ,main="Mid-Marks in Algorithms")
```

Mid-Marks in Algorithms



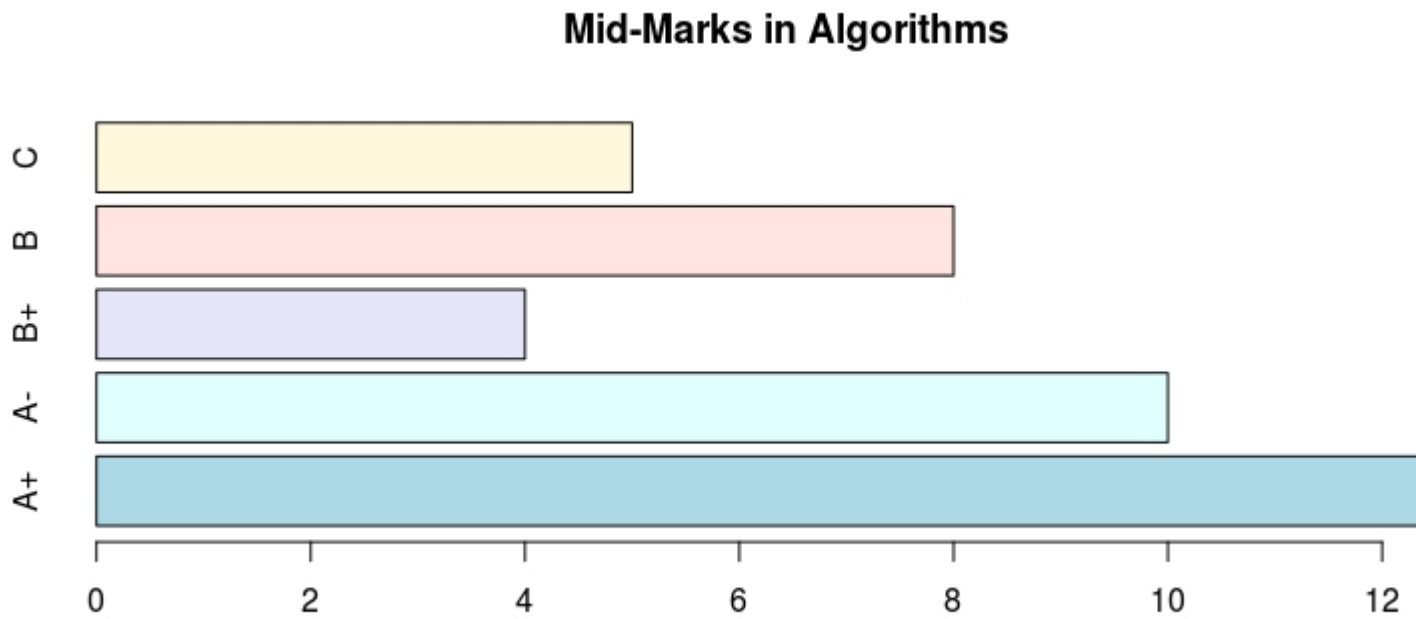
Farbige Balken können mit dem Parameter `col=` gezeichnet werden.

```
> barplot(table(Marks),names.arg=grades,col = c("lightblue",  
  "lightcyan", "lavender", "mistyrose", "cornsilk"),  
  main="Mid-Marks in Algorithms")
```



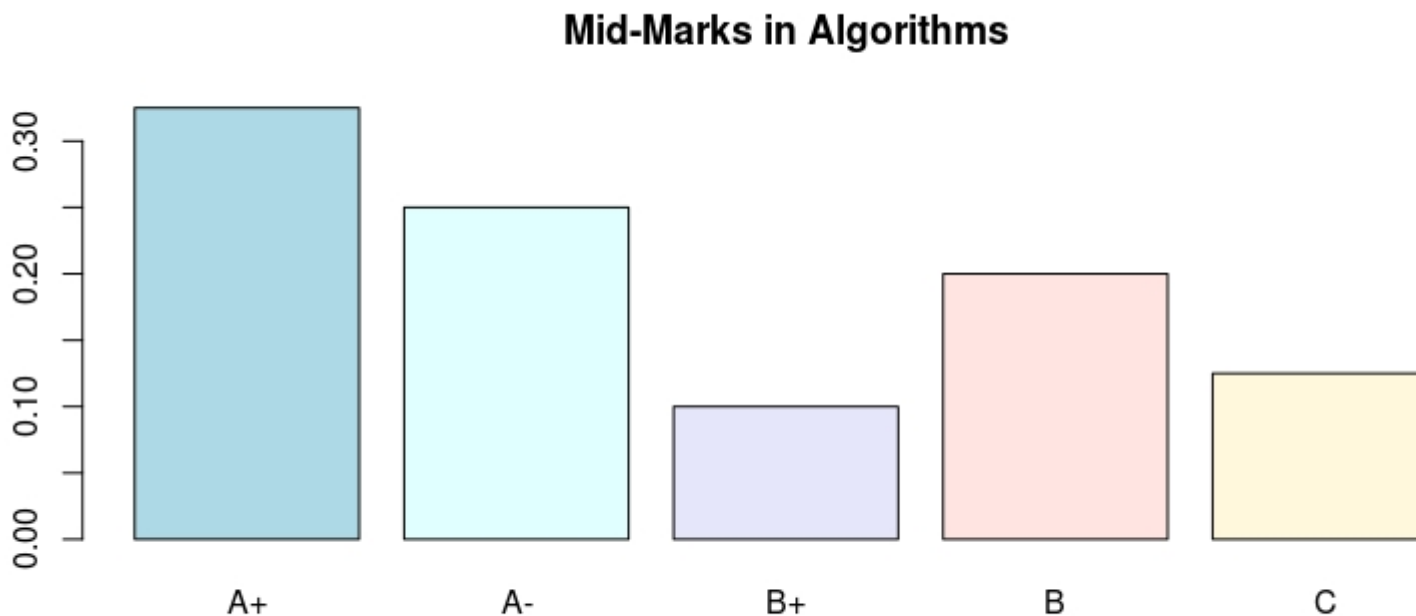
Ein Balkendiagramm mit *horizontalen Balken* erhalten Sie wie folgt:

```
> barplot(table(Marks),names.arg=grades,hORIZ=TRUE,col = c("lightblue",  
  "lightcyan", "lavender", "mistyrose", "cornsilk"),  
  main="Mid-Marks in Algorithms")
```



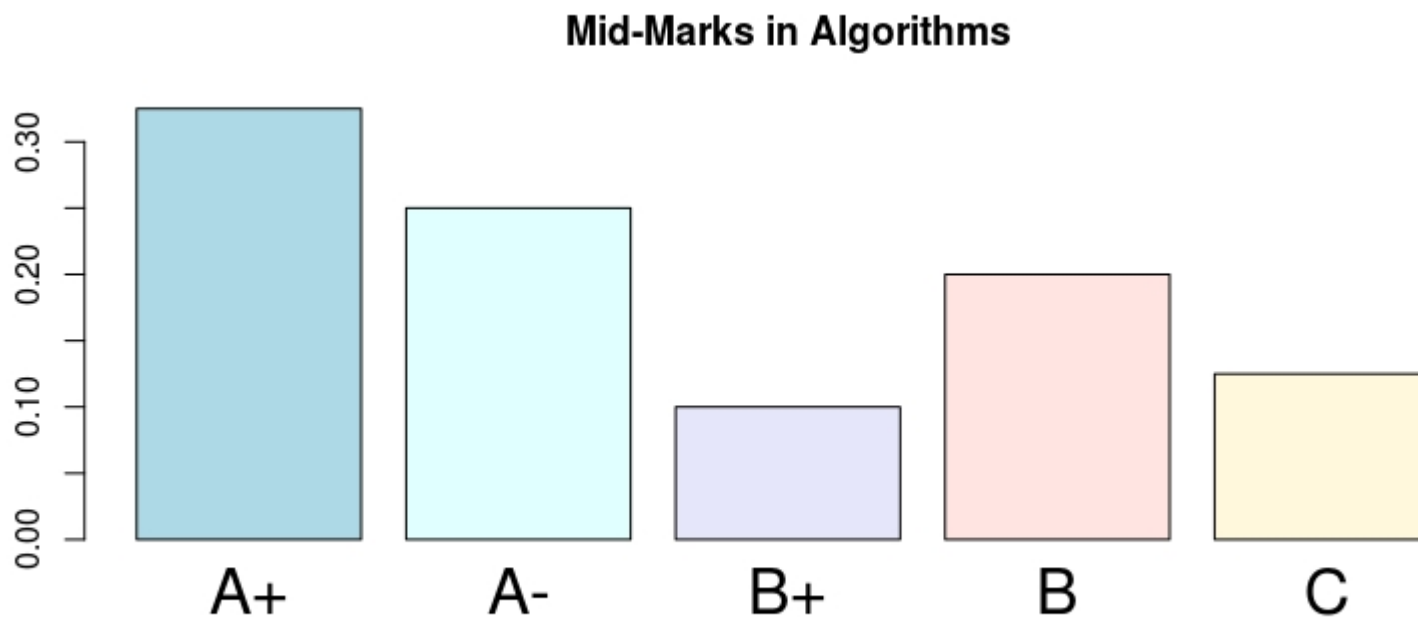
Ein Balkendiagramm mit *Proportionen* auf der y-Achse kann wie folgt erhalten werden:

```
> barplot(prop.table(table(Marks)), names.arg=grades, col = c("lightblue",
  "lightcyan", "lavender", "mistyrose", "cornsilk"),
  main="Mid-Marks in Algorithms")
```



Die Größen der `cex.names` auf der X-Achse können mit `cex.names` Parameter `cex.names` erhöht werden.

```
> barplot(prop.table(table(Marks)),names.arg=grades,col = c("lightblue",
  "lightcyan", "lavender", "mistyrose", "cornsilk"),
  main="Mid-Marks in Algorithms",cex.names=2)
```

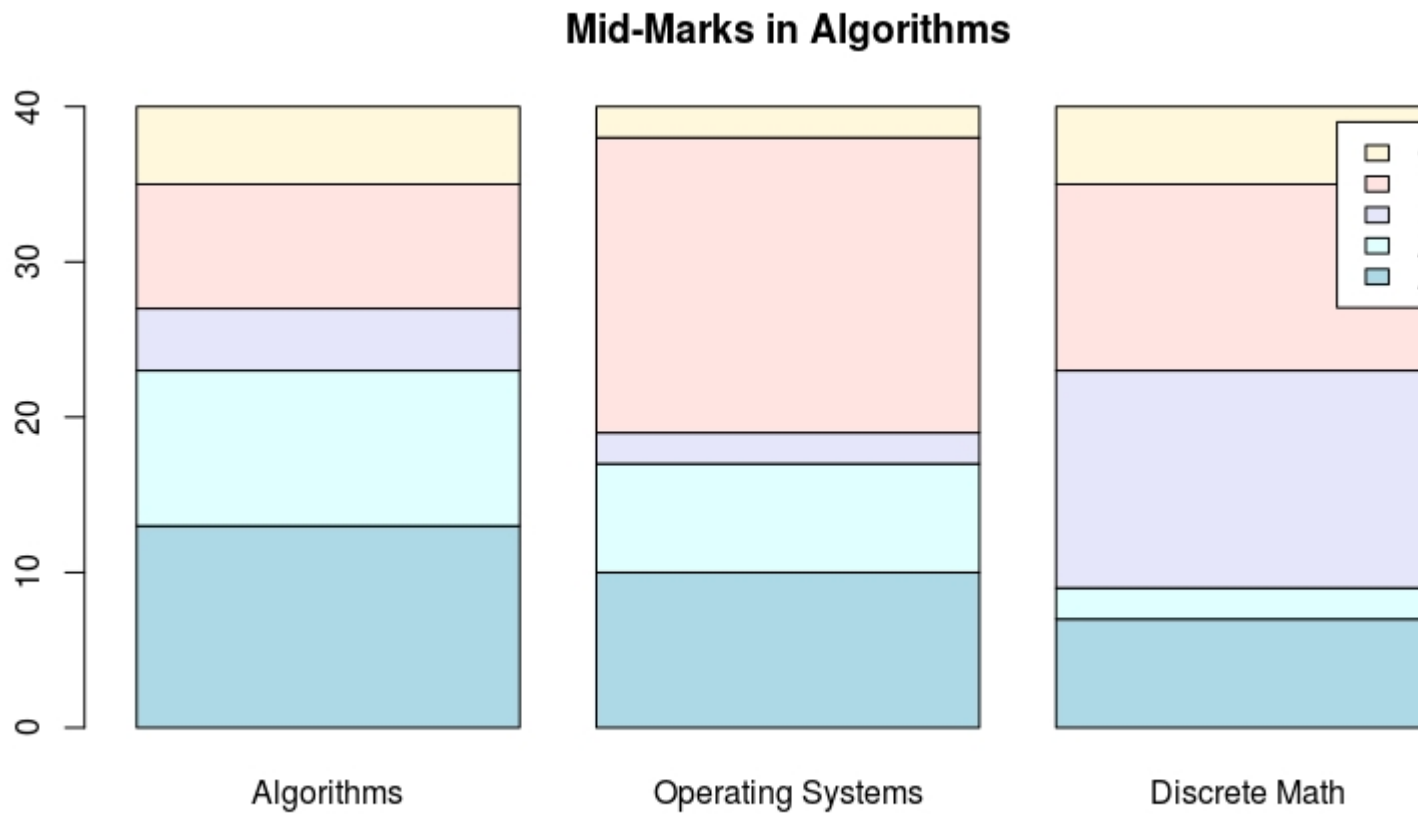


Der `heights` - Parameter der `barplot()` kann eine Matrix sein. Zum Beispiel könnte es sich um eine Matrix handeln, bei der die Spalten die verschiedenen Themen eines Kurses sind, die Zeilen die Bezeichnungen der Noten sein könnten. Betrachten Sie die folgende Matrix:

```
> gradTab
  Algorithms Operating Systems Discrete Math
A-         13             10             7
A+         10             7             2
B           4             2            14
B+          8            19            12
C           5             2             5
```

Um eine gestapelte Leiste zu zeichnen, verwenden Sie einfach den Befehl:

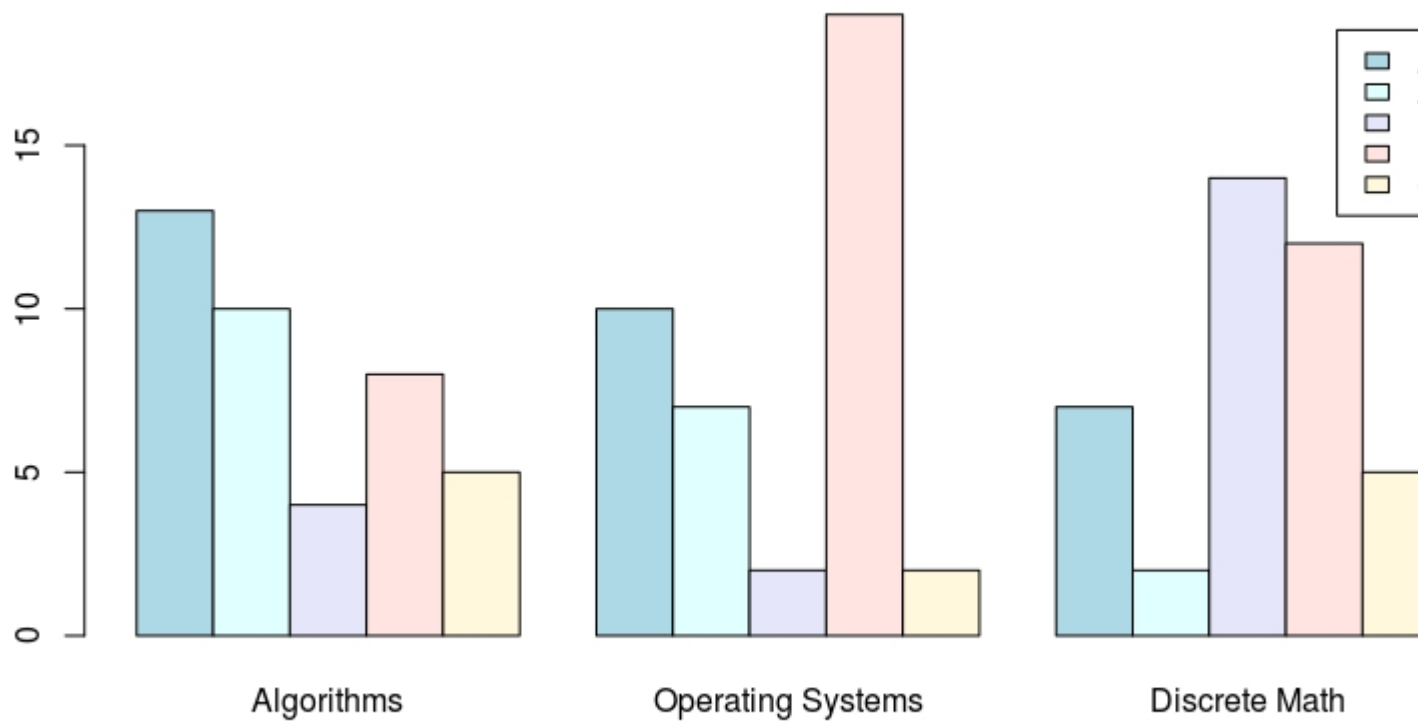
```
> barplot(gradTab,col = c("lightblue","lightcyan",
  "lavender", "mistyrose", "cornsilk"),legend.text = grades,
  main="Mid-Marks in Algorithms")
```



Um nebeneinander liegende Balken zu zeichnen, verwenden Sie den Parameter außerdem `besides` Parameter:

```
> barplot(gradTab,beside = T,col = c("lightblue","lightcyan",
  "lavender", "mistyrose", "cornsilk"),legend.text = grades,
  main="Mid-Marks in Algorithms")
```

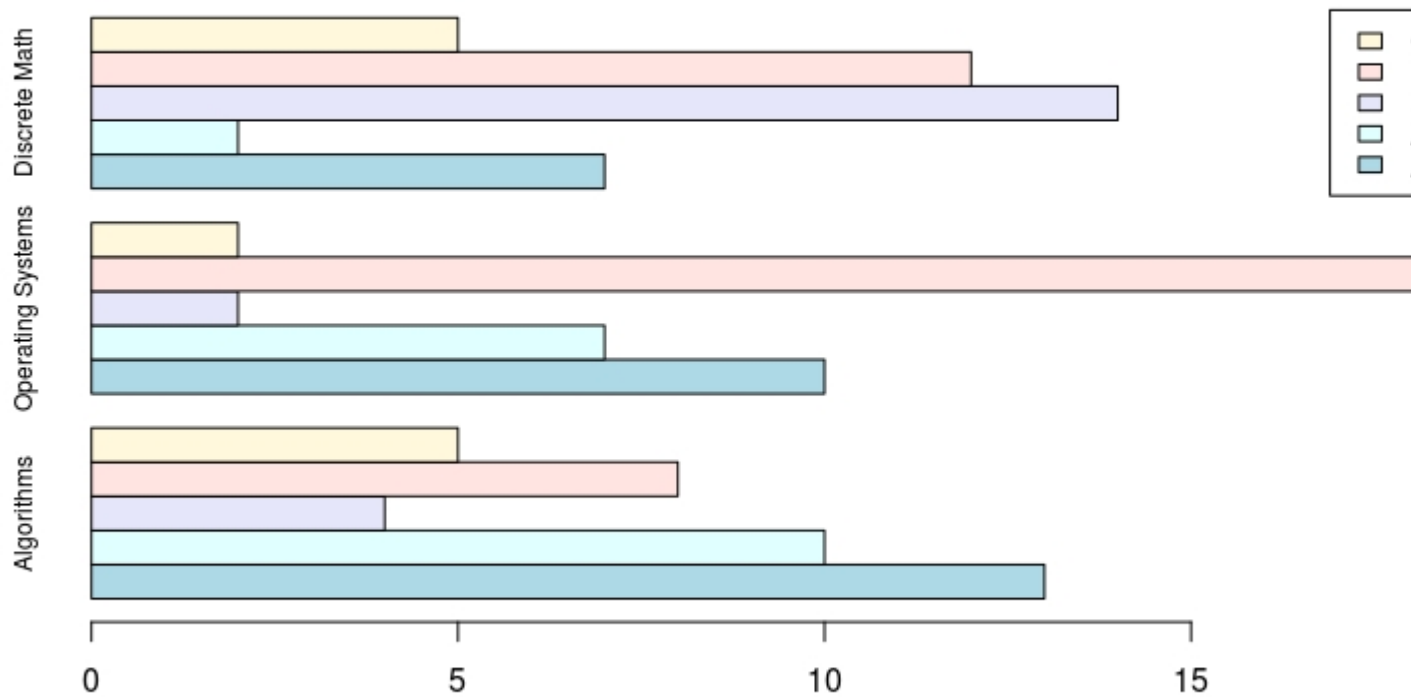
Mid-Marks in Algorithms



Ein horizontales Balkendiagramm kann mit dem Parameter `horiz=T` abgerufen werden:

```
> barplot(gradTab,beside = T,horiz=T,col = c("lightblue","lightcyan",  
"lavender", "mistyrose", "cornsilk"),legend.text = grades,  
cex.names=.75,main="Mid-Marks in Algorithms")
```

Mid-Marks in Algorithms



Balkendiagramm online lesen: <https://riptutorial.com/de/r/topic/8091/balkendiagramm>

Kapitel 12: Basisplot

Parameter

Parameter	Einzelheiten
x	x-Achsenvariable Kann entweder <code>data\$variablex</code> oder <code>data[,x]</code> liefern
y	y-Achsenvariable. Kann entweder <code>data\$variabley</code> oder <code>data[,y]</code> liefern
main	Haupttitel der Handlung
sub	Optionaler Untertitel der Handlung
xlab	Beschriftung für die x-Achse
ylab	Beschriftung für die y-Achse
pch	Eine ganze Zahl oder ein Zeichen, das ein Plot-Symbol anzeigt
col	Ganzzahl oder Zeichenfolge, die die Farbe angibt
type	Art des Grundstücks "p" für Punkte, "l" für Linien, "b" für beide, "c" für den Linienteil "b", "o" für beide "überzeichnet", "h" für "Histogramm" (oder "hohe Dichte") vertikale Linien, "s" für Treppenstufen, "S" für andere Schritte, "n" für kein Plotten

Bemerkungen

Bei den im Abschnitt "Parameter" aufgeführten Elementen handelt es sich um einen kleinen Bruchteil der möglichen Parameter, die von der `par` Funktion geändert oder eingestellt werden können. Siehe `par` für eine vollständigere Liste. Zusätzlich verfügen alle Grafikgeräte, einschließlich der systemspezifischen interaktiven Grafikgeräte, über eine Reihe von Parametern, mit denen die Ausgabe angepasst werden kann.

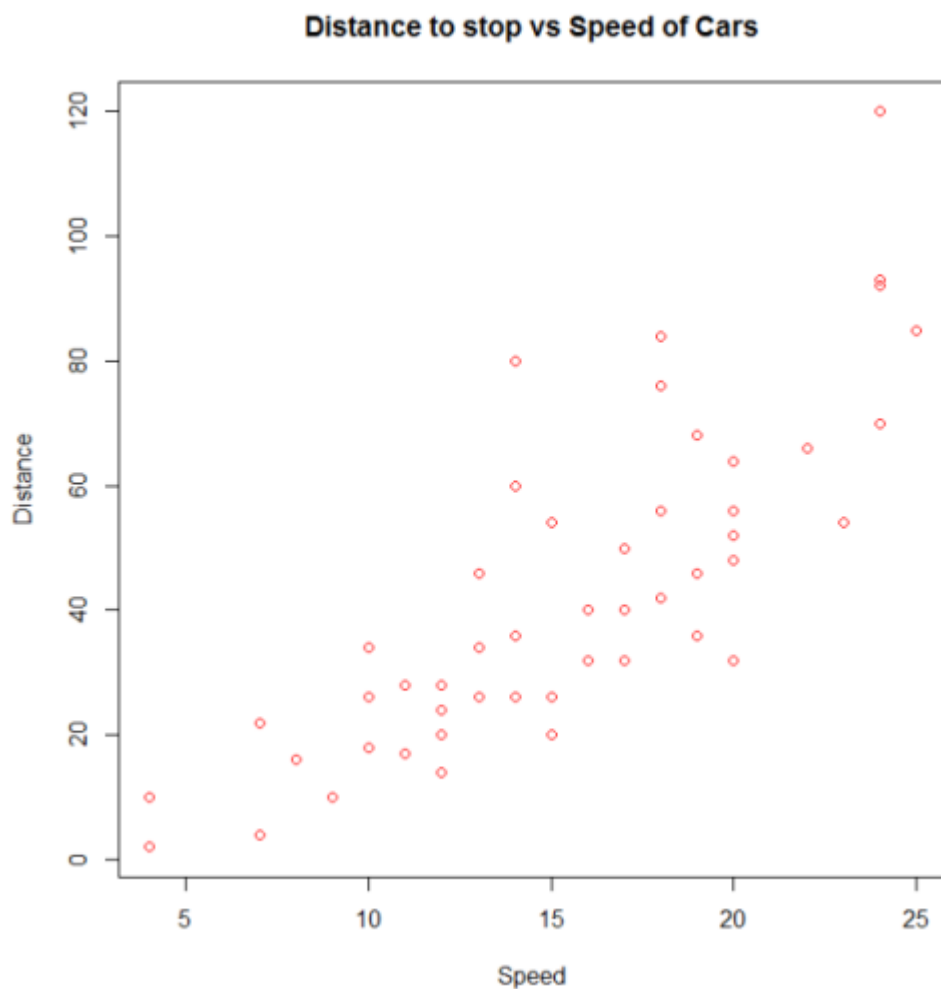
Examples

Grundlegende Darstellung

Ein grundlegendes Diagramm wird durch Aufrufen von `plot()`. Hier verwenden wir den Datenrahmen der eingebauten `cars`, der die Geschwindigkeit der Autos und die zurückgelegten Entfernungen in den 1920er Jahren enthält. (Weitere Informationen zum Datensatz finden Sie in der Hilfe (Autos)).

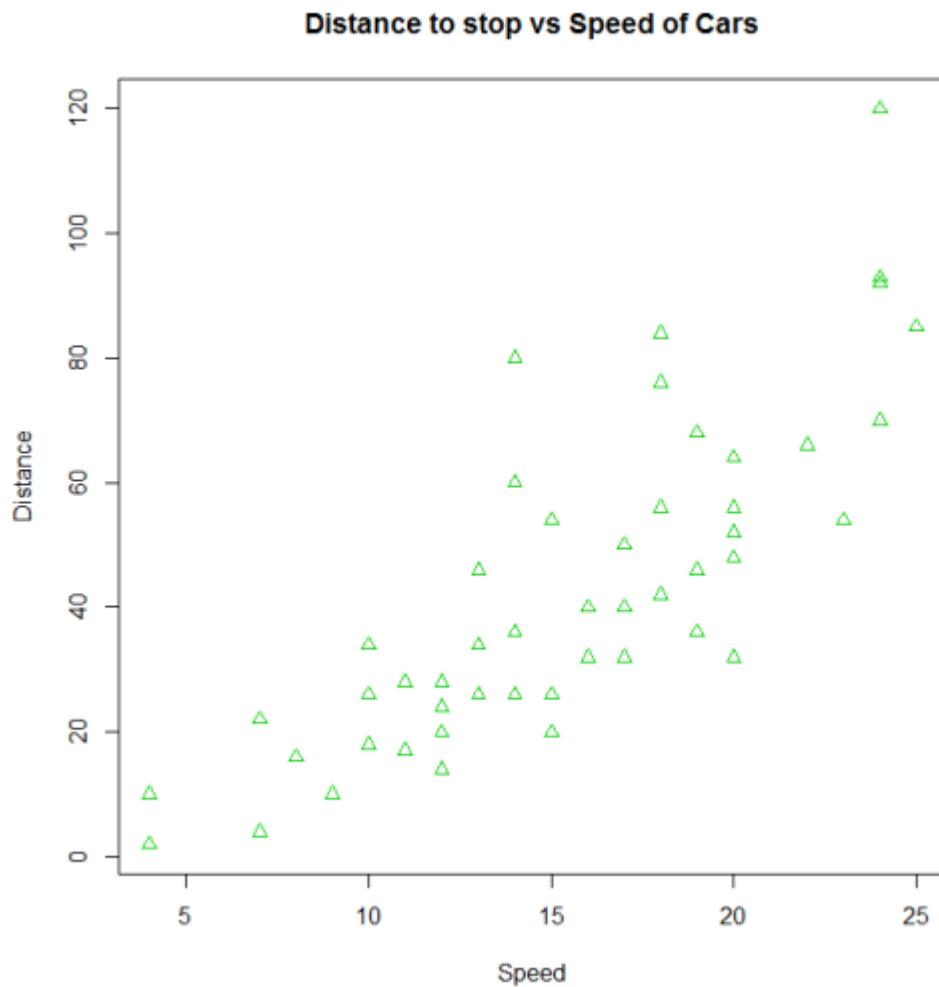
```
plot(x = cars$speed, y = cars$dist, pch = 1, col = 1,
```

```
main = "Distance vs Speed of Cars",  
xlab = "Speed", ylab = "Distance")
```



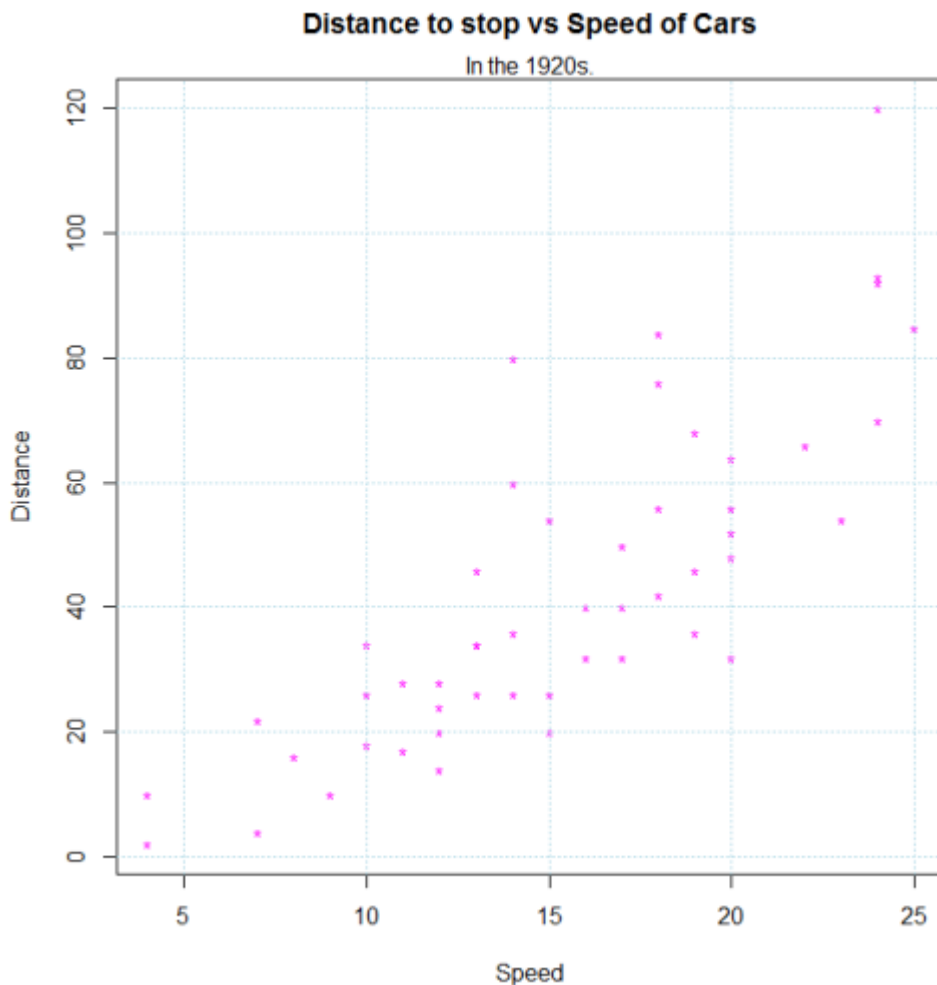
Wir können viele andere Variationen im Code verwenden, um das gleiche Ergebnis zu erzielen. Wir können die Parameter auch ändern, um unterschiedliche Ergebnisse zu erhalten.

```
with(cars, plot(dist~speed, pch = 2, col = 3,  
  main = "Distance to stop vs Speed of Cars",  
  xlab = "Speed", ylab = "Distance"))
```



Zusätzliche Funktionen können zu diesem Diagramm hinzugefügt werden, indem `points()`, `text()`, `mtext()`, `lines()`, `grid()` usw. `mtext()`.

```
plot(dist~speed, pch = "*", col = "magenta", data=cars,
     main = "Distance to stop vs Speed of Cars",
     xlab = "Speed", ylab = "Distance")
mtext("In the 1920s.")
grid(col="lightblue")
```



Matplot

`matplot` ist nützlich, um mehrere Sätze von Beobachtungen von demselben Objekt, insbesondere von einer Matrix, in derselben Grafik schnell zu `matplot`.

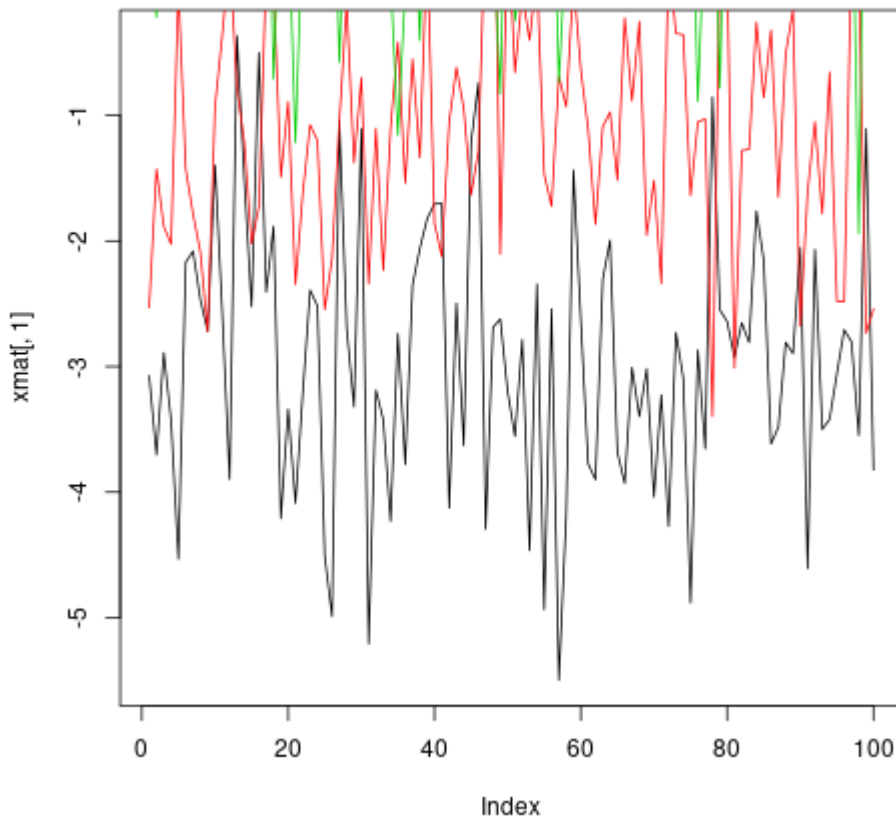
Hier ist ein Beispiel für eine Matrix, die vier Gruppen von Zufallszahlen enthält, die jeweils einen anderen Mittelwert haben.

```
xmat <- cbind(rnorm(100, -3), rnorm(100, -1), rnorm(100, 1), rnorm(100, 3))
head(xmat)
#      [,1]      [,2]      [,3]      [,4]
# [1,] -3.072793 -2.53111494  0.6168063  3.780465
# [2,] -3.702545 -1.42789347 -0.2197196  2.478416
# [3,] -2.890698 -1.88476126  1.9586467  5.268474
# [4,] -3.431133 -2.02626870  1.1153643  3.170689
# [5,] -4.532925  0.02164187  0.9783948  3.162121
# [6,] -2.169391 -1.42699116  0.3214854  4.480305
```

Eine Möglichkeit, alle diese Beobachtungen auf dem gleichen Graphen zeichnen soll man tun `plot` Aufruf durch drei weitere gefolgt `points` oder `lines` Aufrufe.

```
plot(xmat[,1], type = 'l')
lines(xmat[,2], col = 'red')
```

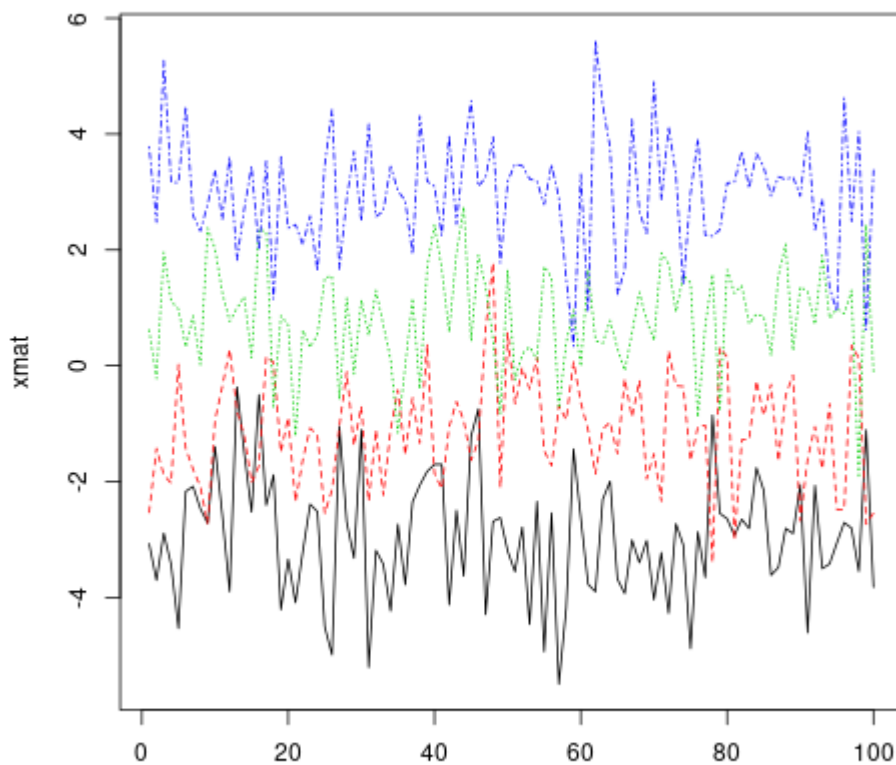
```
lines(xmat[,3], col = 'green')
lines(xmat[,4], col = 'blue')
```



Dies ist jedoch sowohl langwierig als auch problematisch, da unter anderem die Achsengrenzen standardmäßig durch `plot` festgelegt werden, um nur in die erste Spalte zu passen.

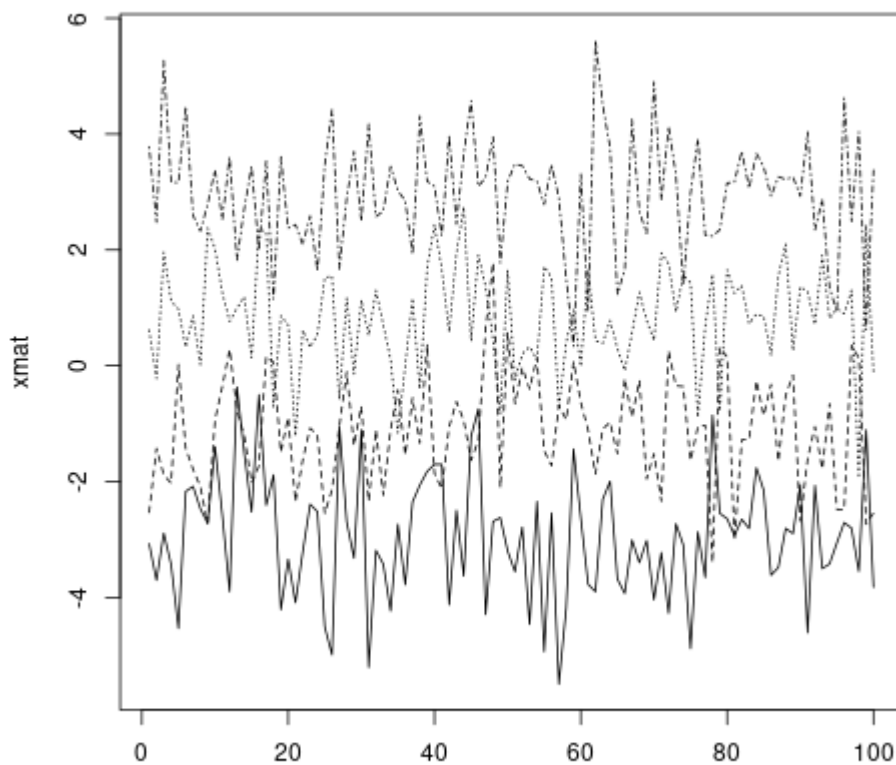
In dieser Situation ist es viel praktischer, die `matplot` Funktion zu verwenden, die nur einen Aufruf erfordert und automatisch die Achsenbegrenzungen berücksichtigt *und* die Ästhetik für jede Spalte ändert, um sie unterscheidbar zu machen.

```
matplot(xmat, type = 'l')
```



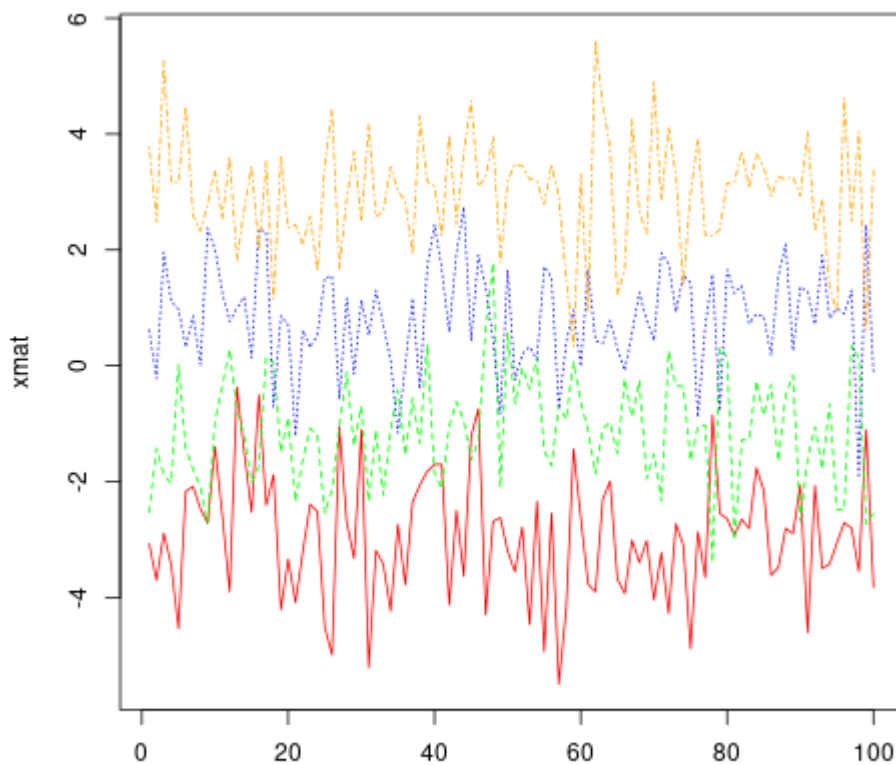
Beachten Sie, dass `matplotlib` standardmäßig sowohl Farbe (`col`) als auch Linientyp (`lty`) `lty` da dies die Anzahl möglicher Kombinationen erhöht, bevor sie wiederholt werden. Jede dieser Ästhetiken (oder beide) kann jedoch auf einen einzigen Wert festgelegt werden ...

```
matplotlib(xmat, type = 'l', col = 'black')
```



... oder einen benutzerdefinierten Vektor (der nach den Standardregeln für den R-Vektor-Recycling auf die Anzahl der Spalten zurückgeführt wird)

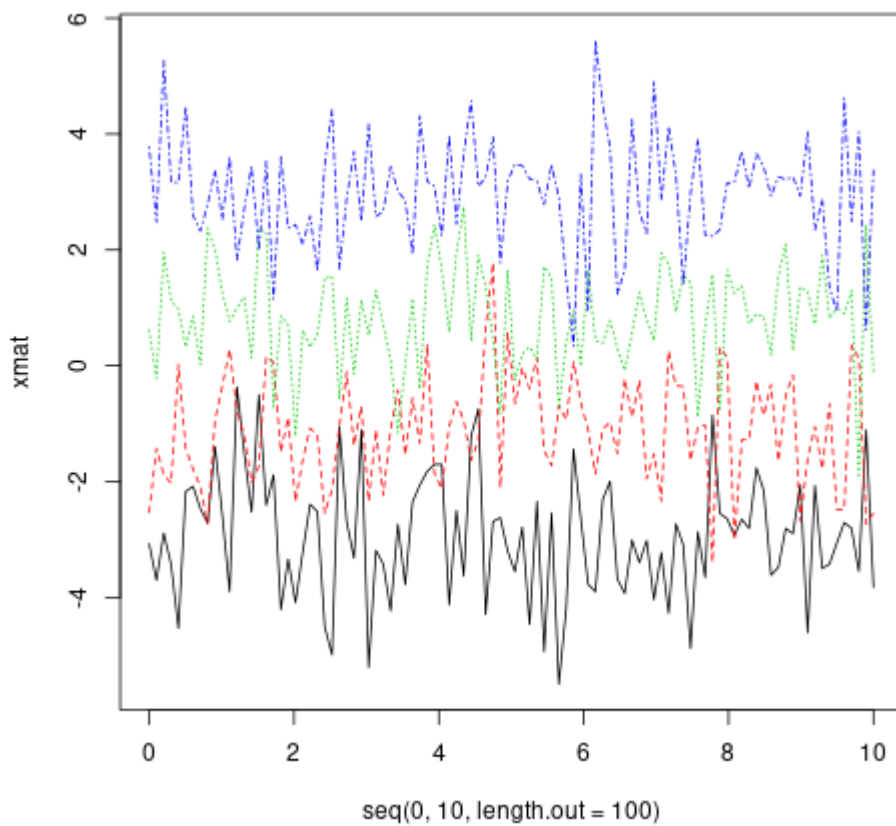
```
matplot(xmat, type = 'l', col = c('red', 'green', 'blue', 'orange'))
```



Grafische Standardparameter, einschließlich `main` , `xlab` , `xmin` , funktionieren genauso wie beim `plot` . Für mehr darüber, siehe `?par` .

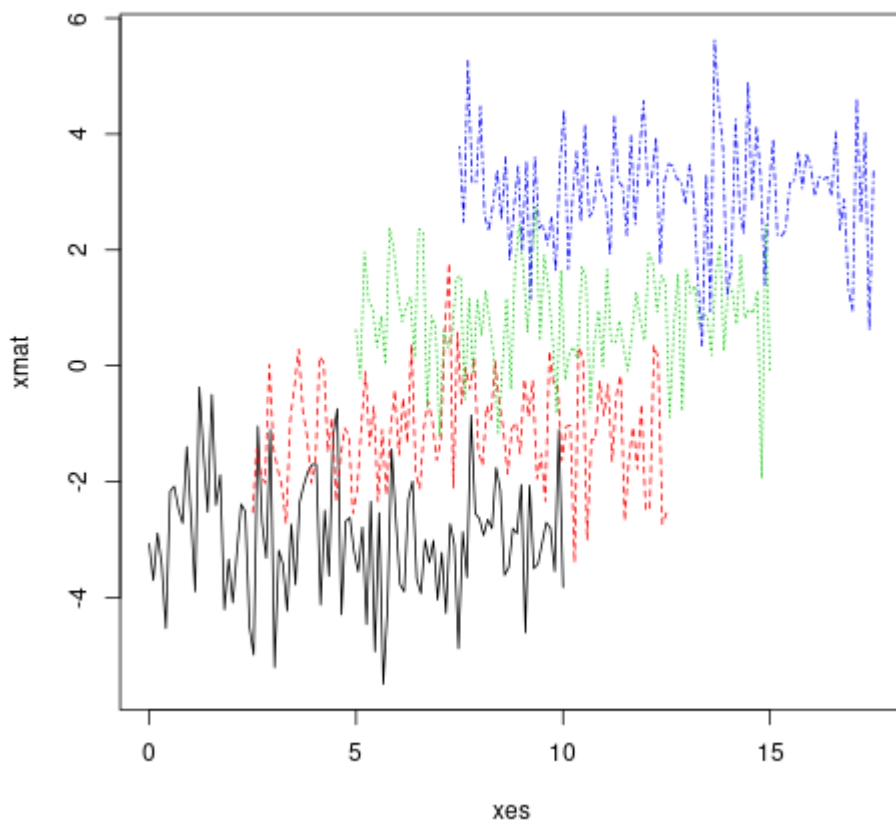
Wie `plot` , gegeben , wenn nur ein Objekt, `matplot` nimmt das ist `y` Variable und verwendet die Indizes für `x` . `x` und `y` können jedoch explizit angegeben werden.

```
matplot(x = seq(0, 10, length.out = 100), y = xmat, type='l')
```

In der Tat können sowohl x als auch y Matrizen sein.

```
xes <- cbind(seq(0, 10, length.out = 100),
             seq(2.5, 12.5, length.out = 100),
             seq(5, 15, length.out = 100),
             seq(7.5, 17.5, length.out = 100))
matplot(x = xes, y = xmat, type = 'l')
```

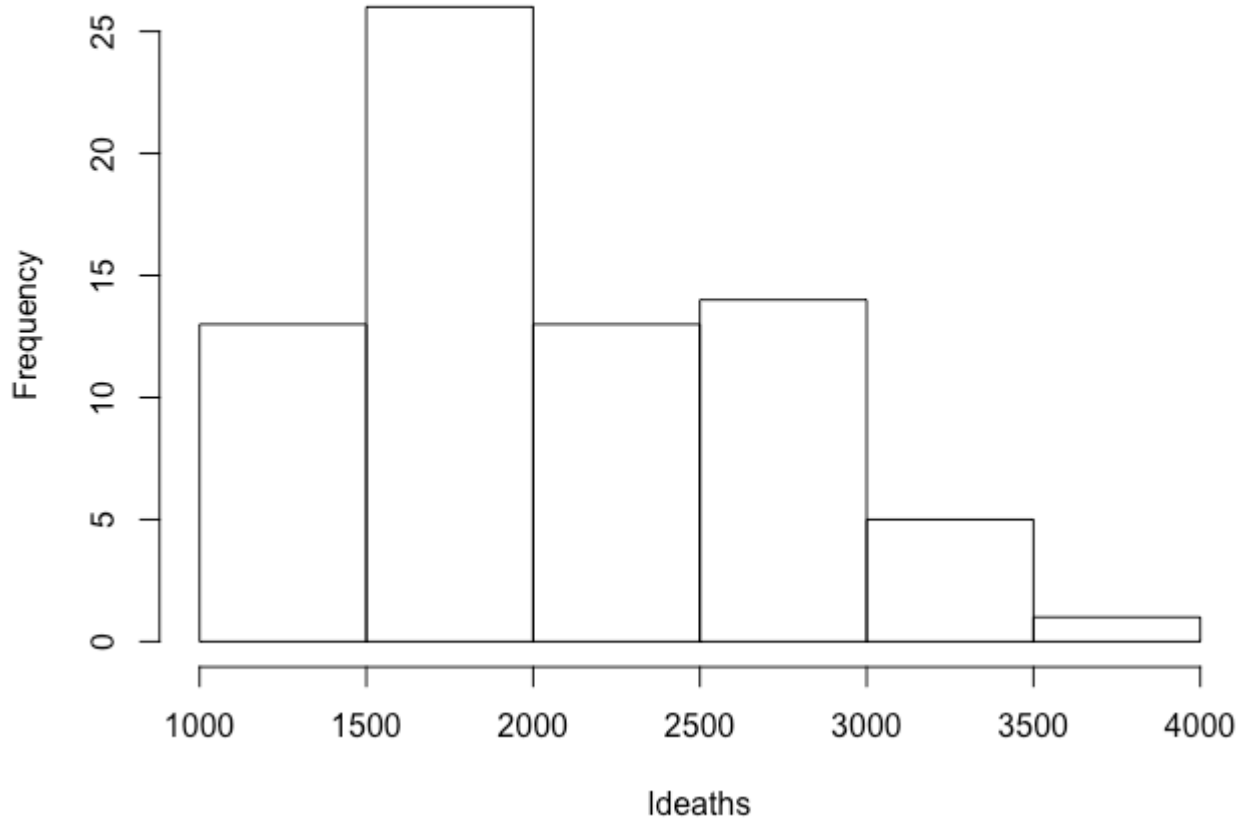


Histogramme

Histogramme ermöglichen eine Pseudo-Darstellung der zugrunde liegenden Verteilung der Daten.

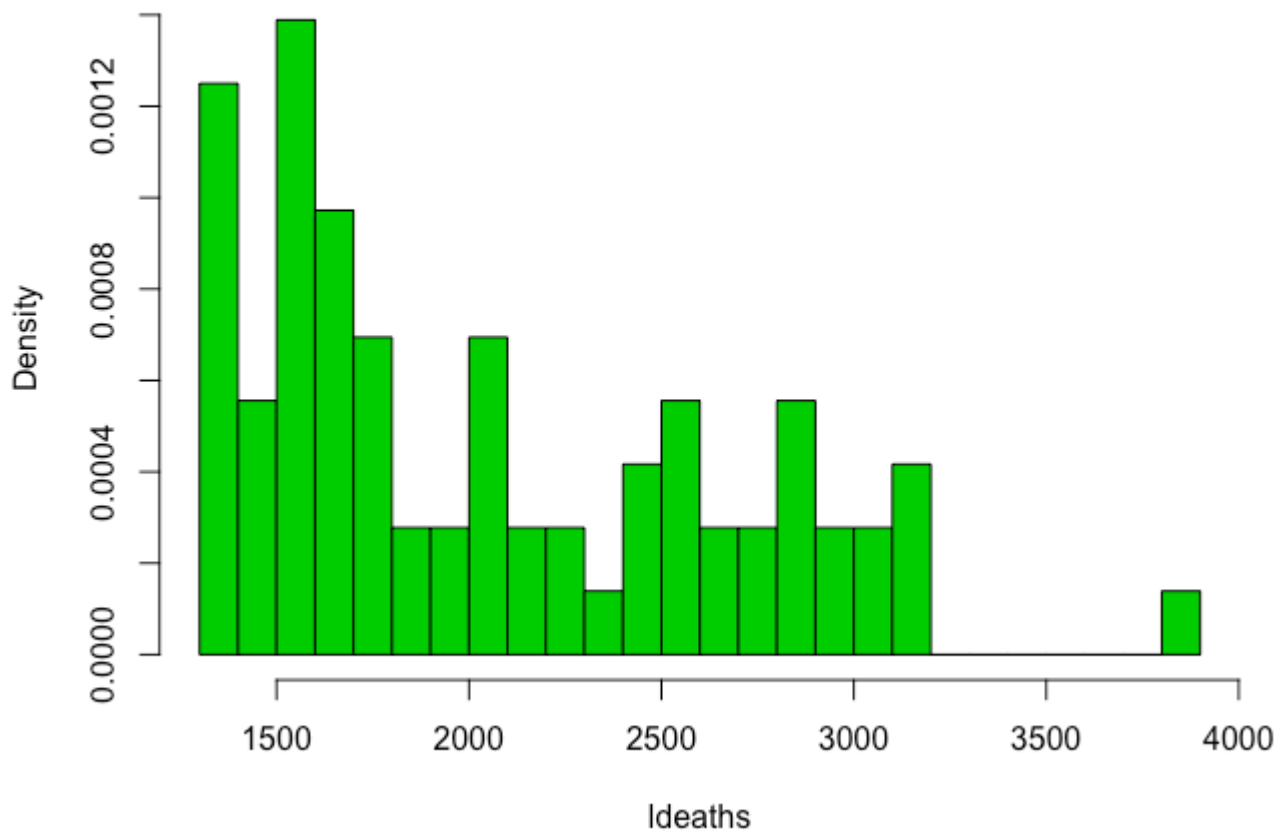
```
hist(ldeaths)
```

Histogram of Ideaths



```
hist(ideaths, breaks = 20, freq = F, col = 3)
```

Histogram of Ideaths



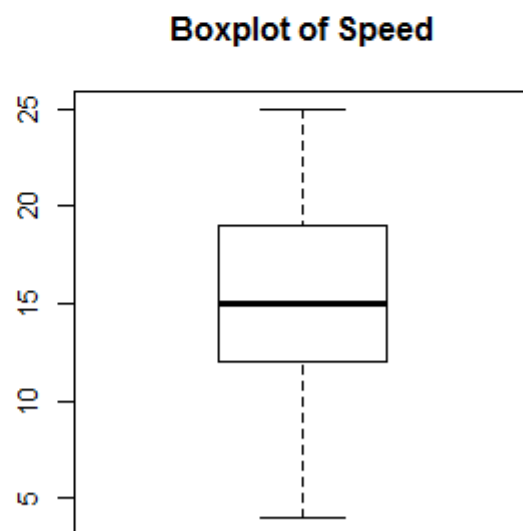
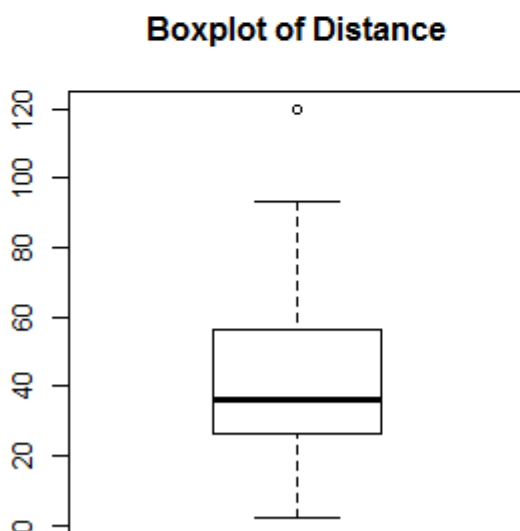
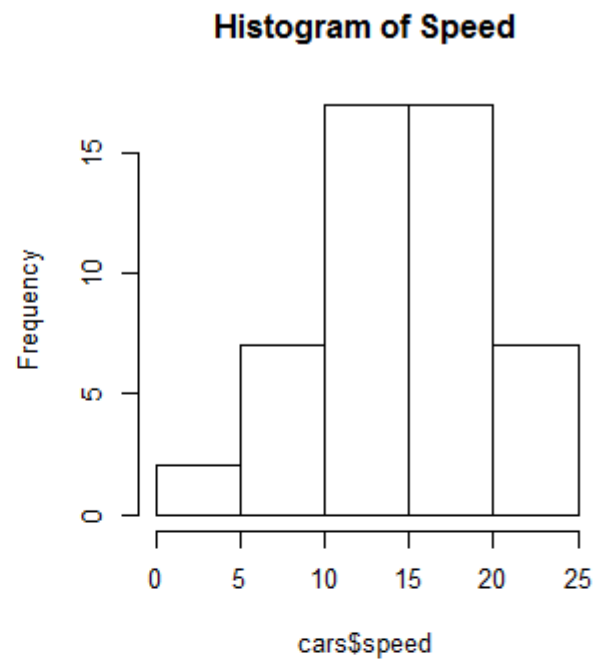
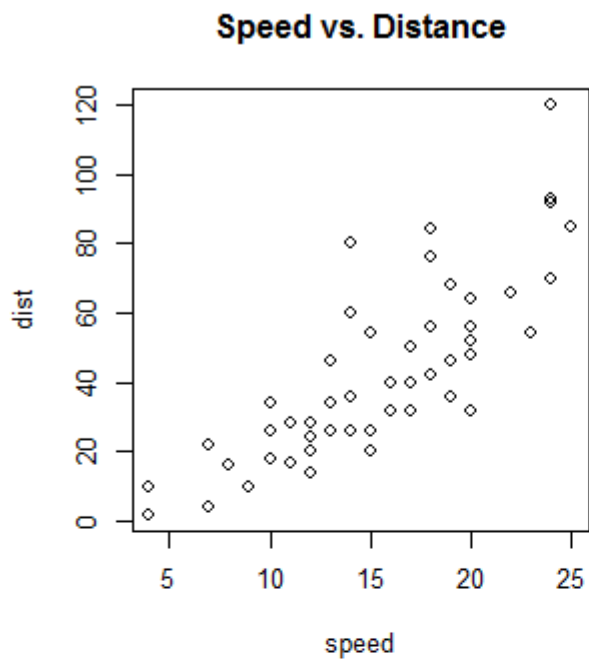
Plots kombinieren

Es ist oft nützlich, mehrere Diagrammtypen in einem Diagramm zu kombinieren (z. B. ein Balkendiagramm neben einem Streudiagramm.) R macht dies mit Hilfe der Funktionen `par()` und `layout()` einfach.

`par()`

`par` verwendet die Argumente `mfrow` oder `mfcol`, um eine Matrix aus `nrows` und `ncols` `c(nrows, ncols)` die als Raster für Ihre Plots dienen. Das folgende Beispiel zeigt, wie vier Diagramme in einem Diagramm kombiniert werden:

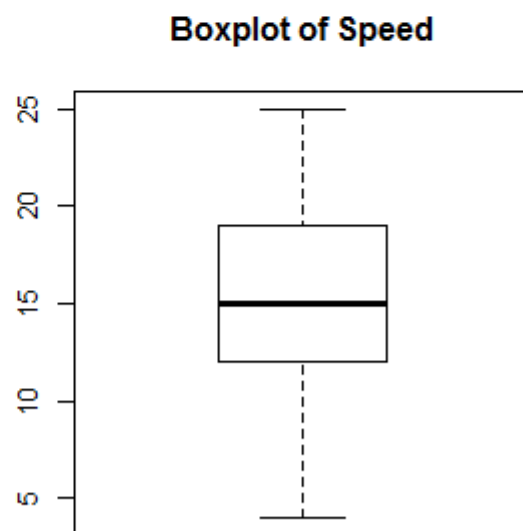
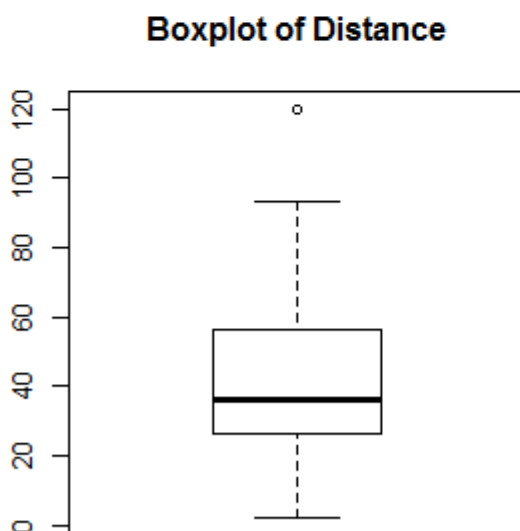
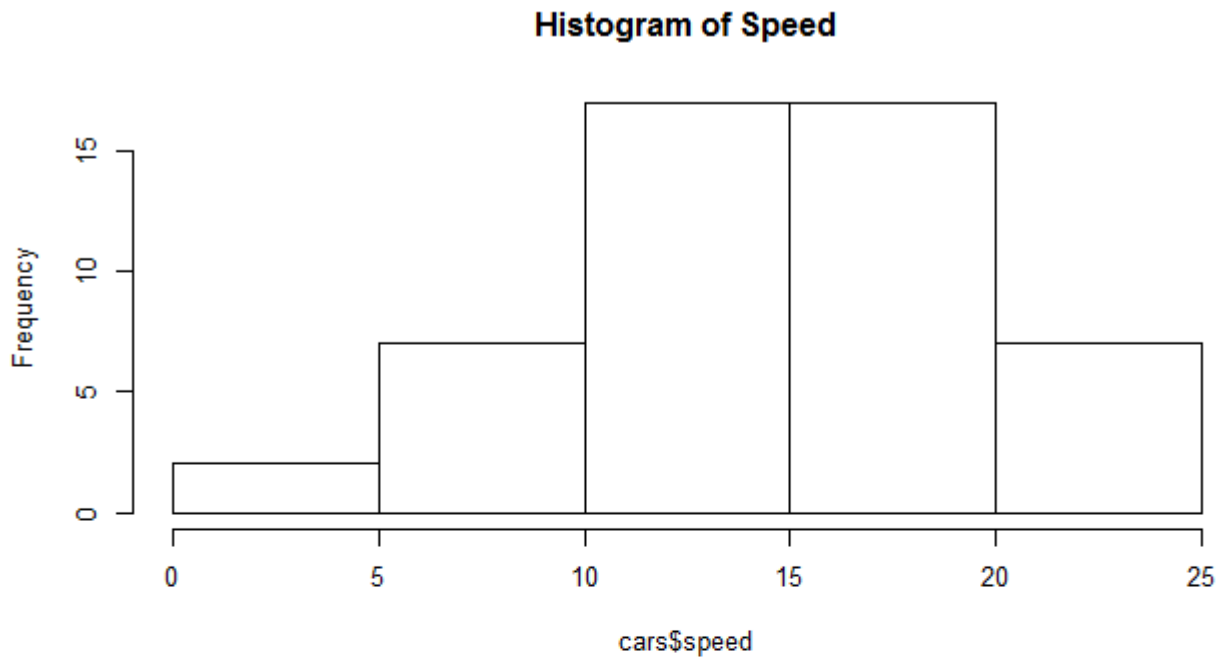
```
par(mfrow=c(2,2))
plot(cars, main="Speed vs. Distance")
hist(cars$speed, main="Histogram of Speed")
boxplot(cars$dist, main="Boxplot of Distance")
boxplot(cars$speed, main="Boxplot of Speed")
```



`layout ()`

Das `layout ()` ist flexibler und ermöglicht es Ihnen, die Position und die Ausdehnung jedes Diagramms innerhalb des endgültigen kombinierten Diagramms anzugeben. Diese Funktion erwartet ein Matrixobjekt als Eingabe:

```
layout(matrix(c(1,1,2,3), 2,2, byrow=T))
hist(cars$speed, main="Histogram of Speed")
boxplot(cars$dist, main="Boxplot of Distance")
boxplot(cars$speed, main="Boxplot of Speed")
```

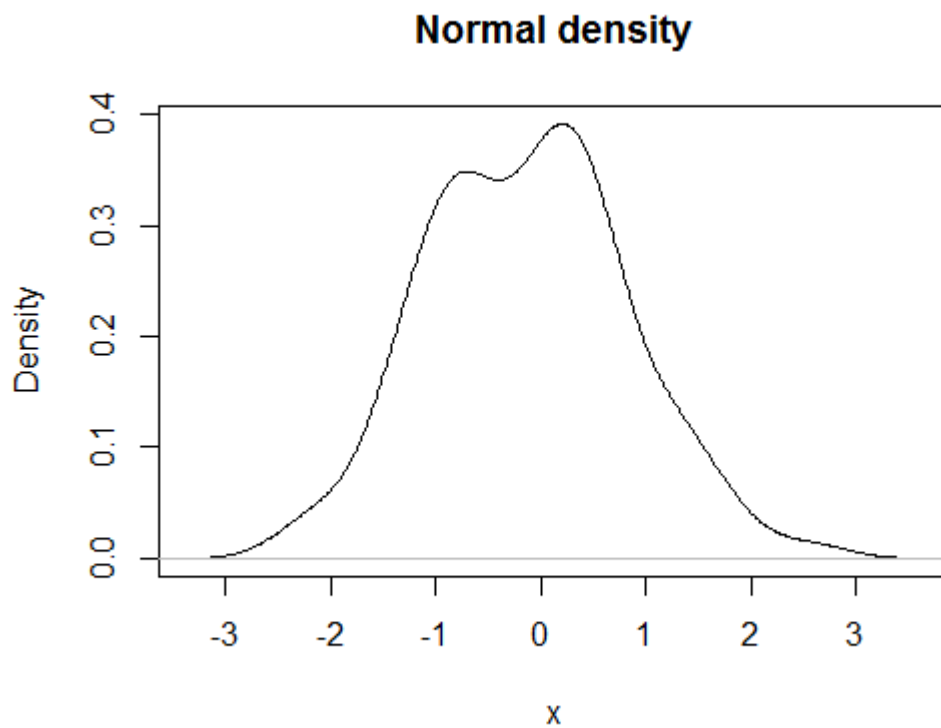


Dichteplot

Eine sehr nützliche und logische Folge von Histogrammen wäre die Darstellung der geglätteten Dichtefunktion einer Zufallsvariablen. Eine durch den Befehl erzeugte Grunddarstellung

```
plot(density(rnorm(100)),main="Normal density",xlab="x")
```

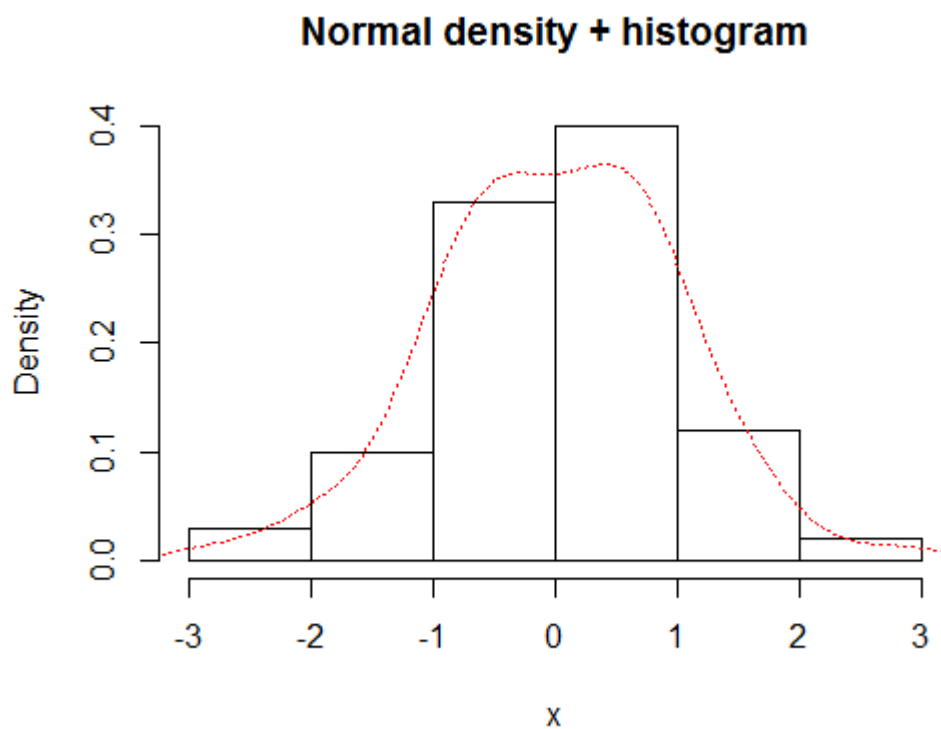
würde aussehen wie



Sie können ein Histogramm und eine Dichtekurve mit überlagern

```
x=rnorm(100)
hist(x,prob=TRUE,main="Normal density + histogram")
lines(density(x),lty="dotted",col="red")
```

was gibt



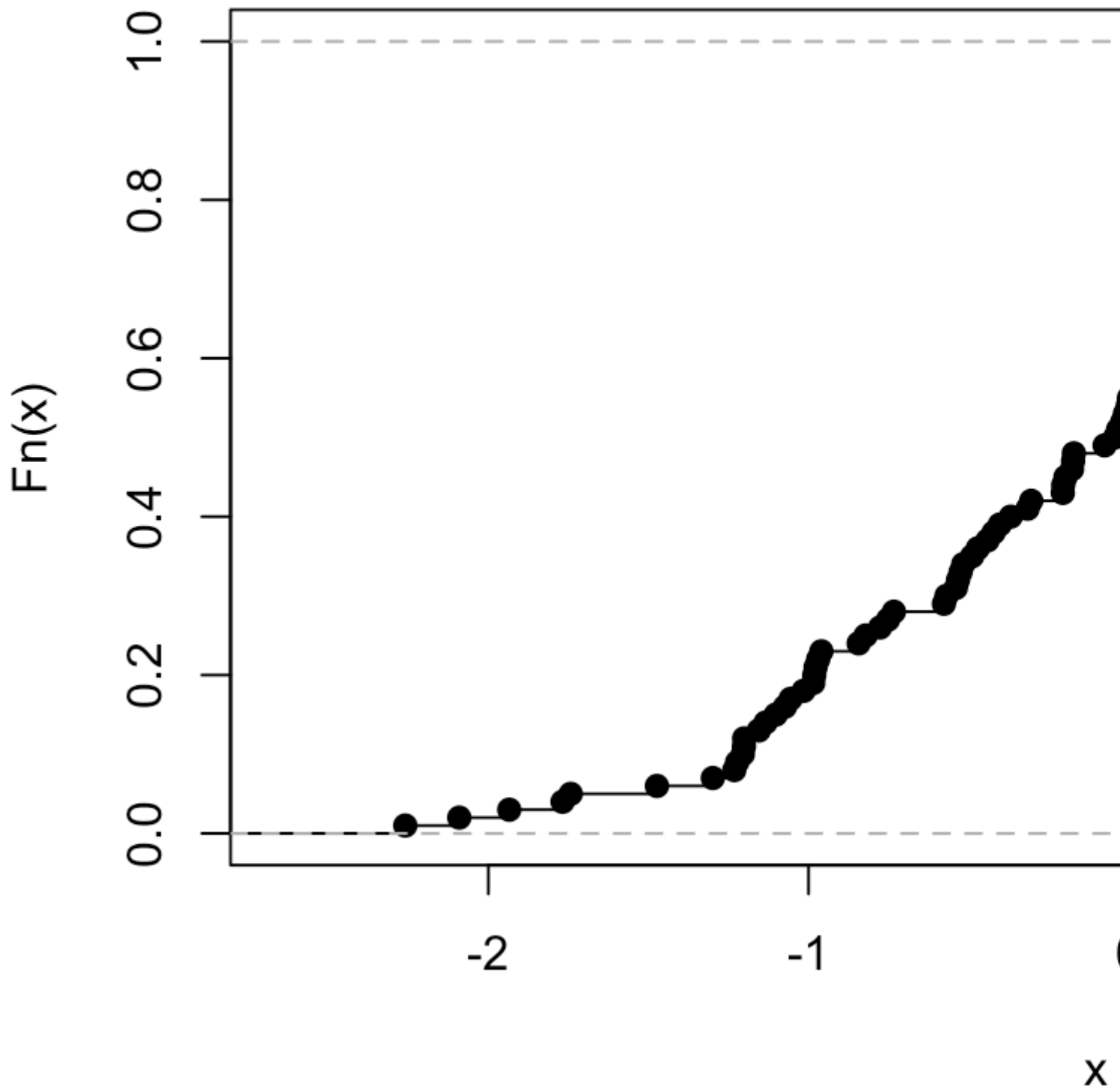
Empirische kumulative Verteilungsfunktion

Eine sehr nützliche und logische Folge von Histogrammen und Dichtediagrammen wäre die empirische kumulative Verteilungsfunktion. Wir können dazu die Funktion `ecdf()` verwenden. Eine durch den Befehl erzeugte Grunddarstellung

```
plot(ecdf(rnorm(100)),main="Cumulative distribution",xlab="x")
```

würde aussehen wie

Cumulative o



Erste Schritte mit R_Plots

- Streudiagramm

Sie haben zwei Vektoren und möchten diese grafisch darstellen.

```
x_values <- rnorm(n = 20 , mean = 5 , sd = 8) #20 values generated from Normal(5,8)
y_values <- rbeta(n = 20 , shape1 = 500 , shape2 = 10) #20 values generated from Beta(500,10)
```

Wenn Sie ein Diagramm `y_values` möchten, das die `y_values` in der vertikalen Achse und die `x_values` in der horizontalen Achse hat, können Sie die folgenden Befehle verwenden:

```
plot(x = x_values, y = y_values, type = "p") #standard scatter-plot
plot(x = x_values, y = y_values, type = "l") # plot with lines
plot(x = x_values, y = y_values, type = "n") # empty plot
```

Sie können `?plot()` in die Konsole eingeben, um weitere Optionen zu erfahren.

- **Box-Plot**

Sie haben einige Variablen und möchten deren Verteilung untersuchen

```
#boxplot is an easy way to see if we have some outliers in the data.

z<- rbeta(20 , 500 , 10) #generating values from beta distribution
z[c(19 , 20)] <- c(0.97 , 1.05) # replace the two last values with outliers
boxplot(z) # the two points are the outliers of variable z.
```

- **Histogramme**

Einfache Möglichkeit, Histogramme zu zeichnen

```
hist(x = x_values) # Histogram for x vector
hist(x = x_values, breaks = 3) #use breaks to set the numbers of bars you want
```

- **Kreisdiagramme**

Wenn Sie die Frequenzen einer Variablen visualisieren möchten, zeichnen Sie einfach einen Kreis

Zuerst müssen wir Daten mit Frequenzen erzeugen, zum Beispiel:

```
P <- c(rep('A' , 3) , rep('B' , 10) , rep('C' , 7) )
t <- table(P) # this is a frequency matrix of variable P
pie(t) # And this is a visual version of the matrix above
```

Basisplot online lesen: <https://riptutorial.com/de/r/topic/1377/basisplot>

Kapitel 13: Benutzereingabe abrufen

Syntax

- `Variable <- readline` (Eingabeaufforderung = "Beliebige Nachricht für Benutzer")
- `name <- readline(prompt = "wie heißt du?")`

Examples

Benutzereingabe in R

Manchmal kann es interessant sein, ein Gespräch zwischen dem Benutzer und dem Programm zu haben, zum Beispiel das [Wirbelpaket](#), das entwickelt wurde, um R in R zu lehren.

Mit dem Befehl `readline` kann nach Benutzereingaben gefragt werden:

```
name <- readline(prompt = "What is your name?")
```

Der Benutzer kann dann eine beliebige Antwort geben, beispielsweise eine Zahl, ein Zeichen, Vektoren und das Scannen des Ergebnisses ist hier, um sicherzustellen, dass der Benutzer eine richtige Antwort gegeben hat. Zum Beispiel:

```
result <- readline(prompt = "What is the result of 1+1?")
while(result!=2){
  readline(prompt = "Wrong answer. What is the result of 1+1?")
}
```

Es ist jedoch zu beachten, dass dieser Code in einer unendlichen Schleife hängen bleibt, da Benutzereingaben als Zeichen gespeichert werden.

Wir müssen es zu einer Zahl zwingen, indem `as.numeric`:

```
result <- as.numeric(readline(prompt = "What is the result of 1+1?"))
while(result!=2){
  readline(prompt = "Wrong answer. What is the result of 1+1?")
}
```

Benutzereingabe abrufen online lesen: <https://riptutorial.com/de/r/topic/5098/benutzereingabe-abrufen>

Kapitel 14: Beschleunigen Sie Code, der stark zu vektorisieren ist

Examples

Beschleunigen Sie mit Rcpp die zähste Vektorisierung für Loops

Betrachten Sie die folgende hard-to-vectorize for-Schleife, die einen Vektor der Länge `len` in dem das erste Element angegeben ist (`first` Element) und jedes Element `xi` gleich `cos(xi-1 + 1)` :

```
repeatedCosPlusOne <- function(first, len) {  
  x <- numeric(len)  
  x[1] <- first  
  for (i in 2:len) {  
    x[i] <- cos(x[i-1] + 1)  
  }  
  return(x)  
}
```

Dieser Code beinhaltet eine for-Schleife mit einer schnellen Operation (`cos(x[i-1]+1)`), die häufig von der Vektorisierung profitiert. Es ist jedoch nicht trivial, diese Operation mit der Basis R zu vektorisieren, da R keine Funktion "kumulativer Cosinus von `x + 1`" hat.

Ein möglicher Ansatz, um diese Funktion zu beschleunigen, wäre die Implementierung in C++ mithilfe des Rcpp-Pakets:

```
library(Rcpp)  
cppFunction("NumericVector repeatedCosPlusOneRcpp(double first, int len) {  
  NumericVector x(len);  
  x[0] = first;  
  for (int i=1; i < len; ++i) {  
    x[i] = cos(x[i-1]+1);  
  }  
  return x;  
}")
```

Dies führt häufig zu erheblichen Beschleunigungen für große Berechnungen, wobei dieselben Ergebnisse erzielt werden:

```
all.equal(repeatedCosPlusOne(1, 1e6), repeatedCosPlusOneRcpp(1, 1e6))  
# [1] TRUE  
system.time(repeatedCosPlusOne(1, 1e6))  
#   user  system elapsed  
# 1.274   0.015   1.310  
system.time(repeatedCosPlusOneRcpp(1, 1e6))  
#   user  system elapsed  
# 0.028   0.001   0.030
```

In diesem Fall generiert der Rcpp-Code in 0,03 Sekunden einen Vektor mit einer Länge von 1

Million anstelle von 1,31 Sekunden mit dem Ansatz der Basis-R.

Beschleunigung für das Schleifen von Bytes durch Kompilieren

Betrachten Sie nach dem Rcpp-Beispiel in diesem Dokumentationseintrag die folgende Tough-to-Vectorize-Funktion, die einen Vektor der Länge `len` wobei das erste Element angegeben wird (`first` Element) und jedes Element `xi` gleich `cos(x{i-1} + 1)` :

```
repeatedCosPlusOne <- function(first, len) {  
  x <- numeric(len)  
  x[1] <- first  
  for (i in 2:len) {  
    x[i] <- cos(x[i-1] + 1)  
  }  
  return(x)  
}
```

Ein einfacher Ansatz, um eine solche Funktion zu beschleunigen, ohne eine einzelne Codezeile umzuschreiben, ist das Bytekompilieren des Codes mithilfe des R-Compile-Pakets:

```
library(compiler)  
repeatedCosPlusOneCompiled <- cmpfun(repeatedCosPlusOne)
```

Die sich ergebende Funktion ist oft wesentlich schneller, liefert aber immer noch die gleichen Ergebnisse:

```
all.equal(repeatedCosPlusOne(1, 1e6), repeatedCosPlusOneCompiled(1, 1e6))  
# [1] TRUE  
system.time(repeatedCosPlusOne(1, 1e6))  
#   user  system elapsed  
# 1.175   0.014   1.201  
system.time(repeatedCosPlusOneCompiled(1, 1e6))  
#   user  system elapsed  
# 0.339   0.002   0.341
```

In diesem Fall beschleunigte das Byte-Kompilieren die Vektorisierung der Vektorisierung mit einem Vektor der Länge 1 Million von 1,20 Sekunden auf 0,34 Sekunden.

Anmerkung

Das Wesen von `repeatedCosPlusOne` als kumulative Anwendung einer einzelnen Funktion kann mit `Reduce` transparenter ausgedrückt werden:

```
iterFunc <- function(init, n, func) {  
  funcs <- replicate(n, func)  
  Reduce(function(., f) f(.), funcs, init = init, accumulate = TRUE)  
}  
repeatedCosPlusOne_vec <- function(first, len) {  
  iterFunc(first, len - 1, function(.) cos(. + 1))  
}
```

`repeatedCosPlusOne_vec` kann als "Vektorisierung" von `repeatedCosPlusOne` . Es kann jedoch erwartet

werden, dass es um den Faktor 2 *langsamer ist* :

```
library(microbenchmark)
microbenchmark(
  repeatedCosPlusOne(1, 1e4),
  repeatedCosPlusOne_vec(1, 1e4)
)
#> Unit: milliseconds
#>              expr      min       lq      mean    median       uq      max
neval cld
#>   repeatedCosPlusOne(1, 10000)  8.349261  9.216724 10.22715 10.23095 11.10817 14.33763
100   a
#> repeatedCosPlusOne_vec(1, 10000) 14.406291 16.236153 17.55571 17.22295 18.59085 24.37059
100   b
```

Beschleunigen Sie Code, der stark zu vektorisieren ist online lesen:

<https://riptutorial.com/de/r/topic/1203/beschleunigen-sie-code--der-stark-zu-vektorisieren-ist>

Kapitel 15: Bewährte Methoden für die Vektorisierung von R-Codes

Examples

Durch Zeilenoperationen

Der Schlüssel bei der Vektorisierung von R-Code besteht darin, "Zeilenoperationen" oder das Methoden-Dispatching von R-Funktionen zu reduzieren oder zu eliminieren.

Das heißt, wenn man sich einem Problem nähert, das auf den ersten Blick "durch Zeilenoperationen" erforderlich ist, wie zum Beispiel die Berechnung der Mittelwerte jeder Zeile, muss man sich fragen:

- Was sind die Klassen der Datensätze, mit denen ich zu tun habe?
- Gibt es einen vorhandenen kompilierten Code, der dies ohne wiederholte Bewertung von R-Funktionen erreichen kann?
- Wenn nicht, kann ich diese Operationen nach Spalten statt nach Zeilen ausführen?
- Lohnt es sich schließlich, viel Zeit darauf zu verwenden, komplizierten vektorisierten Code zu entwickeln, anstatt nur eine einfache `apply` auszuführen? Mit anderen Worten: Sind die Daten groß genug, dass R sie nicht mit einer einfachen Schleife effizient verarbeiten kann?

Abgesehen von der Speicherzuordnung und dem wachsenden Objekt in Schleifen werden wir uns in diesem Beispiel darauf konzentrieren, wie Sie das `apply` Schleifen, das Methoden-Dispatching oder die Neubewertung von R-Funktionen in Schleifen möglicherweise vermeiden.

Eine einfache / einfache Methode zum Berechnen des Mittelwerts pro Zeile wäre:

```
apply(mtcars, 1, mean)
```

	Mazda RX4	Mazda RX4 Wag	Datsun 710	Hornet 4 Drive	Hornet
Sportabout	Valiant	Duster 360			
29.90727	29.98136	23.59818	38.73955		
53.66455	35.04909	59.72000			
Merc 240D	Merc 230	Merc 280	Merc 280C	Merc	
450SE	Merc 450SL	Merc 450SLC			
24.63455	27.23364	31.86000	31.78727		
46.43091	46.50000	46.35000			
Cadillac Fleetwood	Lincoln Continental	Chrysler Imperial	Fiat 128	Honda	
Civic	Toyota Corolla	Toyota Corona			
66.23273	66.05855	65.97227	19.44091		
17.74227	18.81409	24.88864			
Dodge Challenger	AMC Javelin	Camaro Z28	Pontiac Firebird	Fiat	
X1-9	Porsche 914-2	Lotus Europa			
47.24091	46.00773	58.75273	57.37955		
18.92864	24.77909	24.88027			
Ford Pantera L	Ferrari Dino	Maserati Bora	Volvo 142E		
60.97182	34.50818	63.15545	26.26273		

Aber können wir es besser machen? Mal sehen was hier passiert ist:

1. Zuerst haben wir ein `data.frame` in eine `matrix` konvertiert. (Beachten Sie, dass dies innerhalb der `apply` Funktion geschieht.) Dies ist sowohl ineffizient als auch gefährlich. Eine `matrix` kann nicht mehrere Spaltentypen gleichzeitig enthalten. Daher wird eine solche Konvertierung wahrscheinlich zu Informationsverlust und manchmal zu irreführenden Ergebnissen führen (vergleiche `apply(iris, 2, class)` mit `str(iris)` oder mit `sapply(iris, class)`).
2. Zweitens haben wir eine Operation wiederholt, einmal für jede Zeile. Das heißt, wir mussten einige R-Funktionen `nrow(mtcars)` mal auswerten. In diesem speziellen Fall ist der `mean` keine rechenintensive Funktion, daher könnte R wahrscheinlich auch für große Datenmengen leicht damit umgehen. Was passiert, wenn wir die Standardabweichung pro Zeile berechnen müssen (was eine teure Quadratwurzeloperation erfordert)? Was uns zum nächsten Punkt bringt:
3. Wir haben die R-Funktion viele Male ausgewertet, aber vielleicht gibt es bereits eine kompilierte Version dieser Operation?

In der Tat könnten wir einfach tun:

```
rowMeans(mtcars)
```

	Mazda RX4	Mazda RX4 Wag	Datsun 710	Hornet 4 Drive	Hornet
Sportabout	29.90727	29.98136	23.59818	38.73955	
53.66455	35.04909	59.72000			
	Merc 240D	Merc 230	Merc 280	Merc 280C	Merc
450SE	Merc 450SL	Merc 450SLC			
24.63455		27.23364	31.86000	31.78727	
46.43091	46.50000	46.35000			
Cadillac Fleetwood	Lincoln Continental	Chrysler Imperial		Fiat 128	Honda
Civic	Toyota Corolla	Toyota Corona			
66.23273		66.05855	65.97227	19.44091	
17.74227	18.81409	24.88864			
Dodge Challenger	AMC Javelin	Camaro Z28	Pontiac Firebird		Fiat
X1-9	Porsche 914-2	Lotus Europa			
47.24091		46.00773	58.75273	57.37955	
18.92864	24.77909	24.88027			
Ford Pantera L	Ferrari Dino	Maserati Bora	Volvo 142E		
60.97182		34.50818	63.15545	26.26273	

Dies beinhaltet keine zeilenweisen Operationen und daher keine wiederholte Bewertung von R-Funktionen. Ein `data.frame` in eine `matrix` konvertiert. Obwohl `rowMeans` über einen Fehlerbehandlungsmechanismus verfügt, der nicht mit einem Datensatz ausgeführt werden kann, für den er nicht `rowMeans` ist, hat dies dennoch Kosten für die Effizienz.

```
rowMeans(iris)
Error in rowMeans(iris) : 'x' must be numeric
```

Aber können wir es trotzdem besser machen? Wir könnten anstelle einer `mtcars` mit Fehlerbehandlung versuchen, eine andere Methode zu verwenden, die es uns ermöglicht, `mtcars` als Vektor zu verwenden (da `data.frame` im Wesentlichen eine `list` und eine `list` ein `vector`).

```
Reduce('+', mtcars)/ncol(mtcars)
[1] 29.90727 29.98136 23.59818 38.73955 53.66455 35.04909 59.72000 24.63455 27.23364 31.86000
31.78727 46.43091 46.50000 46.35000 66.23273 66.05855
```



```
[17] 65.97227 19.44091 17.74227 18.81409 24.88864 47.24091 46.00773 58.75273 57.37955 18.92864
24.77909 24.88027 60.97182 34.50818 63.15545 26.26273
```

Nun haben wir wegen möglicher Geschwindigkeitssteigerung Spaltennamen und Fehlerbehandlung (einschließlich `NA` Behandlung) verloren.

Ein anderes Beispiel wäre die Berechnung des Mittelwerts nach Gruppe, wobei wir die Basis R verwenden könnten

```
aggregate(. ~ cyl, mtcars, mean)
cyl      mpg      disp      hp      drat      wt      qsec      vs      am      gear
carb
1    4 26.66364 105.1364  82.63636 4.070909 2.285727 19.13727 0.9090909 0.7272727 4.090909
1.545455
2    6 19.74286 183.3143 122.28571 3.585714 3.117143 17.97714 0.5714286 0.4285714 3.857143
3.428571
3    8 15.10000 353.1000 209.21429 3.229286 3.999214 16.77214 0.0000000 0.1428571 3.285714
3.500000
```

Grundsätzlich evaluieren wir eine R-Funktion in einer Schleife, aber die Schleife ist jetzt in einer internen C-Funktion verborgen (es spielt keine Rolle, ob es sich um eine C- oder eine R-Schleife handelt).

Könnten wir es vermeiden? Nun, es gibt eine kompilierte Funktion in R, die `rowsum` heißt, daher könnten wir folgendes tun:

```
rowsum(mtcars[-2], mtcars$cyl)/table(mtcars$cyl)
mpg      disp      hp      drat      wt      qsec      vs      am      gear      carb
4 26.66364 105.1364  82.63636 4.070909 2.285727 19.13727 0.9090909 0.7272727 4.090909 1.545455
6 19.74286 183.3143 122.28571 3.585714 3.117143 17.97714 0.5714286 0.4285714 3.857143 3.428571
8 15.10000 353.1000 209.21429 3.229286 3.999214 16.77214 0.0000000 0.1428571 3.285714 3.500000
```

Allerdings mussten wir auch zuerst in eine Matrix konvertieren.

Zu diesem Punkt stellen wir uns vielleicht die Frage, ob unsere derzeitige Datenstruktur die geeignetste ist. Ist ein `data.frame` die beste `data.frame`? Oder sollte man einfach auf eine `matrix` umsteigen, um die Effizienz zu steigern?

Zeilenoperationen werden immer teurer (sogar in Matrizen), da wir jedes Mal teure Funktionen auswerten. Betrachten wir eine Abweichungsberechnung anhand eines Zeilenbeispiels.

Nehmen wir an, wir haben eine Matrix `m`:

```
set.seed(100)
m <- matrix(sample(1e2), 10)
m
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    8   33   39   86   71  100   81   68   89   84
[2,]   12   16   57   80   32   82   69   11   41   92
[3,]   62   91   53   13   42   31   60   70   98   79
[4,]   66   94   29   67   45   59   20   96   64    1
```

[5,]	36	63	76	6	10	48	85	75	99	2
[6,]	18	4	27	19	44	56	37	95	26	40
[7,]	3	24	21	25	52	51	83	28	49	17
[8,]	46	5	22	43	47	74	35	97	77	65
[9,]	55	54	78	34	50	90	30	61	14	58
[10,]	88	73	38	15	9	72	7	93	23	87

Man könnte einfach tun:

```
apply(m, 1, var)
[1] 871.6556 957.5111 699.2111 941.4333 1237.3333 641.8222 539.7889 759.4333 500.4889
1255.6111
```

Auf der anderen Seite könnte man diese Operation auch vollständig vektorisieren, indem man der Varianzformel folgt

```
RowVar <- function(x) {
  rowSums((x - rowMeans(x))^2) / (dim(x)[2] - 1)
}
RowVar(m)
[1] 871.6556 957.5111 699.2111 941.4333 1237.3333 641.8222 539.7889 759.4333 500.4889
1255.6111
```

Bewährte Methoden für die Vektorisierung von R-Codes online lesen:

<https://riptutorial.com/de/r/topic/3327/bewahrte-methoden-fur-die-vektorisierung-von-r-codes>

Kapitel 16: Bibliographie in RMD

Parameter

Parameter im YAML-Header	Detail
<code>toc</code>	Inhaltsverzeichnis
<code>number_sections</code>	Nummerierung der Abschnitte automatisch
<code>bibliography</code>	Pfad zur Bibliographie-Datei
<code>csl</code>	Pfad zur Style-Datei

Bemerkungen

- Der Zweck dieser Dokumentation besteht darin, eine wissenschaftliche Bibliographie in eine RMD-Datei zu integrieren.
- Um die oben angegebene Dokumentation verwenden zu können, müssen Sie `rmarkdown` in R über `install.packages("rmarkdown")` installieren.
- Manchmal entfernt Rmarkdown die Hyperlinks der Zitate. Die Lösung hierfür ist das Hinzufügen des folgenden Codes zu Ihrem YAML-Header: `link-citations: true`
- Die Bibliographie kann eines der folgenden Formate haben:

Format	Dateierweiterung
MODIFIKATIONEN	.Modifikationen
BibLaTeX	.LätZchen
BibTeX	.bibtex
RIS	.ris
EndNote	.enl
EndNote XML	.xml
ISI	.wos
MEDLINE	.medline
Copac	.copac

Format	Dateierweiterung
JSON citeproc	.json

Examples

Angabe einer Bibliographie und Zitierautoren

Der wichtigste Teil Ihrer RMD-Datei ist der YAML-Header. Zum Verfassen einer wissenschaftlichen Arbeit empfehle ich die Verwendung von PDF-Ausgaben, nummerierten Abschnitten und eines Inhaltsverzeichnisses (toc).

```
---
title: "Writing an academic paper in R"
author: "Author"
date: "Date"
output:
  pdf_document:
    number_sections: yes
toc: yes
bibliography: bibliography.bib
---
```

In diesem Beispiel sieht unsere Datei `bibliography.bib` aus:

```
@ARTICLE{Meyer2000,
  AUTHOR="Bernd Meyer",
  TITLE="A constraint-based framework for diagrammatic reasoning",
  JOURNAL="Applied Artificial Intelligence",
  VOLUME= "14",
  ISSUE = "4",
  PAGES= "327--344",
  YEAR=2000
}
```

Um einen in Ihrer `.bib`-Datei genannten Autor zu zitieren, schreiben Sie `@` und den bibkey, z. B. `Meyer2000`.

```
# Introduction

`@Meyer2000` results in @Meyer2000.

`@Meyer2000 [p. 328]` results in @Meyer2000 [p. 328]

`[@Meyer2000]` results in [@Meyer2000]

`[-@Meyer2000]` results in [-@Meyer2000]

# Summary

# References
```

Das Rendern der RMD-Datei über RStudio (Strg + Umschalt + K) oder über die Konsole

`rmarkdown::render("<path-to-your-RMD-file">)` führt zu folgender Ausgabe:

Writing an academic paper in

Author

Date

Contents

1 Introduction

2 Summary

References

1 Introduction

@Meyer2000 results in Meyer (2000).

@Meyer2000 [p. 328] results in Meyer (2000, 328)

[@Meyer2000] results in (Meyer 2000)

[-@Meyer2000] results in (2000)

2 Summary

References

Meyer, Bernd. 2000. “A Constraint-Based Framework for Diagrammatic Reasoning.” *Artificial Intelligence* 14 (4): 327–44.

Chicago für Zitate und Verweise. Wenn Sie einen anderen Stil verwenden möchten, müssen Sie im CSL-Metadatenfeld eine CSL 1.0-Stildatei angeben. Im Folgenden wird ein häufig verwendeter Zitierstil, der elsevier-Stil, vorgestellt (Download unter <https://github.com/citation-style-language/styles>). Die Style-Datei muss im selben Verzeichnis wie die RMD-Datei gespeichert werden ODER der absolute Pfad zur Datei muss übermittelt werden.

Um einen anderen als den Standardstil zu verwenden, wird folgender Code verwendet:

```
---
title: "Writing an academic paper in R"
author: "Author"
date: "Date"
output:
  pdf_document:
    number_sections: yes
toc: yes
bibliography: bibliography.bib
csl: elsevier-harvard.csl
---

# Introduction

`@Meyer2000` results in @Meyer2000.

`@Meyer2000 [p. 328]` results in @Meyer2000 [p. 328]

`[@Meyer2000]` results in [@Meyer2000]

`[-@Meyer2000]` results in [-@Meyer2000]

# Summary

# Reference
```

Writing an academic paper in R

Author

Date

Contents

1	Introduction	1
2	Summary	1
	Reference	1

1 Introduction

@Meyer2000 results in Meyer (2000).

@Meyer2000 [p. 328] results in Meyer (2000, p. 328)

[@Meyer2000] results in (Meyer, 2000)

[-@Meyer2000] results in (2000)

2 Summary

Reference

Meyer, B., 2000. A constraint-based framework for diagrammatic reasoning. Applied Artificial Intelligence 14, 327–344.

Beachten Sie die Unterschiede zur Ausgabe des Beispiels "Bibliographie angeben und Autoren zitieren".

Bibliographie in RMD online lesen: <https://riptutorial.com/de/r/topic/7606/bibliographie-in-rmd>

Kapitel 17: Box-Plot

Syntax

- `Boxplot(x, ...)` # generische Funktion
- `Boxplot(Formel, Daten = NULL, ..., Teilmenge, na.Aktion = NULL)` ## S3-Methode für Klasse 'Formel'
- `Boxplot(x, ..., Bereich = 1,5, Breite = NULL, Varwidth = FALSE, Notch = FALSE, Gliederung = TRUE, Namen, Plot = TRUE, Rand = par("fg"), col = NULL, log = " ", pars = list(boxwex = 0,8, Staplewex = 0,5, outwex = 0,5), horizontal = FALSE, add = FALSE, at = NULL)` ## Standardmäßige S3-Methode

Parameter

Parameter	Details (Quelldokumentation R)
Formel	eine Formel wie <code>y ~ grp</code> , wobei <code>y</code> ein numerischer Vektor von Datenwerten ist, die entsprechend der Gruppierungsvariablen <code>grp</code> (normalerweise ein Faktor) in Gruppen aufgeteilt werden sollen.
Daten	ein Datenrahmen (oder eine Liste), aus dem die Variablen in der Formel entnommen werden sollen.
Teilmenge	ein optionaler Vektor, der eine Teilmenge von Beobachtungen angibt, die zum Plotten verwendet werden sollen.
na.aktion	eine Funktion, die angibt, was passieren soll, wenn die Daten NAs enthalten. Standardmäßig werden fehlende Werte in der Antwort oder in der Gruppe ignoriert.
Boxwex	ein Skalierungsfaktor, der auf alle Boxen angewendet werden soll. Wenn nur wenige Gruppen vorhanden sind, kann das Erscheinungsbild der Grafik verbessert werden, indem die Kästchen schmaler werden.
Handlung	Wenn <code>TRUE</code> (Standardeinstellung), wird ein Boxplot erstellt. Wenn nicht, werden die Zusammenfassungen zurückgegeben, auf denen die Boxplots basieren.
col	Wenn <code>col</code> nicht null ist, wird angenommen, dass es Farben enthält, die zum Färben der Körper der Boxdiagramme verwendet werden. Standardmäßig haben sie die Hintergrundfarbe.

Examples

Erstellen Sie ein Box-and-Whisker-Diagramm mit `boxplot ()` {graphics}.

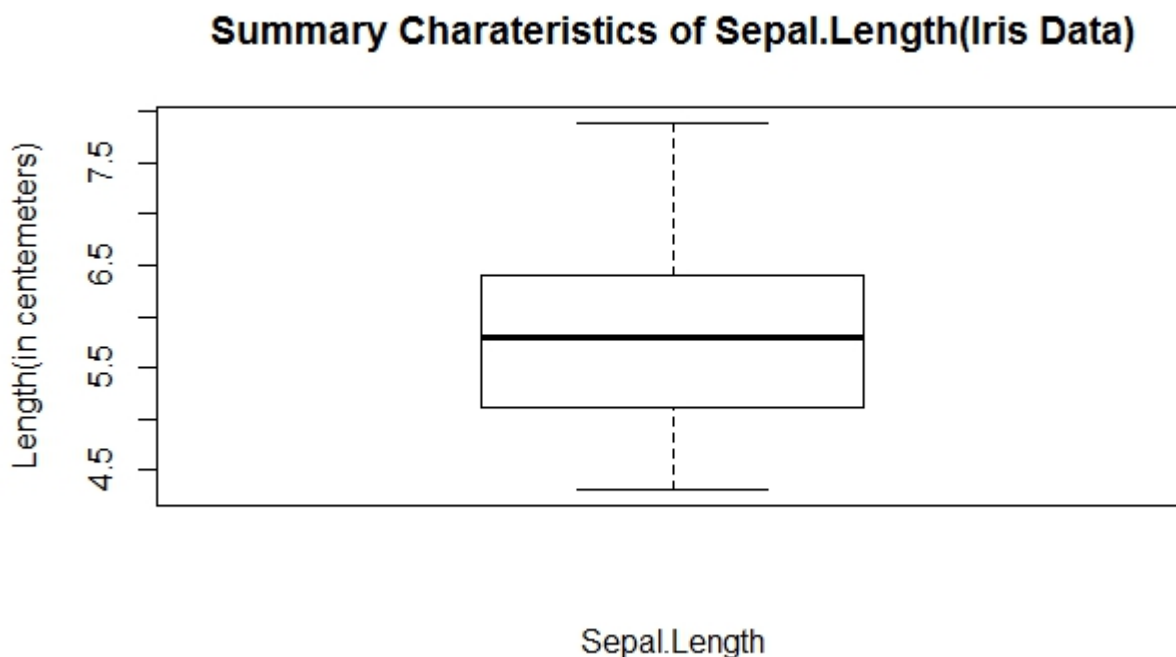
In diesem Beispiel werden die Standardfunktion `boxplot ()` und der `boxplot ()` der `iris` verwendet.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5         1.4         0.2   setosa
2          4.9         3.0         1.4         0.2   setosa
3          4.7         3.2         1.3         0.2   setosa
4          4.6         3.1         1.5         0.2   setosa
5          5.0         3.6         1.4         0.2   setosa
6          5.4         3.9         1.7         0.4   setosa
```

Einfacher Boxplot (Sepal.Length)

Erstellen Sie ein Box-and-Whisker-Diagramm einer numerischen Variablen

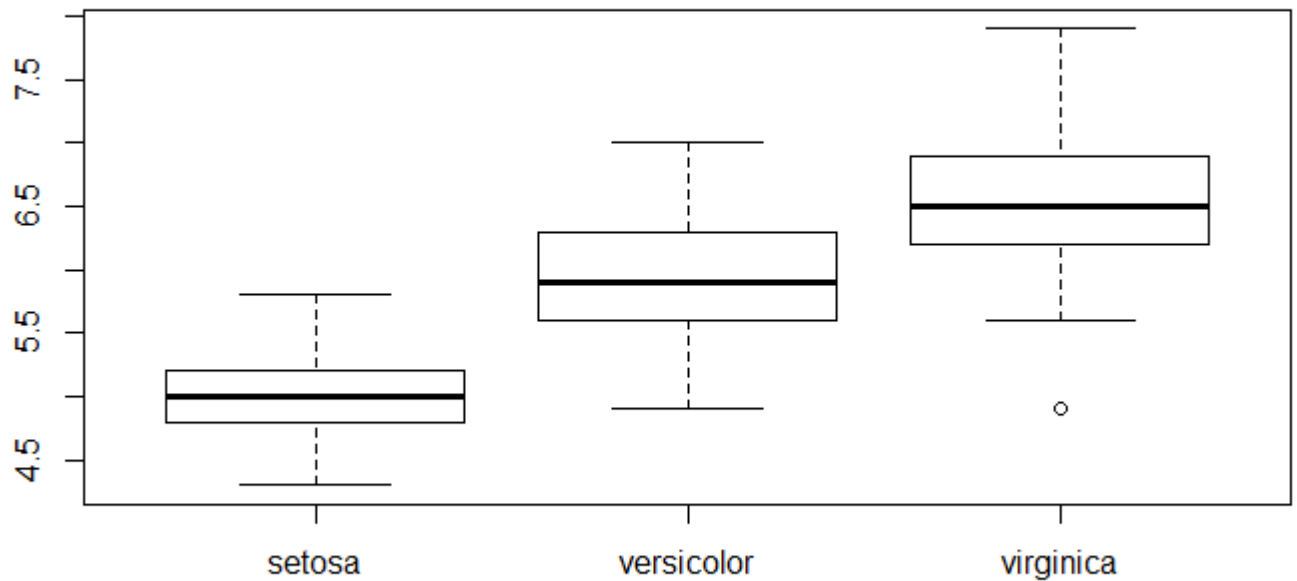
```
boxplot(iris[,1],xlab="Sepal.Length",ylab="Length(in centimeters)",
        main="Summary Charateristics of Sepal.Length(Iris Data)")
```



Boxplot der Kelchlänge nach Arten gruppiert

Erstellen Sie ein Boxplot einer numerischen Variablen, gruppiert nach einer kategorialen Variablen

```
boxplot(Sepal.Length~Species,data = iris)
```

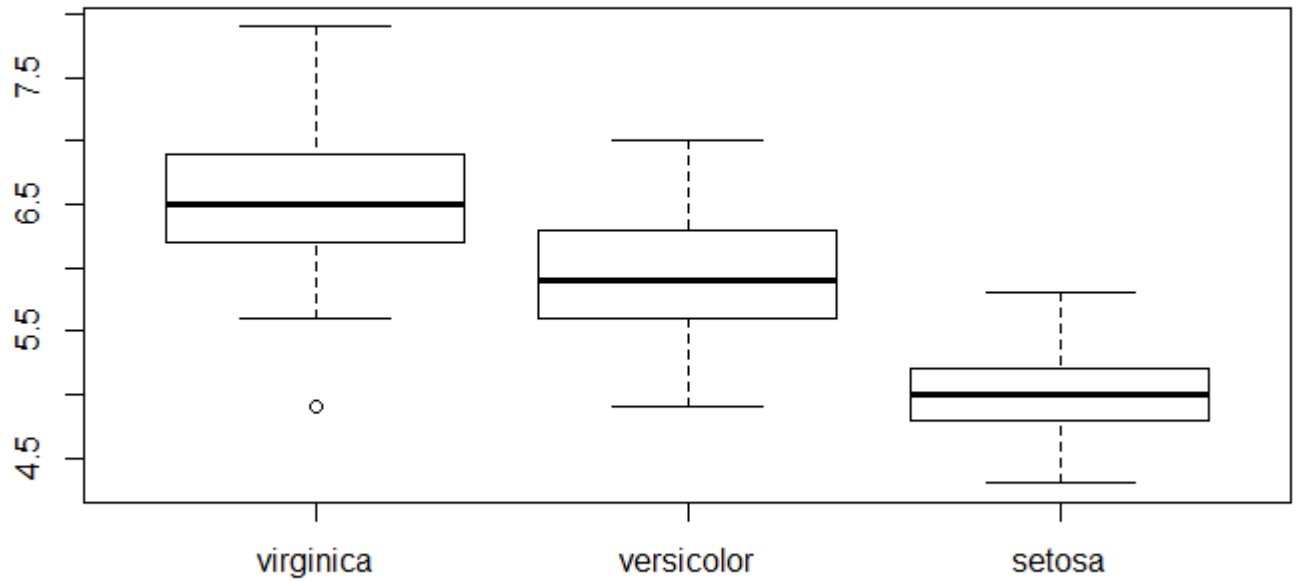


Ordnung bringen

Um die Reihenfolge der Box im Diagramm zu ändern, müssen Sie die Reihenfolge der Ebenen der kategorialen Variablen ändern.

Zum Beispiel, wenn wir die Reihenfolge `virginica - versicolor - setosa`

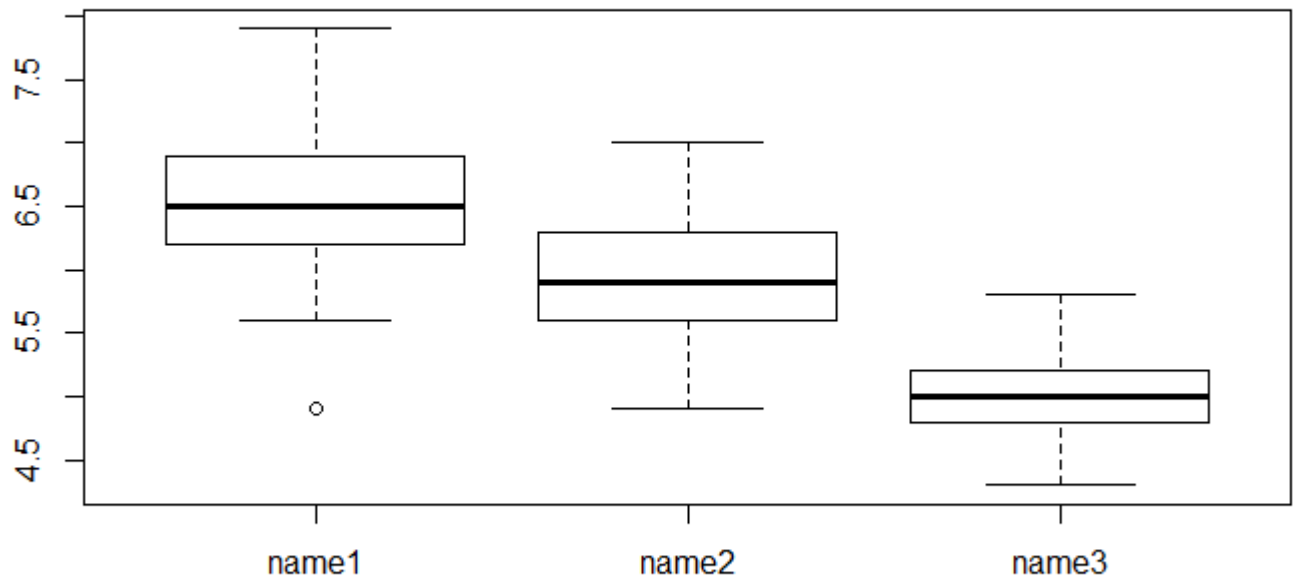
```
newSpeciesOrder <- factor(iris$Species, levels=c("virginica","versicolor","setosa"))  
boxplot(Sepal.Length~newSpeciesOrder,data = iris)
```



Gruppennamen ändern

Wenn Sie Ihren Gruppen einen besseren Namen geben möchten, können Sie den `Names` Parameter verwenden. Es braucht einen Vektor der Größe der Ebenen der kategorialen Variablen

```
boxplot(Sepal.Length~newSpeciesOrder,data = iris,names= c("name1","name2","name3"))
```

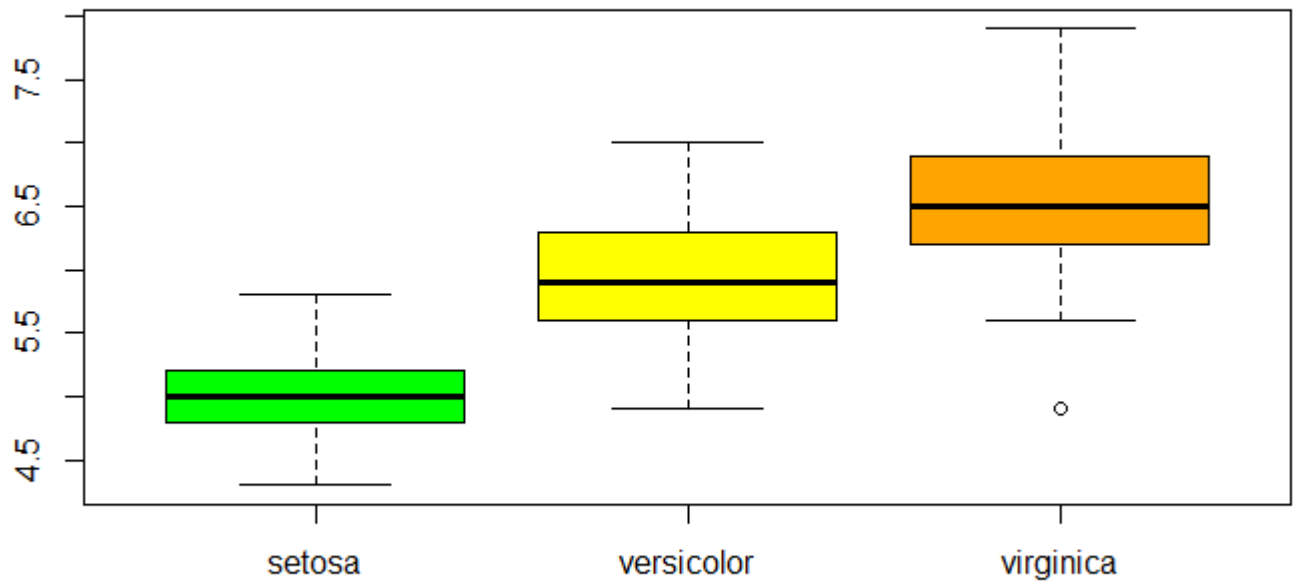


Kleine Verbesserungen

Farbe

`col` : Fügt einen Vektor der Größe der Ebenen der kategorialen Variablen hinzu

```
boxplot(Sepal.Length~Species,data = iris,col=c("green","yellow","orange"))
```

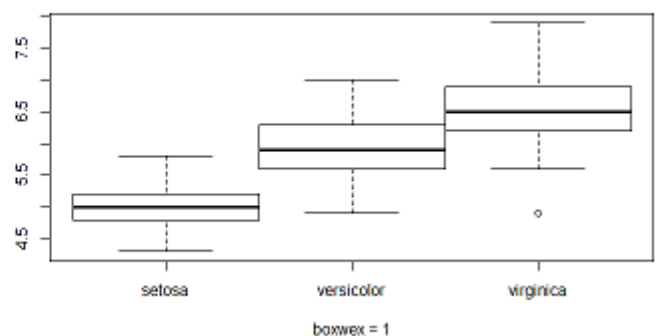
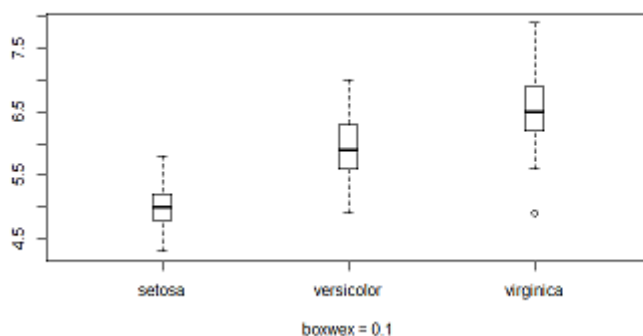


Nähe der Box

`boxwex` : `boxwex` den Abstand zwischen den Boxen fest.

Linke `boxplot(Sepal.Length~Species,data = iris,boxwex = 0.1)`

Rechte `boxplot(Sepal.Length~Species,data = iris,boxwex = 1)`



Sehen Sie sich die Zusammenfassungen an, auf denen die Boxplots basieren. `plot=FALSE`

Um eine Zusammenfassung zu sehen , müssen Sie die Paramater setzen `plot` auf `FALSE` .
Es werden verschiedene Ergebnisse angegeben

```

> boxplot(Sepal.Length~newSpeciesOrder,data = iris,plot=FALSE)
$stats #summary of the numerical variable for the 3 groups
      [,1] [,2] [,3]
[1,]  5.6  4.9  4.3 # extreme value
[2,]  6.2  5.6  4.8 # first quartile limit
[3,]  6.5  5.9  5.0 # median limit
[4,]  6.9  6.3  5.2 # third quartile limit
[5,]  7.9  7.0  5.8 # extreme value

$n #number of observations in each groups
[1] 50 50 50

$confs #extreme value of the notches
      [,1]      [,2]      [,3]
[1,] 6.343588 5.743588 4.910622
[2,] 6.656412 6.056412 5.089378

$out #extreme value
[1] 4.9

$group #group in which are the extreme value
[1] 1

$names #groups names
[1] "virginica" "versicolor" "setosa"

```

Zusätzliche Boxplot-Style-Parameter.

Box

- boxlty - Boxzeilentyp
- boxlwd - Zeilenbreite der Box
- Boxcol - Boxlinienfarbe
- BoxFill - Füllfarbe der Box

Median

- medlty - Median-Linientyp ("leer" für keine Zeile)
- medlwd - mittlere linienbreite
- medcol - mittlere Linienfarbe
- medpch - Mittelwert (NA für kein Symbol)
- medcex - mittlere Punktgröße
- medbg - mittlere Hintergrundfarbe des Punktes

Schnurrhaar

- Whisklty - Whisker-Linientyp
- whisklwd - Breite der Whisker-Linie
- whiskcol - Farbe der Whisker-Linie

Klammer

- Staplelty - Heftlinienart
- staplelwd - Breite der Heftlinie
- Staplecol - Farbe der Grundlinie

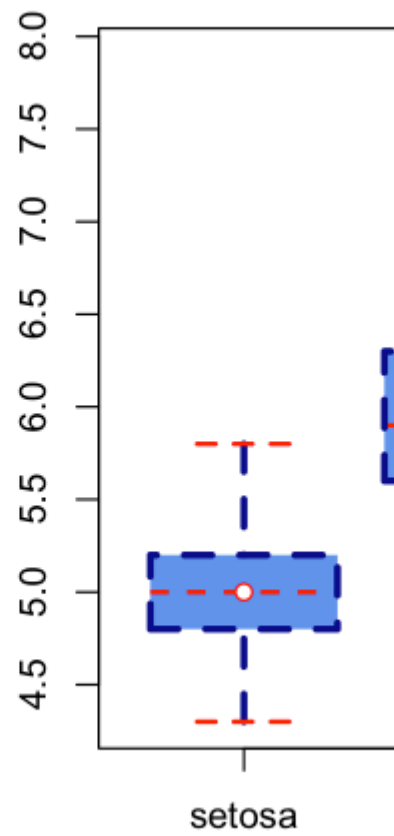
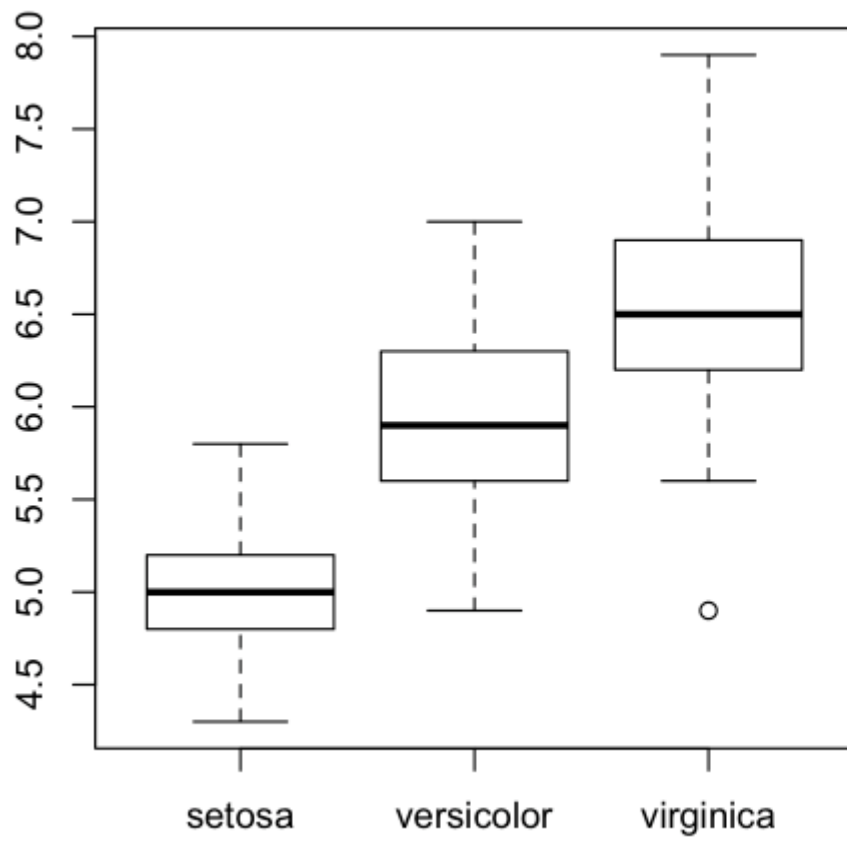
Ausreißer

- Outlty - Outlier-Zeilentyp ("leer" für keine Zeile)
- outlwd - Ausreißerlinienbreite
- Outcol - Ausreißer Linienfarbe
- Outpch - Ausreißerpunkttyp (NA für kein Symbol)
- outcex - Ausreißerpunktgröße
- Outbg - Ausreißerpunkt-Hintergrundfarbe

Beispiel

Standard- und stark modifizierte Diagramme nebeneinander

```
par(mfrow=c(1,2))
# Default
boxplot(Sepal.Length ~ Species, data=iris)
# Modified
boxplot(Sepal.Length ~ Species, data=iris,
        boxlty=2, boxlwd=3, boxfill="cornflowerblue", boxcol="darkblue",
        medlty=2, medlwd=2, medcol="red", medpch=21, medcex=1, medbg="white",
        whisklty=2, whisklwd=3, whiskcol="darkblue",
        staplelty=2, staplelwd=2, staplecol="red",
        outlty=3, outlwd=3, outcol="grey", outpch=NA
        )
```

Box-Plot online lesen: <https://riptutorial.com/de/r/topic/1005/box-plot>

Kapitel 18: Caret

Einführung

`caret` ist ein R-Paket, das die Datenverarbeitung unterstützt, die für maschinelles Lernen erforderlich ist. Es steht für Einstufungs- und Regressionstraining. Beim Erstellen von Modellen für ein reales Dataset müssen neben dem eigentlichen Lernalgorithmus einige andere Aufgaben ausgeführt werden, z. B. das Bereinigen der Daten, das Bearbeiten unvollständiger Beobachtungen, die Validierung unseres Modells mit einem Testset und der Vergleich verschiedener Modelle.

`caret` hilft in diesen Szenarien, unabhängig von den tatsächlich verwendeten Lernalgorithmen.

Examples

Vorverarbeitung

Die Vorverarbeitung in Caret erfolgt über die Funktion `preProcess()`. Bei einem Matrix- oder `preProcess()` `x` wendet `preProcess()` Transformationen auf die Trainingsdaten an, die dann auf `preProcess()` angewendet werden können.

Das Herzstück der Funktion `preProcess()` ist das `method`. Methodenoperationen werden in dieser Reihenfolge angewendet:

1. Nullvarianzfilter
2. Fast-Null-Varianzfilter
3. Box-Cox / Yeo-Johnson / exponentielle Transformation
4. Zentrierung
5. Skalieren
6. Angebot
7. Imputation
8. PCA
9. ICA
10. Räumliches Zeichen

Im Folgenden nehmen wir den `mtcars`-Datensatz und führen Zentrieren, Skalieren und eine Transformation für räumliche Zeichen durch.

```
auto_index <- createDataPartition(mtcars$mpg, p = .8,  
                                  list = FALSE,  
                                  times = 1)  
  
mt_train <- mtcars[auto_index,]  
mt_test  <- mtcars[-auto_index,]  
  
process_mtcars <- preProcess(mt_train, method = c("center", "scale", "spatialSign"))
```

```
mtcars_train_transf <- predict(process_mtcars, mt_train)
mtcars_test_tranf <- predict(process_mtcars, mt_test)
```

Caret online lesen: <https://riptutorial.com/de/r/topic/4271/caret>

Kapitel 19: Code-Profiling

Examples

Systemzeit

Die Systemzeit gibt die CPU-Zeit an, die zum Ausführen eines R-Ausdrucks erforderlich ist.
Beispiel:

```
system.time(print("hello world"))

# [1] "hello world"
#      user  system elapsed
#         0        0         0
```

Sie können größere Codeteile mithilfe von geschweiften Klammern hinzufügen:

```
system.time({
  library(numbers)
  Primes(1,10^5)
})
```

Oder verwenden Sie es zum Testen von Funktionen:

```
fibb <- function (n) {
  if (n < 3) {
    return(c(0,1)[n])
  } else {
    return(fibb(n - 2) + fibb(n -1))
  }
}

system.time(fibb(30))
```

proc.time ()

Im einfachsten `proc.time()` gibt `proc.time()` die gesamte verstrichene CPU-Zeit in Sekunden für den aktuellen Prozess an. Wenn Sie es in der Konsole ausführen, erhalten Sie folgende Ausgabe:

```
proc.time()

#      user      system    elapsed
#  284.507    120.397  515029.305
```

Dies ist besonders nützlich für das Benchmarking bestimmter Codezeilen. Zum Beispiel:

```
t1 <- proc.time()
fibb <- function (n) {
  if (n < 3) {
```

```

        return(c(0,1)[n])
    } else {
        return(fibb(n - 2) + fibb(n -1))
    }
}
print("Time one")
print(proc.time() - t1)

t2 <- proc.time()
fibb(30)

print("Time two")
print(proc.time() - t2)

```

Dies ergibt die folgende Ausgabe:

```

source('~/.active-rstudio-document')

# [1] "Time one"
#   user  system elapsed
#     0      0      0

# [1] "Time two"
#   user  system elapsed
# 1.534  0.012  1.572

```

`system.time()` ist ein Wrapper für `proc.time()`, der die abgelaufene Zeit für einen bestimmten Befehl / Ausdruck zurückgibt.

```

print(t1 <- system.time(replicate(1000,12^2)))
##   user  system elapsed
## 0.000   0.000   0.002

```

Beachten Sie, dass das zurückgegebene Objekt der Klasse `proc.time` etwas komplizierter ist, als es auf der Oberfläche erscheint:

```

str(t1)
## Class 'proc_time'  Named num [1:5] 0 0 0.002 0 0
##   ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...

```

Linienprofilierung

Ein Paket für das Line-Profiling ist [Lineprof](#), das von Hadley Wickham geschrieben und verwaltet wird. Hier eine `auto.arima` Demonstration, wie es mit `auto.arima` im Prognosepaket funktioniert:

```

library(lineprof)
library(forecast)

l <- lineprof(auto.arima(AirPassengers))
shine(l)

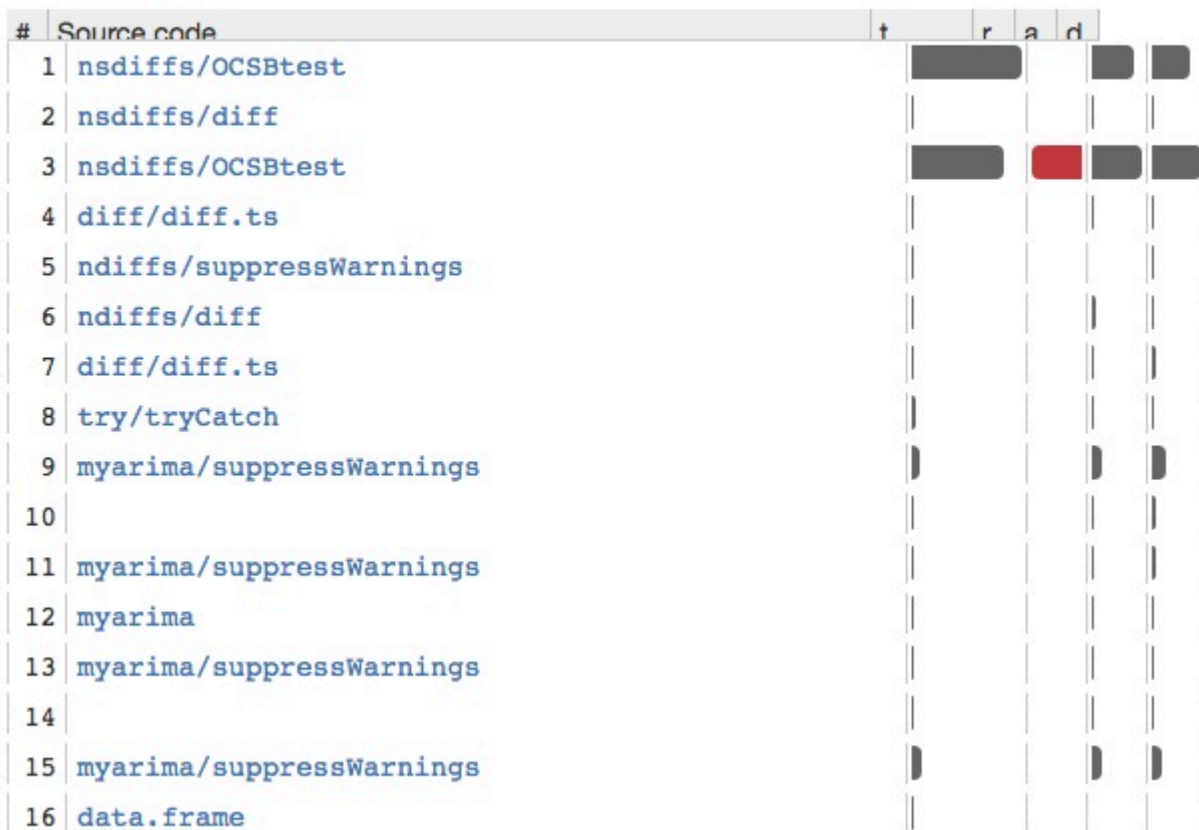
```

Dadurch erhalten Sie eine glänzende App, mit der Sie tiefer in jeden Funktionsaufruf einsteigen können. So können Sie mit Leichtigkeit sehen, was dazu führt, dass Ihr R-Code langsamer wird.

Unten sehen Sie einen Screenshot der glänzenden App:

Line profiling

Back



Mikrobankmark

Mikrobankmarkierung ist nützlich, um die Zeit für ansonsten schnelle Verfahren abzuschätzen. Stellen Sie sich beispielsweise vor, wie viel Zeit benötigt wird, um Hallo Welt zu drucken.

```
system.time(print("hello world"))

# [1] "hello world"
#      user  system elapsed
#       0      0      0
```

Dies liegt daran, dass `system.time` im Wesentlichen eine Wrapper-Funktion für `proc.time`, die in Sekunden misst. Da der Druck "Hallo Welt" weniger als eine Sekunde dauert, scheint die Zeit weniger als eine Sekunde zu sein, dies ist jedoch nicht der Fall. Um dies zu sehen, können wir das Paket `microbenchmark` verwenden:

```
library(microbenchmark)
microbenchmark(print("hello world"))

# Unit: microseconds
#      expr      min       lq      mean  median       uq      max neval
# print("hello world") 26.336 29.984 44.11637 44.6835 45.415 158.824   100
```

Nachdem wir `print("hello world")` 100 Mal ausgeführt haben, betrug die durchschnittliche Zeit tatsächlich 44 Mikrosekunden. (Beachten Sie, dass beim Ausführen dieses Codes "Hallo Welt" 100 Mal auf die Konsole gedruckt wird.)

Wir können dies mit einem gleichwertigen Verfahren, `cat("hello world\n")`, um zu sehen, ob es schneller ist als `print("hello world")`:

```
microbenchmark(cat("hello world\n"))

# Unit: microseconds
#      expr      min       lq     mean  median      uq      max  neval
# cat("hello world\n") 14.093 17.6975 23.73829 19.319 20.996 119.382   100
```

In diesem Fall ist `cat()` fast doppelt so schnell wie `print()`.

Alternativ können zwei Verfahren innerhalb desselben `microbenchmark` Aufrufs verglichen werden:

```
microbenchmark(print("hello world"), cat("hello world\n"))
# Unit: microseconds
# expr      min       lq     mean  median      uq      max  neval
# print("hello world") 29.122 31.654 39.64255 34.5275 38.852 192.779   100
# cat("hello world\n")  9.381 12.356 13.83820 12.9930 13.715  52.564   100
```

Benchmarking mit Mikrobankmark

Sie können das [microbenchmark](#) Paket verwenden, um ein "präzises Timing der Ausdrucksauswertung unter einer Millisekunde" durchzuführen.

In [diesem Beispiel vergleichen](#) wir die Geschwindigkeiten von sechs äquivalenten `data.table` Ausdrücken zum Aktualisieren von Elementen in einer Gruppe basierend auf einer bestimmten Bedingung.

Genauer:

Eine `data.table` mit 3 Spalten: `id`, `time` und `status`. Für jede ID möchte ich den Datensatz mit der maximalen Zeit suchen. Wenn für diesen Datensatz der Status `true` ist, möchte ich ihn auf `false` setzen, wenn die `Zeit > 7` ist

```
library(microbenchmark)
library(data.table)

set.seed(20160723)
dt <- data.table(id = c(rep(seq(1:10000), each = 10)),
                 time = c(rep(seq(1:10000), 10)),
                 status = c(sample(c(TRUE, FALSE), 10000*10, replace = TRUE)))
setkey(dt, id, time) ## create copies of the data so the 'updates-by-reference' don't affect
other expressions
dt1 <- copy(dt)
dt2 <- copy(dt)
dt3 <- copy(dt)
dt4 <- copy(dt)
dt5 <- copy(dt)
dt6 <- copy(dt)
```

```

microbenchmark(

  expression_1 = {
    dt1[ dt1[order(time), .I[.N], by = id]$V1, status := status * time < 7 ]
  },

  expression_2 = {
    dt2[,status := c(.SD[.N, status], .SD[.N, status * time > 7]), by = id]
  },

  expression_3 = {
    dt3[dt3[, .N, by = id][,cumsum(N)], status := status * time > 7]
  },

  expression_4 = {
    y <- dt4[, .SD[.N], by=id]
    dt4[y, status := status & time > 7]
  },

  expression_5 = {
    y <- dt5[, .SD[.N, .(time, status)], by = id][time > 7 & status]
    dt5[y, status := FALSE]
  },

  expression_6 = {
    dt6[ dt6[, .I == .I[which.max(time)], by = id]$V1 & time > 7, status := FALSE]
  },

  times = 10L ## specify the number of times each expression is evaluated
)

# Unit: milliseconds
#      expr      min       lq      mean     median        uq      max neval
# expression_1 11.646149 13.201670 16.808399 15.643384 18.78640 26.321346    10
# expression_2 8051.898126 8777.016935 9238.323459 8979.553856 9281.93377 12610.869058    10
# expression_3   3.208773   3.385841   4.207903   4.089515   4.70146   5.654702    10
# expression_4  15.758441  16.247833  20.677038  19.028982  21.04170  36.373153    10
# expression_5 7552.970295 8051.080753 8702.064620 8861.608629 9308.62842 9722.234921    10
# expression_6  18.403105  18.812785  22.427984  21.966764  24.66930  28.607064    10

```

Die Ausgabe zeigt, dass `expression_3` in diesem Test der schnellste ist.

Verweise

[data.table - Spalten hinzufügen und ändern](#)

[data.table - spezielle Gruppierungssymbole in data.table](#)

[Code-Profiling online lesen: https://riptutorial.com/de/r/topic/2149/code-profiling](https://riptutorial.com/de/r/topic/2149/code-profiling)

Kapitel 20: Daten reinigen

Einführung

Das Reinigen von Daten in R ist für jede Analyse von größter Bedeutung. Unabhängig von den Daten, die Sie haben, sei es durch Messungen, die Sie im Feld gemacht oder aus dem Web entnommen haben, ist es sehr wahrscheinlich, dass Sie sie umformen, transformieren oder filtern müssen, damit sie für Ihre Analyse geeignet sind. In dieser Dokumentation werden die folgenden Themen behandelt: - Entfernen von Beobachtungen mit fehlenden Daten - Faktorisieren von Daten - Entfernen unvollständiger Zeilen

Examples

Fehlende Daten aus einem Vektor entfernen

Zuerst können wir einen Vektor mit dem Namen Vector1 erstellen:

```
set.seed(123)
Vector1 <- rnorm(20)
```

Und fügen Sie fehlende Daten hinzu:

```
set.seed(123)
Vector1[sample(1:length(Vector1), 5)] <- NA
```

Jetzt können wir die `is.na`-Funktion verwenden, um den Vektor zu subsetieren

```
Vector1 <- Vector1[!is.na(Vector1)]
```

Der resultierende Vektor hat nun die NAs des ursprünglichen Vector1 entfernt

Unvollständige Zeilen entfernen

Es kann vorkommen, dass Sie einen Datenrahmen haben und alle Zeilen entfernen möchten, die möglicherweise einen NA-Wert enthalten. Dazu ist die Funktion `complete.cases` die beste Option.

Wir werden die ersten 6 Zeilen des *Luftqualitäts*- Datasets verwenden, um ein Beispiel zu machen, da es bereits NAs hat

```
x <- head(airquality)
```

Dies hat zwei Zeilen mit NAs in der Solar.R-Spalte. Um diese zu entfernen, führen wir die folgenden Schritte aus

```
x_no_NA <- x[complete.cases(x),]
```

Der resultierende Datenrahmen `x_no_NA` *enthält* nur vollständige Zeilen ohne NAs

Daten reinigen online lesen: <https://riptutorial.com/de/r/topic/8165/daten-reinigen>

Kapitel 21: Daten zwischen langen und breiten Formularen umformen

Einführung

In R werden Tabellendaten in [Datenrahmen](#) gespeichert. In diesem Thema werden die verschiedenen Möglichkeiten zum Transformieren einer einzelnen Tabelle beschrieben.

Bemerkungen

Hilfreiche Pakete

- [Umformen, Stapeln und Teilen](#) mit `data.table`
- [Umformen mit Tidy](#)
- `splitstackshape`

Examples

Die Umformfunktion

Die flexibelste Basis-R-Funktion zum Umformen von Daten ist `reshape` Umformen. Siehe `?reshape` für seine Syntax.

```
# create unbalanced longitudinal (panel) data set
set.seed(1234)
df <- data.frame(identifier=rep(1:5, each=3),
                  location=rep(c("up", "down", "left", "up", "center"), each=3),
                  period=rep(1:3, 5), counts=sample(35, 15, replace=TRUE),
                  values=runif(15, 5, 10))[-c(4,8,11),]
```

```
df
```

	identifier	location	period	counts	values
1	1	up	1	4	9.186478
2	1	up	2	22	6.431116
3	1	up	3	22	6.334104
5	2	down	2	31	6.161130
6	2	down	3	23	6.583062
7	3	left	1	1	6.513467
9	3	left	3	24	5.199980
10	4	up	1	18	6.093998
12	4	up	3	20	7.628488
13	5	center	1	10	9.573291
14	5	center	2	33	9.156725
15	5	center	3	11	5.228851

Beachten Sie, dass der `data.frame` unsymmetrisch ist, d. H. Einheit 2 eine Beobachtung in der ersten Periode fehlt, während in den Einheiten 3 und 4 Beobachtungen in der zweiten Periode

fehlen. Beachten Sie außerdem, dass es zwei Variablen gibt, die sich über die Zeiträume hinweg unterscheiden: Anzahl und Werte, und zwei Variablen, die nicht variieren: Bezeichner und Ort.

Lang bis breit

Um das data.frame in das Breitformat umzuwandeln,

```
# reshape wide on time variable
df.wide <- reshape(df, idvar="identifizier", timevar="period",
                  v.names=c("values", "counts"), direction="wide")
df.wide
  identifizier location values.1 counts.1 values.2 counts.2 values.3 counts.3
1           1        up 9.186478         4 6.431116         22 6.334104         22
5           2       down      NA         NA 6.161130         31 6.583062         23
7           3       left 6.513467         1      NA         NA 5.199980         24
10          4        up 6.093998        18      NA         NA 7.628488         20
13          5      center 9.573291        10 9.156725         33 5.228851         11
```

Beachten Sie, dass die fehlenden Zeiträume mit NAs ausgefüllt werden.

Bei der Weiterverformung gibt das Argument "v.names" die Spalten an, die sich im Laufe der Zeit ändern. Wenn die Standortvariable nicht erforderlich ist, kann sie vor dem Umformen mit dem Argument "drop" gelöscht werden. Wenn Sie die einzige nicht variierende / nicht-id-Spalte aus dem data.frame löschen, wird das Argument v.names nicht mehr benötigt.

```
reshape(df, idvar="identifizier", timevar="period", direction="wide",
        drop="location")
```

Weit bis lang

Um lange mit dem aktuellen df.wide umzugestalten, ist eine minimale Syntax

```
reshape(df.wide, direction="long")
```

Dies ist jedoch in der Regel schwieriger:

```
# remove "." separator in df.wide names for counts and values
names(df.wide)[grep("\\.", names(df.wide))] <-
  gsub("\\.", "", names(df.wide)[grep("\\.", names(df.wide))])
```

Jetzt erzeugt die einfache Syntax einen Fehler bezüglich undefinierter Spalten.

Mit Spaltennamen, die schwieriger für die reshape Funktion automatisch zu analysieren, ist es manchmal notwendig, das „Varying“ Argument hinzuzufügen, die erzählte reshape zu gruppieren bestimmte Variablen im Wide - Format für die Umwandlung in Langformat. Dieses Argument enthält eine Liste von Vektoren von Variablennamen oder Indizes.

```
reshape(df.wide, idvar="identifizier",
        varying=list(c(3,5,7), c(4,6,8)), direction="long")
```

Bei der Umformung von long kann das Argument "v.names" angegeben werden, um die resultierenden variablen Variablen umzubenennen.

Manchmal kann die Angabe "Variieren" vermieden werden, indem das Argument "sep" verwendet wird, das der `reshape` mitteilt, welcher Teil des Variablennamens das Wertargument und das Zeitargument angibt.

Daten umformen

Oft sind Daten in Tabellen enthalten. Im Allgemeinen kann man diese Tabellendaten in Breit- und Langformate unterteilen. In einem breiten Format hat jede Variable eine eigene Spalte.

Person	Höhe (cm)	Alter [Jahr]
Alison	178	20
Bob	174	45
Carl	182	31

Manchmal ist es jedoch günstiger, ein langes Format zu verwenden, in dem alle Variablen in einer Spalte und die Werte in einer zweiten Spalte enthalten sind.

Person	Variable	Wert
Alison	Höhe (cm)	178
Bob	Höhe (cm)	174
Carl	Höhe (cm)	182
Alison	Alter [Jahr]	20
Bob	Alter [Jahr]	45
Carl	Alter [Jahr]	31

Base R sowie Pakete von Drittanbietern können verwendet werden, um diesen Prozess zu vereinfachen. Für jede Option wird das `mtcars` Dataset verwendet. Dieses Dataset hat standardmäßig ein langes Format. Damit die Pakete funktionieren, werden wir die Zeilennamen als erste Spalte einfügen.

```
mtcars # shows the dataset
data <- data.frame(observation=row.names(mtcars),mtcars)
```

Basis R

In Base R gibt es zwei Funktionen, die zum Konvertieren zwischen Breit- und Langformat verwendet werden können: `stack()` und `unstack()` .

```
long <- stack(data)
long # this shows the long format
wide <- unstack(long)
wide # this shows the wide format
```

Diese Funktionen können jedoch für komplexere Anwendungsfälle sehr komplex werden. Glücklicherweise gibt es andere Möglichkeiten, Pakete von Drittanbietern zu verwenden.

Das Tidyr-Paket

Dieses Paket verwendet `gather()` zum Konvertieren von Wide in Long und `spread()` , um von Long nach Wide zu konvertieren.

```
library(tidyr)
long <- gather(data, variable, value, 2:12) # where variable is the name of the
# variable column, value indicates the name of the value column and 2:12 refers to
# the columns to be converted.
long # shows the long result
wide <- spread(long, variable, value)
wide # shows the wide result (~data)
```

Das Paket data.table

Das Paket `data.table` erweitert die Funktionen von `reshape2` und verwendet die Funktion `melt()` , um von Wide zu Long zu `dcast()` und `dcast()` , um von Long nach Wide zu gelangen.

```
library(data.table)
long <- melt(data, 'observation', 2:12, 'variable', 'value')
long # shows the long result
wide <- dcast(long, observation ~ variable)
wide # shows the wide result (~data)
```

Daten zwischen langen und breiten Formularen umformen online lesen:

<https://riptutorial.com/de/r/topic/2904/daten-zwischen-langen-und-breiten-formularen-umformen>

Kapitel 22: Datenerfassung

Einführung

Holen Sie sich Daten direkt in eine R-Sitzung. Eine der schönen Eigenschaften von R ist die einfache Datenerfassung. Es gibt mehrere Möglichkeiten, Daten mit R-Paketen zu verbreiten.

Examples

Eingebaute Datensätze

R verfügt über eine umfangreiche Sammlung eingebauter Datensätze. In der Regel werden sie zu Unterrichtszwecken verwendet, um schnell und leicht reproduzierbare Beispiele zu erstellen. Es gibt eine schöne Webseite mit den eingebauten Datensätzen:

<https://vincentarelbundock.github.io/Rdatasets/datasets.html>

Beispiel

Schweizer Fertilitäts- und sozioökonomische Indikatoren (1888) Daten. Untersuchen wir den Unterschied in der Fruchtbarkeit basierend auf der ländlichen Entwicklung und der Dominanz der katholischen Bevölkerung.

```
library(tidyverse)

swiss %>%
  ggplot(aes(x = Agriculture, y = Fertility,
             color = Catholic > 50))+
  geom_point()+
  stat_ellipse()
```

Fertility

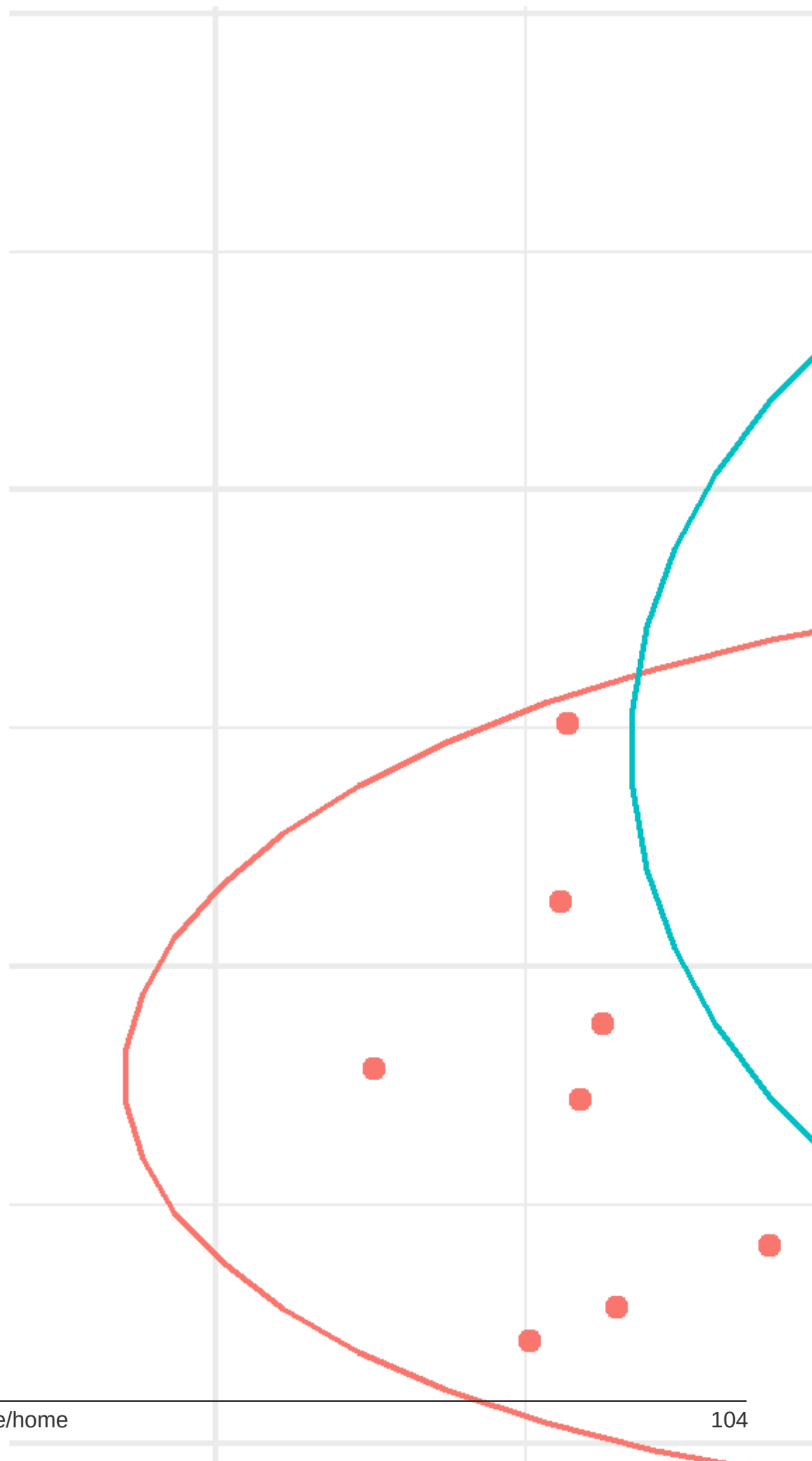
110

90

70

50

104



verfügt, werden nicht alle relevanten Datensätze gefunden. Dies ist bequemer, wenn Sie den Code eines Datensatzes manuell auf der Eurostat-Website durchsuchen: [Länder-](#) oder [Regionaldatenbank](#) . Wenn der automatisierte Download nicht funktioniert, können die Daten manuell über die [Bulk Download-Funktion](#) abgerufen werden.

```
library(tidyverse)
library(lubridate)
library(forcats)
library(eurostat)
library(geofacet)
library(viridis)
library(ggthemes)
library(extrafont)

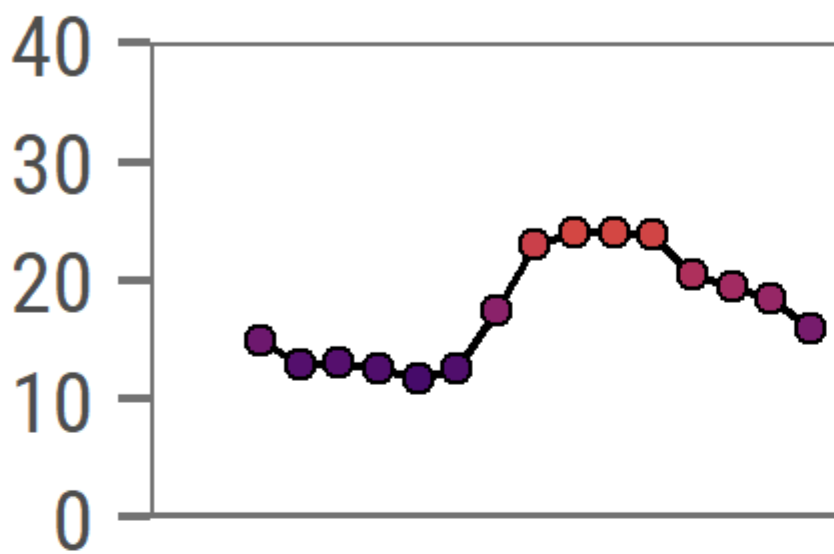
# download NEET data for countries
neet <- get_eurostat("edat_lfse_22")

neet %>%
  filter(geo %>% paste %>% nchar == 2,
         sex == "T", age == "Y18-24") %>%
  group_by(geo) %>%
  mutate(avg = values %>% mean()) %>%
  ungroup() %>%
  ggplot(aes(x = time %>% year(),
             y = values))+
  geom_path(aes(group = 1))+
  geom_point(aes(fill = values), pch = 21)+
  scale_x_continuous(breaks = seq(2000, 2015, 5),
                    labels = c("2000", "'05", "'10", "'15"))+
  scale_y_continuous(expand = c(0, 0), limits = c(0, 40))+
  scale_fill_viridis("NEET, %", option = "B")+
  facet_geo(~ geo, grid = "eu_grid1")+
  labs(x = "Year",
       y = "NEET, %",
       title = "Young people neither in employment nor in education and training in
Europe",
       subtitle = "Data: Eurostat Regional Database, 2000-2016",
       caption = "ikashnitsky.github.io")+
  theme_few(base_family = "Roboto Condensed", base_size = 15)+
  theme(axis.text = element_text(size = 10),
        panel.spacing.x = unit(1, "lines"),
        legend.position = c(0, 0),
        legend.justification = c(0, 0))
```

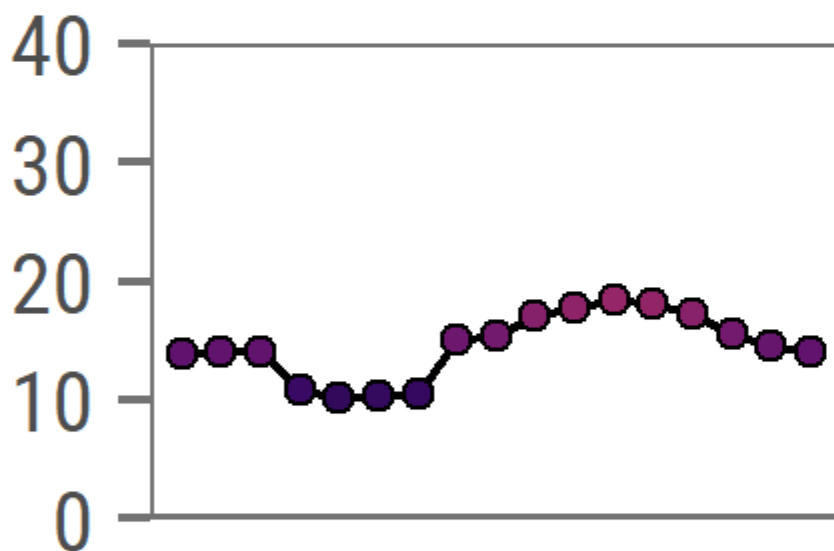
Young people neither

Data: Eurostat Regional D

IE



UK



[Max-Planck-Instituts für demografische Forschung](https://www.mortality.org/) , das Daten zur menschlichen Mortalität für jene Länder erfasst und vorverarbeitet, in denen mehr oder weniger zuverlässige Statistiken verfügbar sind.

```
# load required packages
library(tidyverse)
library(extrafont)
library(HMDHFDplus)

country <- getHMDcountries()

exposures <- list()
for (i in 1:length(country)) {
  cnt <- country[i]
  exposures[[cnt]] <- readHMDweb(cnt, "Exposures_1x1", user_hmd, pass_hmd)
  # let's print the progress
  paste(i,'out of',length(country))
} # this will take quite a lot of time
```

Bitte beachten Sie, dass die Argumente `user_hmd` und `pass_hmd` die Anmeldeinformationen auf der Website der Human Mortality Database sind. Um auf die Daten zugreifen zu können, müssen Sie ein Konto bei <http://www.mortality.org/> erstellen und Ihre eigenen Anmeldeinformationen für die Funktion `readHMDweb()` .

```
sr_age <- list()

for (i in 1:length(exposures)) {
  di <- exposures[[i]]
  sr_agei <- di %>% select(Year, Age, Female, Male) %>%
    filter(Year %in% 2012) %>%
    select(-Year) %>%
    transmute(country = names(exposures)[i],
              age = Age, sr_age = Male / Female * 100)
  sr_age[[i]] <- sr_agei
}
sr_age <- bind_rows(sr_age)

# remove optional populations
sr_age <- sr_age %>% filter(!country %in% c("FRACNP", "DEUTE", "DEUTW", "GBRCENW", "GBR_NP"))

# summarize all ages older than 90 (too jerky)
sr_age_90 <- sr_age %>% filter(age %in% 90:110) %>%
  group_by(country) %>% summarise(sr_age = mean(sr_age, na.rm = T)) %>%
  ungroup() %>% transmute(country, age=90, sr_age)

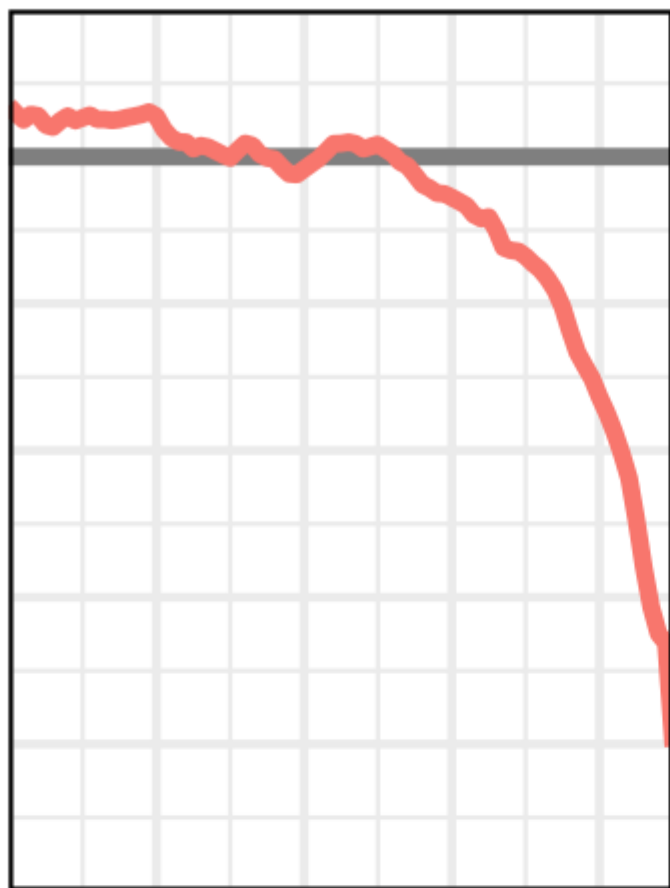
df_plot <- bind_rows(sr_age %>% filter(!age %in% 90:110), sr_age_90)

# finally - plot
df_plot %>%
  ggplot(aes(age, sr_age, color = country, group = country))+
  geom_hline(yintercept = 100, color = 'grey50', size = 1)+
  geom_line(size = 1)+
  scale_y_continuous(limits = c(0, 120), expand = c(0, 0), breaks = seq(0, 120, 20))+
  scale_x_continuous(limits = c(0, 90), expand = c(0, 0), breaks = seq(0, 80, 20))+
  xlab('Age')+
  ylab('Sex ratio, males per 100 females')+
  facet_wrap(~country, ncol=6)+
  theme_minimal(base_family = "Roboto Condensed", base_size = 15)+
```

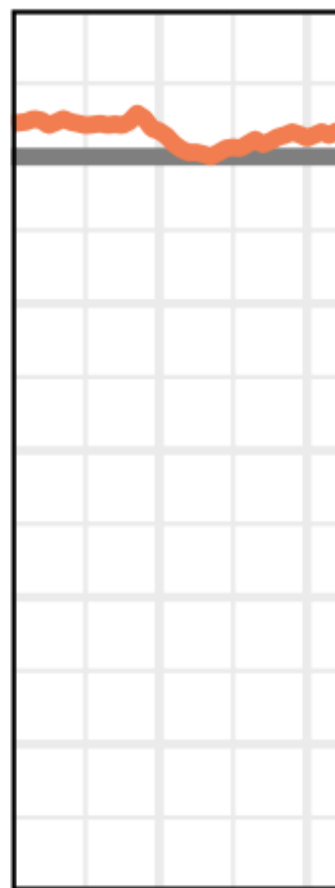
```
theme(legend.position='none',  
      panel.border = element_rect(size = .5, fill = NA))
```

AUT

120
100
80
60
40
20
0

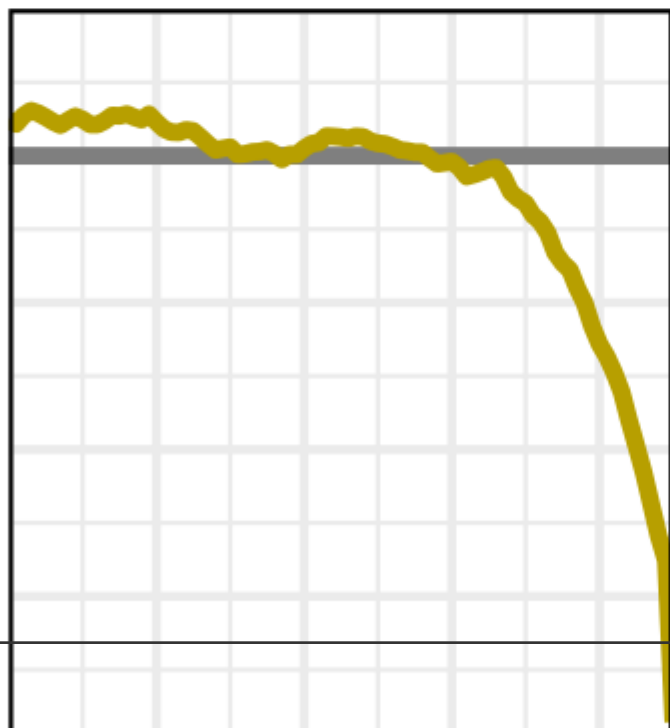


BI

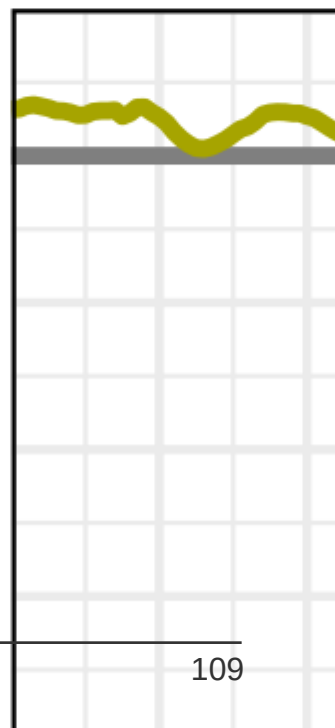


DNK

120
100
80
60
40
20
0



ES



Kapitel 23: Datenrahmen

Syntax

- `data.frame(..., row.names = NULL, check.rows = FALSE, check.names = TRUE, stringsAsFactors = default.stringsAsFactors())`
- `as.data.frame(x, row.names = NULL, optional = FALSE, ...)` # generische Funktion
- `as.data.frame(x, ..., stringsAsFactors = default.stringsAsFactors())` # S3-Methode für Klasse 'Zeichen'
- `as.data.frame(x, row.names = NULL, optional = FALSE, ..., stringsAsFactors = default.stringsAsFactors())` # S3-Methode für Klasse 'Matrix'
- `is.data.frame(x)`

Examples

Erstellen Sie ein leeres data.frame

Ein Datenrahmen ist eine besondere Art von Liste: sie ist *rechteckig*. Jedes Element (Spalte) der Liste hat dieselbe Länge und jede Zeile hat einen "Zeilenamen". Jede Spalte hat ihre eigene Klasse, aber die Klasse einer Spalte kann sich von der Klasse einer anderen Spalte unterscheiden (im Gegensatz zu einer Matrix, bei der alle Elemente dieselbe Klasse haben müssen).

Ein `data.frame` kann grundsätzlich keine Zeilen und keine Spalten haben:

```
> structure(list(character()), class = "data.frame")
NULL
<0 rows> (or 0-length row.names)
```

Das ist aber ungewöhnlich. Es ist üblicher, dass ein `data.frame` viele Spalten und viele Zeilen hat. Hier ist ein Datenrahmen mit drei Zeilen und zwei Spalten (`a` ist numerische Klasse und `b` ist Zeichenklasse):

```
> structure(list(a = 1:3, b = letters[1:3]), class = "data.frame")
[1] a b
<0 rows> (or 0-length row.names)
```

Damit der `data.frame` gedruckt werden kann, müssen wir einige Zeilenamen angeben. Hier verwenden wir nur die Zahlen 1: 3:

```
> structure(list(a = 1:3, b = letters[1:3]), class = "data.frame", row.names = 1:3)
  a b
1 1 a
2 2 b
```

```
3 3 c
```

Jetzt wird es offensichtlich, dass wir einen Datenrahmen mit 3 Zeilen und 2 Spalten haben. Sie können dies mit `nrow()`, `ncol()` und `dim()` überprüfen:

```
> x <- structure(list(a = numeric(3), b = character(3)), class = "data.frame", row.names = 1:3)
> nrow(x)
[1] 3
> ncol(x)
[1] 2
> dim(x)
[1] 3 2
```

R bietet zwei weitere Funktionen (neben `structure()`), mit denen ein `data.frame` erstellt werden kann. Die erste heißt intuitiv `data.frame()`. Es überprüft, ob die von Ihnen angegebenen Spaltennamen gültig sind, dass die Listenelemente alle die gleiche Länge haben, und liefert einige automatisch generierte Zeilenamen. Das bedeutet, dass die Ausgabe von `data.frame()` jetzt immer genau das ist, was Sie erwarten:

```
> str(data.frame("a a a" = numeric(3), "b-b-b" = character(3)))
'data.frame': 3 obs. of 2 variables:
 $ a.a.a: num 0 0 0
 $ b.b.b: Factor w/ 1 level " ": 1 1 1
```

Die andere Funktion heißt `as.data.frame()`. Dies kann verwendet werden, um ein Objekt, das kein `data.frame` ist, zu einem `data.frame` zu zwingen, indem es durch `data.frame()`. Als Beispiel betrachten wir eine Matrix:

```
> m <- matrix(letters[1:9], nrow = 3)
> m
      [,1] [,2] [,3]
[1,] "a"  "d"  "g"
[2,] "b"  "e"  "h"
[3,] "c"  "f"  "i"
```

Und das Ergebnis:

```
> as.data.frame(m)
  V1 V2 V3
1  a  d  g
2  b  e  h
3  c  f  i
> str(as.data.frame(m))
'data.frame': 3 obs. of 3 variables:
 $ V1: Factor w/ 3 levels "a","b","c": 1 2 3
 $ V2: Factor w/ 3 levels "d","e","f": 1 2 3
 $ V3: Factor w/ 3 levels "g","h","i": 1 2 3
```

Zeilen und Spalten aus einem Datenrahmen subetting

Syntax für Zeilen und Spalten zuzugreifen: `[ , ]` und `$`

In diesem Thema wird die am häufigsten verwendete Syntax für den Zugriff auf bestimmte Zeilen und Spalten eines Datenrahmens behandelt. Diese sind

- Wie eine `matrix` mit `data[rows, columns]` Klammern `data[rows, columns]`
 - Verwenden Sie Zeilen- und Spaltennummern
 - Spalten- (und Zeilen-) Namen verwenden
- Wie eine `list` :
 - Mit einfachen Klammern `data[columns]` , um einen Datenrahmen zu erhalten
 - Mit doppelten Klammern `data[[one_column]]` erhalten Sie einen Vektor
- Mit `$` für eine einzelne Spalte `data$column_name`

Wir werden den eingebauten `mtcars` - `mtcars` zur Veranschaulichung verwenden.

Wie eine Matrix: `data[rows, columns]`

Mit numerischen Indizes

Mithilfe der integrierten `mtcars` können Zeilen und Spalten mit `[ ] mtcars` mit einem Komma extrahiert werden. Indizes vor dem Komma sind Zeilen:

```
# get the first row
mtcars[1, ]
# get the first five rows
mtcars[1:5, ]
```

Ähnlich sind nach dem Komma Spalten:

```
# get the first column
mtcars[, 1]
# get the first, third and fifth columns:
mtcars[, c(1, 3, 5)]
```

Wie oben gezeigt, werden alle Zeilen ausgewählt, wenn entweder Zeilen oder Spalten leer bleiben. `mtcars[1, ]` gibt die erste Zeile mit *allen* Spalten an.

Mit Spaltennamen (und Zeilenamen)

Dies ist bisher identisch mit dem Zugriff auf Zeilen und Spalten von Matrizen. Bei `data.frame`s ist es meistens vorzuziehen, einen Spaltennamen für einen Spaltenindex zu verwenden. Dies geschieht durch die Verwendung eines `character` mit dem Spaltennamen anstelle von `numeric` mit einer Spaltennummer:

```
# get the mpg column
```



```
mtcars[, "mpg"]  
# get the mpg, cyl, and disp columns  
mtcars[, c("mpg", "cyl", "disp")]
```

Obwohl weniger üblich, können auch Zeilenamen verwendet werden:

```
mtcars["Mazda Rx4", ]
```

Zeilen und Spalten zusammen

Die Zeilen- und Spaltenargumente können zusammen verwendet werden:

```
# first four rows of the mpg column  
mtcars[1:4, "mpg"]  
  
# 2nd and 5th row of the mpg, cyl, and disp columns  
mtcars[c(2, 5), c("mpg", "cyl", "disp")]
```

Eine Warnung zu den Abmessungen:

Wenn Sie mit diesen Methoden mehrere Spalten extrahieren, erhalten Sie einen Datenrahmen zurück. Wenn Sie jedoch eine *einzelne* Spalte extrahieren, wird unter den Standardoptionen ein Vektor und kein Datenrahmen angezeigt.

```
## multiple columns returns a data frame  
class(mtcars[, c("mpg", "cyl")])  
# [1] "data.frame"  
## single column returns a vector  
class(mtcars[, "mpg"])  
# [1] "numeric"
```

Es gibt zwei Möglichkeiten, dies zu umgehen. Eine besteht darin, den Datenrahmen als Liste zu behandeln (siehe unten), die andere besteht darin, das Argument `drop = FALSE` hinzuzufügen. Dies weist R an, "die nicht verwendeten Dimensionen nicht zu löschen":

```
class(mtcars[, "mpg", drop = FALSE])  
# [1] "data.frame"
```

Beachten Sie, dass Matrizen auf dieselbe Weise funktionieren. Standardmäßig ist eine einzelne Spalte oder Zeile ein Vektor. Wenn Sie `drop = FALSE` angeben, können Sie sie als einspaltige oder einzeilige Matrix beibehalten.

Wie eine Liste

Datenframes sind im Wesentlichen `list`, dh sie sind eine Liste von Spaltenvektoren (die alle dieselbe Länge haben müssen). Listen können mit einfachen Klammern `[` für eine Unterliste oder mit doppelten Klammern `[[` für ein einzelnes Element] Teilmengen sein.

Mit Klammern `data[columns]`

Wenn Sie einfache Klammern und keine Kommas verwenden, erhalten Sie die Spalte zurück, da Datenrahmen Spaltenlisten sind.

```
mtcars["mpg"]
mtcars[c("mpg", "cyl", "disp")]
my_columns <- c("mpg", "cyl", "hp")
mtcars[my_columns]
```

Einzelne Klammern *wie eine Liste* oder einzelne Klammern *wie eine Matrix*

Der Unterschied zwischen `data[columns]` und `data[, columns]` besteht darin, dass bei der Behandlung von `data.frame` als `list` (kein Komma in Klammern) das zurückgegebene Objekt *ein* `data.frame` . Wenn Sie den `data.frame` mit einem Komma wie eine `matrix` behandeln, wird beim Auswählen einer einzelnen Spalte ein Vektor zurückgegeben. Wenn Sie jedoch mehrere Spalten auswählen, wird ein `data.frame` .

```
## When selecting a single column
## like a list will return a data frame
class(mtcars["mpg"])
# [1] "data.frame"
## like a matrix will return a vector
class(mtcars[, "mpg"])
# [1] "numeric"
```

Mit doppelten Klammern `data[[one_column]]`

Um eine einzelne Spalte *als Vektor* zu extrahieren, wenn Sie Ihren `data.frame` als `list` , können Sie doppelte Klammern `[[` . Dies funktioniert nur für eine einzelne Spalte.

```
# extract a single column by name as a vector
mtcars[["mpg"]]

# extract a single column by name as a data frame (as above)
mtcars["mpg"]
```

Mit `$` auf Spalten zugreifen

Eine einzelne Spalte kann mit der magischen Verknüpfung `$` extrahiert werden, ohne einen in Anführungszeichen gesetzten Spaltennamen zu verwenden:

```
# get the column "mpg"
mtcars$mpg
```

Spalten, auf die mit `$` zugegriffen wird, sind immer Vektoren und keine Datenrahmen.

Nachteile von `$` für den Zugriff auf Spalten

Das `$` kann eine bequeme Verknüpfung sein, insbesondere wenn Sie in einer Umgebung arbeiten (z. B. RStudio), die in diesem Fall den Spaltennamen automatisch vervollständigt. `$` Hat jedoch auch Nachteile: Es verwendet eine *nicht standardmäßige Bewertung*, um die Notwendigkeit von Anführungszeichen zu vermeiden. **Dies** bedeutet, dass es *nicht funktioniert*, wenn der Spaltenname in einer Variablen gespeichert ist.

```
my_column <- "mpg"
# the below will not work
mtcars$my_column
# but these will work
mtcars[, my_column] # vector
mtcars[my_column]   # one-column data frame
mtcars[[my_column]] # vector
```

Aufgrund dieser Bedenken wird `$` am besten in *interaktiven* R-Sitzungen verwendet, wenn die Spaltennamen konstant sind. Für den *programmatischen* Gebrauch, beispielsweise beim Schreiben einer verallgemeinerbaren Funktion, die für verschiedene Datensätze mit unterschiedlichen Spaltennamen verwendet wird, sollte `$` vermieden werden.

Beachten Sie außerdem, dass das Standardverhalten darin besteht, partiellen Abgleich nur beim Extrahieren von rekursiven Objekten (außer Umgebungen) mit `$`

```
# give you the values of "mpg" column
# as "mtcars" has only one column having name starting with "m"
mtcars$m
# will give you "NULL"
# as "mtcars" has more than one columns having name starting with "d"
mtcars$d
```

Erweiterte Indexierung: Negative und logische Indizes

Wenn wir die Option haben, Zahlen für einen Index zu verwenden, können wir auch negative Zahlen verwenden, um bestimmte Indizes wegzulassen, oder einen booleschen (logischen) Vektor, um genau anzugeben, welche Elemente beibehalten werden sollen.

Negative Indizes lassen Elemente aus

```
mtcars[1, ] # first row
mtcars[-1, ] # everything but the first row
mtcars[-(1:10), ] # everything except the first 10 rows
```

Logische Vektoren zeigen bestimmte Elemente an, die beibehalten werden sollen

Wir können eine Bedingung wie `<`, um einen logischen Vektor zu generieren und nur die Zeilen zu

extrahieren, die die Bedingung erfüllen:

```
# logical vector indicating TRUE when a row has mpg less than 15
# FALSE when a row has mpg >= 15
test <- mtcars$mpg < 15

# extract these rows from the data frame
mtcars[test, ]
```

Wir können den Schritt des Speicherns der Zwischenvariablen auch umgehen

```
# extract all columns for rows where the value of cyl is 4.
mtcars[mtcars$cyl == 4, ]
# extract the cyl, mpg, and hp columns where the value of cyl is 4
mtcars[mtcars$cyl == 4, c("cyl", "mpg", "hp")]
```

Komfortfunktionen zum Bearbeiten von data.frames

Einige praktische Funktionen zum Bearbeiten von `data.frames` sind `subset()`, `transform()`, `with()` und `within()`.

Teilmenge

Mit der Funktion `subset()` können Sie einen `data.frame` bequemer `data.frame` (subset funktioniert auch mit anderen Klassen):

```
subset(mtcars, subset = cyl == 6, select = c("mpg", "hp"))
      mpg  hp
Mazda RX4    21.0 110
Mazda RX4 Wag 21.0 110
Hornet 4 Drive 21.4 110
Valiant      18.1 105
Merc 280      19.2 123
Merc 280C     17.8 123
Ferrari Dino  19.7 175
```

Im obigen Code fragen wir nur nach den Zeilen, in denen `cyl == 6` und nach den Spalten `mpg` und `hp`. Sie können dasselbe Ergebnis erzielen, indem Sie `[]` mit dem folgenden Code verwenden:

```
mtcars[mtcars$cyl == 6, c("mpg", "hp")]
```

verwandeln

Die Funktion `transform()` ist eine praktische Funktion zum Ändern von Spalten in einem `data.frame`. Im folgenden Code wird dem `mtcars data.frame` beispielsweise eine weitere Spalte mit dem Namen `mpg2` mit dem Ergebnis von `mpg^2 data.frame`:

```
mtcars <- transform(mtcars, mpg2 = mpg^2)
```

mit und innerhalb

Sowohl `with()` als auch `within()` Sie Ausdrücke in der Umgebung von `data.frame` auswerten. `data.frame` ermöglicht eine etwas sauberere Syntax und erspart Ihnen die Verwendung von `$` oder `[]`.

Wenn Sie beispielsweise mehrere Spalten im `airquality data.frame` erstellen, ändern und / oder entfernen `data.frame`:

```
aq <- within(airquality, {  
  lOzone <- log(Ozone) # creates new column  
  Month <- factor(month.abb[Month]) # changes Month Column  
  cTemp <- round((Temp - 32) * 5/9, 1) # creates new column  
  S.cT <- Solar.R / cTemp # creates new column  
  rm(Day, Temp) # removes columns  
})
```

Einführung

Datenrahmen sind wahrscheinlich die Datenstruktur, die Sie in Ihren Analysen am häufigsten verwenden. Ein Datenrahmen ist eine spezielle Art von Liste, in der Vektoren gleicher Länge unterschiedlicher Klassen gespeichert sind. Sie erstellen Datenrahmen mit der Funktion `data.frame`. Das folgende Beispiel zeigt dies, indem ein numerischer und ein Zeichenvektor in einem Datenrahmen kombiniert werden. Es nutzt den `:` Operator, der einen Vektor erstellen alle Zahlen von 1 bis 3.

```
df1 <- data.frame(x = 1:3, y = c("a", "b", "c"))  
df1  
##      x y  
## 1 1 a  
## 2 2 b  
## 3 3 c  
class(df1)  
## [1] "data.frame"
```

Datenrahmenobjekte werden nicht mit Anführungszeichen gedruckt, sodass die Klasse der Spalten nicht immer offensichtlich ist.

```
df2 <- data.frame(x = c("1", "2", "3"), y = c("a", "b", "c"))  
df2  
##      x y  
## 1 1 a  
## 2 2 b  
## 3 3 c
```

Ohne weitere Untersuchung können die "x" -Spalten in `df1` und `df2` nicht unterschieden werden. Mit der `str` Funktion können Objekte detaillierter als Klassen beschrieben werden.

```
str(df1)  
## 'data.frame':    3 obs. of  2 variables:  
##  $ x: int  1 2 3
```

```
## $ y: Factor w/ 3 levels "a","b","c": 1 2 3
str(df2)
## 'data.frame':    3 obs. of  2 variables:
## $ x: Factor w/ 3 levels "1","2","3": 1 2 3
## $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

Hier sehen Sie, dass `df1` ein `data.frame` ist und hat 3 Beobachtungen von 2 Variablen „x“ und „y“. Dann wird Ihnen gesagt, dass „x“ den Datentyp Integer hat (nicht wichtig für diese Klasse, aber für unsere Zwecke verhält es sich wie eine Zahl) und „y“ ist ein Faktor mit drei Ebenen (eine andere Datenklasse, über die wir nicht sprechen). **Es ist wichtig zu beachten, dass Datenrahmen standardmäßig Zeichen zu Faktoren zwingen.** Das Standardverhalten kann mit dem Parameter `stringsAsFactors` geändert werden:

```
df3 <- data.frame(x = 1:3, y = c("a", "b", "c"), stringsAsFactors = FALSE)
str(df3)
## 'data.frame':    3 obs. of  2 variables:
## $ x: int  1 2 3
## $ y: chr  "a" "b" "c"
```

Jetzt ist die Spalte „y“ ein Zeichen. Wie oben erwähnt, muss jede „Spalte“ eines Datenrahmens dieselbe Länge haben. Der Versuch, ein `data.frame` aus Vektoren mit unterschiedlichen Längen zu erstellen, führt zu einem Fehler. (Führen Sie `data.frame(x = 1:3, y = 1:4)`, um den resultierenden Fehler `data.frame(x = 1:3, y = 1:4)` zu sehen.)

Als Testfälle für Datenrahmen werden einige Daten standardmäßig von R bereitgestellt. Eine davon ist `Iris`, die wie folgt geladen wird:

```
mydataframe <- iris
str(mydataframe)
```

Konvertieren Sie mit `do.call` in einer Liste gespeicherte Daten in einen einzelnen Datenrahmen

Wenn Sie Ihre Daten in einer Liste gespeichert haben und diese Liste in einen `do.call` konvertieren möchten, können Sie dies mit der `do.call` Funktion `do.call`. Es ist jedoch wichtig, dass alle Listenelemente die gleiche Länge haben, um ein unbeabsichtigtes Wiederverwenden von Werten zu verhindern.

```
dataList <- list(1:3, 4:6, 7:9)
dataList
# [[1]]
# [1] 1 2 3
#
# [[2]]
# [1] 4 5 6
#
# [[3]]
# [1] 7 8 9

dataframe <- data.frame(do.call(rbind, dataList))
dataframe
#   X1 X2 X3
```

```
# 1 1 2 3
# 2 4 5 6
# 3 7 8 9
```

Es funktioniert auch, wenn Ihre Liste selbst aus Datenrahmen besteht.

```
dataframeList <- list(data.frame(a = 1:2, b = 1:2, c = 1:2),
                      data.frame(a = 3:4, b = 3:4, c = 3:4))

dataframeList
# [[1]]
#   a b c
# 1 1 1 1
# 2 2 2 2

# [[2]]
#   a b c
# 1 3 3 3
# 2 4 4 4

dataframe <- do.call(rbind, dataframeList)
dataframe
#   a b c
# 1 1 1 1
# 2 2 2 2
# 3 3 3 3
# 4 4 4 4
```

Konvertieren Sie alle Spalten eines data.frame in eine Zeichenklasse

Eine übliche Aufgabe besteht darin, alle Spalten eines data.frames zur Erleichterung der Bearbeitung in eine Zeichenklasse zu konvertieren, z. B. beim Senden von data.frames an ein RDBMS oder beim Zusammenführen von data.frames, die Faktoren enthalten, bei denen die Ebenen zwischen den eingegebenen data.frames variieren können .

Die beste Zeit dafür ist, wenn die Daten eingelesen werden. Fast alle Eingabemethoden, die Datenrahmen erstellen, verfügen über `stringsAsFactors` die auf `FALSE` .

Wenn die Daten bereits erstellt wurden, können Faktorspalten wie unten dargestellt in Zeichenspalten konvertiert werden.

```
bob <- data.frame(jobs = c("scientist", "analyst"),
                 pay = c(160000, 100000), age = c(30, 25))
str(bob)
```

```
'data.frame':    2 obs. of  3 variables:
 $ jobs: Factor w/ 2 levels "analyst","scientist": 2 1
 $ pay : num  160000 100000
 $ age : num   30 25
```

```
# Convert *all columns* to character
bob[] <- lapply(bob, as.character)
str(bob)
```

```
'data.frame': 2 obs. of 3 variables:
 $ jobs: chr "scientist" "analyst"
 $ pay : chr "160000" "1e+05"
 $ age : chr "30" "25"
```

```
# Convert only factor columns to character
bob[] <- lapply(bob, function(x) {
  if is.factor(x) x <- as.character(x)
  return(x)
})
```

Zeilen nach Spaltenwerten unterteilen

Integrierte Funktionen können `rows` mit `columns`, die die Bedingungen erfüllen.

```
df <- data.frame(item = c(1:10),
  price_Elasticity = c(-0.57667, 0.03205, -0.04904, 0.10342, 0.04029,
    0.0742, 0.1669, 0.0313, 0.22204, 0.06158),
  total_Margin = c(-145062, 98671, 20576, -56382, 207623, 43463, 1235,
    34521, 146553, -74516))
```

So finden Sie `rows` mit `price_Elasticity > 0`:

```
df[df$price_Elasticity > 0, ]
```

	item	price_Elasticity	total_Margin
2	2	0.03205	98671
4	4	0.10342	-56382
5	5	0.04029	207623
6	6	0.07420	43463
7	7	0.16690	1235
8	8	0.03130	34521
9	9	0.22204	146553
10	10	0.06158	-74516

Teilmenge basierend auf `price_Elasticity > 0` und `total_Margin > 0`:

```
df[df$price_Elasticity > 0 & df$total_Margin > 0, ]
```

	item	price_Elasticity	total_Margin
2	2	0.03205	98671
5	5	0.04029	207623
6	6	0.07420	43463
7	7	0.16690	1235
8	8	0.03130	34521
9	9	0.22204	146553

Datenrahmen online lesen: <https://riptutorial.com/de/r/topic/438/datenrahmen>

Kapitel 24: Datentabelle

Einführung

Data.table ist ein Paket, das die Funktionalität von Datenrahmen von der Basis R aus erweitert und insbesondere deren Leistung und Syntax verbessert. Weitere [Informationen finden Sie](#) im Docs-Bereich des Pakets unter [Erste Schritte mit data.table](#) .

Syntax

- `DT[i, j, by]`
DT [wo, wählen Sie | update | do, by]
- `DT[...][...]`
ketten
- ##### Shortcuts, special functions and special symbols inside DT[...]
- `()`
in mehreren Argumenten ersetzt list ()
- `J ()`
in i ersetzt list ()
- `:=`
in j, eine Funktion zum Hinzufügen oder Ändern von Spalten
- `.N`
in i die Gesamtzahl der Zeilen
in j die Anzahl der Zeilen in einer Gruppe
- `.ICH`
in j der Vektor der Zeilennummern in der Tabelle (gefiltert nach i)
- `.SD`
in j, die aktuelle Untermenge der Daten
ausgewählt durch das Argument .SDcols
- `.GRP`
in j der aktuelle Index der Untermenge der Daten
- `.DURCH`
in j die Liste der by-Werte für die aktuelle Datenuntermenge
- `V1, V2, ...`
Standardnamen für nicht benannte Spalten, die in j erstellt wurden
- ##### Joins inside DT[...]
- `DT1 [DT2, Ein, J]`
zwei Tische verbinden
- `ich.*`
Sonderpräfix für DT2-Spalten nach dem Join
- `von = .EACHI`
spezielle Option nur mit einem Join verfügbar
- `DT1 [! DT2, Ein, J]`
Anti-Join von zwei Tabellen
- `DT1 [DT2, ein, rollen, j]`

- `# Verbinden Sie zwei Tabellen und rollen Sie in der letzten Spalte von on =`
- `##### Reshaping, stacking and splitting`
- Schmelze (DT, id.vars, measure.vars)
 - `# in langes Format umwandeln`
 - `# Für mehrere Spalten verwenden Sie measure.vars = pattern (...).`
- dcast (DT, Formel)
 - `# in ein Breitformat umwandeln`
- rbind (DT1, DT2, ...)
- rbindlist (DT_list, idcol)
 - `# Stack aufgezählte data.tables`
- Split (DT, von)
 - `# eine Datentabelle in eine Liste aufteilen`
- `##### Some other functions specialized for data.tables`
- Foverlaps
 - `# Überlappungen`
- verschmelzen
 - `# eine andere Möglichkeit, zwei Tabellen zu verbinden`
- einstellen
 - `# eine andere Möglichkeit, Spalten hinzuzufügen oder zu ändern`
- fintersect, fsetdiff, funion, fsetequal, einzigartig, dupliziert, anyDuplicated
 - `# Mengenoperationsoperationen mit Zeilen als Elementen`
- uniqueN
 - `Anzahl der verschiedenen Zeilen`
- rowidv (DT, Spalten)
 - `# Zeilen-ID (1 bis .N) innerhalb jeder Gruppe, bestimmt durch Spalten`
- rleidv (DT, cols)
 - `# Gruppen-ID (1 bis .GRP) innerhalb jeder Gruppe, bestimmt durch Anzahl von Spalten`
- Verschiebung (DT, n, Typ = c ("Lag", "Lead"))
 - `# Einen Schichtoperator auf jede Spalte anwenden`
- setorder, setcolororder, setnames, setkey, setindex, setattr
 - `# Attribute und Reihenfolge nach Verweis ändern`

Bemerkungen

Installation und Unterstützung

So installieren Sie das Paket "data.table":

```
# install from CRAN
install.packages("data.table")

# or install development version
install.packages("data.table", type = "source", repos =
"http://Rdatatable.github.io/data.table")

# and to revert from devel to CRAN, the current version must first be removed
```

```
remove.packages("data.table")
install.packages("data.table")
```

Auf der [offiziellen Website des Pakets](#) finden Sie Wiki-Seiten, die Ihnen den Einstieg erleichtern, sowie Listen mit Präsentationen und Artikeln aus dem Internet. Bevor Sie eine Frage stellen - hier bei StackOverflow oder anderswo - lesen Sie bitte [die Support-Seite](#) .

Paket laden

Viele der Funktionen in den obigen Beispielen sind im Namespace `data.table` enthalten. Um sie zu verwenden, müssen Sie zuerst eine Zeile wie die `library(data.table)` hinzufügen oder ihren vollständigen Pfad verwenden, beispielsweise `data.table::fread` anstatt nur `fread` . Für Hilfe zu den einzelnen Funktionen ist die Syntax - `help("fread")` oder `?fread` . Wenn das Paket nicht geladen ist, verwenden Sie erneut den vollständigen Namen wie `?data.table::fread` .

Examples

Eine Datentabelle anlegen

Eine `data.table` ist eine erweiterte Version der `data.frame`-Klasse von Basis R. Das `class()` - Attribut ist daher der Vektor `"data.table" "data.frame"` Funktionen, die auch auf einem `data.frame` funktionieren, werden verwendet Arbeit mit einer Datentabelle. Es gibt viele Möglichkeiten, eine `data.table` zu erstellen, zu laden oder zu zwingen.

Bauen

Vergessen Sie nicht, das Paket `data.table` zu installieren und zu aktivieren

```
library(data.table)
```

Es gibt einen Konstruktor mit demselben Namen:

```
DT <- data.table(
  x = letters[1:5],
  y = 1:5,
  z = (1:5) > 3
)
#      x y      z
# 1: a 1 FALSE
# 2: b 2 FALSE
# 3: c 3 FALSE
# 4: d 4  TRUE
# 5: e 5  TRUE
```

Im Gegensatz zu `data.frame` `data.table` `data.frame` `data.table` nicht zu Faktoren:

```
sapply(DT, class)
#           x           y           z
# "character" "integer" "logical"
```

Einlesen

Wir können aus einer Textdatei lesen:

```
dt <- fread("my_file.csv")
```

Im `read.csv` `fread` liest `fread` Strings als Strings und nicht als Faktoren.

Ändern Sie einen Datenrahmen

Aus Gründen der Effizienz bietet `data.table` eine Möglichkeit, ein `data.frame` oder eine Liste zu ändern, um eine `data.table` an Ort und Stelle zu machen (ohne eine Kopie zu erstellen oder den Speicherort zu ändern):

```
# example data.frame
DF <- data.frame(x = letters[1:5], y = 1:5, z = (1:5) > 3)
# modification
setDT(DF)
```

Beachten Sie, dass wir das Ergebnis nicht `<-` zuweisen, da das Objekt `DF` vor Ort geändert wurde. Die Klassenattribute des `data.frame` bleiben erhalten:

```
sapply(DF, class)
#           x           y           z
# "factor" "integer" "logical"
```

Erzwingen Sie ein Objekt in data.table

Wenn Sie über eine `list`, einen `data.frame` oder eine `data.table`, sollten Sie die Funktion `setDT` zum Konvertieren in eine `data.table` da sie die Konvertierung nach Verweis `as.data.table` anstatt eine Kopie zu `as.data.table` (wie `as.data.table`). Dies ist wichtig, wenn Sie mit großen Datensätzen arbeiten.

Wenn Sie ein anderes R-Objekt haben (z. B. eine Matrix), müssen Sie `as.data.table`, um es in eine `data.table`.

```
mat <- matrix(0, ncol = 10, nrow = 10)

DT <- as.data.table(mat)
# or
DT <- data.table(mat)
```

Spalten hinzufügen und ändern

`DT[where, select|update|do, by]` Syntax wird verwendet, um mit Spalten einer `data.table` zu arbeiten.

- "Wo" ist das `i` Argument
- Der Teil "select | update | do" ist das Argument `j`

Diese beiden Argumente werden normalerweise nach Position statt nach Name übergeben.

Unsere Beispieldaten sind unten

```
mtcars = data.table(mtcars, keep.rownames = TRUE)
```

Ganze Spalten bearbeiten

Verwenden Sie den Operator `:=` in `j`, um neue Spalten zuzuweisen:

```
mtcars[, mpg_sq := mpg^2]
```

Spalten entfernen, indem Sie `NULL`:

```
mtcars[, mpg_sq := NULL]
```

Fügen Sie mehrere Spalten hinzu, indem Sie das multivariate Format des Operators `:=` verwenden:

```
mtcars[, `:=`(mpg_sq = mpg^2, wt_sqrt = sqrt(wt))]  
# or  
mtcars[, c("mpg_sq", "wt_sqrt") := .(mpg^2, sqrt(wt))]
```

Wenn die Spalten abhängig sind und nacheinander definiert werden müssen, gibt es eine Möglichkeit:

```
mtcars[, c("mpg_sq", "mpg2_hp") := .(temp1 <- mpg^2, temp1/hp)]
```

Die Syntax `.()` Wird verwendet, wenn die rechte Seite von `LHS := RHS` eine Liste von Spalten ist.

Verwenden Sie für dynamisch bestimmte Spaltennamen Klammern:

```
vn = "mpg_sq"  
mtcars[, (vn) := mpg^2]
```

Spalten können auch mit `set` geändert werden, obwohl dies selten notwendig ist:

```
set(mtcars, j = "hp_over_wt", v = mtcars$hp/mtcars$wt)
```

Untermengen von Spalten bearbeiten

Verwenden Sie das Argument "i", um die Zeilen in den Untermengen zu sortieren, in denen "Änderungen" vorgenommen werden sollen:

```
mtcars[1:3, newvar := "Hello"]  
# or  
set(mtcars, j = "newvar", i = 1:3, v = "Hello")
```

Wie in einem data.frame können wir eine Teilmenge mit Zeilennummern oder logischen Tests vornehmen. Es ist auch möglich, einen "Join" in i, die kompliziertere Aufgabe wird jedoch in einem anderen Beispiel behandelt.

Spaltenattribute bearbeiten

Funktionen, die Attribute bearbeiten, z. B. `levels<-` oder `names<-`, ersetzen ein Objekt tatsächlich durch eine modifizierte Kopie. Auch wenn nur eine Spalte in einer Datentabelle verwendet wird, wird das gesamte Objekt kopiert und ersetzt.

Um ein Objekt ohne Kopien zu ändern, verwenden Sie `setnames`, um die Spaltennamen von data.table oder data.frame und `setattr` zu ändern, um ein Attribut für ein beliebiges Objekt zu ändern.

```
# Print a message to the console whenever the data.table is copied  
tracemem(mtcars)  
mtcars[, cyl2 := factor(cyl)]  
  
# Neither of these statements copy the data.table  
setnames(mtcars, old = "cyl2", new = "cyl_fac")  
setattr(mtcars$cyl_fac, "levels", c("four", "six", "eight"))  
  
# Each of these statements copies the data.table  
names(mtcars)[names(mtcars) == "cyl_fac"] <- "cf"  
levels(mtcars$cf) <- c("IV", "VI", "VIII")
```

Beachten Sie, dass diese Änderungen durch Verweis vorgenommen werden, sodass sie *global sind*. Wenn Sie sie innerhalb einer Umgebung ändern, wirkt sich dies auf das Objekt in allen Umgebungen aus.

```
# This function also changes the levels in the global environment  
edit_levels <- function(x) setattr(x, "levels", c("low", "med", "high"))  
edit_levels(mtcars$cyl_factor)
```

Spezielle Symbole in data.table

.SD

`.SD` bezieht sich auf die Teilmenge der `data.table` für jede Gruppe, ausgenommen alle Spalten, die in verwendet `by` .

`.SD` zusammen mit `lapply` verwendet werden, um jede Funktion auf mehrere Spalten nach Gruppe in einer `data.table`

Wir werden weiterhin dasselbe eingebaute Dataset, `mtcars` :

```
mtcars = data.table(mtcars) # Let's not include rownames to keep things simpler
```

Mittelwert aller Spalten im Datensatz nach *Zylinderanzahl* , `cyl` :

```
mtcars[, lapply(.SD, mean), by = cyl]

#   cyl      mpg      disp      hp      drat      wt      qsec      vs      am      gear
carb
#1:   6 19.74286 183.3143 122.28571 3.585714 3.117143 17.97714 0.5714286 0.4285714 3.857143
3.428571
#2:   4 26.66364 105.1364  82.63636 4.070909 2.285727 19.13727 0.9090909 0.7272727 4.090909
1.545455
#3:   8 15.10000 353.1000 209.21429 3.229286 3.999214 16.77214 0.0000000 0.1428571 3.285714
3.500000
```

Neben `cyl` gibt es im Datensatz weitere Kategoriespalten wie `vs` , `am` , `gear` und `carb` . Es macht keinen Sinn, den `mean` dieser Spalten zu nehmen. Lassen Sie uns diese Spalten ausschließen. Hier kommt das `.SDcols` ins `.SDcols` .

.Scols

`.SDcols` gibt die Spalten der `data.table` , die in `.SD` .

Mittel aller Spalten (continuous Spalten) in der Datenmenge durch die *Anzahl der Zahnräder* `gear` und die *Anzahl der Zylinder*, `cyl` , angeordnet durch `gear` und `cyl` :

```
# All the continuous variables in the dataset
cols_chosen <- c("mpg", "disp", "hp", "drat", "wt", "qsec")

mtcars[order(gear, cyl), lapply(.SD, mean), by = .(gear, cyl), .SDcols = cols_chosen]

#   gear cyl      mpg      disp      hp      drat      wt      qsec
#1:   3   4 21.500 120.1000  97.0000 3.700000 2.465000 20.0100
#2:   3   6 19.750 241.5000 107.5000 2.920000 3.337500 19.8300
#3:   3   8 15.050 357.6167 194.1667 3.120833 4.104083 17.1425
#4:   4   4 26.925 102.6250  76.0000 4.110000 2.378125 19.6125
#5:   4   6 19.750 163.8000 116.5000 3.910000 3.093750 17.6700
#6:   5   4 28.200 107.7000 102.0000 4.100000 1.826500 16.8000
#7:   5   6 19.700 145.0000 175.0000 3.620000 2.770000 15.5000
#8:   5   8 15.400 326.0000 299.5000 3.880000 3.370000 14.5500
```

Vielleicht möchten wir den `mean` nach Gruppen berechnen. Um den Mittelwert für alle Autos im Datensatz zu berechnen, geben wir die `by` Variable nicht an.

```
mtcars[, lapply(.SD, mean), .SDcols = cols_chosen]

#      mpg      disp      hp      drat      wt      qsec
#1: 20.09062 230.7219 146.6875 3.596563 3.21725 17.84875
```

Hinweis:

- Es ist nicht notwendig, vorher `cols_chosen` zu definieren. `.SDcols` können `.SDcols` direkt übernehmen
- `.SDcols` können auch direkt einen Vektor von Spaltennummern aufnehmen. Im obigen Beispiel wäre dies `mtcars[, lapply(.SD, mean), .SDcols = c(1,3:7)]`

.N

`.N` ist die Abkürzung für die Anzahl der Zeilen in einer Gruppe.

```
iris[, .(count=.N), by=Species]

#      Species count
#1:      setosa     50
#2: versicolor     50
#3:  virginica     50
```

Code schreiben, der mit `data.frame` und `data.table` kompatibel ist

Unterschiede in der Subsetting-Syntax

A `data.table` ist eines von mehreren zweidimensionalen Datenstrukturen in R neben `data.frame`, `matrix` und (2D) `array`. Alle diese Klassen verwenden eine sehr ähnliche, jedoch nicht identische Syntax für das Subsetting, das Schema `A[rows, cols]`.

Betrachten Sie die folgenden Daten, die in einer `matrix`, einem `data.frame` und einer `data.table`:

```
ma <- matrix(rnorm(12), nrow=4, dimnames=list(letters[1:4], c('X', 'Y', 'Z')))
df <- as.data.frame(ma)
dt <- as.data.table(ma)

ma[2:3] #---> returns the 2nd and 3rd items, as if 'ma' were a vector (because it is!)
df[2:3] #---> returns the 2nd and 3rd columns
dt[2:3] #---> returns the 2nd and 3rd rows!
```

Wenn Sie sicher sein möchten, was zurückgegeben wird, ist es besser, *explizit zu sein*.

Um bestimmte **Zeilen zu erhalten**, fügen Sie nach dem Bereich ein Komma hinzu:

```
ma[2:3, ] # \
df[2:3, ] # }---> returns the 2nd and 3rd rows
dt[2:3, ] # /
```


Wenn Sie jedoch Teilmengen von **Spalten** erstellen möchten, werden einige Fälle unterschiedlich interpretiert. Alle drei können auf dieselbe Weise Teilmengen mit Ganzzahlen- oder Zeichenindizes sein, die *nicht* in einer Variablen gespeichert sind.

```
ma[, 2:3]          # \
df[, 2:3]          # \
dt[, 2:3]          # }---> returns the 2nd and 3rd columns
ma[, c("Y", "Z")]  # /
df[, c("Y", "Z")]  # /
dt[, c("Y", "Z")]  # /
```

Sie unterscheiden sich jedoch für nicht benannte Variablennamen

```
mycols <- 2:3
ma[, mycols]        # \
df[, mycols]        # }---> returns the 2nd and 3rd columns
dt[, mycols, with = FALSE] # /

dt[, mycols]        # ---> Raises an error
```

Im letzten Fall wird `mycols` als Name einer Spalte ausgewertet. Da `dt` keine Spalte namens `mycols`, wird ein Fehler `mycols`.

Anmerkung: Bei Versionen des `data.table` Pakets vor 1.9.8 unterschied sich dieses Verhalten geringfügig. Alles, was sich im Spaltenindex befindet, wurde unter Verwendung von `dt` als Umgebung bewertet. Also würden sowohl `dt[, 2:3]` als auch `dt[, mycols]` den Vektor `2:3`. Für den zweiten Fall wird kein Fehler `mycols`, da die Variable `mycols` in der übergeordneten Umgebung vorhanden ist.

Strategien zur Aufrechterhaltung der Kompatibilität mit `data.frame` und `data.table`

Es gibt viele Gründe, Code zu schreiben, der garantiert mit `data.frame` und `data.table`. Möglicherweise müssen Sie `data.frame`, oder Sie müssen Code `data.frame`, der nicht weiß, wie er verwendet wird. Es gibt also einige Hauptstrategien, um dies in der Reihenfolge der Bequemlichkeit zu erreichen:

1. Verwenden Sie eine Syntax, die für beide Klassen gleich ist.
2. Verwenden Sie eine allgemeine Funktion, die dasselbe tut wie die kürzeste Syntax.
3. `data.table`, dass `data.table` sich wie `data.frame` (Beispiel: Aufruf der spezifischen Methode `print.data.frame`).
4. Behandle sie als `list`, was sie letztendlich sind.
5. Konvertieren Sie die Tabelle in einen `data.frame` bevor Sie etwas tun (schlechte Idee, wenn es sich um eine große Tabelle handelt).
6. Konvertieren Sie die Tabelle in `data.table`, wenn Abhängigkeiten keine `data.table`.

Teilmengenzeilen Es ist einfach, verwenden Sie einfach die `[, ]` Auswahl *mit* dem Komma:

```
A[1:10, ]  
A[A$var > 17, ] # A[var > 17, ] just works for data.table
```

Teilmengenspalten Wenn Sie eine einzelne Spalte wünschen, verwenden Sie die `$` oder `[[ ]]` :

```
A$var  
colname <- 'var'  
A[[colname]]  
A[[1]]
```

Wenn Sie möchten, dass mehrere Spalten einheitlich erfasst werden, müssen Sie ein wenig ansprechen:

```
B <- `[.data.frame`](A, 2:4)  
  
# We can give it a better name  
select <- `[.data.frame`  
B <- select(A, 2:4)  
C <- select(A, c('foo', 'bar'))
```

Teilmenge 'indizierte' Zeilen Während `data.frame` hat `row.names` , `data.table` hat sein einzigartiges `key` Am besten vermeiden Sie `row.names` vollständig und nutzen die vorhandenen Optimierungen im Fall von `data.table` Möglichkeit.

```
B <- A[A$var != 0, ]  
# or...  
B <- with(A, A[var != 0, ]) # data.table will silently index A by var before subsetting  
  
stuff <- c('a', 'c', 'f')  
C <- A[match(stuff, A$name), ] # really worse than: setkey(A); A[stuff, ]
```

Holen Sie sich eine 1-Spalten-Tabelle, erhalten Sie eine Zeile als Vektor. Das ist einfach mit dem, was wir bisher gesehen haben:

```
B <- select(A, 2) #---> a table with just the second column  
C <- unlist(A[1, ]) #---> the first row as a vector (coerced if necessary)
```

Schlüssel in data.table setzen

Ja, Sie müssen SETKEY vor 1.9.6 einstellen

In der Vergangenheit (vor 1.9.6) wurde Ihre `data.table` beschleunigt, indem Sie Spalten als Schlüssel für die Tabelle `data.table` , insbesondere für große Tabellen. [Siehe [Intro-Vignette Seite 5](#) der Version von September 2015, bei der die Suchgeschwindigkeit 544-mal besser war.] Möglicherweise finden Sie älteren Code, der diese Einstellungstasten mit 'setkey' verwendet oder beim Einrichten der Tabelle eine Spalte 'key =' setzt.

```
library(data.table)  
DT <- data.table(  
  x = letters[1:5],  
  y = 5:1,
```

```

  z = (1:5) > 3
)

#> DT
#   x y      z
#1: a 5 FALSE
#2: b 4 FALSE
#3: c 3 FALSE
#4: d 2  TRUE
#5: e 1  TRUE

```

`setkey` Sie Ihren Schlüssel mit dem Befehl `setkey` . Sie können einen Schlüssel mit mehreren Spalten haben.

```
setkey(DT, y)
```

Überprüfen Sie den Schlüssel Ihrer Tabelle in `tables()`

```

tables()

> tables()
      NAME NROW NCOL MB COLS  KEY
[1,] DT      5    3  1 x,y,z y
Total: 1MB

```

Beachten Sie, dass dadurch Ihre Daten neu sortiert werden.

```

#> DT
#   x y      z
#1: e 1  TRUE
#2: d 2  TRUE
#3: c 3 FALSE
#4: b 4 FALSE
#5: a 5 FALSE

```

Jetzt ist es unnötig

Vor v1.9.6 mussten Sie für bestimmte Operationen einen Schlüssel festgelegt haben, insbesondere das Verbinden von Tabellen. Die Entwickler von `data.table` haben eine `"on="` - Funktion eingeführt, die die Abhängigkeit von Schlüsseln ersetzen kann. Siehe [SO-Antwort hier für eine ausführliche Diskussion](#).

Im Januar 2017 haben die Entwickler eine [Vignette um Sekundärindizes geschrieben](#), die die "Ein"-Syntax erläutert und die Identifizierung anderer Spalten für eine schnelle Indizierung ermöglicht.

Sekundärindizes erstellen?

In ähnlicher Weise wie `key` können Sie `setindex(DT, key.col)` oder `setindexv(DT, "key.col.string")` , wobei `DT` Ihre `data.table` ist. Entfernen Sie alle Indizes mit `setindex(DT, NULL)` .

Sehen Sie Ihre Sekundärindizes mit `indices(DT)` .

Warum Sekundärindizes?

Dies **sortiert** die Tabelle nicht (im Gegensatz zu `key`), ermöglicht jedoch eine schnelle Indizierung mit der `"on"` -Syntax. Beachten Sie, dass es nur einen Schlüssel geben kann. Sie können jedoch mehrere sekundäre Indizes verwenden, sodass Sie die Tabelle nicht erneut eingeben und neu sortieren müssen. Dadurch wird Ihr Subsetting beschleunigt, wenn Sie die Spalten ändern, für die Sie einen Subset erstellen möchten.

Erinnern Sie sich, im obigen Beispiel war `y` der Schlüssel für die Tabelle `DT`:

```
DT
# x y      z
# 1: e 1  TRUE
# 2: d 2  TRUE
# 3: c 3 FALSE
# 4: b 4 FALSE
# 5: a 5 FALSE

# Let us set x as index
setindex(DT, x)

# Use indices to see what has been set
indices(DT)
# [1] "x"

# fast subset using index and not keyed column
DT["c", on = "x"]
# x y      z
#1: c 3 FALSE

# old way would have been rekeying DT from y to x, doing subset and
# perhaps keying back to y (now we save two sorts)
# This is a toy example above but would have been more valuable with big data sets
```

Datentabelle online lesen: <https://riptutorial.com/de/r/topic/849/datentabelle>

Kapitel 25: Datum und Uhrzeit

Einführung

R enthält Klassen für Datumsangaben, Datumszeiten und Zeitunterschiede. `?DateTimeClasses` `?difftime` `?Dates`, `?DateTimeClasses`, `?difftime` und im Abschnitt "Weitere `?difftime`" dieser Dokumente. Verwandte Dokumente: [Datums-](#) und [Datumszeit-Klassen](#).

Bemerkungen

Klassen

- [POSIXct](#)

Eine Datum-Uhrzeit-Klasse, `POSIXct`, speichert die Zeit als Sekunden seit UNIX-Epoche am `1970-01-01 00:00:00 UTC`. `Sys.Time()` ist das Format, das beim `Sys.Time()` der aktuellen Uhrzeit mit `Sys.Time()`.

- [POSIXlt](#)

In einer Datums- / Zeitklasse wird eine Liste mit Tag, Monat, Jahr, Stunde, Minute, Sekunde usw. gespeichert. Dies ist das von `strptime` zurückgegebene `strptime`.

- [Datum](#) Die einzige Datumsklasse speichert das Datum als Gleitkommazahl.

Datum / Uhrzeit-Format auswählen

`POSIXct` ist die einzige Option in der Welt von UNIX. Es ist schneller und benötigt weniger Speicher als `POSIXlt`.

```
origin = as.POSIXct("1970-01-01 00:00:00", format = "%Y-%m-%d %H:%M:%S", tz = "UTC")

origin
## [1] "1970-01-01 UTC"

origin + 47
## [1] "1970-01-01 00:00:47 UTC"

as.numeric(origin)      # At epoch
## 0

as.numeric(Sys.time()) # Right now (output as of July 21, 2016 at 11:47:37 EDT)
## 1469116057

posixlt = as.POSIXlt(Sys.time(), format = "%Y-%m-%d %H:%M:%S", tz = "America/Chicago")

# Conversion to POSIXct
```

```

posixct = as.POSIXct(posixlt)
posixct

# Accessing components
posixlt$sec    # Seconds 0-61
posixlt$min    # Minutes 0-59
posixlt$hour   # Hour 0-23
posixlt$mday   # Day of the Month 1-31
posixlt$mon    # Months after the first of the year 0-11
posixlt$year   # Years since 1900.

ct = as.POSIXct("2015-05-25")
lt = as.POSIXlt("2015-05-25")

object.size(ct)
# 520 bytes
object.size(lt)
# 1816 bytes

```

Spezialisierte Pakete

- Jederzeit
- data.table IDate und ITime
- schnelle Zeit
- [Lubridat](#)
- Nanotime

Examples

Aktuelles Datum und Uhrzeit

R kann auf das aktuelle Datum, die Uhrzeit und die Zeitzone zugreifen:

```

Sys.Date()           # Returns date as a Date object

## [1] "2016-07-21"

Sys.time()           # Returns date & time at current locale as a POSIXct object

## [1] "2016-07-21 10:04:39 CDT"

as.numeric(Sys.time()) # Seconds from UNIX Epoch (1970-01-01 00:00:00 UTC)

## [1] 1469113479

Sys.timezone()       # Time zone at current location

## [1] "Australia/Melbourne"

```

Verwenden Sie `OlsonNames()` , um die Zeitzonennamen in der Olson / IANA-Datenbank auf dem aktuellen System anzuzeigen:

```
str(OlsonNames())
## chr [1:589] "Africa/Abidjan" "Africa/Accra" "Africa/Addis_Ababa" "Africa/Algiers"
"Africa/Asmara" "Africa/Asmera" "Africa/Bamako" ...
```

Gehe zum Ende des Monats

Nehmen wir an, wir wollen bis zum letzten Tag des Monats gehen. Diese Funktion hilft Ihnen dabei:

```
eom <- function(x, p=as.POSIXlt(x)) as.Date(modifyList(p, list(mon=p$mon + 1, mday=0)))
```

Prüfung:

```
x <- seq(as.POSIXct("2000-12-10"), as.POSIXct("2001-05-10"), by="months")
> data.frame(before=x, after=eom(x))
  before      after
1 2000-12-10 2000-12-31
2 2001-01-10 2001-01-31
3 2001-02-10 2001-02-28
4 2001-03-10 2001-03-31
5 2001-04-10 2001-04-30
6 2001-05-10 2001-05-31
>
```

Verwenden eines Datums in einem String-Format:

```
> eom('2000-01-01')
[1] "2000-01-31"
```

Zum ersten Tag des Monats gehen

Angenommen, wir möchten zum ersten Tag eines bestimmten Monats gehen:

```
date <- as.Date("2017-01-20")

> as.POSIXlt(cut(date, "month"))
[1] "2017-01-01 EST"
```

Verschieben Sie ein Datum um mehrere Monate konsistent

Lassen Sie uns sagen, dass wir zu einem bestimmten Zeitpunkt eine verschiebende `num` Monate. Wir können die folgende Funktion definieren, die das Paket `mondate`:

```
moveNumOfMonths <- function(date, num) {
  as.Date(mondate(date) + num)
}
```

Es bewegt sich konsistent den Monatsteil des Datums und passt den Tag an, falls sich das Datum auf den letzten Tag des Monats bezieht.

Zum Beispiel:

Einen Monat zurück:

```
> moveNumOfMonths("2017-10-30",-1)
[1] "2017-09-30"
```

Zurück zwei Monate:

```
> moveNumOfMonths("2017-10-30",-2)
[1] "2017-08-30"
```

Zwei Monate weiterleiten:

```
> moveNumOfMonths("2017-02-28", 2)
[1] "2017-04-30"
```

Es bewegt sich zwei Monate nach dem letzten Tag im Februar, also dem letzten Tag im April.

Sehen wir uns an, wie es für Vorwärts- und Rückwärtsoperationen am letzten Tag des Monats funktioniert:

```
> moveNumOfMonths("2016-11-30", 2)
[1] "2017-01-31"
> moveNumOfMonths("2017-01-31", -2)
[1] "2016-11-30"
```

Da der November 30 Tage hat, erhalten wir bei der Rückwärtsoperation das gleiche Datum, aber:

```
> moveNumOfMonths("2017-01-30", -2)
[1] "2016-11-30"
> moveNumOfMonths("2016-11-30", 2)
[1] "2017-01-31"
```

Da der Januar 31 Tage hat, wird der letzte Tag des Monats Januar angezeigt.

Datum und Uhrzeit online lesen: <https://riptutorial.com/de/r/topic/1157/datum-und-uhrzeit>

Kapitel 26: Datum-Zeit-Klassen (POSIXct und POSIXlt)

Einführung

R enthält zwei Datums- / Zeitklassen - POSIXct und POSIXlt - siehe `?DateTimeClasses` .

Bemerkungen

Fallstricke

Bei POSIXct zeigt Mitternacht nur Datum und Zeitzone an, obwohl die volle Zeit noch gespeichert ist.

verwandte Themen

- [Datum und Uhrzeit](#)

Spezialisierte Pakete

- [Lubridat](#)

Examples

Datum-Uhrzeit-Objekte formatieren und drucken

```
# test date-time object
options(digits.secs = 3)
d = as.POSIXct("2016-08-30 14:18:30.58", tz = "UTC")

format(d,"%S") # 00-61 Second as integer
## [1] "30"

format(d,"%OS") # 00-60.99... Second as fractional
## [1] "30.579"

format(d,"%M") # 00-59 Minute
## [1] "18"

format(d,"%H") # 00-23 Hours
## [1] "14"

format(d,"%I") # 01-12 Hours
```

```
## [1] "02"

format(d,"%p") # AM/PM Indicator
## [1] "PM"

format(d,"%z") # Signed offset
## [1] "+0000"

format(d,"%Z") # Time Zone Abbreviation
## [1] "UTC"
```

?strptime Informationen zu den ?strptime sowie zu anderen Formaten.

Analysieren von Zeichenfolgen in Datum-Uhrzeit-Objekten

Die Funktionen zum Analysieren einer Zeichenfolge in POSIXct und POSIXlt verwenden ähnliche Parameter und geben ein ähnlich aussehendes Ergebnis zurück. Es gibt jedoch Unterschiede in der Speicherung des Datums und der Uhrzeit. siehe Anmerkungen."

```
as.POSIXct("11:38", # time string
           format = "%H:%M") # formatting string
## [1] "2016-07-21 11:38:00 CDT"
strptime("11:38", # identical, but makes a POSIXlt object
         format = "%H:%M")
## [1] "2016-07-21 11:38:00 CDT"

as.POSIXct("11 AM",
           format = "%I %p")
## [1] "2016-07-21 11:00:00 CDT"
```

Beachten Sie, dass Datum und Zeitzone berechnet werden.

```
as.POSIXct("11:38:22", # time string without timezone
           format = "%H:%M:%S",
           tz = "America/New_York") # set time zone
## [1] "2016-07-21 11:38:22 EDT"

as.POSIXct("2016-07-21 00:00:00",
           format = "%F %T") # shortcut tokens for "%Y-%m-%d" and "%H:%M:%S"
```

?strptime Informationen zu den ?strptime .

Anmerkungen

Fehlende Elemente

- Wenn kein Datumselement geliefert wird, wird das vom aktuellen Datum verwendet.
- Wenn kein Zeitelement geliefert wird, wird das von Mitternacht verwendet, dh 0s.
- Wenn weder im String noch im Parameter `tz` Zeitzone angegeben wird, wird die lokale Zeitzone verwendet.

Zeitzone

- Die akzeptierten Werte von `tz` hängen vom Standort ab.
 - CST wird mit "CST6CDT" oder "America/Chicago"
- Für unterstützte Orte und Zeitzone verwenden Sie:
 - In R: `OlsonNames()`
 - Alternativ versuchen Sie es in R: `system("cat $R_HOME/share/zoneinfo/zone.tab")`
- Diese Speicherorte werden von der [Internet Assigned Numbers Authority \(IANA\)](#) vergeben.
 - [Liste der Zeitzone der TZ-Datenbank \(Wikipedia\)](#)
 - [IANA TZ Data \(2016e\)](#)

Datum-Uhrzeit-Arithmetik

Um die Zeit zu addieren / subtrahieren, verwenden Sie `POSIXct`, da die Zeiten in Sekunden gespeichert werden

```
## adding/subtracting times - 60 seconds
as.POSIXct("2016-01-01") + 60
# [1] "2016-01-01 00:01:00 AEDT"

## adding 3 hours, 14 minutes, 15 seconds
as.POSIXct("2016-01-01") + ( (3 * 60 * 60) + (14 * 60) + 15)
# [1] "2016-01-01 03:14:15 AEDT"
```

Formal kann mit `as.difftime` Zeiträume angegeben werden, die einem Datums- oder Datumszeitobjekt hinzugefügt werden sollen. Z.B:

```
as.POSIXct("2016-01-01") +
  as.difftime(3, units="hours") +
  as.difftime(14, units="mins") +
  as.difftime(15, units="secs")
# [1] "2016-01-01 03:14:15 AEDT"
```

Um die Differenz zwischen Datum und `difftime()` verwenden Sie `difftime()` für Unterschiede in Sekunden, Minuten, Stunden, Tagen oder Wochen.

```
# using POSIXct objects
difftime(
  as.POSIXct("2016-01-01 12:00:00"),
  as.POSIXct("2016-01-01 11:59:59"),
  unit = "secs")
# Time difference of 1 secs
```

Um Sequenzen von Datum und `seq.POSIXt()` zu erzeugen, verwenden Sie `seq.POSIXt()` oder einfach `seq`.

Datum-Zeit-Klassen (`POSIXct` und `POSIXlt`) online lesen:

<https://riptutorial.com/de/r/topic/9027/datum-zeit-klassen--posixct-und-posixlt->

Kapitel 27: Debuggen

Examples

Browser verwenden

Die `browser` kann wie ein Haltepunkt verwendet werden: Die Codeausführung wird an dem Punkt angehalten, an dem sie aufgerufen wird. Der Benutzer kann dann Variablenwerte prüfen, beliebigen R-Code ausführen und den Code Zeile für Zeile durchgehen.

Sobald der `browser()` im Code gedrückt wird, wird der interaktive Interpreter gestartet. Jeder R-Code kann wie gewohnt ausgeführt werden. Außerdem sind folgende Befehle vorhanden:

Befehl	Bedeutung
c	Browser verlassen und Programm fortsetzen
f	Aktuelle Schleife oder Funktion beenden \
n	Step Over (nächste Anweisung auswerten, Funktionsaufrufe übergehen)
s	Step Into (nächste Anweisung auswerten, in Funktionsaufrufe einsteigen)
woher	Stack-Trace drucken
r	Rufen Sie den "Resume" -Start auf
Q	Browser verlassen und beenden

Zum Beispiel könnten wir ein Skript haben wie

```
toDebug <- function() {  
  a = 1  
  b = 2  
  
  browser()  
  
  for(i in 1:100) {  
    a = a * b  
  }  
}  
  
toDebug()
```

Beim Ausführen des obigen Skripts sehen wir zunächst etwas wie

```
Called from: toDebug  
Browser[1]>
```

Wir könnten dann mit der Eingabeaufforderung so interagieren,

```
Called from: toDebug
Browser[1]> a
[1] 1
Browser[1]> b
[1] 2
Browse[1]> n
debug at #7: for (i in 1:100) {
  a = a * b
}
Browse[2]> n
debug at #8: a = a * b
Browse[2]> a
[1] 1
Browse[2]> n
debug at #8: a = a * b
Browse[2]> a
[1] 2
Browse[2]> Q
```

`browser()` kann auch als Teil einer Funktionskette verwendet werden:

```
mtcars %>% group_by(cyl) %>% {browser() }
```

Debug verwenden

Sie können eine beliebige Funktion für das Debuggen mit `debug` einstellen.

```
debug(mean)
mean(1:3)
```

Alle nachfolgenden Aufrufe der Funktion werden in den Debugging-Modus versetzt. Sie können dieses Verhalten mit `undebug` deaktivieren.

```
undebug(mean)
mean(1:3)
```

Wenn Sie wissen, dass Sie den Debugging-Modus einer Funktion nur einmal `debugonce` möchten, sollten Sie `debugonce` .

```
debugonce(mean)
mean(1:3)
mean(1:3)
```

Debuggen online lesen: <https://riptutorial.com/de/r/topic/1695/debuggen>

Kapitel 28: Die Charakterklasse

Einführung

Zeichen sind das, was andere Sprachen "String-Vektoren" nennen.

Bemerkungen

verwandte Themen

Muster

- [Reguläre Ausdrücke \(Regex\)](#)
- [Musterabgleich und Ersatz](#)
- [Strsplit-Funktion](#)

Eingabe und Ausgabe

- [Zeichenfolgen lesen und schreiben](#)

Examples

Zwang

Um zu überprüfen, ob ein Wert ein Zeichen ist, verwenden Sie die Funktion `is.character()`. Um eine Variable in ein Zeichen zu zwingen, verwenden Sie die Funktion `as.character()`.

```
x <- "The quick brown fox jumps over the lazy dog"
class(x)
[1] "character"
is.character(x)
[1] TRUE
```

Beachten Sie, dass Numerik in Zeichen umgewandelt werden kann. Wenn Sie jedoch versuchen, ein Zeichen in eine Zahl umzuwandeln, kann dies zu `NA`.

```
as.numeric("2")
[1] 2
as.numeric("fox")
[1] NA
Warning message:
NAs introduced by coercion
```

Die Charakterklasse online lesen: <https://riptutorial.com/de/r/topic/9017/die-charakterklasse>

Kapitel 29: Die Date-Klasse

Bemerkungen

verwandte Themen

- [Datum und Uhrzeit](#)

Durcheinandergebrannte Notizen

- `Date` : Speichert die Zeit in Tagen seit der UNIX-Epoche am `1970-01-01` . mit negativen Werten für frühere Termine.
- Es wird als ganze Zahl dargestellt (in der internen Darstellung jedoch nicht erzwungen).
- Sie werden immer nach den Regeln des aktuellen Gregorianischen Kalenders gedruckt, auch wenn der Kalender vor langer Zeit nicht verwendet wurde.
- Zeitzonen werden nicht `POSIXct` , daher sollte die Zeit für `POSIXct` oder `POSIXlt` Objekte nicht `POSIXlt` werden.
- `sys.Date()` gibt ein Objekt der Klasse `Date`

Weitere Notizen

- [lubridate](#) , `s_ymd` , `mdy` usw. sind Alternativen zu `as.Date` , die auch für `Date` - Klasse analysieren; Siehe [Analysieren von Datumsangaben und Datumsangaben aus Zeichenfolgen mit Lubridat](#) .
- [data.table](#) , `s` experimentelle `iDate` Klasse abgeleitet ist von und ist meist austauschbar mit `Datum`, sondern wird als ganz Zahl anstelle von Doppel gespeichert.

Examples

Datumsangaben

Um `Dates` zu formatieren, verwenden wir die Funktion `format(date, format="%Y-%m-%d")` entweder mit dem `POSIXct` (von `as.POSIXct()`) oder `POSIXlt` (von `as.POSIXlt()`).

```
d = as.Date("2016-07-21") # Current Date Time Stamp

format(d, "%a")           # Abbreviated Weekday
## [1] "Thu"

format(d, "%A")           # Full Weekday
## [1] "Thursday"

format(d, "%b")           # Abbreviated Month
```

```
## [1] "Jul"

format(d,"%B")          # Full Month
## [1] "July"

format(d,"%m")          # 00-12 Month Format
## [1] "07"

format(d,"%d")          # 00-31 Day Format
## [1] "21"

format(d,"%e")          # 0-31 Day Format
## [1] "21"

format(d,"%Y")          # 00-99 Year
## [1] "16"

format(d,"%Y")          # Year with Century
## [1] "2016"
```

Weitere `?strptime` .

Termine

Um eine Variable zu einem Datum zu zwingen, verwenden Sie die Funktion `as.Date()` .

```
> x <- as.Date("2016-8-23")
> x
[1] "2016-08-23"
> class(x)
[1] "Date"
```

Mit der Funktion `as.Date()` können Sie ein `as.Date()` . Der Standardwert ist `%Y-%m-%d` , also Jahr-Monat-Tag.

```
> as.Date("23-8-2016", format="%d-%m-%Y") # To read in an European-style date
[1] "2016-08-23"
```

Die Formatzeichenfolge kann entweder in Anführungszeichen oder in Anführungszeichen gesetzt werden. : Daten werden in der Regel in einer Vielzahl von Formen, wie zum Ausdruck `"dm-yy"` oder `"dm-YYYY"` oder `"md-yy"` oder `"md-YYYY"` oder `"YYYY-md"` oder `"YYYY-dm"` . Diese Formate können auch ausgedrückt werden, indem `"-"` durch `"/"` . Ferner werden Daten auch in den Formen ausgedrückt, beispielsweise `"6. November 1986"` oder `"6. November 1986"` oder `"6. November 1986"` oder `"6. November 1986"` und so weiter. Die Funktion **`as.Date()`** akzeptiert alle derartigen Zeichenfolgen. Wenn wir das entsprechende Format der Zeichenfolge erwähnen, gibt sie immer das Datum in der Form `"YYYY-md"` .

Angenommen, wir haben eine Datumszeichenfolge `"9-6-1962"` im Format `"%d-%m-%Y"` .

```
#
# It tries to interprets the string as YYYY-m-d
#
> as.Date("9-6-1962")
```



```
[1] "0009-06-19"      #interprets as "%Y-%m-%d"
>
as.Date("9/6/1962")
[1] "0009-06-19"      #again interprets as "%Y-%m-%d"
>
# It has no problem in understanding, if the date is in form YYYY-m-d or YYYY/m/d
#
> as.Date("1962-6-9")
[1] "1962-06-09"      # no problem
> as.Date("1962/6/9")
[1] "1962-06-09"      # no problem
>
```

Durch die Angabe des korrekten Formats der Eingabezeichenfolge können wir die gewünschten Ergebnisse erzielen. Wir verwenden die folgenden Codes, um die Formate für die Funktion **as.Date ()** anzugeben.

Formatcode	Bedeutung
%d	Tag
%m	Monat
%y	Jahr zweistellig
%Y	Jahr 4-stellig
%b	abgekürzter Monat in 3 Zeichen
%B	vollständiger Name des Monats

Betrachten Sie das folgende Beispiel, indem Sie den **Formatparameter** angeben:

```
> as.Date("9-6-1962", format="%d-%m-%Y")
[1] "1962-06-09"
>
```

Der Parametername **Format** kann weggelassen werden.

```
> as.Date("9-6-1962", "%d-%m-%Y")
[1] "1962-06-09"
>
```

In manchen Fällen werden die Namen der Monate, die mit den ersten drei Zeichen abgekürzt wurden, beim Schreiben der Daten verwendet. In diesem Fall verwenden wir den Formatbezeichner **%b**.

```
> as.Date("6Nov1962", "%d%b%Y")
[1] "1962-11-06"
>
```

Beachten Sie, dass zwischen den Elementen in der Datumszeichenfolge keine '-' oder '/' oder Leerzeichen stehen. Die Formatzeichenfolge sollte genau mit dieser Eingabezeichenfolge übereinstimmen. Betrachten Sie das folgende Beispiel:

```
> as.Date("6 Nov, 1962", "%d %b, %Y")
[1] "1962-11-06"
>
```

Beachten Sie, dass in der Datumszeichenfolge ein Komma und somit auch in der Formatangabe ein Komma enthalten ist. Wenn in der Formatzeichenfolge kein Komma angegeben wird, führt dies zu einer `NA`. Ein Beispiel für die Verwendung des Formatbezeichners `%B` ist wie folgt:

```
> as.Date("October 12, 2016", "%B %d, %Y")
[1] "2016-10-12"
>
> as.Date("12 October, 2016", "%d %B, %Y")
[1] "2016-10-12"
>
```

`%y` Format `%y` ist systemspezifisch und sollte daher mit Vorsicht verwendet werden. Andere mit dieser Funktion verwendete Parameter sind **origin** und **tz** (Zeitzone).

Zeichenketten in Datumsobjekte analysieren

R enthält eine Date-Klasse, die mit `as.Date()` und eine Zeichenfolge oder einen Vektor von Zeichenfolgen verwendet. Wenn das Datum nicht im Datumsformat `YYYY-MM-DD` (ISO 8601) `strptime`, eine Formatierungszeichenfolge mit `strptime` Tokens.

```
as.Date('2016-08-01')      # in ISO format, so does not require formatting string
## [1] "2016-08-01"

as.Date('05/23/16', format = '%m/%d/%y')
## [1] "2016-05-23"

as.Date('March 23rd, 2016', '%B %drd, %Y')      # add separators and literals to format
## [1] "2016-03-23"

as.Date(' 2016-08-01  foo')      # leading whitespace and all trailing characters are ignored
## [1] "2016-08-01"

as.Date(c('2016-01-01', '2016-01-02'))
# [1] "2016-01-01" "2016-01-02"
```

Die Date-Klasse online lesen: <https://riptutorial.com/de/r/topic/9015/die-date-klasse>

Kapitel 30: Die logische Klasse

Einführung

Logisch ist ein Modus (und eine implizite Klasse) für Vektoren.

Bemerkungen

Stenografie

`TRUE`, `FALSE` und `NA` sind die einzigen Werte für logische Vektoren. und alle drei sind reservierte Wörter. `T` und `F` können in einer reinen R-Sitzung eine Abkürzung für `TRUE` und `FALSE`, aber weder `T` noch `F` sind reserviert. Daher kann die Zuweisung von Nicht-Standardwerten zu diesen Namen zu Problemen führen.

Examples

Logische Operatoren

Es gibt zwei Arten von logischen Operatoren: solche, die Vektoren beliebiger Länge annehmen und zurückgeben (elementweise Operatoren: `!`, `|`, `&`, `xor()`) und diejenigen, die nur das erste Element in jedem Argument (`&&`, `||`) auswerten. Die zweite Sortierung wird hauptsächlich als `cond` Argument für die `if` Funktion verwendet.

Logischer Operator	Bedeutung	Syntax
<code>!</code>	Nicht	<code>! x</code>
<code>&</code>	elementweise (vektoriert) und	<code>x und y</code>
<code>&&</code>	und (nur ein Element)	<code>x && y</code>
<code> </code>	elementweise (vektoriert) oder	<code>x y</code>
<code> </code>	oder (nur ein Element)	<code>x y</code>
<code>xor</code>	elementweise (vektoriert) exklusives ODER	<code>xor (x, y)</code>

Beachten Sie, dass der `||` Der Operator wertet die linke Bedingung aus und wenn die linke Bedingung `TRUE` ist, wird die rechte Seite niemals ausgewertet. Dies kann Zeit sparen, wenn die erste das Ergebnis einer komplexen Operation ist. Der Operator `&&` gibt ebenfalls `FALSE` zurück, ohne das zweite Argument auszuwerten, wenn das erste Element des ersten Arguments `FALSE` ist.

```
> x <- 5
> x > 6 || stop("X is too small")
Error: X is too small
> x > 3 || stop("X is too small")
[1] TRUE
```

Um zu prüfen, ob ein Wert ein logischer Wert ist, können Sie die Funktion `is.logical()` verwenden.

Zwang

Um eine Variable in eine logische Funktion zu zwingen, verwenden Sie die Funktion `as.logical()` .

```
> x <- 2
> z <- x > 4
> z
[1] FALSE
> class(x)
[1] "numeric"
> as.logical(2)
[1] TRUE
```

Beim Anwenden von `as.numeric()` auf ein Logisches wird ein Double zurückgegeben. `NA` ist ein logischer Wert, und ein logischer Operator mit einer `NA` gibt `NA` wenn das Ergebnis mehrdeutig ist.

Interpretation von NAs

Weitere Informationen finden Sie unter [Fehlende Werte](#) .

```
> TRUE & NA
[1] NA
> FALSE & NA
[1] FALSE
> TRUE || NA
[1] TRUE
> FALSE || NA
[1] NA
```

Die logische Klasse online lesen: <https://riptutorial.com/de/r/topic/9016/die-logische-klasse>

Kapitel 31: dplyr

Bemerkungen

dplyr ist eine Iteration von plyr, die flexible "Verb" -basierte Funktionen zur Bearbeitung von Daten in R bereitstellt. Die neueste Version von dplyr kann von CRAN heruntergeladen werden

```
install.packages("dplyr")
```

Das Schlüsselobjekt in dplyr ist eine Tabelle, eine Darstellung einer tabellarischen Datenstruktur. Derzeit unterstützt dplyr (Version 0.5.0):

- Datenrahmen
- Datentabellen
- SQLite
- PostgreSQL / Redshift
- MySQL / MariaDB
- Große frage
- MonetDB
- Datenwürfel mit Arrays (teilweise Implementierung)

Examples

dplyr's einzelne Tabellenverben

dplyr führt eine Grammatik der Datenmanipulation in R. Es bietet eine konsistente Schnittstelle, um mit Daten zu arbeiten, egal wo sie gespeichert sind: `data.frame`, `data.table` oder einer `database`. Die wichtigsten Teile von `dplyr` werden mit **Rcpp geschrieben**, wodurch das Arbeiten mit In-Memory-Daten sehr schnell ist.

`dplyr` Philosophie ist es, kleine Funktionen zu haben, die eine Sache gut machen. Die fünf einfachen Funktionen (`filter`, `arrange`, `select`, `mutate` und `summarise`) können verwendet werden, um neue Möglichkeiten zur Beschreibung von Daten aufzuzeigen. In Kombination mit `group_by` können diese Funktionen zum Berechnen von gruppenweisen Zusammenfassungsstatistiken verwendet werden.

Syntax-Gemeinsamkeiten

Alle diese Funktionen haben eine ähnliche Syntax:

- Das erste Argument für alle diese Funktionen ist immer ein Datenrahmen
- Spalten können direkt mit bloßen Variablennamen referenziert werden (dh ohne `$`).
- Diese Funktionen ändern die Originaldaten nicht selbst, dh sie haben keine Nebenwirkungen. Daher sollten die Ergebnisse immer in einem Objekt gespeichert werden.

Wir werden das integrierte `mtcars`- Dataset verwenden, um die einzelnen Tabellenverben von `dplyr` zu `dplyr` . Bevor Sie den Typ der `mtcars` in `tbl_df` (da der Druck dadurch sauberer wird), fügen Sie die `rownames` des Datasets als Spalte hinzu, indem `rownames_to_column` Funktion `rownames_to_column` aus dem `tibble`- Paket verwenden.

```
library(dplyr) # This documentation was written using version 0.5.0

mtcars_tbl <- as_data_frame(tibble::rownames_to_column(mtcars, "cars"))

# examine the structure of data
head(mtcars_tbl)

# A tibble: 6 x 12
#       cars      mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
#       <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1      Mazda RX4    21.0     6   160   110   3.90  2.620  16.46     0     1     4     4
#2      Mazda RX4 Wag  21.0     6   160   110   3.90  2.875  17.02     0     1     4     4
#3      Datsun 710    22.8     4   108    93   3.85  2.320  18.61     1     1     4     1
#4    Hornet 4 Drive  21.4     6   258   110   3.08  3.215  19.44     1     0     3     1
#5 Hornet Sportabout  18.7     8   360   175   3.15  3.440  17.02     0     0     3     2
#6      Valiant     18.1     6   225   105   2.76  3.460  20.22     1     0     3     1
```

Filter

`filter` hilft bei der Untermenge von Zeilen, die bestimmten Kriterien entsprechen. Das erste Argument ist der Name von `data.frame` und das zweite (und nachfolgende) Argument sind die Kriterien, nach denen die Daten gefiltert werden (diese Kriterien sollten entweder `TRUE` oder `FALSE`).

Teilmenge alle Autos, die 4 Zylinder haben - `cyl` :

```
filter(mtcars_tbl, cyl == 4)

# A tibble: 11 x 12
#       cars      mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
#       <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1    Datsun 710    22.8     4  108.0    93   3.85  2.320  18.61     1     1     4     1
#2    Merc 240D    24.4     4  146.7    62   3.69  3.190  20.00     1     0     4     2
#3    Merc 230     22.8     4  140.8    95   3.92  3.150  22.90     1     0     4     2
#4    Fiat 128     32.4     4   78.7    66   4.08  2.200  19.47     1     1     4     1
#5   Honda Civic    30.4     4   75.7    52   4.93  1.615  18.52     1     1     4     2
# ... with 6 more rows
```

Wir können mehrere durch Kommas getrennte Kriterien übergeben. Die Wagen mit 4 oder 6 Zylindern - `cyl` und 5 Gängen - müssen `gear` :

```
filter(mtcars_tbl, cyl == 4 | cyl == 6, gear == 5)

# A tibble: 3 x 12
#       cars      mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
#       <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1 Porsche 914-2    26.0     4  120.3    91   4.43  2.140  16.7     0     1     5     2
#2 Lotus Europa    30.4     4   95.1   113   3.77  1.513  16.9     1     1     5     2
#3 Ferrari Dino    19.7     6  145.0   175   3.62  2.770  15.5     0     1     5     6
```

`filter` wählt Zeilen nach Kriterien aus, um Zeilen nach Position auszuwählen, verwenden Sie `slice`. `slice` `data.frame` nur zwei Argumente: Das erste ist ein `data.frame` und das zweite ist ein ganzzahliger `data.frame`.

So wählen Sie die Zeilen 6 bis 9 aus:

```
slice(mtcars_tbl, 6:9)

# A tibble: 4 x 12
#   cars      mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
#   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1  Valiant  18.1     6  225.0   105  2.76  3.46  20.22     1   0     3     1
#2 Duster 360  14.3     8  360.0   245  3.21  3.57  15.84     0   0     3     4
#3  Merc 240D  24.4     4  146.7    62  3.69  3.19  20.00     1   0     4     2
#4  Merc 230  22.8     4  140.8    95  3.92  3.15  22.90     1   0     4     2
```

Oder:

```
slice(mtcars_tbl, -c(1:5, 10:n()))
```

Dies ergibt dieselbe Ausgabe wie `slice(mtcars_tbl, 6:9)`

`n()` für die Anzahl der Beobachtungen in der aktuellen Gruppe

ordnen

`arrange` werden die Daten nach einer oder mehreren angegebenen Variablen sortiert. Genau wie beim vorherigen Verb (und allen anderen Funktionen in `dplyr`) ist das erste Argument ein `data.frame`, und die folgenden Argumente werden zum Sortieren der Daten verwendet. Wenn mehr als eine Variable übergeben wird, werden die Daten zuerst nach der ersten und dann nach der zweiten Variablen usw. sortiert.

So bestellen Sie die Daten mit *PS* - `hp`

```
arrange(mtcars_tbl, hp)

# A tibble: 32 x 12
#   cars      mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
#   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1  Honda Civic  30.4     4   75.7    52  4.93  1.615  18.52     1   1     4     2
#2  Merc 240D  24.4     4  146.7    62  3.69  3.190  20.00     1   0     4     2
#3 Toyota Corolla 33.9     4   71.1    65  4.22  1.835  19.90     1   1     4     1
#4   Fiat 128  32.4     4   78.7    66  4.08  2.200  19.47     1   1     4     1
#5   Fiat X1-9  27.3     4   79.0    66  4.08  1.935  18.90     1   1     4     1
#6  Porsche 914-2 26.0     4  120.3    91  4.43  2.140  16.70     0   1     5     2
# ... with 26 more rows
```

Um die Daten nach *Meilen pro Gallone* (`mpg` in absteigender Reihenfolge zu `arrange`, gefolgt von der *Anzahl der Zylinder* - `cyl` :

```
arrange(mtcars_tbl, desc(mpg), cyl)
```

```
# A tibble: 32 x 12
#       cars      mpg    cyl  disp    hp  drat    wt   qsec    vs    am  gear  carb
#       <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1 Toyota Corolla  33.9     4  71.1    65  4.22  1.835  19.90     1     1     4     1
#2   Fiat 128      32.4     4  78.7    66  4.08  2.200  19.47     1     1     4     1
#3   Honda Civic  30.4     4  75.7    52  4.93  1.615  18.52     1     1     4     2
#4   Lotus Europa  30.4     4  95.1   113  3.77  1.513  16.90     1     1     5     2
#5   Fiat X1-9     27.3     4  79.0    66  4.08  1.935  18.90     1     1     4     1
#6  Porsche 914-2  26.0     4 120.3    91  4.43  2.140  16.70     0     1     5     2
# ... with 26 more rows
```

wählen

`select` wird verwendet, um nur eine Teilmenge von Variablen auszuwählen. Um nur `mpg`, `disp`, `wt`, `qsec` und `vs` aus `mtcars_tbl`:

```
select(mtcars_tbl, mpg, disp, wt, qsec, vs)
```

```
# A tibble: 32 x 5
#       mpg  disp    wt  qsec    vs
#       <dbl> <dbl> <dbl> <dbl> <dbl>
#1  21.0 160.0  2.620 16.46     0
#2  21.0 160.0  2.875 17.02     0
#3  22.8 108.0  2.320 18.61     1
#4  21.4 258.0  3.215 19.44     1
#5  18.7 360.0  3.440 17.02     0
#6  18.1 225.0  3.460 20.22     1
# ... with 26 more rows
```

: Notation kann verwendet werden, um aufeinanderfolgende Spalten auszuwählen. So wählen Sie Spalten von `cars` über `disp` und `vs` durch `carb`:

```
select(mtcars_tbl, cars:disp, vs:carb)
```

```
# A tibble: 32 x 8
#       cars      mpg    cyl  disp    vs    am  gear  carb
#       <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1   Mazda RX4  21.0     6 160.0     0     1     4     4
#2   Mazda RX4 Wag 21.0     6 160.0     0     1     4     4
#3   Datsun 710  22.8     4 108.0     1     1     4     1
#4  Hornet 4 Drive 21.4     6 258.0     1     0     3     1
#5 Hornet Sportabout 18.7     8 360.0     0     0     3     2
#6    Valiant    18.1     6 225.0     1     0     3     1
# ... with 26 more rows
```

oder `select(mtcars_tbl, -(hp:qsec))`

Bei Datensätzen, die mehrere Spalten enthalten, kann es mühsam sein, mehrere Spalten nach Namen auszuwählen. Um das Leben zu erleichtern, gibt es eine Reihe von `starts_with()` (z. B. `starts_with()`, `ends_with()`, `contains()`, `matches()`, `num_range()`, `one_of()` und `everything()`), die in `select`. Weitere `?select_helpers` zur Verwendung finden Sie unter `?select_helpers` und `?select`.

Hinweis : Wenn Sie in `select()` direkt auf Spalten verweisen, verwenden wir bloße Spaltennamen. In Bezug auf Spalten in Hilfsfunktionen sollten jedoch Anführungszeichen verwendet werden.

Umbenennen von Spalten während der Auswahl:

```
select(mtcars_tbl, cylinders = cyl, displacement = disp)

# A tibble: 32 x 2
#   cylinders displacement
#   <dbl>         <dbl>
#1         6         160.0
#2         6         160.0
#3         4         108.0
#4         6         258.0
#5         8         360.0
#6         6         225.0
# ... with 26 more rows
```

Wie erwartet werden alle anderen Variablen gelöscht.

Um Spalten umzubenennen, ohne andere Variablen zu löschen, verwenden Sie `rename` :

```
rename(mtcars_tbl, cylinders = cyl, displacement = disp)

# A tibble: 32 x 12
#   cars      mpg cylinders displacement    hp  drat    wt  qsec    vs
#   <chr> <dbl>    <dbl>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1  Mazda RX4   21.0         6         160.0   110  3.90  2.620  16.46    0
#2  Mazda RX4 Wag 21.0         6         160.0   110  3.90  2.875  17.02    0
#3   Datsun 710  22.8         4         108.0    93  3.85  2.320  18.61    1
#4  Hornet 4 Drive 21.4         6         258.0   110  3.08  3.215  19.44    1
#5  Hornet Sportabout 18.7         8         360.0   175  3.15  3.440  17.02    0
#6    Valiant   18.1         6         225.0   105  2.76  3.460  20.22    1
# ... with 26 more rows, and 3 more variables: am <dbl>, gear <dbl>, carb <dbl>
```

mutieren

`mutate` kann verwendet werden, um den Daten neue Spalten hinzuzufügen. Wie alle anderen Funktionen in `dplyr` fügt `dplyr` die neu erstellten Spalten nicht zu den ursprünglichen Daten hinzu. Spalten werden am Ende des `data.frame` .

```
mutate(mtcars_tbl, weight_ton = wt/2, weight_pounds = weight_ton * 2000)

# A tibble: 32 x 14
#   cars      mpg  cyl disp    hp  drat    wt  qsec    vs  am gear carb
#   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1  Mazda RX4   21.0     6  160.0   110  3.90  2.620  16.46    0    1    4    4
#   weight_ton weight_pounds
#1  1.3100      2620
#2  Mazda RX4 Wag 21.0     6  160.0   110  3.90  2.875  17.02    0    1    4    4
#   weight_ton weight_pounds
#2  1.4375      2875
#3   Datsun 710  22.8     4  108.0    93  3.85  2.320  18.61    1    1    4    1
#   weight_ton weight_pounds
#3  1.1600      2320
```

```
#4      Hornet 4 Drive  21.4      6 258.0    110  3.08 3.215 19.44      1      0      3      1
1.6075      3215
#5 Hornet Sportabout  18.7      8 360.0    175  3.15 3.440 17.02      0      0      3      2
1.7200      3440
#6      Valiant  18.1      6 225.0    105  2.76 3.460 20.22      1      0      3      1
1.7300      3460
# ... with 26 more rows
```

Beachten Sie die Verwendung von `weight_ton` beim Erstellen von `weight_pounds`. Im Gegensatz zu Base R, `mutate` ermöglicht es uns, auf den Spalten zu verweisen, die wir gerade erstellt, um für einen nachfolgenden Vorgang verwendet werden.

Um nur die neu erstellten Spalten beizubehalten, verwenden Sie `transmute` statt `mutate`:

```
transmute(mtcars_tbl, weight_ton = wt/2, weight_pounds = weight_ton * 2000)

# A tibble: 32 x 2
#   weight_ton weight_pounds
#   <dbl>      <dbl>
#1     1.3100         2620
#2     1.4375         2875
#3     1.1600         2320
#4     1.6075         3215
#5     1.7200         3440
#6     1.7300         3460
# ... with 26 more rows
```

zusammenfassen

`summarise` berechnet die Statistikstatistik von Variablen, indem mehrere Werte auf einen einzigen Wert reduziert werden. Es kann mehrere Statistiken berechnen und wir können diese Zusammenfassungsspalten in derselben Anweisung benennen.

So berechnen Sie den *Mittelwert* und die *Standardabweichung* von `mpg` und `disp` aller Autos im Datensatz:

```
summarise(mtcars_tbl, mean_mpg = mean(mpg), sd_mpg = sd(mpg),
           mean_disp = mean(disp), sd_disp = sd(disp))

# A tibble: 1 x 4
#   mean_mpg sd_mpg mean_disp sd_disp
#   <dbl>   <dbl>   <dbl>   <dbl>
#1  20.09062 6.026948  230.7219 123.9387
```

gruppiere nach

`group_by` können gruppenweise Operationen an Daten ausgeführt werden. Wenn die oben definierten Verben auf diese gruppierten Daten angewendet werden, werden sie automatisch auf jede Gruppe separat angewendet.

mean und sd von mpg nach cyl :

```
by_cyl <- group_by(mtcars_tbl, cyl)
summarise(by_cyl, mean_mpg = mean(mpg), sd_mpg = sd(mpg))

# A tibble: 3 x 3
#   cyl mean_mpg sd_mpg
#   <dbl>   <dbl>   <dbl>
#1     4  26.66364  4.509828
#2     6  19.74286  1.453567
#3     8  15.10000  2.560048
```

Alles zusammenfügen

Wir wählen die Spalten von `cars` über `hp` und `gear`, ordnen die Zeilen nach `cyl` und vom höchsten bis zum niedrigsten `mpg`, gruppieren die Daten nach `gear` und schließlich haben nur diese Autos die `mpg > 20` und `hp > 75`

```
selected <- select(mtcars_tbl, cars:hp, gear)
ordered <- arrange(selected, cyl, desc(mpg))
by_cyl <- group_by(ordered, gear)
filter(by_cyl, mpg > 20, hp > 75)

Source: local data frame [9 x 6]
Groups: gear [3]

#   cars      mpg   cyl  disp    hp  gear
#   <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
#1 Lotus Europa  30.4     4   95.1   113     5
#2 Porsche 914-2  26.0     4  120.3    91     5
#3 Datsun 710    22.8     4  108.0    93     4
#4 Merc 230     22.8     4  140.8    95     4
#5 Toyota Corona 21.5     4  120.1    97     3
# ... with 4 more rows
```

Vielleicht sind wir nicht an den Zwischenergebnissen interessiert, wir können das gleiche Ergebnis wie oben erzielen, indem wir die Funktionsaufrufe umschließen:

```
filter(
  group_by(
    arrange(
      select(
        mtcars_tbl, cars:hp
      ), cyl, desc(mpg)
    ), cyl
  ), mpg > 20, hp > 75
)
```

Dies kann etwas schwierig zu lesen sein. `dplyr` Vorgänge können also mit dem Operator `pipe` `%>%` verkettet werden. Der obige Code übersetzt nach:

```
mtcars_tbl %>%
```

```
select(cars:hp) %>%
  arrange(cyl, desc(mpg)) %>%
  group_by(cyl) %>%
  filter(mpg > 20, hp > 75)
```

fassen Sie mehrere Spalten zusammen

dplyr **bietet** `summarise_all()` , um Funktionen auf alle (nicht gruppierenden) Spalten anzuwenden.

So ermitteln Sie die Anzahl der unterschiedlichen Werte für jede Spalte:

```
mtcars_tbl %>%
  summarise_all(n_distinct)

# A tibble: 1 x 12
#   cars   mpg   cyl  disp    hp  drat    wt  qsec    vs  am  gear  carb
#   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
#1    32    25     3    27    22    22    29    30     2     2     3     6
```

So ermitteln Sie die Anzahl der unterschiedlichen Werte für jede Spalte nach `cyl` :

```
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_all(n_distinct)

# A tibble: 3 x 12
#   cyl   cars   mpg  disp    hp  drat    wt  qsec    vs  am  gear  carb
#   <dbl> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
#1     4    11     9    11    10    10    11    11     2     2     3     2
#2     6     7     6     5     4     5     6     7     2     2     3     3
#3     8    14    12    11     9    11    13    14     1     2     2     4
```

Beachten Sie, dass wir nur die `group_by` Anweisung hinzufügen `group_by` und der Rest des Codes derselbe ist. Die Ausgabe besteht jetzt aus drei Zeilen - einer für jeden eindeutigen Wert von `cyl` .

Um `summarise` spezifische mehr Spalten, verwenden `summarise_at`

```
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_at(c("mpg", "disp", "hp"), mean)

# A tibble: 3 x 4
#   cyl    mpg    disp    hp
#   <dbl> <dbl> <dbl> <dbl>
#1     4 26.66364 105.1364 82.63636
#2     6 19.74286 183.3143 122.28571
#3     8 15.10000 353.1000 209.21429
```

`helper ( ?select_helpers )` anstelle von Spaltennamen verwendet werden , um bestimmte Spalten auswählen

Um mehrere Funktionen anzuwenden, übergeben Sie die Funktionsnamen entweder als

Zeichenvektor:

```
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_at(c("mpg", "disp", "hp"),
               c("mean", "sd"))
```

oder wickeln Sie sie in `funcs` :

```
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_at(c("mpg", "disp", "hp"),
               funcs(mean, sd))

# A tibble: 3 x 7
#   cyl mpg_mean disp_mean hp_mean mpg_sd disp_sd hp_sd
#   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
#1     4 26.66364 105.1364 82.63636 4.509828 26.87159 20.93453
#2     6 19.74286 183.3143 122.28571 1.453567 41.56246 24.26049
#3     8 15.10000 353.1000 209.21429 2.560048 67.77132 50.97689
```

Spaltennamen werden jetzt mit Funktionsnamen versehen, um sie voneinander zu trennen. Um dies zu ändern, übergeben Sie den Namen, der mit der Funktion angehängt werden soll:

```
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_at(c("mpg", "disp", "hp"),
               c(Mean = "mean", SD = "sd"))

mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_at(c("mpg", "disp", "hp"),
               funcs(Mean = mean, SD = sd))

# A tibble: 3 x 7
#   cyl mpg_Mean disp_Mean hp_Mean mpg_SD disp_SD hp_SD
#   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
#1     4 26.66364 105.1364 82.63636 4.509828 26.87159 20.93453
#2     6 19.74286 183.3143 122.28571 1.453567 41.56246 24.26049
#3     8 15.10000 353.1000 209.21429 2.560048 67.77132 50.97689
```

Um bedingte Spalten auszuwählen, verwenden Sie `summarise_if` :

Nehmen Sie die `mean` aller Spalten, die `numeric` von gruppiert `cyl` :

```
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise_if(is.numeric, mean)

# A tibble: 3 x 11
#   cyl   mpg   disp  hp  drat    wt  qsec
#   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#1     4 26.66364 105.1364 82.63636 4.070909 2.285727 19.13727
#2     6 19.74286 183.3143 122.28571 3.585714 3.117143 17.97714
#3     8 15.10000 353.1000 209.21429 3.229286 3.999214 16.77214
```

```
# ... with 4 more variables: vs <dbl>, am <dbl>, gear <dbl>,  
#   carb <dbl>
```

Einige Variablen sind jedoch diskret und der `mean` dieser Variablen ist nicht sinnvoll.

Den `mean` von nur kontinuierlichen Variablen nach `cyl` :

```
mtcars_tbl %>%  
  group_by(cyl) %>%  
  summarise_if(function(x) is.numeric(x) & n_distinct(x) > 6, mean)  
  
# A tibble: 3 x 7  
#   cyl      mpg      disp      hp      drat      wt      qsec  
#   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  
#1     4 26.66364 105.1364  82.63636 4.070909 2.285727 19.13727  
#2     6 19.74286 183.3143 122.28571 3.585714 3.117143 17.97714  
#3     8 15.10000 353.1000 209.21429 3.229286 3.999214 16.77214
```

Teilmengenbeobachtung (Zeilen)

`dplyr::filter()` - Wählen Sie eine Teilmenge von Zeilen in einem `dplyr::filter()` , die logische Kriterien erfüllen:

```
dplyr::filter(iris, Sepal.Length > 7)  
#   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
#   1           7.1         3.0          5.9         2.1 virginica  
#   2           7.6         3.0          6.6         2.1 virginica  
#   3           7.3         2.9          6.3         1.8 virginica  
#   4           7.2         3.6          6.1         2.5 virginica  
#   5           7.7         3.8          6.7         2.2 virginica  
#   6           7.7         2.6          6.9         2.3 virginica  
#   7           7.7         2.8          6.7         2.0 virginica  
#   8           7.2         3.2          6.0         1.8 virginica  
#   9           7.2         3.0          5.8         1.6 virginica  
#  10           7.4         2.8          6.1         1.9 virginica  
#  11           7.9         3.8          6.4         2.0 virginica  
#  12           7.7         3.0          6.1         2.3 virginica
```

`dplyr::distinct()` - Doppelte Zeilen entfernen:

```
distinct(iris, Sepal.Length, .keep_all = TRUE)  
#   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
#   1           5.1         3.5          1.4         0.2 setosa  
#   2           4.9         3.0          1.4         0.2 setosa  
#   3           4.7         3.2          1.3         0.2 setosa  
#   4           4.6         3.1          1.5         0.2 setosa  
#   5           5.0         3.6          1.4         0.2 setosa  
#   6           5.4         3.9          1.7         0.4 setosa  
#   7           4.4         2.9          1.4         0.2 setosa  
#   8           4.8         3.4          1.6         0.2 setosa  
#   9           4.3         3.0          1.1         0.1 setosa  
#  10           5.8         4.0          1.2         0.2 setosa  
#  11           5.7         4.4          1.5         0.4 setosa
```

#	12	5.2	3.5	1.5	0.2	setosa
#	13	5.5	4.2	1.4	0.2	setosa
#	14	4.5	2.3	1.3	0.3	setosa
#	15	5.3	3.7	1.5	0.2	setosa
#	16	7.0	3.2	4.7	1.4	versicolor
#	17	6.4	3.2	4.5	1.5	versicolor
#	18	6.9	3.1	4.9	1.5	versicolor
#	19	6.5	2.8	4.6	1.5	versicolor
#	20	6.3	3.3	4.7	1.6	versicolor
#	21	6.6	2.9	4.6	1.3	versicolor
#	22	5.9	3.0	4.2	1.5	versicolor
#	23	6.0	2.2	4.0	1.0	versicolor
#	24	6.1	2.9	4.7	1.4	versicolor
#	25	5.6	2.9	3.6	1.3	versicolor
#	26	6.7	3.1	4.4	1.4	versicolor
#	27	6.2	2.2	4.5	1.5	versicolor
#	28	6.8	2.8	4.8	1.4	versicolor
#	29	7.1	3.0	5.9	2.1	virginica
#	30	7.6	3.0	6.6	2.1	virginica
#	31	7.3	2.9	6.3	1.8	virginica
#	32	7.2	3.6	6.1	2.5	virginica
#	33	7.7	3.8	6.7	2.2	virginica
#	34	7.4	2.8	6.1	1.9	virginica
#	35	7.9	3.8	6.4	2.0	virginica

Aggregation mit dem Operator %>% (Pipe)

Der Pipe- **Operator** (%>%) kann in Kombination mit `dplyr` Funktionen verwendet werden. In diesem Beispiel verwenden wir das `mtcars` Dataset (weitere Informationen finden Sie in der `help("mtcars")`), um zu zeigen, wie ein Datenrahmen zusammengefasst wird und um den Daten Variablen hinzuzufügen.

```
library(dplyr)
library(magrittr)
df <- mtcars
df$cars <- rownames(df) #just add the cars names to the df
df <- df[,c(ncol(df),1:(ncol(df)-1))] # and place the names in the first column
```

1. Summieren Sie die Daten

Für die Berechnung von Statistiken verwenden wir die Funktionen `summarize` und die entsprechenden Funktionen. In diesem Fall wird `n()` verwendet, um die Anzahl der Fälle zu zählen.

```
df %>%
  summarize(count=n(),mean_mpg = mean(mpg, na.rm = TRUE),
            min_weight = min(wt),max_weight = max(wt))

# count mean_mpg min_weight max_weight
#1      32 20.09062      1.513      5.424
```

2. Berechnen Sie Statistiken nach Gruppe

Es ist möglich, die Statistiken nach Datengruppen zu berechnen. In diesem Fall nach *Anzahl der*

Zylinder und Anzahl der Vorwärtsgänge

```
df %>%
  group_by(cyl, gear) %>%
  summarize(count=n(), mean_mpg = mean(mpg, na.rm = TRUE),
            min_weight = min(wt), max_weight = max(wt))

# Source: local data frame [8 x 6]
# Groups: cyl [?]
#
#   cyl  gear count mean_mpg min_weight max_weight
#   <dbl> <dbl> <int>    <dbl>    <dbl>    <dbl>
#1     4     3     1    21.500     2.465     2.465
#2     4     4     8    26.925     1.615     3.190
#3     4     5     2    28.200     1.513     2.140
#4     6     3     2    19.750     3.215     3.460
#5     6     4     4    19.750     2.620     3.440
#6     6     5     1    19.700     2.770     2.770
#7     8     3    12    15.050     3.435     5.424
#8     8     5     2    15.400     3.170     3.570
```

Beispiele für NSE- und String-Variablen in dplyr

`dplyr` verwendet Non-Standard Evaluation (NSE), weshalb wir normalerweise die Variablennamen ohne Anführungszeichen verwenden können. Manchmal müssen wir während der Daten-Pipeline jedoch unsere Variablennamen aus anderen Quellen beziehen, z. B. aus einem glänzenden Auswahlfeld. Bei Funktionen wie `select` können Sie einfach `select_`, um eine String-Variable zur Auswahl zu verwenden

```
variable1 <- "Sepal.Length"
variable2 <- "Sepal.Width"
iris %>%
  select_(variable1, variable2) %>%
  head(n=5)
#   Sepal.Length Sepal.Width
# 1           5.1          3.5
# 2           4.9          3.0
# 3           4.7          3.2
# 4           4.6          3.1
# 5           5.0          3.6
```

Wenn Sie jedoch andere Funktionen wie Zusammenfassen oder Filtern verwenden `interp` müssen Sie die `interp` Funktion aus dem `lazyeval` Paket verwenden

```
variable1 <- "Sepal.Length"
variable2 <- "Sepal.Width"
variable3 <- "Species"
iris %>%
  select_(variable1, variable2, variable3) %>%
  group_by_(variable3) %>%
  summarize_(mean1 = lazyeval::interp(~mean(var), var = as.name(variable1)), mean2 =
    lazyeval::interp(~mean(var), var = as.name(variable2)))
#   Species mean1 mean2
#   <fctr> <dbl> <dbl>
# 1 setosa 5.006 3.428
```



```
# 2 versicolor 5.936 2.770  
# 3 virginica 6.588 2.974
```

dplyr online lesen: <https://riptutorial.com/de/r/topic/4250/dplyr>

Kapitel 32: Durchführen eines Permutationstests

Examples

Eine ziemlich allgemeine Funktion

Wir werden den integrierten [Zahnwachstumsdatensatz verwenden](#). Wir sind daran interessiert, ob es einen statistisch signifikanten Unterschied im Zahnwachstum gibt, wenn den Meerschweinchen Vitamin C gegen Orangensaft gegeben wird.

Hier ist das vollständige Beispiel:

```
teethVC = ToothGrowth[ToothGrowth$supp == 'VC',]
teethOJ = ToothGrowth[ToothGrowth$supp == 'OJ',]

permutationTest = function(vectorA, vectorB, testStat){
  N = 10^5
  fullSet = c(vectorA, vectorB)
  lengthA = length(vectorA)
  lengthB = length(vectorB)
  trials <- replicate(N,
                      {index <- sample(lengthB + lengthA, size = lengthA, replace = FALSE)
                       testStat((fullSet[index]), fullSet[-index]) } )

  trials
}
vec1 =teethVC$len;
vec2 =teethOJ$len;
subtractMeans = function(a, b){ return (mean(a) - mean(b))}
result = permutationTest(vec1, vec2, subtractMeans)
observedMeanDifference = subtractMeans(vec1, vec2)
result = c(result, observedMeanDifference)
hist(result)
abline(v=observedMeanDifference, col = "blue")
pValue = 2*mean(result <= (observedMeanDifference))
pValue
```

Nachdem wir die CSV eingelesen haben, definieren wir die Funktion

```
permutationTest = function(vectorA, vectorB, testStat){
  N = 10^5
  fullSet = c(vectorA, vectorB)
  lengthA = length(vectorA)
  lengthB = length(vectorB)
  trials <- replicate(N,
                      {index <- sample(lengthB + lengthA, size = lengthA, replace = FALSE)
                       testStat((fullSet[index]), fullSet[-index]) } )

  trials
}
```

Diese Funktion nimmt zwei Vektoren, mischt ihren Inhalt zusammen und führt dann die Funktion

`testStat` für die gemischten Vektoren aus. Das Ergebnis von `teststat` wird zu den `trials` hinzugefügt. Dies ist der Rückgabewert.

Das macht $N = 10^5$ mal. Beachten Sie, dass der Wert N durchaus ein Parameter für die Funktion gewesen sein könnte.

Dadurch bleiben neue Daten, `trials`, die Mittel, die sich ergeben könnten, wenn wirklich keine Beziehung zwischen den beiden Variablen besteht.

Nun definieren Sie unsere Teststatistik:

```
subtractMeans = function(a, b){ return (mean(a) - mean(b)) }
```

Führen Sie den Test durch:

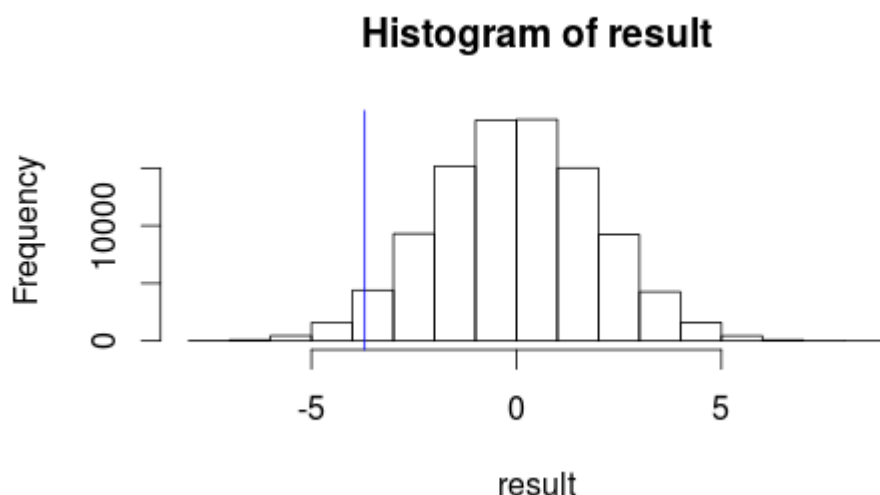
```
result = permutationTest(vec1, vec2, subtractMeans)
```

Berechnen Sie unsere tatsächlich beobachtete mittlere Differenz:

```
observedMeanDifference = subtractMeans(vec1, vec2)
```

Mal sehen, wie unsere Beobachtung in einem Histogramm unserer Teststatistik aussieht.

```
hist(result)
abline(v=observedMeanDifference, col = "blue")
```



Es sieht nicht so *aus, als* ob unser beobachtetes Ergebnis sehr wahrscheinlich zufällig auftritt ...

Wir wollen den p-Wert berechnen, die Wahrscheinlichkeit des ursprünglichen beobachteten Ergebnisses, wenn zwischen den beiden Variablen keine Beziehung besteht.

```
pValue = 2*mean(result >= (observedMeanDifference))
```

Lassen Sie uns das ein bisschen brechen:

```
result >= (observedMeanDifference)
```

Erzeugt einen booleschen Vektor wie:

```
FALSE TRUE FALSE FALSE TRUE FALSE ...
```

Bei `TRUE` ist der `result` jedes Mal größer oder gleich dem `observedMean`.

Die Funktion `mean` wird dieses Vektors als interpretieren 1 für `TRUE` und 0 für `FALSE`, und geben Sie uns den Prozentsatz von 1, s in der Mischung, dh die Anzahl der Male, unsere neu gemischt Vektor mittlere Differenz übertroffen oder erreicht, was wir beobachtet.

Schließlich multiplizieren wir mit 2, da die Verteilung unserer Teststatistik hochsymmetrisch ist und wir wirklich wissen wollen, welche Ergebnisse "extremer" sind als unser beobachtetes Ergebnis.

Alles, was übrig bleibt, ist die Ausgabe des p-Werts, der 0.06093939. Die Interpretation dieses Wertes ist subjektiv, aber ich würde sagen, dass Vitamin C das Zahnwachstum wesentlich mehr fördert als Orangensaft.

Durchführen eines Permutationstests online lesen:

<https://riptutorial.com/de/r/topic/3216/durchfuehren-eines-permutationstests>

Kapitel 33: E / A für das Binärformat von R

Examples

Rds- und RData-Dateien (Rda)

`.rds` und `.Rdata` Dateien (auch als `.rda` Dateien bezeichnet) können verwendet werden, um R-Objekte in einem von R native Format zu speichern. Das Speichern dieser Art hat mehrere Vorteile, wenn es mit nicht nativen Speicheransätzen `write.table`, z.

- Es ist schneller, die Daten auf R wiederherzustellen
- Es speichert R-spezifische Informationen in den Daten (z. B. Attribute, Variablentypen usw.).

`saveRDS` / `readRDS` nur ein einzelnes R-Objekt. Sie sind jedoch flexibler als der Multi-Objekt-Speicheransatz, da der Objektname des wiederhergestellten Objekts nicht mit dem Objektnamen identisch sein muss, als das Objekt gespeichert wurde.

Mit einer `.rds`-Datei, zum Beispiel zum Speichern des `iris` Datasets würden wir Folgendes verwenden:

```
saveRDS(object = iris, file = "my_data_frame.rds")
```

So laden Sie die Daten wieder in:

```
iris2 <- readRDS(file = "my_data_frame.rds")
```

Um mehrere Objekte zu speichern, können Sie `save()` und als `.Rdata`.

Beispiel, um 2 Datenrahmen zu speichern: Iris und Autos

```
save(iris, cars, file = "myIrisAndCarsData.Rdata")
```

Laden:

```
load("myIrisAndCarsData.Rdata")
```

Umgebung

Mit den Funktionen `save` und `load` können Sie die Umgebung angeben, in der das Objekt gehostet wird:

```
save(iris, cars, file = "myIrisAndCarsData.Rdata", envir = foo <- new.env())  
load("myIrisAndCarsData.Rdata", envir = foo)  
foo$cars
```

```
save(iris, cars, file = "myIrisAndCarsData.Rdata", envir = foo <- new.env())  
load("myIrisAndCarsData.Rdata", envir = foo)  
foo$cars
```

E / A für das Binärformat von R online lesen: <https://riptutorial.com/de/r/topic/5540/e---a-fur-das-binärformat-von-r>

Kapitel 34: E / A für Datenbanktabellen

Bemerkungen

Spezialisierte Pakete

- RMySQL
- [RODBC](#)

Examples

Daten aus MySQL-Datenbanken lesen

Allgemeines

Mit dem Paket [RMySQL](#) können Sie MySQL- und MariaDB-Datenbanken einfach abfragen und das Ergebnis in einem R-Datenrahmen speichern:

```
library(RMySQL)

mydb <- dbConnect(MySQL(), user='user', password='password', dbname='dbname', host='127.0.0.1')

queryString <- "SELECT * FROM table1 t1 JOIN table2 t2 on t1.id=t2.id"
query <- dbSendQuery(mydb, queryString)
data <- fetch(query, n=-1) # n=-1 to return all results
```

Grenzen verwenden

Es ist auch möglich, ein Limit festzulegen, z. B. nur die ersten 100.000 Zeilen. Ändern Sie dazu einfach die SQL-Abfrage hinsichtlich des gewünschten Grenzwerts. Das erwähnte Paket wird diese Optionen berücksichtigen. Beispiel:

```
queryString <- "SELECT * FROM table1 limit 100000"
```

Daten aus MongoDB-Datenbanken lesen

Um Daten aus einer MongoDB-Datenbank in einen R-Datenrahmen zu laden, verwenden Sie die Bibliothek [MongoLite](#) :

```
# Use MongoLite library:
#install.packages("mongolite")
library(jsonlite)
```

```
library(mongolite)

# Connect to the database and the desired collection as root:
db <- mongo(collection = "Tweets", db = "TweetCollector", url =
"mongodb://USERNAME:PASSWORD@HOSTNAME")

# Read the desired documents i.e. Tweets inside one dataframe:
documents <- db$find(limit = 100000, skip = 0, fields = '{ "_id" : false, "Text" : true }')
```

Der Code stellt eine Verbindung zum Server `HOSTNAME` als `USERNAME` mit `PASSWORD` , versucht, die Datenbank `TweetCollector` zu öffnen und die Sammlungs- `Tweets` lesen. Die Abfrage versucht, das Feld bzw. die Spalte `Text` zu lesen.

Das Ergebnis ist ein Datenrahmen mit Spalten als Ergebnisdatensatz. In diesem Beispiel enthält der Datenrahmen die Spalte `Text` , z. B. `documents$Text` .

E / A für Datenbanktabellen online lesen: <https://riptutorial.com/de/r/topic/5537/e---a-fur-datenbanktabellen>

Kapitel 35: E / A für Fremdtabellen (Excel, SAS, SPSS, Stata)

Examples

Daten mit rio importieren

Eine sehr einfache Möglichkeit, Daten aus vielen gängigen Dateiformaten zu importieren, ist mit [rio](#). Dieses Paket enthält eine Funktion `import()`, die viele häufig verwendete Datenimportfunktionen umschließt und somit eine Standardschnittstelle bereitstellt. Es funktioniert einfach, indem Sie einen Dateinamen oder eine URL an `import()`:

```
import("example.csv")      # comma-separated values
import("example.tsv")      # tab-separated values
import("example.dta")      # Stata
import("example.sav")      # SPSS
import("example.sas7bdat")  # SAS
import("example.xlsx")     # Excel
```

`import()` kann auch aus komprimierten Verzeichnissen, URLs (HTTP oder HTTPS) und der Zwischenablage lesen. Eine umfassende Liste aller unterstützten Dateiformate ist im [rio-Paket github-Repository](#) verfügbar.

Es ist sogar möglich, einige weitere Parameter anzugeben, die sich auf das bestimmte Dateiformat beziehen, das Sie lesen möchten, und diese direkt an die Funktion `import()`:

```
import("example.csv", format = ",") #for csv file where comma is used as separator
import("example.csv", format = ";") #for csv file where semicolon is used as separator
```

Excel-Dateien importieren

Es gibt mehrere R-Pakete zum Lesen von Excel-Dateien, die jeweils unterschiedliche Sprachen oder Ressourcen verwenden, wie in der folgenden Tabelle zusammengefasst:

R paket	Verwendet
xlsx	Java
XLconnect	Java
openxlsx	C ++
readxl	C ++
RODBC	ODBC

R paket	Verwendet
gdata	Perl

Für die Pakete, die Java oder ODBC verwenden, ist es wichtig, Details zu Ihrem System zu kennen, da in Abhängigkeit von Ihrer R-Version und dem Betriebssystem Kompatibilitätsprobleme auftreten können. Wenn Sie beispielsweise R 64-Bit verwenden, müssen Sie auch über Java 64-Bit verfügen, um `xlsx` oder `XLconnect` .

Nachfolgend finden Sie einige Beispiele zum Lesen von Excel-Dateien mit jedem Paket. Beachten Sie, dass viele Pakete die gleichen oder sehr ähnliche Funktionsnamen haben. Daher ist es hilfreich, das Paket explizit anzugeben, wie `package::function` . Das Paket `openxlsx` erfordert eine vorherige Installation von RTools.

Lesen von Excel-Dateien mit dem xlsx-Paket

```
library(xlsx)
```

Der Index oder Name des Blattes ist zum Importieren erforderlich.

```
xlsx::read.xlsx("Book1.xlsx", sheetIndex=1)

xlsx::read.xlsx("Book1.xlsx", sheetName="Sheet1")
```

Lesen von Excel-Dateien mit dem XLconnect-Paket

```
library(XLConnect)
wb <- XLConnect::loadWorkbook("Book1.xlsx")

# Either, if Book1.xlsx has a sheet called "Sheet1":
sheet1 <- XLConnect::readWorksheet(wb, "Sheet1")
# Or, more generally, just get the first sheet in Book1.xlsx:
sheet1 <- XLConnect::readWorksheet(wb, getSheets(wb)[1])
```

`XLConnect` importiert automatisch die in `Book1.xlsx` eingebetteten `Book1.xlsx` Excel- `Book1.xlsx` . Dies ist nützlich, wenn Sie Ihr Arbeitsmappenobjekt formatieren und ein perfekt formatiertes Excel-Dokument exportieren möchten. Zunächst müssen Sie die gewünschten `Book1.xlsx` in `Book1.xlsx` und diese beispielsweise als `myHeader` , `myBody` und `myPcts` . Nach dem Laden der Arbeitsmappe in R (siehe oben):

```
Headerstyle <- XLConnect::getCellStyle(wb, "myHeader")
Bodystyle <- XLConnect::getCellStyle(wb, "myBody")
Pctsstyle <- XLConnect::getCellStyle(wb, "myPcts")
```

Die Zellstile werden jetzt in Ihrer `R` Umgebung gespeichert. Um die Zellenstile bestimmten Bereichen Ihrer Daten zuzuordnen, müssen Sie den Bereich definieren und dann den Stil zuweisen:

```
Headerrange <- expand.grid(row = 1, col = 1:8)
Bodyrange <- expand.grid(row = 2:6, col = c(1:5, 8))
Pctrange <- expand.grid(row = 2:6, col = c(6, 7))

XLConnect::setCellStyle(wb, sheet = "sheet1", row = Headerrange$row,
                        col = Headerrange$col, cellstyle = Headerstyle)
XLConnect::setCellStyle(wb, sheet = "sheet1", row = Bodyrange$row,
                        col = Bodyrange$col, cellstyle = Bodystyle)
XLConnect::setCellStyle(wb, sheet = "sheet1", row = Pctrange$row,
                        col = Pctrange$col, cellstyle = Pctstyle)
```

Beachten Sie, dass `XLConnect` einfach ist, die Formatierung jedoch extrem langsam werden kann. Eine wesentlich schnellere, aber umständlichere Formatierungsoption bietet `openxlsx`.

Lesen von Excel-Dateien mit dem Paket `openxlsx`

Excel-Dateien können mit dem Paket `openxlsx` importiert werden

```
library(openxlsx)

openxlsx::read.xlsx("spreadsheet1.xlsx", colNames=TRUE, rowNames=TRUE)

#colNames: If TRUE, the first row of data will be used as column names.
#rowNames: If TRUE, first column of data will be used as row names.
```

Das Blatt, das in `R` gelesen werden soll, kann ausgewählt werden, entweder durch seine Position in der Bereitstellung `sheet` Argumente:

```
openxlsx::read.xlsx("spreadsheet1.xlsx", sheet = 1)
```

oder indem Sie ihren Namen angeben:

```
openxlsx::read.xlsx("spreadsheet1.xlsx", sheet = "Sheet1")
```

Außerdem kann `openxlsx` in einem `openxlsx` erkennen. Um die automatische Erkennung von Datumsangaben zu ermöglichen, sollte ein Argument `detectDates` auf `TRUE`:

```
openxlsx::read.xlsx("spreadsheet1.xlsx", sheet = "Sheet1", detectDates= TRUE)
```

Lesen von Excel-Dateien mit dem Paket

readxl

Excel-Dateien können als `readxl` mit dem Paket `readxl` in R `readxl` werden.

```
library(readxl)
```

Es kann sowohl `.xls` als auch `.xlsx` Dateien lesen.

```
readxl::read_excel("spreadsheet1.xls")
readxl::read_excel("spreadsheet2.xlsx")
```

Das zu importierende Blatt kann nach Nummer oder Name angegeben werden.

```
readxl::read_excel("spreadsheet.xls", sheet = 1)
readxl::read_excel("spreadsheet.xls", sheet = "summary")
```

Das Argument `col_names = TRUE` setzt die erste Zeile als Spaltennamen.

```
readxl::read_excel("spreadsheet.xls", sheet = 1, col_names = TRUE)
```

Das Argument `col_types` kann verwendet werden, um die `col_types` in den Daten als Vektor anzugeben.

```
readxl::read_excel("spreadsheet.xls", sheet = 1, col_names = TRUE,
  col_types = c("text", "date", "numeric", "numeric"))
```

Lesen von Excel-Dateien mit dem RODBC-Paket

Excel-Dateien können mit dem ODBC-Excel-Treiber gelesen werden, der eine Schnittstelle zu Access Database Engine (ACE) von Windows (ehemals JET) herstellt. Mit dem RODBC-Paket kann R eine Verbindung zu diesem Treiber herstellen und Arbeitsmappen direkt abfragen. Es wird angenommen, dass Arbeitsblätter Spaltenüberschriften in der ersten Zeile mit Daten in organisierten Spalten mit ähnlichen Typen beibehalten. **HINWEIS:** Dieser Ansatz ist nur auf Windows / PC-Maschinen beschränkt, da JET / ACE-DLL-Dateien installiert sind und auf anderen Betriebssystemen nicht verfügbar sind.

```
library(RODBC)

xlconn <- odbcDriverConnect('Driver={Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)};
  DBQ=C:\\Path\\To\\Workbook.xlsx')

df <- sqlQuery(xlconn, "SELECT * FROM [SheetName$]")
close(xlconn)
```

Excel-Arbeitsblätter können auf diese Weise mit einer SQL-Engine verbunden werden, ähnlich wie Datenbanktabellen einschließlich `JOIN` und `UNION` Operationen. Die Syntax folgt dem JET / ACE SQL-Dialekt. **ANMERKUNG:** Nur Datenzugriffs-DML-Anweisungen, speziell `SELECT` können für Arbeitsmappen ausgeführt werden, die als nicht aktualisierbare Abfragen betrachtet werden.

```
joindf <- sqlQuery(xlconn, "SELECT t1.*, t2.* FROM [Sheet1$] t1
                           INNER JOIN [Sheet2$] t2
                           ON t1.[ID] = t2.[ID]")

uniondf <- sqlQuery(xlconn, "SELECT * FROM [Sheet1$]
                           UNION
                           SELECT * FROM [Sheet2$]")
```

Sogar andere Arbeitsmappen können von demselben ODBC-Kanal abgefragt werden, der auf eine aktuelle Arbeitsmappe verweist:

```
otherwkbkdf <- sqlQuery(xlconn, "SELECT * FROM
                                [Excel 12.0 Xml;HDR=Yes;
                                Database=C:\\Path\\To\\Other\\Workbook.xlsx].[Sheet1$];")
```

Lesen von Excel-Dateien mit dem gdata-Paket

Beispiel hier

Lesen und Schreiben von Stata-, SPSS- und SAS-Dateien

Mit den Paketen `foreign` und `haven` können Dateien aus verschiedenen anderen Statistikpaketen wie Stata, SPSS und SAS sowie zugehöriger Software importiert und exportiert werden. Es gibt eine `read` für jede der unterstützten Datentypen, um die Dateien zu importieren.

```
# loading the packages
library(foreign)
library(haven)
library(readstata13)
library(Hmisc)
```

Einige Beispiele für die häufigsten Datentypen:

```
# reading Stata files with `foreign`
read.dta("path\to\your\data")
# reading Stata files with `haven`
read_dta("path\to\your\data")
```

Das `foreign` Paket kann in Stata (DTA) Dateien für die Versionen von Stata 7-12 lesen. Gemäß der Entwicklungsseite ist die `read.dta` mehr oder weniger eingefroren und wird zum Lesen in Version 13+ nicht aktualisiert. Für neuere Versionen von Stata können Sie entweder das Paket `readstata13` oder den `haven`. Für `readstata13` sind die Dateien

```
# reading recent Stata (13+) files with `readstata13`
read.dta13("path\to\your\data")
```

Zum Einlesen von SPSS- und SAS-Dateien

```
# reading SPSS files with `foreign`
read.spss("path\to\your\data.sav", to.data.frame = TRUE)
# reading SPSS files with `haven`
read_spss("path\to\your\data.sav")
read_sav("path\to\your\data.sav")
read_por("path\to\your\data.por")

# reading SAS files with `foreign`
read.ssd("path\to\your\data")
# reading SAS files with `haven`
read_sas("path\to\your\data")
# reading native SAS files with `Hmisc`
sas.get("path\to\your\data") #requires access to saslib
# Reading SA XPORT format ( *.XPT ) files
sasxport.get("path\to\your\data.xpt") # does not require access to SAS executable
```

Das **SAScii** Paket bietet Funktionen, die SAS SET- **SAScii** akzeptieren und eine Textdatei erstellen, die mit `read.fwf` verarbeitet werden `read.fwf` . Es hat sich als sehr robust für den Import großer öffentlich veröffentlichter Datensätze erwiesen. Der Support ist unter <https://github.com/ajdamico/SAScii> verfügbar

Um Datenrahmen in andere Statistikpakete zu exportieren, können Sie die Schreibfunktionen `write.foreign()` . Dadurch werden 2 Dateien geschrieben, eine mit den Daten und eine mit Anweisungen, die das andere Paket zum Lesen der Daten benötigt.

```
# writing to Stata, SPSS or SAS files with `foreign`
write.foreign(dataframe, datafile, codefile,
              package = c("SPSS", "Stata", "SAS"), ...)
write.foreign(dataframe, "path\to\data\file", "path\to\instruction\file", package = "Stata")

# writing to Stata files with `foreign`
write.dta(dataframe, "file", version = 7L,
          convert.dates = TRUE, tz = "GMT",
          convert.factors = c("labels", "string", "numeric", "codes"))

# writing to Stata files with `haven`
write_dta(dataframe, "path\to\your\data")

# writing to Stata files with `readstata13`
save.dta13(dataframe, file, data.label = NULL, time.stamp = TRUE,
            convert.factors = TRUE, convert.dates = TRUE, tz = "GMT",
            add.rownames = FALSE, compress = FALSE, version = 117,
            convert.underscore = FALSE)

# writing to SPSS files with `haven`
write_sav(dataframe, "path\to\your\data")
```

Von der SPSS gespeicherte Datei kann auf folgende Weise mit `read.spss` gelesen werden:

```
foreign::read.spss('data.sav', to.data.frame=TRUE, use.value.labels=FALSE,
```

```

        use.missings=TRUE, reencode='UTF-8')
# to.data.frame if TRUE: return a data frame
# use.value.labels if TRUE: convert variables with value labels into R factors with those
levels
# use.missings if TRUE: information on user-defined missing values will be used to set the
corresponding values to NA.
# reencode character strings will be re-encoded to the current locale. The default, NA, means
to do so in a UTF-8 locale, only.

```

Importieren oder Exportieren einer Federdatei

Feather ist eine Implementierung von **Apache Arrow**, die zum Speichern von Datenrahmen in einer sprachenunabhängigen Weise entwickelt wurde, während Metadaten (z. B. Datumsklassen) beibehalten werden, wodurch die Interoperabilität zwischen Python und R erhöht wird. Durch das Lesen einer Federdatei wird ein Halbbild erzeugt, kein Standard-Datenrahmen.

```

library(feather)

path <- "filename.feather"
df <- mtcars

write_feather(df, path)

df2 <- read_feather(path)

head(df2)
## A tibble: 6 x 11
##   mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  21.0     6   160   110   3.90  2.620  16.46     0     1     4     4
## 2  21.0     6   160   110   3.90  2.875  17.02     0     1     4     4
## 3  22.8     4   108    93   3.85  2.320  18.61     1     1     4     1
## 4  21.4     6   258   110   3.08  3.215  19.44     1     0     3     1
## 5  18.7     8   360   175   3.15  3.440  17.02     0     0     3     2
## 6  18.1     6   225   105   2.76  3.460  20.22     1     0     3     1

head(df)
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6  160  110  3.90  2.620 16.46 0  1   4    4
## Mazda RX4 Wag  21.0   6  160  110  3.90  2.875 17.02 0  1   4    4
## Datsun 710     22.8   4  108   93  3.85  2.320 18.61 1  1   4    1
## Hornet 4 Drive  21.4   6  258  110  3.08  3.215 19.44 1  0   3    1
## Hornet Sportabout 18.7   8  360  175  3.15  3.440 17.02 0  0   3    2
## Valiant        18.1   6  225  105  2.76  3.460 20.22 1  0   3    1

```

Die aktuelle Dokumentation enthält diese Warnung:

Hinweis für Benutzer: Feder sollte als Alpha-Software behandelt werden. Insbesondere das Dateiformat wird sich voraussichtlich im kommenden Jahr weiterentwickeln. Verwenden Sie Feather nicht zur Langzeitspeicherung von Daten.

E / A für Fremdtabellen (Excel, SAS, SPSS, Stata) online lesen:

<https://riptutorial.com/de/r/topic/5536/e---a-fur-fremdtabellen--excel--sas--spss--stata->

Kapitel 36: E / A für geographische Daten (Shapefiles usw.)

Einführung

Siehe auch [Einführung in geografische Karten](#) sowie [Eingabe und Ausgabe](#)

Examples

Importieren und Exportieren von Shapefiles

Mit dem Paket `rgdal` ist es möglich, Shapfiles mit R zu importieren und zu exportieren. Mit der Funktion `readOGR` können Shapfiles importiert werden. Wenn Sie eine Datei von zB ArcGIS importieren möchten, ist das erste Argument `dsn` der Pfad zu dem Ordner, der das Shapefile enthält. `layer` ist der Name des Shapefiles ohne Dateiendung (nur `map` und nicht `map.shp`).

```
library(rgdal)
readOGR(dsn = "path\to\the\folder\containing\the\shapefile", layer = "map")
```

Um ein Shapefile zu exportieren, verwenden Sie die Funktion `writeOGR` . Das erste Argument ist das in R. `dsn` erzeugte räumliche Objekt und die `layer` sind die gleichen wie oben. Das obligatorische Argument ist der Treiber, mit dem das Shapefile erstellt wird. Die Funktion `ogrDrivers()` listet alle verfügbaren Treiber auf. Wenn Sie ein Shapfile nach ArcGis oder QGis exportieren möchten, können Sie `driver = "ESRI Shapefile"` .

```
writeOGR(Rmap, dsn = "path\to\the\folder\containing\the\shapefile", layer = "map",
        driver = "ESRI Shapefile" )
```

`tmap` Paket hat eine sehr praktische Funktion `read_shape()` , die einen Wrapper für `rgdal::readOGR()` . Die Funktion `read_shape()` vereinfacht das Importieren eines Shapefiles `read_shape()` . Nach unten ist `tmap` ziemlich schwer.

E / A für geographische Daten (Shapefiles usw.) online lesen:

<https://riptutorial.com/de/r/topic/5538/e---a-fur-geographische-daten--shapefiles-usw-->

Kapitel 37: E / A für Rasterbilder

Einführung

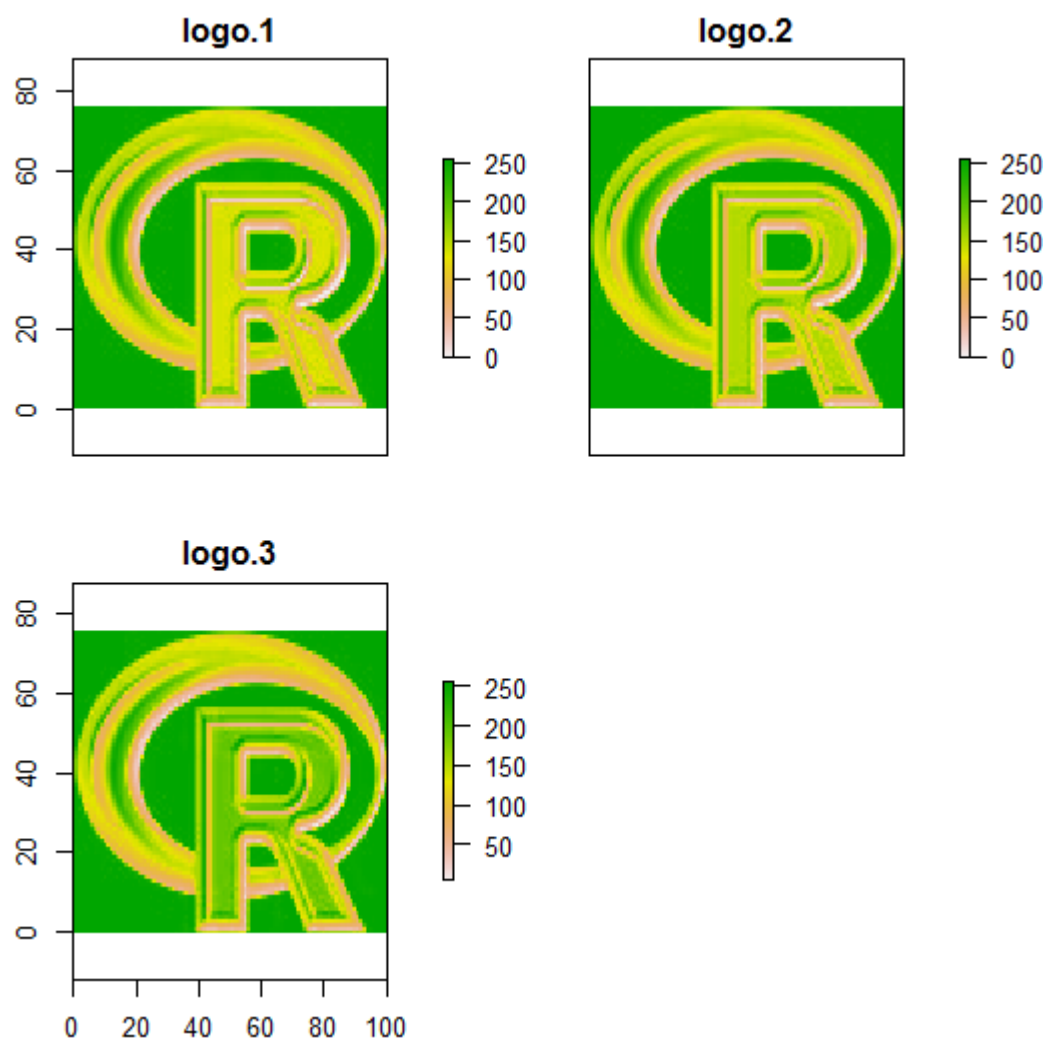
Siehe auch [Raster- und Bildanalyse](#) sowie [Eingabe und Ausgabe](#)

Examples

Laden Sie ein Multilayer-Raster

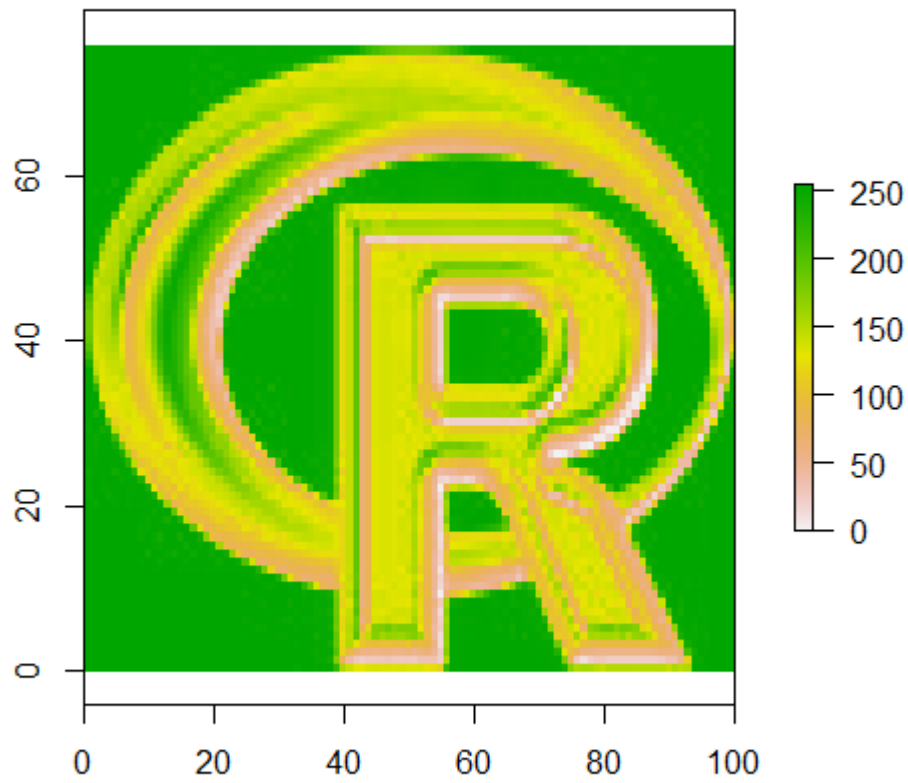
Das R-Logo ist eine Multilayer-Raster-Datei (rot, grün, blau).

```
library(raster)
r <- stack("C:/Program Files/R/R-3.2.3/doc/html/logo.jpg")
plot(r)
```



Die einzelnen Layer des `RasterStack` Objekts können mit `[[` .

```
plot(r[[1]])
```



E / A für Rasterbilder online lesen: <https://riptutorial.com/de/r/topic/5539/e---a-fur-rasterbilder>

Kapitel 38: Einführung in geographische Karten

Einführung

Siehe auch [E / A für geografische Daten](#)

Examples

Grundlegende Kartenerstellung mit `map()` aus den Paketkarten

Die Funktion `map()` aus den Package- `maps` bietet einen einfachen Ausgangspunkt zum Erstellen von Maps mit R.

Eine grundlegende Weltkarte kann wie folgt gezeichnet werden:

```
require(maps)
map()
```



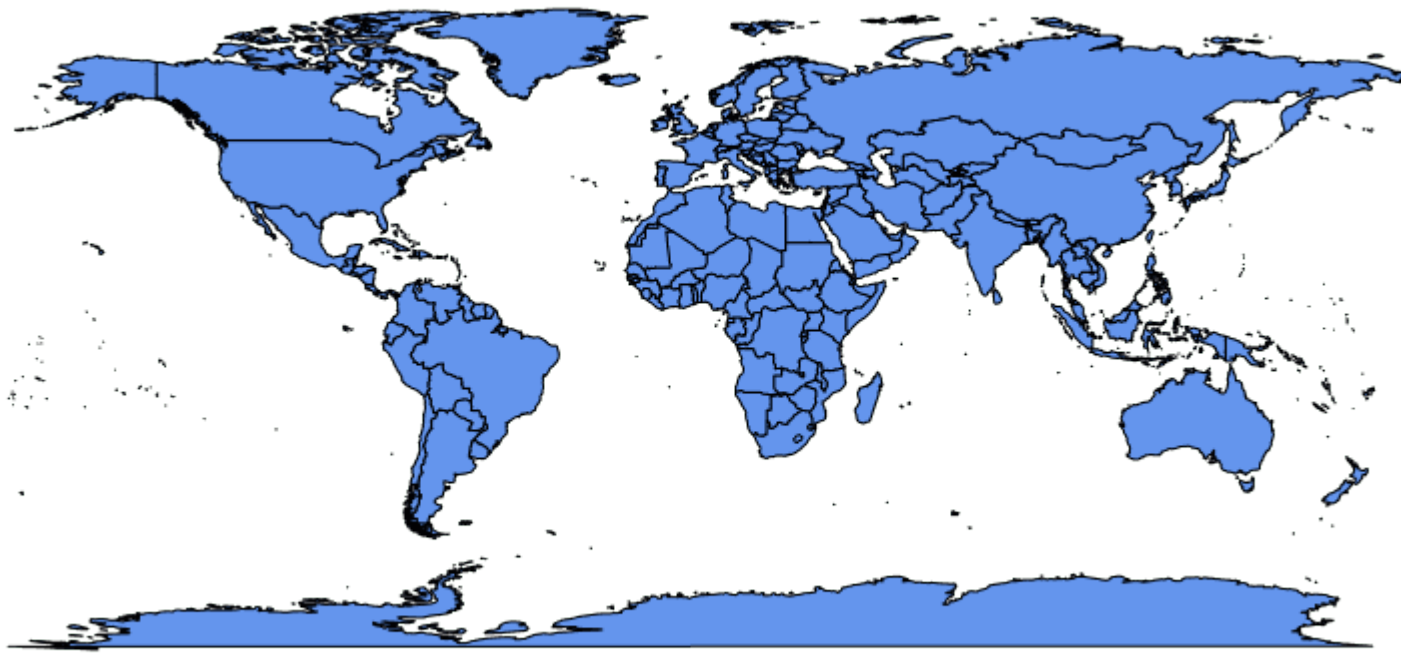
Die Farbe der Kontur kann geändert werden, indem der Farbparameter `col` entweder auf den Zeichennamen oder den Hexadezimalwert einer Farbe gesetzt wird:

```
require(maps)
map(col = "cornflowerblue")
```



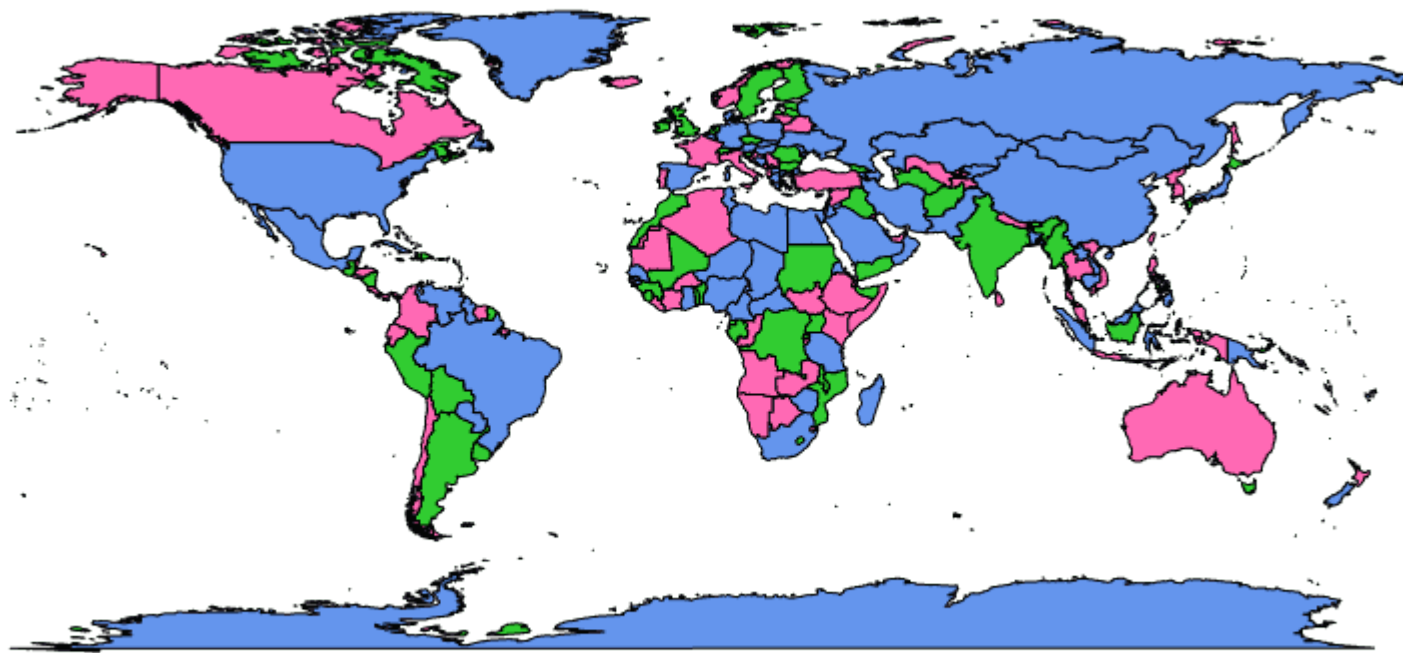
Landmassen mit der Farbe zu füllen `col` wir einstellen können `fill = TRUE` :

```
require(maps)
map(fill = TRUE, col = c("cornflowerblue"))
```



Ein Vektor mit beliebiger Länge kann an `col` wenn auch `fill = TRUE` gesetzt ist:

```
require(maps)
map(fill = TRUE, col = c("cornflowerblue", "limegreen", "hotpink"))
```



Im obigen Beispiel werden Farben aus `col` beliebig Polygonen in der Karte zugewiesen, die Regionen darstellen, und Farben werden wiederverwendet, wenn weniger Farben als Polygone vorhanden sind.

Wir können die Farbkodierung auch verwenden, um eine statistische Variable darzustellen, die optional in einer Legende beschrieben werden kann. Eine als solche erstellte Karte wird als "Choropleth" bezeichnet.

Das folgende Choropleth-Beispiel legt das erste Argument von `map()`, bei dem es sich bei der database um "county" und "state", um die Arbeitslosigkeit mithilfe von Daten aus den integrierten Datensätzen `unemp` und `county.fips` während die `unemp county.fips` überlagert werden:

```
require(maps)
if(require(mapproj)) { # mapproj is used for projection="polyconic"
  # color US county map by 2009 unemployment rate
  # match counties to map using FIPS county codes
  # Based on J's solution to the "Choropleth Challenge"
  # Code improvements by Hack-R (hack-r.github.io)

  # load data
  # unemp includes data for some counties not on the "lower 48 states" county
  # map, such as those in Alaska, Hawaii, Puerto Rico, and some tiny Virginia
  # cities
  data(unemp)
  data(county.fips)
```

```

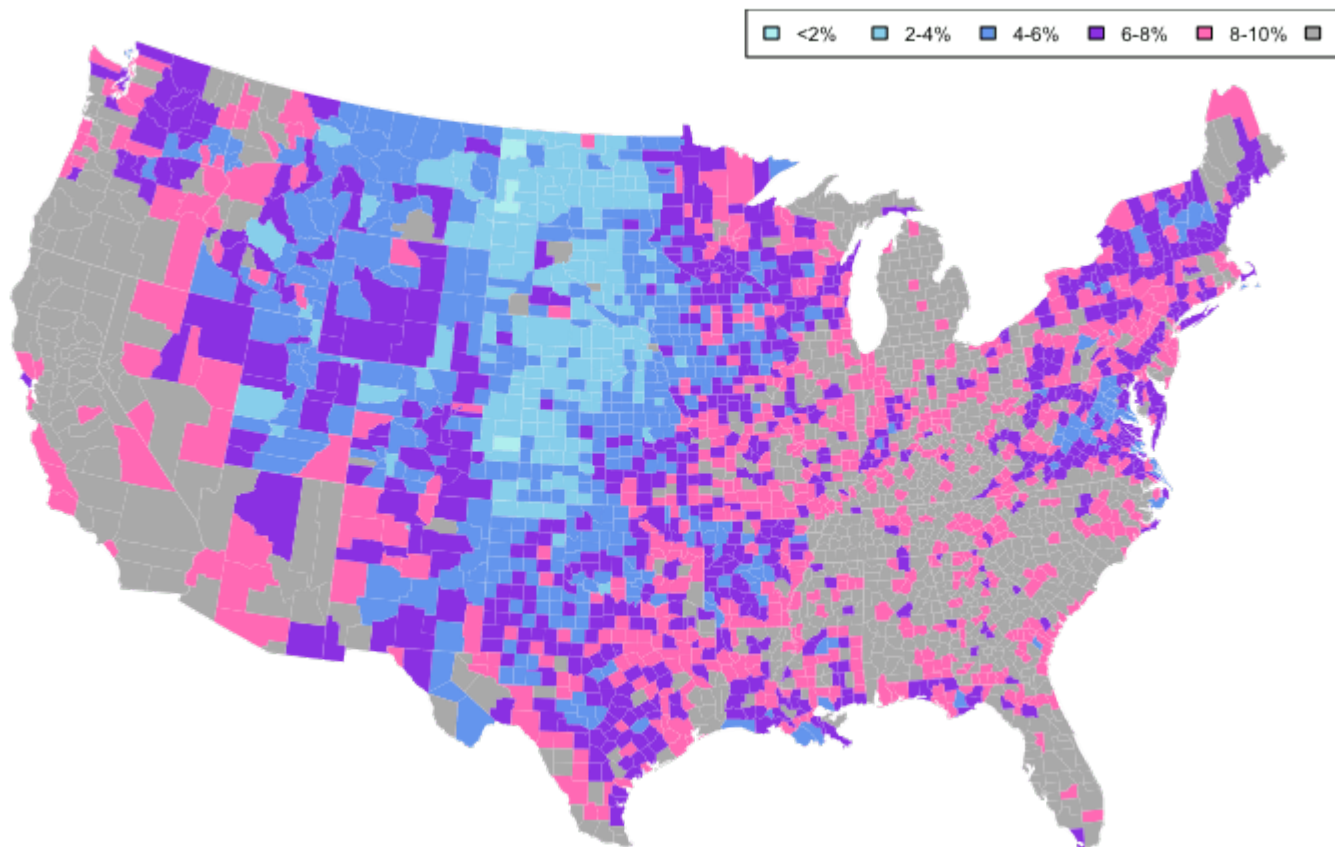
# define color buckets
colors = c("paleturquoise", "skyblue", "cornflowerblue", "blueviolet", "hotpink",
"darkgrey")
unemp$colorBuckets <- as.numeric(cut(unemp$unemp, c(0, 2, 4, 6, 8, 10, 100)))
leg.txt <- c("<2%", "2-4%", "4-6%", "6-8%", "8-10%", ">10%")

# align data with map definitions by (partial) matching state,county
# names, which include multiple polygons for some counties
cnty.fips <- county.fips$fips[match(map("county", plot=FALSE)$names,
                                     county.fips$polynome)]
colorsmatched <- unemp$colorBuckets[match(cnty.fips, unemp$fips)]

# draw map
par(mar=c(1, 1, 2, 1) + 0.1)
map("county", col = colors[colorsmatched], fill = TRUE, resolution = 0,
    lty = 0, projection = "polyconic")
map("state", col = "white", fill = FALSE, add = TRUE, lty = 1, lwd = 0.1,
    projection="polyconic")
title("unemployment by county, 2009")
legend("topright", leg.txt, horiz = TRUE, fill = colors, cex=0.6)
}

```

unemployment by county, 2009



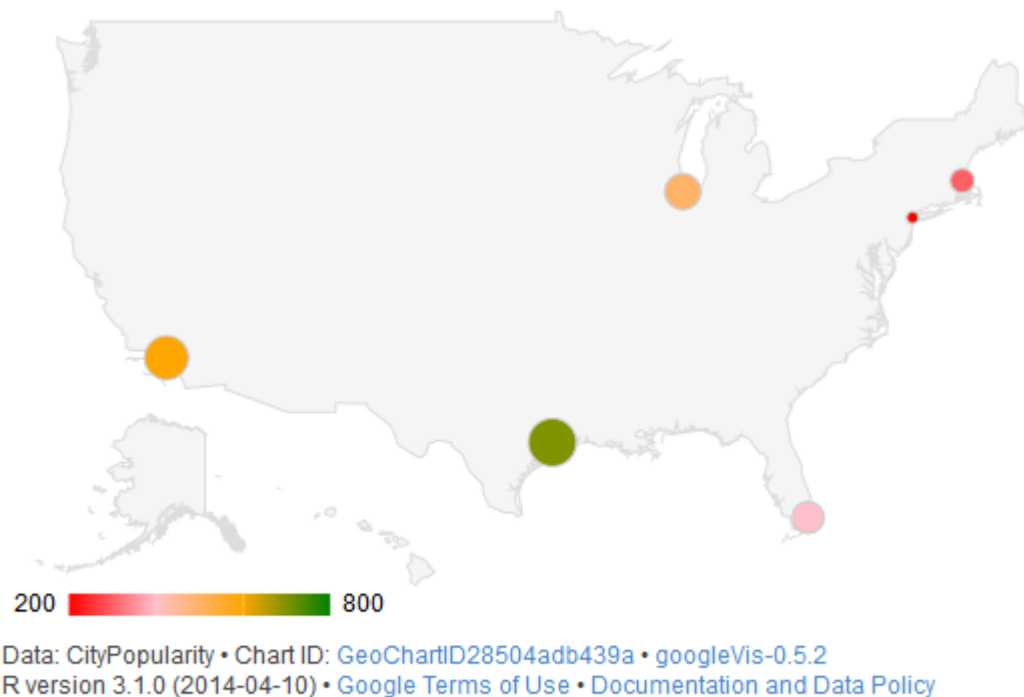
50 Landeskarten und erweiterte Choroplethen mit Google Viz

Eine häufig gestellte [Frage](#) ist, wie man räumlich getrennte geografische Regionen auf derselben Karte nebeneinander stellt (kombiniert), wie im Fall eines Choropleths, der alle 50 amerikanischen Staaten beschreibt (das Festland mit Alaska und Hawaii nebeneinander).

Das Erstellen einer attraktiven Karte mit 50 Zuständen ist beim Einsatz von Google Maps einfach. Zu den Schnittstellen der Google-API gehören die Pakete `googleVis`, `ggmap` und `RgoogleMaps`.

```
require(googleVis)

G4 <- gvisGeoChart(CityPopularity, locationvar='City', colorvar='Popularity',
                  options=list(region='US', height=350,
                              displayMode='markers',
                              colorAxis="{values:[200,400,600,800],
                              colors:['red', 'pink', 'orange','green']}" )
plot(G4)
```

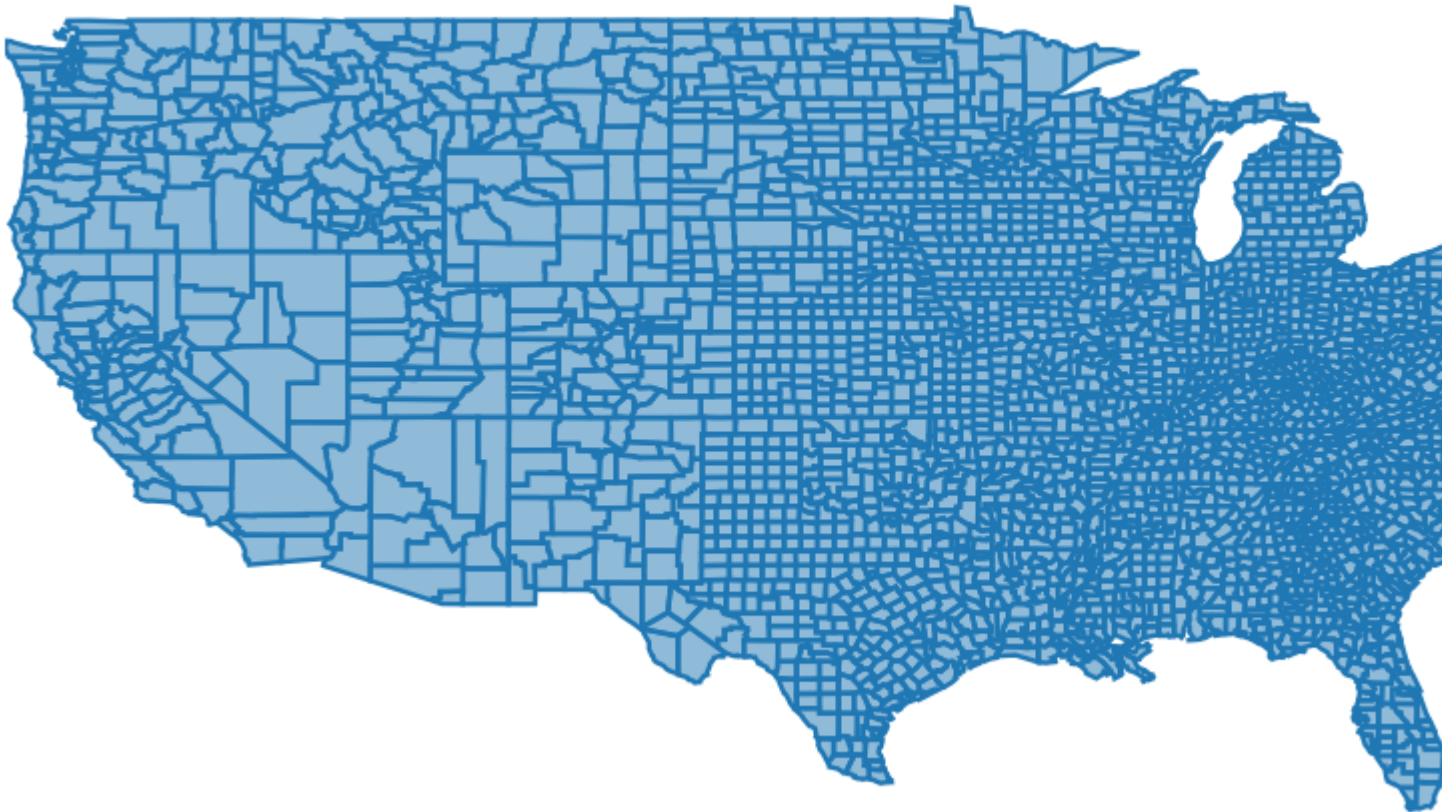


Die Funktion `gvisGeoChart()` erfordert weit weniger Codierung, um einen Choropleth zu erstellen, im Vergleich zu älteren Mapping-Methoden wie `map()` aus den Package- `maps`. Der `colorvar` Parameter ermöglicht das einfache Einfärben einer statistischen Variablen auf einer durch den `locationvar` Parameter festgelegten Ebene. Die verschiedenen Optionen übergeben `options` als Liste ermöglichen die Anpassung von Details der Karte wie Größe (`height`), Form (`markers`) und Farbcodierung (`colorAxis` und `colors`).

Interaktive Kartenpläne

Das Paket enthält mehrere Arten von interaktiven `plotly`, einschließlich Karten. Es gibt mehrere Möglichkeiten, eine Karte in `plotly` zu erstellen. `plot_ly()` Sie die Kartendaten entweder selbst an (über `plot_ly()` oder `ggplotly()`), verwenden Sie die "native" Zuordnungsfunktion von `plot_geo()` über `plot_geo()` oder `plot_mapbox()` oder auch eine Kombination aus beiden. Ein Beispiel für die eigene Bereitstellung der Karte wäre:


```
library(plotly)
map_data("county") %>%
  group_by(group) %>%
  plot_ly(x = ~long, y = ~lat) %>%
  add_polygons() %>%
  layout(
    xaxis = list(title = "", showgrid = FALSE, showticklabels = FALSE),
    yaxis = list(title = "", showgrid = FALSE, showticklabels = FALSE)
  )
```



Für eine Kombination beider Ansätze tauschen Sie `plot_ly()` für `plot_geo()` oder `plot_mapbox()` im obigen Beispiel aus. Weitere Beispiele finden Sie im [ausführlichen Buch](#).

Das nächste Beispiel ist ein "streng nativer" Ansatz, bei dem das Attribut `layout.geo` verwendet wird, um die Ästhetik und die [Zoomstufe](#) der Karte [festzulegen](#). Es verwendet auch die Datenbank `world.cities` aus `maps`, um die brasilianischen Städte zu filtern und sie über der "nativen" Karte zu zeichnen.

Die `poph : poph` ist ein Text mit der Stadt und ihrer Bevölkerung (was beim Mauszeiger angezeigt wird); `q` ist ein geordneter Faktor aus dem Quantil der Bevölkerung. `ge` hat Informationen für das Layout der Karten. Weitere Informationen finden Sie in der [Paketdokumentation](#).

```
library(maps)
dfb <- world.cities[world.cities$country.etc=="Brazil",]
library(plotly)
dfb$poph <- paste(dfb$name, "Pop", round(dfb$pop/1e6,2), " millions")
```

```
dfb$q <- with(dfb, cut(pop, quantile(pop), include.lowest = T))
levels(dfb$q) <- paste(c("1st", "2nd", "3rd", "4th"), "Quantile")
dfb$q <- as.ordered(dfb$q)

ge <- list(
  scope = 'south america',
  showland = TRUE,
  landcolor = toRGB("gray85"),
  subunitwidth = 1,
  countrywidth = 1,
  subunitcolor = toRGB("white"),
  countrycolor = toRGB("white")
)

plot_geo(dfb, lon = ~long, lat = ~lat, text = ~poph,
  marker = ~list(size = sqrt(pop/10000) + 1, line = list(width = 0)),
  color = ~q, locationmode = 'country names') %>%
  layout(geo = ge, title = 'Populations<br>(Click legend to toggle)')
```



Dynamische HTML-Karten mit Broschüre erstellen

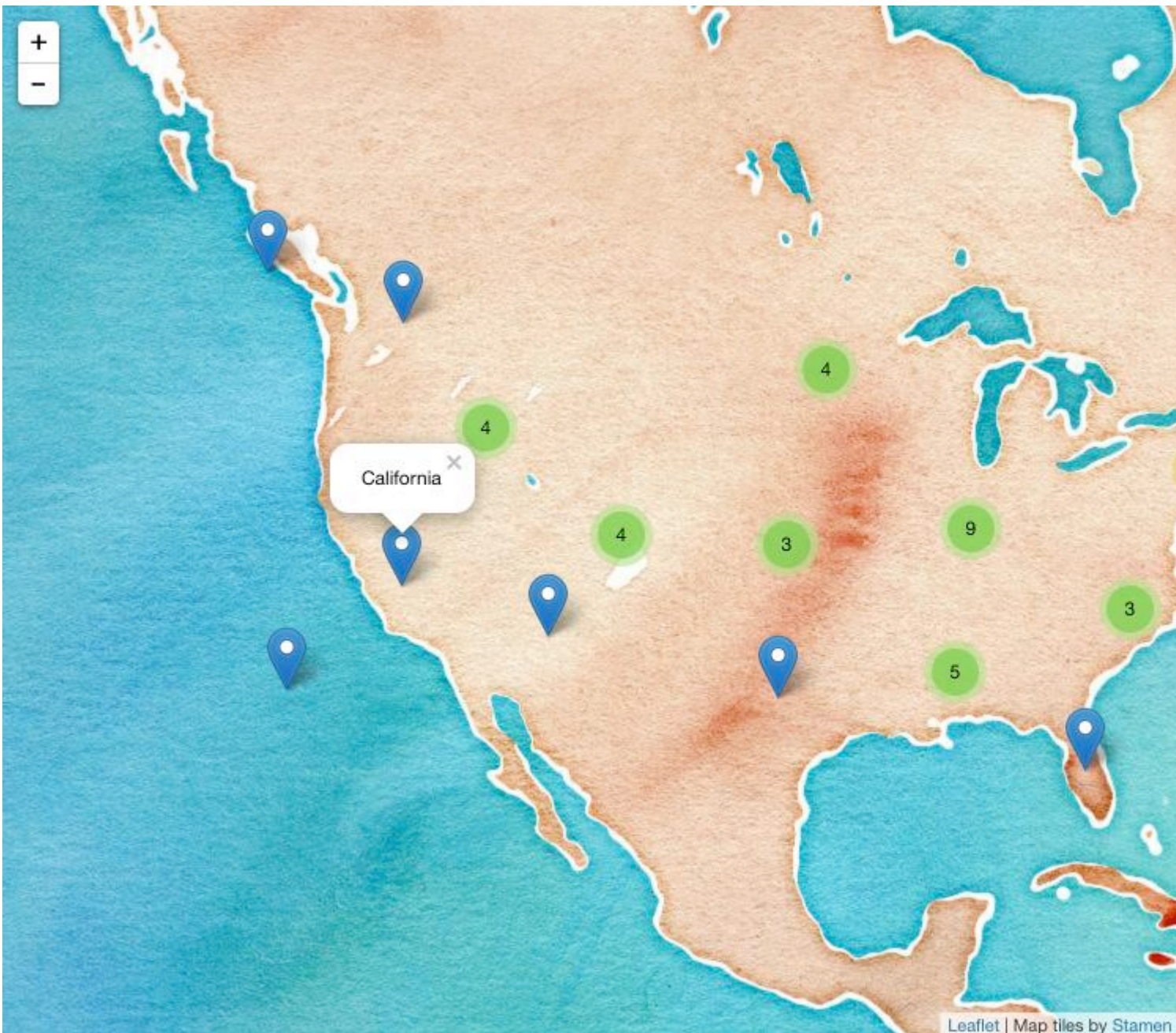
[Leaflet](#) ist eine Open-Source-JavaScript-Bibliothek zum Erstellen dynamischer Karten für das Web. RStudio schrieb R Bindungen für Leaflet, erhältlich durch sein [leaflet - htmlwidgets Paket](#), mit eingebauter [htmlwidgets](#). Prospektkarten lassen sich gut in die Ökosysteme [RMarkdown](#) und [Shiny integrieren](#).

Die Schnittstelle wird [verrohrt](#), mit einer `leaflet()` Funktion, um eine Karte und Folgefunktionen hinzuzufügen (oder zu entfernen) Kartenebenen zu initialisieren. Es gibt viele Arten von Ebenen, von Markern mit Popups bis hin zu Polygonen zum Erstellen von Choroplethenkarten. Auf Variablen, die in dem `data.frame` an `leaflet()` werden, wird über Funktionsstil `~` quotation zugegriffen.

So `state.name state.center` [Datasets](#) `state.name` und `state.center`:

```
library(leaflet)

data.frame(state.name, state.center) %>%
  leaflet() %>%
  addProviderTiles('Stamen.Watercolor') %>%
  addMarkers(lng = ~x, lat = ~y,
             popup = ~state.name,
             clusterOptions = markerClusterOptions())
```



(Screenshot; klicken Sie für eine dynamische Version.)

Dynamische Prospektkarten in glänzenden Anwendungen

Das [Merkblatt](#) - Paket ist so ausgelegt , um [mit glänzenden integrieren](#)

Im **ui** rufen Sie `leafletOutput()` und in dem Server , den Sie aufrufen `renderLeaflet()`

```
library(shiny)
library(leaflet)

ui <- fluidPage(
  leafletOutput("my_leaf")
)

server <- function(input, output, session){

  output$my_leaf <- renderLeaflet({
```



```

    leaflet() %>%
      addProviderTiles('Hydda.Full') %>%
      setView(lat = -37.8, lng = 144.8, zoom = 10)

  })

}

shinyApp(ui, server)

```

Reaktive Eingaben, die sich auf den Ausdruck `renderLeaflet` auswirken, `renderLeaflet` jedoch, dass die gesamte Map jedes Mal neu gezeichnet wird, wenn das reaktive Element aktualisiert wird.

Um eine Karte zu ändern, die bereits läuft, sollten Sie die Funktion `leafletProxy()` verwenden.

Normalerweise verwenden Sie `leaflet`, um die statischen Aspekte der Karte zu erstellen, und `leafletProxy`, um die dynamischen Elemente zu verwalten, zum Beispiel:

```

library(shiny)
library(leaflet)

ui <- fluidPage(
  sliderInput(inputId = "slider",
    label = "values",
    min = 0,
    max = 100,
    value = 0,
    step = 1),
  leafletOutput("my_leaf")
)

server <- function(input, output, session){
  set.seed(123456)
  df <- data.frame(latitude = sample(seq(-38.5, -37.5, by = 0.01), 100),
    longitude = sample(seq(144.0, 145.0, by = 0.01), 100),
    value = seq(1,100))

  ## create static element
  output$my_leaf <- renderLeaflet({

    leaflet() %>%
      addProviderTiles('Hydda.Full') %>%
      setView(lat = -37.8, lng = 144.8, zoom = 8)

  })

  ## filter data
  df_filtered <- reactive({
    df[df$value >= input$slider, ]
  })

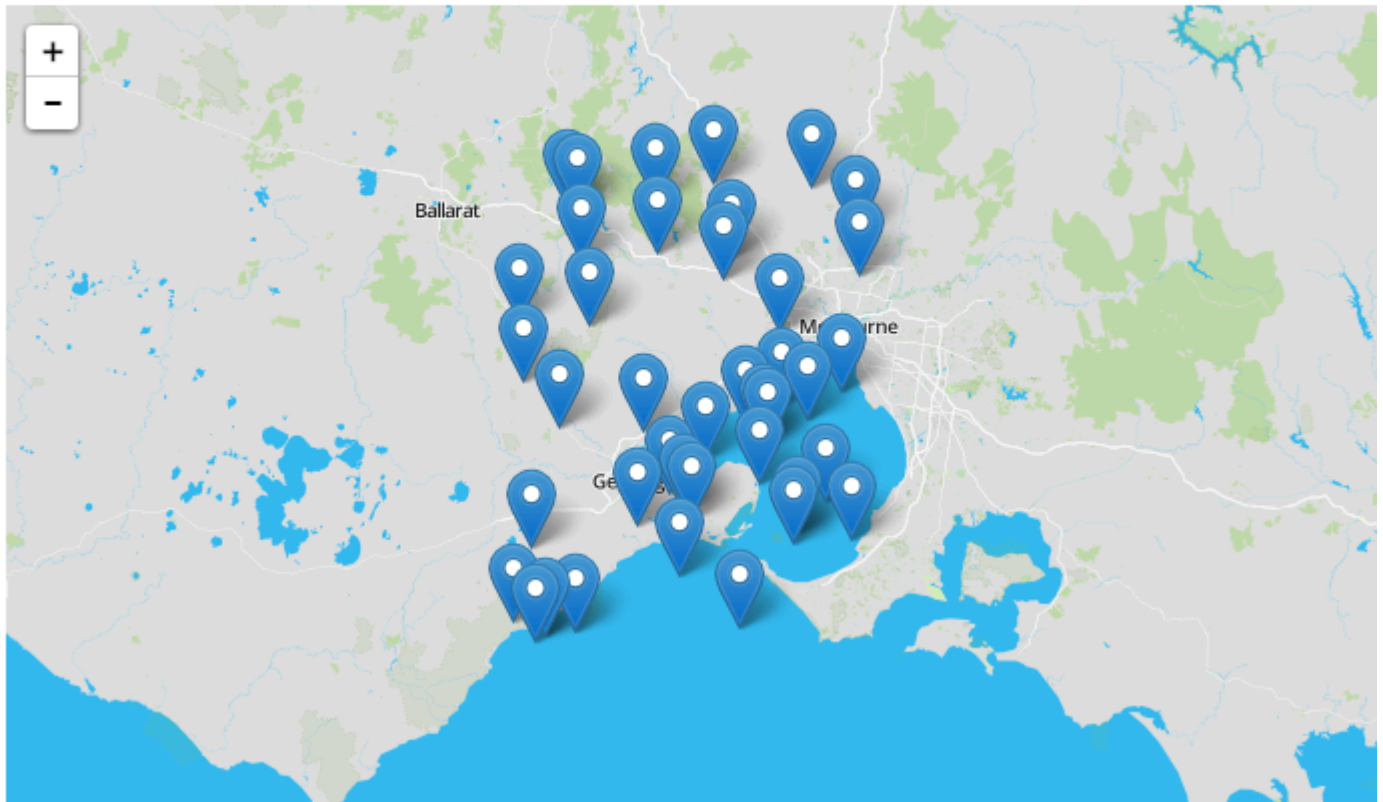
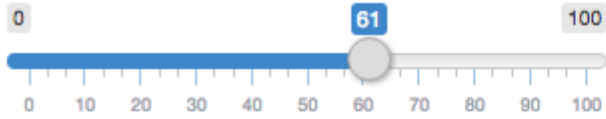
  ## respond to the filtered data
  observe({

    leafletProxy(mapId = "my_leaf", data = df_filtered()) %>%
      clearMarkers() %>%    ## clear previous markers
      addMarkers()
  })
}

```

```
  })  
  
}  
  
shinyApp(ui, server)
```

values



Einführung in geographische Karten online lesen:

<https://riptutorial.com/de/r/topic/1372/einfuehrung-in-geographische-karten>

Kapitel 39: Eingabe und Ausgabe

Bemerkungen

Verwenden Sie `file.path` , um `file.path` zum Lesen oder Schreiben zu `file.path` .

Verwenden Sie `dir` zu sehen, welche Dateien sich in einem Verzeichnis befinden.

Examples

Datenrahmen lesen und schreiben

Datenrahmen sind die tabellarische Datenstruktur von R. Sie können auf verschiedene Weise beschrieben oder gelesen werden.

Dieses Beispiel zeigt einige häufige Situationen. Siehe die Links am Ende für weitere Ressourcen.

Schreiben

Stellen Sie vor dem Erstellen der Beispieldaten sicher, dass Sie sich in einem Ordner befinden, in den Sie schreiben möchten. Führen Sie `getwd()` , um den Ordner zu überprüfen, in dem Sie sich befinden, und lesen Sie `?setwd` wenn Sie Ordner wechseln müssen.

```
set.seed(1)
for (i in 1:3)
  write.table(
    data.frame(id = 1:2, v = sample(letters, 2)),
    file = sprintf("file201%s.csv", i)
  )
```

Jetzt haben wir drei ähnlich formatierte CSV-Dateien auf der Festplatte.

lesen

Wir haben drei ähnlich formatierte Dateien (aus dem letzten Abschnitt) zum Einlesen. Da diese Dateien miteinander zusammenhängen, sollten wir sie nach dem Einlesen zusammen in einer `list` speichern:

```
file_names = c("file2011.csv", "file2012.csv", "file2013.csv")
file_contents = lapply(setNames(file_names, file_names), read.table)

# $file2011.csv
#   id v
# 1  1 g
# 2  2 j
```

```
#
# $file2012.csv
#   id v
# 1  1 o
# 2  2 w
#
# $file2013.csv
#   id v
# 1  1 f
# 2  2 w
```

Um mit dieser Liste von Dateien zu arbeiten, überprüfen Sie zuerst die Struktur mit `str(file_contents)` und lesen Sie dann, wie Sie die Liste mit `?rbind` oder über die Liste mit `?lapply` .

Weitere Ressourcen

Schauen Sie sich `?read.table` und `?write.table` an, um dieses Beispiel zu erweitern. Ebenfalls:

- [R-Binärformate \(für Tabellen und andere Objekte\)](#)
- [Nur-Text-Tabellenformate](#)
 - durch Trennzeichen getrennte CSVs
 - durch Tabulatorzeichen getrennte TSVs
 - Formate mit fester Breite
- Sprachunabhängige binäre Tabellenformate
 - Feder
- [Ausländische Tabellen- und Tabellenformate](#)
 - SAS
 - SPSS
 - Stata
 - Excel
- [Relationale Datenbanktabellenformate](#)
 - MySQL
 - SQLite
 - PostgreSQL

Eingabe und Ausgabe online lesen: <https://riptutorial.com/de/r/topic/5543/eingabe-und-ausgabe>

Kapitel 40: Erinnerung an Beispiele

Einführung

Dieses Thema ist als Andenken an die R-Sprache ohne Text mit selbsterklärenden Beispielen gedacht.

Jedes Beispiel soll so kurz wie möglich sein.

Examples

Datentypen

Vektoren

```
a <- c(1, 2, 3)
b <- c(4, 5, 6)
mean_ab <- (a + b) / 2

d <- c(1, 0, 1)
only_1_3 <- a[d == 1]
```

Matrizen

```
mat <- matrix(c(1,2,3,4), nrow = 2, ncol = 2)
dimnames(mat) <- list(c(), c("a", "b", "c"))
mat[,] == mat
```

Datenrahmen

```
df <- data.frame(qualifiers = c("Buy", "Sell", "Sell"),
                 symbols = c("AAPL", "MSFT", "GOOGL"),
                 values = c(326.0, 598.3, 201.5))
df$symbols == df[[2]]
df$symbols == df[["symbols"]]
df[[2, 1]] == "AAPL"
```

Listen

```
l <- list(a = 500, "aaa", 98.2)
length(l) == 3
class(l[1]) == "list"
```

```
class(l[[1]]) == "numeric"
class(l$a) == "numeric"
```

Umgebungen

```
env <- new.env()
env[["foo"]] = "bar"
env2 <- env
env2[["foo"]] = "BAR"

env[["foo"]] == "BAR"
get("foo", envir = env) == "BAR"
rm("foo", envir = env)
env[["foo"]] == NULL
```

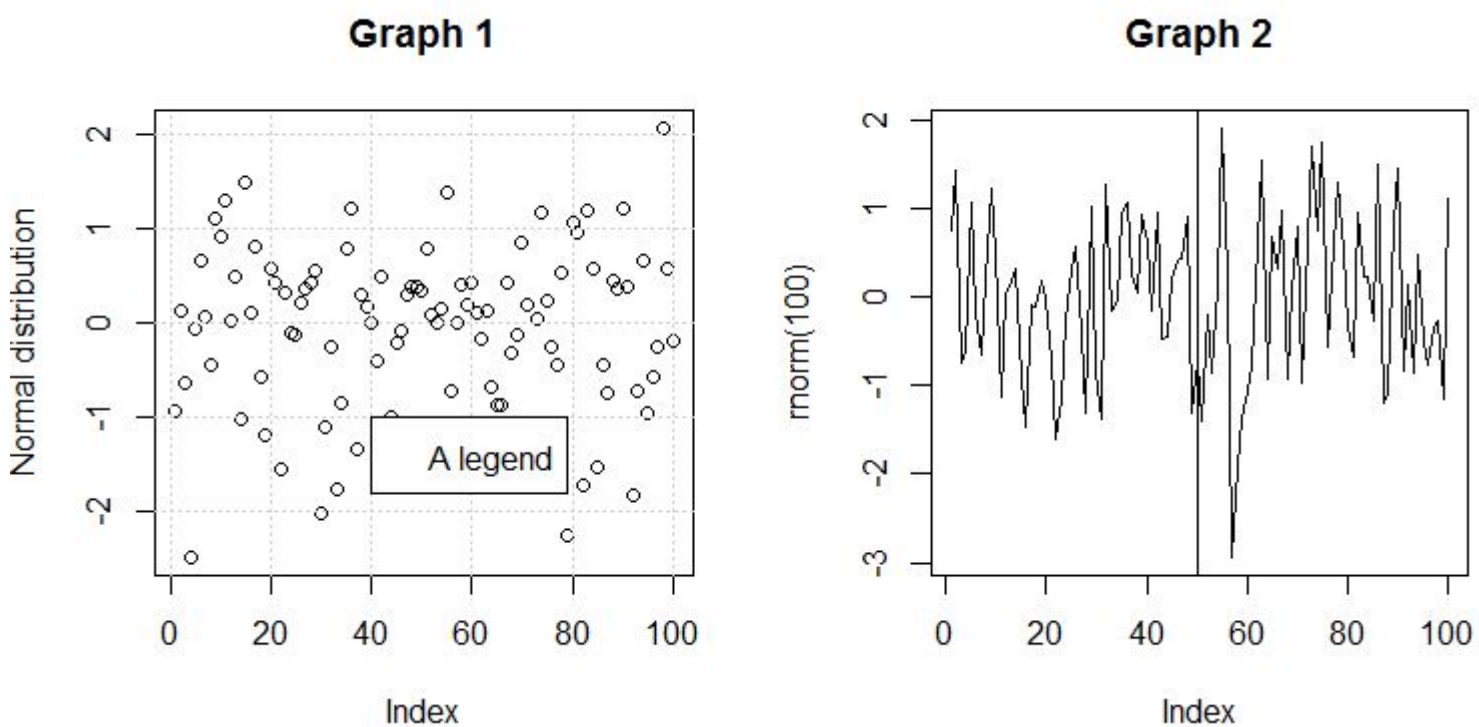
Plotten (mit Plot)

```
# Creates a 1 row - 2 columns format
par(mfrow=c(1,2))

plot(rnorm(100), main = "Graph 1", ylab = "Normal distribution")
grid()
legend(x = 40, y = -1, legend = "A legend")

plot(rnorm(100), main = "Graph 2", type = "l")
abline(v = 50)
```

Ergebnis:



Häufig verwendete Funktionen

```
# Create 100 standard normals in a vector
x <- rnorm(100, mean = 0, sd = 1)

# Find the length of a vector
length(x)

# Compute the mean
mean(x)

# Compute the standard deviation
sd(x)

# Compute the median value
median(x)

# Compute the range (min, max)
range(x)

# Sum an iterable
sum(x)

# Cumulative sum (x[1], x[1]+x[2], ...)
cumsum(x)

# Display the first 3 elements
head(3, x)

# Display min, 1st quartile, median, mean, 3rd quartile, max
summary(x)

# Compute successive difference between elements
diff(x)

# Create a range from 1 to 10 step 1
1:10

# Create a range from 1 to 10 step 0.1
seq(1, 10, 0.1)

# Print a string
print("hello world")
```

Erinnerung an Beispiele online lesen: <https://riptutorial.com/de/r/topic/10827/erinnerung-an-beispiele>

Kapitel 41: Erstellen von Berichten mit RMarkdown

Examples

Tabellen drucken

Es gibt mehrere Pakete, die die Ausgabe von Datenstrukturen in Form von HTML- oder LaTeX-Tabellen ermöglichen. Sie unterscheiden sich meist in der Flexibilität.

Hier verwende ich die Pakete:

- Knitr
- xtable
- Kuppler

Für HTML-Dokumente

```
---
title: "Printing Tables"
author: "Martin Schmelzer"
date: "29 Juli 2016"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(knitr)
library(xtable)
library(pander)
df <- mtcars[1:4,1:4]
```

# Print tables using `kable`
```{r, 'kable'}
kable(df)
```

# Print tables using `xtable`
```{r, 'xtable', results='asis'}
print(xtable(df), type="html")
```

# Print tables using `pander`
```{r, 'pander'}
pander(df)
```
```

Printing Tables

Martin Schmelzer

29 Juli 2016

Print tables using kable

```
kable(df)
```

	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160	110
Mazda RX4 Wag	21.0	6	160	110
Datsun 710	22.8	4	108	93
Hornet 4 Drive	21.4	6	258	110

Print tables using xtable

```
print(xtable(df), type="html")
```

	mpg	cyl	disp	hp
Mazda RX4	21.000000	6	160.000000	110.00
Mazda RX4 Wag	21.000000	6	160.000000	110.00
Datsun 710	22.800000	4	108.000000	93.00
Hornet 4 Drive	21.400000	6	258.000000	110.00

Print tables using pander

```
pander(df)
```

	mpg	cyl	disp	hp
Mazda RX4	21	6	160	110
Mazda RX4 Wag	21	6	160	110
Datsun 710	22.8	4	108	93
Hornet 4 Drive	21.4	6	258	110

Für PDF-Dokumente

```
---
title: "Printing Tables"
author: "Martin Schmelzer"
date: "29 Juli 2016"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(knitr)
library(xtable)
library(pander)
df <- mtcars[1:4,1:4]
```

# Print tables using `kable`
```{r, 'kable'}
kable(df)
```

# Print tables using `xtable`
```{r, 'xtable', results='asis'}
print(xtable(df, caption="My Table"))
```

# Print tables using `pander`
```{r, 'pander'}
pander(df)
```
```

Print tables using kable

```
kable(mf)
```

	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160	110
Mazda RX4 Wag	21.0	6	160	110
Datsun 710	22.8	4	108	93
Hornet 4 Drive	21.4	6	258	110

Print tables using xtable

```
print(xtable(mf, options = "My Table"))
```

% latex table generated in R 3.3.1 by xtable 1.8-2 package % Fri Jul 29 10:18:04 2016

	mpg	cyl	disp	hp
Mazda RX4	21.00	6.00	160.00	110.00
Mazda RX4 Wag	21.00	6.00	160.00	110.00
Datsun 710	22.80	4.00	108.00	93.00
Hornet 4 Drive	21.40	6.00	258.00	110.00

Table 2: My Table

Print tables using pander

```
pander(mf)
```

	mpg	cyl	disp	hp
Mazda RX4	21	6	160	110
Mazda RX4 Wag	21	6	160	110
Datsun 710	22.8	4	108	93
Hornet 4 Drive	21.4	6	258	110

Wie kann ich stoppen, dass xtable den Kommentar vor jeder Tabelle druckt?

```
options(xtable.comment = FALSE)
```

Einschließlich LaTeX Preamble-Befehle

Es gibt zwei Möglichkeiten, LaTeX-Präambelbefehle (z. B. `\usepackage`) in ein RMarkdown-Dokument aufzunehmen.

1. Verwendung der YAML-Option `header-includes` :

```
---
title: "Including LaTeX Preamble Commands in RMarkdown"
header-includes:
  - \renewcommand{\familydefault}{\cmss}
  - \usepackage[cm, slantedGreek]{sfmath}
  - \usepackage[T1]{fontenc}
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, external=T)
```

# Section 1

As you can see, this text uses the Computer Modern Font!
```

Including LaTeX Preamble Commands in RMarkdown

Section 1

As you can see, this text uses the Computer Modern Font!

2. Einschließen externer Befehle mit `includes in_header`

```
---
title: "Including LaTeX Preamble Commands in RMarkdown"
output:
  pdf_document:
    includes:
      in_header: includes.tex
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, external=T)
```

# Section 1

As you can see, this text uses the Computer Modern Font!
```

Der Inhalt von `Includes.tex` sind die gleichen drei Befehle, die wir mit `header-includes` .

Eine ganz neue Vorlage schreiben

Eine mögliche dritte Möglichkeit besteht darin, eine eigene LaTeX-Vorlage zu schreiben und sie der `template` hinzuzufügen. Dies deckt jedoch viel mehr als nur die Präambel ab.

```
---
title: "My Template"
author: "Martin Schmelzer"
output:
  pdf_document:
    template: myTemplate.tex
---
```

Einschließlich Bibliographien

Ein `bibtex`-Katalog kann leicht in die YAML-Optionsbibliographie aufgenommen werden

`bibliography`: Ein bestimmter Stil für die Bibliographie kann mit dem `biblio-style`: hinzugefügt werden `biblio-style`: Die Referenzen werden am Ende des Dokuments hinzugefügt.

```
---
title: "Including Bibliography"
author: "John Doe"
output: pdf_document
bibliography: references.bib
---

# Abstract

@R_Core_Team_2016

# References
```

Abstract

R Core Team (2016)

References

R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.

Grundlegende R-Markdown-Belegstruktur

R-Markdown-Codeabschnitte

R-Markdown ist eine Markdown-Datei mit eingebetteten Blöcken von R-Code, die als *Chunks* bezeichnet werden. Es gibt zwei Arten von R-Code- **Blöcken** : **Inline** und **Block** .

Inline- Blöcke werden mit der folgenden Syntax hinzugefügt:

```
`r 2*2`
```

Sie werden ausgewertet und ihre Ausgabe-Antwort eingefügt.

Block Stücke haben eine andere Syntax:

```
```${r name, echo=TRUE, include=TRUE, ...}  

2*2

```
```

Und sie haben verschiedene Möglichkeiten. Hier sind die wichtigsten (aber es gibt viele andere):

- **echo** (boolean) steuert, ob der Code in chunk in das Dokument aufgenommen wird
- **include** (boolean) Steuerelemente, ob die Ausgabe im Dokument enthalten sein soll
- **fig.width** (numerisch) legt die Breite der Ausgabefiguren fest
- **fig.height** (numerisch) legt die Höhe der Ausgabefiguren fest
- **fig.cap** (Zeichen) legt die Bildunterschriften fest

Sie werden in einem einfachen `tag=value` Format wie im obigen Beispiel geschrieben.

Beispiel für ein R-Markdown-Dokument

Nachfolgend finden Sie ein grundlegendes Beispiel für eine R-Markdown-Datei, die veranschaulicht, wie R-Code-Blöcke in R-Markdown eingebettet werden.

```
# Title #  
  
This is plain markdown text.
```



```

```{r code, include=FALSE, echo=FALSE}

Just declare variables

income <- 1000
taxes <- 125

...

My income is: `r income` dollars and I payed `r taxes` dollars in taxes.

Below is the sum of money I will have left:

```{r gain, include=TRUE, echo=FALSE}

gain <- income-taxes

gain

...

```{r plotOutput, include=TRUE, echo=FALSE, fig.width=6, fig.height=6}

pie(c(income,taxes), label=c("income", "taxes"))

...

```

## Konvertierung von R-Markdown in andere Formate

Mit dem R `knitr` Paket können R-Blöcke in der R-Markdown-Datei ausgewertet und in eine reguläre Markdown-Datei umgewandelt werden.

Die folgenden Schritte sind erforderlich, um die R-Markdown-Datei in pdf / html umzuwandeln:

1. Konvertieren R-Abschlags - Datei Abschlags - Datei mit `knitr` .
2. Konvertieren Sie die erhaltene Markdown-Datei mit speziellen Tools wie *Pandoc* in pdf / html.

Neben dem obigen `knitr` Paket gibt es die Wrapper-Funktionen `knit2html()` und `knit2pdf()` , mit deren Hilfe das endgültige Dokument ohne Zwischenschritt der manuellen Umwandlung in das Markdown-Format erstellt werden kann:

Wenn die obige Beispieldatei als `income.Rmd` gespeichert wurde, kann sie mit den folgenden R-Befehlen in eine `pdf` Datei konvertiert werden:

```

library(knitr)
knit2pdf("income.Rmd", "income.pdf")

```

Das endgültige Dokument wird dem folgenden ähneln.

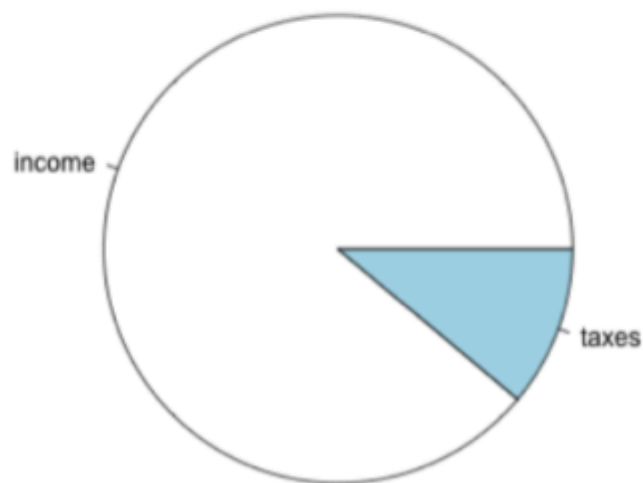
## Title

This is **plain markdown** text.

My income is: 1000 dollars and I payed 125 dollars in taxes.

Below is the sum of money I will have left:

```
[1] 875
```



Erstellen von Berichten mit RMarkdown online lesen:

<https://riptutorial.com/de/r/topic/4572/erstellen-von-berichten-mit-rmarkdown>

---

# Kapitel 42: Extrahieren und Auflisten von Dateien in komprimierten Archiven

## Examples

### Extrahieren von Dateien aus einem ZIP-Archiv

Das Entpacken eines ZIP-Archivs erfolgt mit der `unzip` Funktion aus dem Paket `utils` (in der Basis R enthalten).

```
unzip(zipfile = "bar.zip", exdir = "./foo")
```

Dadurch werden alle Dateien in `"bar.zip"` in das Verzeichnis `"foo"` extrahiert, das bei Bedarf erstellt wird. Die Tilde-Erweiterung erfolgt automatisch von Ihrem Arbeitsverzeichnis aus. Alternativ können Sie den gesamten Pfadnamen an die zip-Datei übergeben.

### Dateien in einem ZIP-Archiv auflisten

Das Auflisten der Dateien in einem ZIP-Archiv erfolgt mit der `unzip` Funktion aus dem Paket `utils` (das in der Basis R enthalten ist).

```
unzip(zipfile = "bar.zip", list = TRUE)
```

Dadurch werden alle Dateien in `"bar.zip"` und keine extrahiert. Die Tilde-Erweiterung erfolgt automatisch von Ihrem Arbeitsverzeichnis aus. Alternativ können Sie den gesamten Pfadnamen an die zip-Datei übergeben.

### Dateien in einem .tar-Archiv auflisten

Die Auflistung der Dateien in einem tar-Archiv erfolgt mit der Funktion `untar` aus dem Paket `utils` (in der Basis R enthalten).

```
untar(zipfile = "bar.tar", list = TRUE)
```

Dadurch werden alle Dateien in `"bar.tar"` und keine extrahiert. Die Tilde-Erweiterung erfolgt automatisch von Ihrem Arbeitsverzeichnis aus. Alternativ können Sie den gesamten Pfadnamen an das Tarfile übergeben.

### Extrahieren von Dateien aus einem .tar-Archiv

Das Extrahieren von Dateien aus einem tar-Archiv erfolgt mit der `untar` Funktion aus dem Paket `utils` (das in der Basis R enthalten ist).

```
untar(tarfile = "bar.tar", exdir = "./foo")
```

Dadurch werden alle Dateien in "bar.tar" in das Verzeichnis "foo" extrahiert, das bei Bedarf erstellt wird. Die Tilde-Erweiterung erfolgt automatisch von Ihrem Arbeitsverzeichnis aus. Alternativ können Sie den gesamten Pfadnamen an das Tarfile übergeben.

## Extrahieren Sie alle ZIP-Archive in einem Verzeichnis

Mit einer einfachen `for` Schleife können alle ZIP-Archive in einem Verzeichnis extrahiert werden.

```
for (i in dir(pattern=".zip$"))
 unzip(i)
```

Die Funktion `dir` erzeugt einen Zeichenvektor mit den Namen der Dateien in einem Verzeichnis, das dem durch Muster angegebenen Regexp- `pattern` . Dieser Vektor wird mit dem Index `i` durchgeschleift, wobei die Funktion zum `unzip` ZIP-Archive verwendet wird.

Extrahieren und Auflisten von Dateien in komprimierten Archiven online lesen:

<https://riptutorial.com/de/r/topic/4323/extrahieren-und-auflisten-von-dateien-in-komprimierten-archiven>

# Kapitel 43: Faktoren

## Syntax

1. Faktor (x = Zeichen (), Ebenen, Beschriftungen = Ebenen, ausschließen = NA, geordnet = ist.geordnet (X), nmax = NA)
2. Run `?factor` oder Online- [Dokumentation](#) .

## Bemerkungen

Ein Objekt mit dem Klasse - `factor` ist ein Vektor mit einem bestimmten Satz von Eigenschaften.

1. Es wird intern als `integer` gespeichert.
2. Es verwaltet ein `levels` , das die Zeichendarstellung der Werte zeigt.
3. Ihre Klasse wird als `factor` gespeichert

Lassen Sie uns zur Veranschaulichung aus einem Satz von Farben einen Vektor von 1.000 Beobachtungen erzeugen.

```
set.seed(1)
Color <- sample(x = c("Red", "Blue", "Green", "Yellow"),
 size = 1000,
 replace = TRUE)
Color <- factor(Color)
```

Wir können jede der oben aufgeführten Eigenschaften von `Color` beobachten:

```
1. It is stored internally as an `integer` vector
typeof(Color)
```

```
[1] "integer"
```

```
2. It maintains a `levels` attribute the shows the character representation of the values.
3. Its class is stored as `factor`
attributes(Color)
```

```
$levels
[1] "Blue" "Green" "Red" "Yellow"

$class
[1] "factor"
```

Der Hauptvorteil eines Faktorobjekts ist die Effizienz bei der Datenspeicherung. Eine Ganzzahl erfordert weniger Speicherplatz als ein Zeichen. Eine solche Effizienz war äußerst wünschenswert, wenn viele Computer über weitaus geringere Ressourcen als derzeitige Computer verfügten (für eine genauere Beschreibung der Gründe für die Verwendung von Faktoren siehe [stringsAsFactors : eine nicht autorisierte Biografie](#) ). Der Unterschied bei der

Speichernutzung ist sogar in unserem `Color` Objekt sichtbar. Wie Sie sehen können, erfordert das Speichern von `Color` als Zeichen etwa 1,7-mal so viel Speicher wie das Faktorobjekt.

```
Amount of memory required to store Color as a factor.
object.size(Color)
```

```
4624 bytes
```

```
Amount of memory required to store Color as a character
object.size(as.character(Color))
```

```
8232 bytes
```

---

## Die Ganzzahl der Ebene zuordnen

Während die interne Berechnung von Faktoren das Objekt als Ganzzahl sieht, ist die gewünschte Darstellung für den menschlichen Konsum die Zeichenebene. Zum Beispiel,

```
head(Color)
```

```
[1] Blue Blue Green Yellow Red Yellow
Levels: Blue Green Red Yellow
```

ist für das menschliche Verständnis einfacher als

```
head(as.numeric(Color))
```

```
[1] 1 1 2 4 3 4
```

Eine ungefähre Darstellung, wie R die Zeichendarstellung an den internen Integerwert anpasst, ist:

```
head(levels(Color)[as.numeric(Color)])
```

```
[1] "Blue" "Blue" "Green" "Yellow" "Red" "Yellow"
```

Vergleichen Sie diese Ergebnisse mit

```
head(Color)
```

```
[1] Blue Blue Green Yellow Red Yellow
Levels: Blue Green Red Yellow
```

# Moderne Nutzung von Faktoren

Im Jahr 2007 führte R eine Hashmethode für Zeichen ein, um die Speicherlast von Zeichenvektoren zu reduzieren (ref: [stringsAsFactors : a Unauthorized Biography](#) ). Wenn wir festgestellt haben, dass Zeichen 1,7-mal mehr Speicherplatz benötigen als Faktoren, wurde dies in einer aktuellen Version von R berechnet, was bedeutet, dass der Speicherverbrauch von Zeichenvektoren vor 2007 noch mehr belastend war.

Aufgrund des Hash-Verfahrens in modernen R und der weitaus größeren Speicherressourcen in modernen Computern wurde das Problem der Speichereffizienz beim Speichern von Zeichenwerten auf ein sehr kleines Problem reduziert. Die vorherrschende Einstellung in der R-Gemeinschaft ist in den meisten Situationen eine Bevorzugung von Zeichenvektoren gegenüber Faktoren. Die Hauptursachen für die Abkehr von Faktoren sind

1. Die Zunahme unstrukturierter und / oder lose kontrollierter Zeichendaten
2. Die Tendenz von Faktoren, sich nicht wie gewünscht zu verhalten, wenn der Benutzer vergisst, dass er mit einem Faktor und nicht mit einem Charakter zu tun hat

Im ersten Fall macht es keinen Sinn, Freitext oder offene Antwortfelder als Faktoren zu speichern, da es unwahrscheinlich ist, dass ein Muster vorhanden ist, das mehr als eine Beobachtung pro Ebene zulässt. Alternativ, wenn die Datenstruktur nicht sorgfältig gesteuert wird, ist es möglich, mehrere Ebenen zu erhalten, die derselben Kategorie entsprechen (z. B. "blau", "blau" und "blau"). In solchen Fällen ziehen es viele vor, diese Diskrepanzen als Zeichen zu verwalten, bevor sie in einen Faktor umgewandelt werden (wenn überhaupt eine Konvertierung erfolgt).

Im zweiten Fall, wenn der Benutzer der Meinung ist, dass er mit einem Zeichenvektor arbeitet, reagieren bestimmte Methoden möglicherweise nicht wie erwartet. Dieses grundlegende Verständnis kann beim Debuggen von Skripts und Codes zu Verwirrung und Frustration führen. Genau genommen kann dies als Verschulden des Benutzers betrachtet werden. Die meisten Benutzer verzichten jedoch gerne auf Faktoren und vermeiden diese Situationen insgesamt.

## Examples

### Grundlegende Faktorenbildung

Faktoren sind eine Möglichkeit, kategoriale Variablen in R darzustellen. Ein Faktor wird intern als **Vektor von Ganzzahlen** gespeichert. Die eindeutigen Elemente des gelieferten Zeichenvektors werden als *Stufen* des Faktors bezeichnet. Wenn die Ebenen nicht vom Benutzer angegeben werden, generiert R standardmäßig die Menge der eindeutigen Werte im Vektor, sortiert diese Werte alphanumerisch und verwendet sie als Ebenen.

```
charvar <- rep(c("n", "c"), each = 3)
f <- factor(charvar)
f
levels(f)

> f
```

```
[1] n n n c c c
Levels: c n
> levels(f)
[1] "c" "n"
```

Wenn Sie die Reihenfolge der Ebenen ändern möchten, können Sie die Ebenen manuell mit einer Option festlegen:

```
levels(factor(charvar, levels = c("n", "c")))

> levels(factor(charvar, levels = c("n", "c")))
[1] "n" "c"
```

Faktoren haben eine Reihe von Eigenschaften. Beispielsweise können Ebenen Beschriftungen erhalten:

```
> f <- factor(charvar, levels=c("n", "c"), labels=c("Newt", "Capybara"))
> f
[1] Newt Newt Newt Capybara Capybara Capybara
Levels: Newt Capybara
```

Eine weitere Eigenschaft, die zugewiesen werden kann, ist die Anordnung des Faktors:

```
> Weekdays <- factor(c("Monday", "Wednesday", "Thursday", "Tuesday", "Friday", "Sunday",
"Saturday"))
> Weekdays
[1] Monday Wednesday Thursday Tuesday Friday Sunday Saturday
Levels: Friday Monday Saturday Sunday Thursday Tuesday Wednesday
> Weekdays <- factor(Weekdays, levels=c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"), ordered=TRUE)
> Weekdays
[1] Monday Wednesday Thursday Tuesday Friday Sunday Saturday
Levels: Monday < Tuesday < Wednesday < Thursday < Friday < Saturday < Sunday
```

Wenn eine Stufe des Faktors nicht mehr verwendet wird, können Sie sie mit der Funktion `droplevels()` :

```
> Weekend <- subset(Weekdays, Weekdays == "Saturday" | Weekdays == "Sunday")
> Weekend
[1] Sunday Saturday
Levels: Monday < Tuesday < Wednesday < Thursday < Friday < Saturday < Sunday
> Weekend <- droplevels(Weekend)
> Weekend
[1] Sunday Saturday
Levels: Saturday < Sunday
```

## Faktorstufen mit einer Liste konsolidieren

Es gibt Zeiten, in denen es wünschenswert ist, die Faktorstufen in weniger Gruppen zusammenzufassen, möglicherweise aufgrund spärlicher Daten in einer der Kategorien. Es kann auch vorkommen, wenn Sie unterschiedliche Schreibweisen oder Großschreibung für die Kategorienamen verwenden. Betrachten Sie den Faktor als Beispiel



```
set.seed(1)
colorful <- sample(c("red", "Red", "RED", "blue", "Blue", "BLUE", "green", "gren"),
 size = 20,
 replace = TRUE)
colorful <- factor(colorful)
```

Da R die Groß- und Kleinschreibung berücksichtigt, erscheint eine Frequenztabelle dieses Vektors wie folgt.

```
table(colorful)
```

```
colorful
blue Blue BLUE green gren red Red RED
 3 1 4 2 4 1 3 2
```

Diese Tabelle stellt jedoch nicht die tatsächliche Verteilung der Daten dar, und die Kategorien können effektiv auf drei Typen reduziert werden: Blau, Grün und Rot. Drei Beispiele werden bereitgestellt. Die erste veranschaulicht eine scheinbar naheliegende Lösung, die jedoch keine wirkliche Lösung bietet. Die zweite gibt eine funktionierende Lösung, ist aber ausführlich und rechenintensiv. Die dritte ist keine offensichtliche Lösung, aber sie ist relativ kompakt und recheneffizient.

## Ebenen über `factor` konsolidieren ( `factor_approach` )

```
factor(as.character(colorful),
 levels = c("blue", "Blue", "BLUE", "green", "gren", "red", "Red", "RED"),
 labels = c("Blue", "Blue", "Blue", "Green", "Green", "Red", "Red", "Red"))
```

```
[1] Green Blue Red Red Blue Red Red Red Blue Red Green Green Green
Blue Red Green
[17] Red Green Green Red
Levels: Blue Blue Blue Green Green Red Red Red
Warning message:
In `levels<-`(`*tmp*`, value = if (nl == nL) as.character(labels) else
paste0(labels, " :
 duplicated levels in factors are deprecated
```

Beachten Sie, dass es doppelte Ebenen gibt. Wir haben immer noch drei Kategorien für "Blau", was unsere Aufgabe der Konsolidierung der Ebenen nicht vollständig erfüllt. Darüber hinaus gibt es eine Warnung, dass doppelte Ebenen veraltet sind, was bedeutet, dass dieser Code in Zukunft einen Fehler erzeugen kann.

## Ebenen mit `ifelse` konsolidieren ( `ifelse_approach` )

```
factor(ifelse(colorful %in% c("blue", "Blue", "BLUE"),
 "Blue",
 ifelse(colorful %in% c("green", "gren"),
 "Green",
 "Red")))
```

```
[1] Green Blue Red Red Blue Red Red Red Blue Red Green Green Green
Blue Red Green
[17] Red Green Green Red
Levels: Blue Green Red
```

Dieser Code generiert das gewünschte Ergebnis, erfordert jedoch die Verwendung geschachtelter `ifelse` Anweisungen. Obwohl an diesem Ansatz nichts falsch ist, kann das Verwalten verschachtelter `ifelse` Anweisungen eine langwierige Aufgabe sein und muss sorgfältig ausgeführt werden.

## `list_approach` mit einer Liste `list_approach ( list_approach )`

Eine weniger offensichtliche Möglichkeit zur Konsolidierung von Ebenen besteht in der Verwendung einer Liste, in der der Name jedes Elements der gewünschte Kategorienname ist und das Element ein Zeichenvektor der Ebenen des Faktors ist, der der gewünschten Kategorie zugeordnet werden sollte. Dies hat den zusätzlichen Vorteil, dass Sie direkt mit dem `levels` des Faktors arbeiten, ohne neue Objekte zuweisen zu müssen.

```
levels(colorful) <-
 list("Blue" = c("blue", "Blue", "BLUE"),
 "Green" = c("green", "gren"),
 "Red" = c("red", "Red", "RED"))
```

```
[1] Green Blue Red Red Blue Red Red Red Blue Red Green Green Green
Blue Red Green
[17] Red Green Green Red
Levels: Blue Green Red
```

## Benchmarking für jeden Ansatz

Die für die Ausführung dieser Ansätze erforderliche Zeit ist unten zusammengefasst. (Aus Platzgründen wird der Code zum Generieren dieser Zusammenfassung nicht angezeigt.)

```
Unit: microseconds
 expr min lq mean median uq max neval cld
 factor 78.725 83.256 93.26023 87.5030 97.131 218.899 100 b
 ifelse 104.494 107.609 123.53793 113.4145 128.281 254.580 100 c
list_approach 49.557 52.955 60.50756 54.9370 65.132 138.193 100 a
```

Der `ifelse` etwa doppelt so schnell wie der `list_approach` Ansatz. Mit Ausnahme von sehr, sehr großen Datenmengen werden die Unterschiede in der Ausführungszeit jedoch wahrscheinlich entweder in Mikrosekunden oder Millisekunden gemessen. Bei so kleinen Zeitunterschieden muss die Effizienz nicht die Entscheidung über den zu verwendenden Ansatz bestimmen. Verwenden Sie stattdessen einen vertrauten und komfortablen Ansatz, den Sie und Ihre Mitarbeiter bei der zukünftigen Überprüfung verstehen werden.

## Faktoren

**Faktoren** sind eine Methode zur Darstellung kategorialer Variablen in R. Wenn ein Vektor `x` dessen Werte mit `as.character()` in Zeichen umgewandelt werden können, `as.character()` die Standardargumente für `factor()` und `as.factor()` jedem einzelnen Element von `x` eine Ganzzahl zu der Vektor sowie ein Ebenenattribut und ein Beschriftungsattribut. `Level` sind die Werte, die `x` möglicherweise annehmen kann, und `Labels` können entweder das angegebene Element sein oder vom Benutzer festgelegt werden.

Um zu zeigen, wie Faktoren funktionieren, erstellen wir einen Faktor mit Standardattributen, benutzerdefinierten Ebenen und benutzerdefinierten Ebenen und Beschriftungen.

```
standard
factor(c(1,1,2,2,3,3))
[1] 1 1 2 2 3 3
Levels: 1 2 3
```

Es kann vorkommen, dass der Benutzer weiß, dass die Anzahl der möglichen Werte, die ein Faktor annehmen kann, größer ist als die aktuellen Werte im Vektor. Dazu weisen wir die Stufen selbst in `factor()` .

```
factor(c(1,1,2,2,3,3),
 levels = c(1,2,3,4,5))
[1] 1 1 2 2 3 3
Levels: 1 2 3 4 5
```

Zu Stilzwecken möchte der Benutzer möglicherweise jeder Ebene Beschriftungen zuweisen. Standardmäßig sind Beschriftungen die Zeichendarstellung der Ebenen. Hier weisen wir für jede der möglichen Ebenen im Faktor Beschriftungen zu.

```
factor(c(1,1,2,2,3,3),
 levels = c(1,2,3,4,5),
 labels = c("Fox", "Dog", "Cow", "Brick", "Dolphin"))
[1] Fox Fox Dog Dog Cow Cow
Levels: Fox Dog Cow Brick Dolphin
```

Normalerweise können Faktoren nur mit `==` und `!=` Verglichen werden, wenn die Faktoren die gleichen Pegel haben. Der folgende Vergleich von Faktoren schlägt fehl, obwohl sie gleich erscheinen, da die Faktoren unterschiedliche Faktorstufen haben.

```
factor(c(1,1,2,2,3,3), levels = c(1,2,3)) == factor(c(1,1,2,2,3,3), levels = c(1,2,3,4,5))
Error in Ops.factor(factor(c(1, 1, 2, 2, 3, 3), levels = c(1, 2, 3)), :
 level sets of factors are different
```

Dies ist sinnvoll, da aufgrund der zusätzlichen Pegel in der RHS nicht genügend Informationen zu jedem Faktor vorhanden sind, um sie sinnvoll miteinander zu vergleichen.

Die Operatoren `<`, `<=`, `>` und `>=` nur für geordnete Faktoren verwendet werden. Diese können kategoriale Werte darstellen, die noch eine lineare Reihenfolge haben. Ein geordneter Faktor kann erstellt werden, indem für die `ordered = TRUE` das `ordered = TRUE` Argument `factor` wird oder einfach die `ordered` Funktion verwendet wird.

```
x <- factor(1:3, labels = c('low', 'medium', 'high'), ordered = TRUE)
print(x)
[1] low medium high
Levels: low < medium < high

y <- ordered(3:1, labels = c('low', 'medium', 'high'))
print(y)
[1] high medium low
Levels: low < medium < high

x < y
[1] TRUE FALSE FALSE
```

Weitere Informationen finden Sie in der [Factor-Dokumentation](#) .

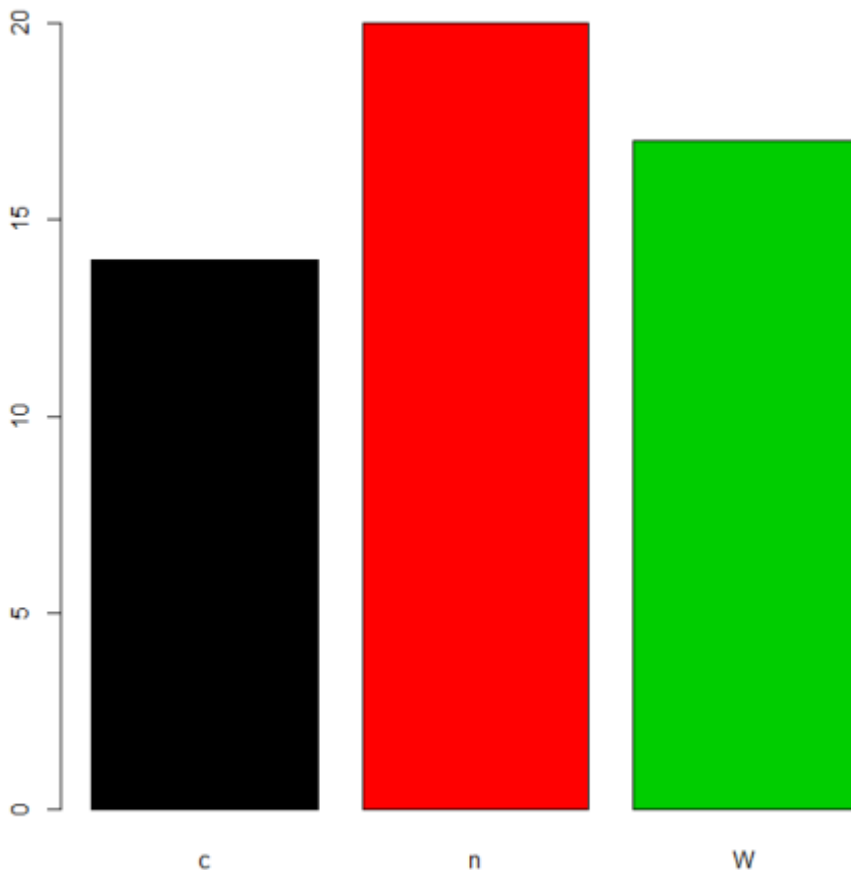
## Faktoren ändern und neu ordnen

Wenn Faktoren mit `as.character` erstellt werden, werden die `levels` von `as.character` auf die Eingaben angewendeten `as.character` gebildet und alphabetisch geordnet.

```
charvar <- rep(c("W", "n", "c"), times=c(17,20,14))
f <- factor(charvar)
levels(f)
[1] "c" "n" "W"
```

In einigen Situationen ist die Behandlung der Standardreihenfolge der `levels` (alphabetische / lexikalische Reihenfolge) akzeptabel. Zum Beispiel, wenn man justs wollen `plot` die Frequenzen, wird dies das Ergebnis sein:

```
plot(f, col=1:length(levels(f)))
```



Wenn Sie jedoch eine andere Reihenfolge der `levels` wünschen, müssen Sie dies im Parameter "`levels`" oder "`labels`" angeben (wobei darauf zu achten ist, dass sich die Bedeutung von "Reihenfolge" hier von den *geordneten* Faktoren unterscheidet, siehe unten). Es gibt viele Alternativen, um diese Aufgabe abhängig von der Situation auszuführen.

### 1. Definieren Sie den Faktor neu

Wenn es möglich ist, können wir den Faktor mithilfe des `levels` in der gewünschten Reihenfolge neu erstellen.

```
ff <- factor(charvar, levels = c("n", "W", "c"))
levels(ff)
[1] "n" "W" "c"

gg <- factor(charvar, levels = c("W", "c", "n"))
levels(gg)
[1] "W" "c" "n"
```

Wenn sich die Eingangspegel von den gewünschten Ausgangspegeln unterscheiden, verwenden wir den `labels` Parameter, der bewirkt, dass der `levels` Parameter ein "Filter" für akzeptable Eingabewerte wird, aber die endgültigen Werte für "Level" für den Faktor-Vektor als Argument belassen `labels` :

```
fm <- factor(as.numeric(f), levels = c(2,3,1),
 labels = c("nn", "WW", "cc"))
levels(fm)
[1] "nn" "WW" "cc"

fm <- factor(LETTERS[1:6], levels = LETTERS[1:4], # only 'A'-'D' as input
 labels = letters[1:4]) # but assigned to 'a'-'d'
fm
[1] a b c d <NA> <NA>
#Levels: a b c d
```

## 2. Verwenden Sie die `relevel` Funktion

Wenn eine bestimmte `level` die erste sein muss, können wir `relevel`. Dies geschieht beispielsweise im Rahmen der statistischen Analyse, wenn eine `base` zum Testen der Hypothese erforderlich ist.

```
g<-relevel(f, "n") # moves n to be the first level
levels(g)
[1] "n" "c" "W"
```

Wie zu bestätigen ist, sind `f` und `g` gleich

```
all.equal(f, g)
[1] "Attributes: < Component \"levels\": 2 string mismatches >"
all.equal(f, g, check.attributes = F)
[1] TRUE
```

## 3. Umordnungsfaktoren

Es gibt Fälle, in denen wir müssen `reorder` die `levels` auf einer Reihe basieren, ein Teilergebnis, eine berechnete Statistik oder frühere Berechnungen. Lassen Sie uns basierend auf den **Frequenzen** der `levels` neu ordnen

```
table(g)
g
n c W
20 14 17
```

Die `reorder` ist generisch (siehe `help(reorder)`), benötigt aber in diesem Zusammenhang: `x`, in diesem Fall den Faktor; `x` ein numerischer Wert der gleichen Länge wie `x`; und `FUN`, eine Funktion, die auf `x` anzuwenden ist und nach dem Pegel des `x` berechnet wird, der die Reihenfolge der `levels`, standardmäßig erhöht. Das Ergebnis ist derselbe Faktor, wenn die Ebenen neu angeordnet werden.

```
g.ord <- reorder(g, rep(1, length(g)), FUN=sum) #increasing
levels(g.ord)
[1] "c" "W" "n"
```

Um die Reihenfolge zu verringern, betrachten wir negative Werte (`-1`)

```
g.ord.d <- reorder(g,rep(-1,length(g)), FUN=sum)
levels(g.ord.d)
[1] "n" "W" "c"
```

Wieder ist der Faktor derselbe wie bei den anderen.

```
data.frame(f,g,g.ord,g.ord.d)[seq(1,length(g),by=5),] #just same lines
f g g.ord g.ord.d
1 W W W W
6 W W W W
11 W W W W
16 W W W W
21 n n n n
26 n n n n
31 n n n n
36 n n n n
41 c c c c
46 c c c c
51 c c c c
```

Wenn es eine **quantitative Variable** gibt, die sich auf die Faktorvariable bezieht, könnten wir andere Funktionen verwenden, um die `levels` neu zu ordnen. Lets nehmen die `iris` (`help("iris")` für weitere Informationen), zum Neuordnen der `Species` Faktor sein Mittelwert mit `Sepal.Width`.

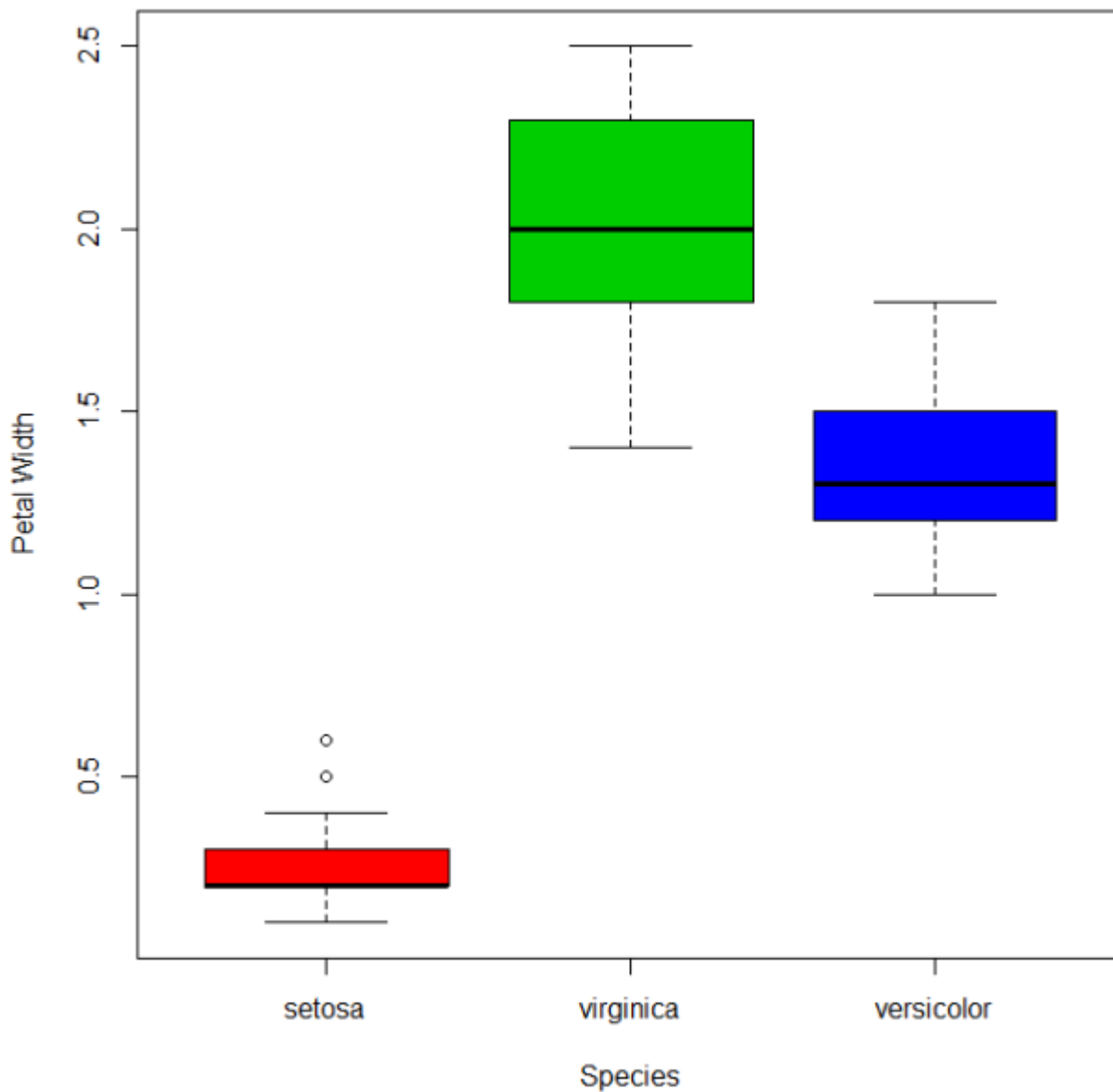
```
miris <- iris #help("iris") # copy the data
with(miris, tapply(Sepal.Width,Species,mean))
setosa versicolor virginica
3.428 2.770 2.974

miris$Species.o<-with(miris,reorder(Species,-Sepal.Width))
levels(miris$Species.o)
[1] "setosa" "virginica" "versicolor"
```

Das übliche `boxplot` (sprich: `with(miris, boxplot(Petal.Width~Species))`) zeigt die Spezies in dieser Reihenfolge: *setosa*, *versicolor* und *virginica*. `Sepal.Width` den geordneten Faktor verwenden, erhalten wir die Art nach ihrem mittleren `Sepal.Width`:

```
boxplot(Petal.Width~Species.o, data = miris,
 xlab = "Species", ylab = "Petal Width",
 main = "Iris Data, ordered by mean sepal width", varwidth = TRUE,
 col = 2:4)
```

Iris Data, ordered by mean sepal width



Außerdem können Sie die Namen von `levels` ändern, sie in Gruppen zusammenfassen oder neue `levels` hinzufügen. Dafür verwenden wir die Funktion der gleichen `levels`.

```
f1<-f
levels(f1)
[1] "c" "n" "w"
levels(f1) <- c("upper","upper","CAP") #rename and grouping
levels(f1)
[1] "upper" "CAP"

f2<-f1
levels(f2) <- c("upper","CAP", "Number") #add Number level, which is empty
levels(f2)
[1] "upper" "CAP" "Number"
f2[length(f2):(length(f2)+5)]<- "Number" # add cases for the new level
table(f2)
f2
```



```
upper CAP Number
33 17 6

f3<-f1
levels(f3) <- list(G1 = "upper", G2 = "CAP", G3 = "Number") # The same using list
levels(f3)
[1] "G1" "G2" "G3"
f3[length(f3):(length(f3)+6)]<-"G3" ## add cases for the new level
table(f3)
f3
G1 G2 G3
33 17 7
```

## - geordnete Faktoren

Schließlich wissen wir, dass `ordered` Faktoren sich von den `factors` unterscheiden. Der erste wird zur Darstellung von *Ordinaldaten verwendet*, der zweite für die Arbeit mit *Solldaten*. Zunächst ist es nicht sinnvoll, die Reihenfolge der `levels` für geordnete Faktoren zu ändern, aber wir können die `labels` ändern.

```
ordvar<-rep(c("Low", "Medium", "High"), times=c(7,2,4))

of<-ordered(ordvar,levels=c("Low", "Medium", "High"))
levels(of)
[1] "Low" "Medium" "High"

of1<-of
levels(of1)<- c("LOW", "MEDIUM", "HIGH")
levels(of1)
[1] "LOW" "MEDIUM" "HIGH"
is.ordered(of1)
[1] TRUE
of1
[1] LOW LOW LOW LOW LOW LOW LOW LOW MEDIUM MEDIUM HIGH HIGH HIGH HIGH

Levels: LOW < MEDIUM < HIGH
```

## Wiederherstellungsfaktoren von Null

### Problem

Faktoren werden verwendet, um Variablen darzustellen, die Werte aus einer Reihe von Kategorien annehmen, die als Pegel in R bekannt sind. Zum Beispiel könnte ein Experiment durch das Energieniveau einer Batterie mit vier Stufen charakterisiert werden: leer, niedrig, normal und voll. Für fünf verschiedene Probenahmestellen könnten diese Werte wie folgt definiert werden:

**voll , voll , normal , leer , niedrig**

In Datenbanken oder anderen Informationsquellen erfolgt die Handhabung dieser Daten normalerweise über willkürliche, ganzzahlige Indizes, die den Kategorien oder Ebenen zugeordnet sind. Wenn wir davon ausgehen, dass wir für das gegebene Beispiel die Indizes wie folgt zuordnen würden: 1 = leer, 2 = niedrig, 3 = normal, 4 = voll, dann könnten die 5 Abtastwerte wie folgt codiert werden:

**4, 4, 3, 1, 2**

Es kann vorkommen, dass Sie aus Ihrer Informationsquelle, z. B. einer Datenbank, nur die codierte Liste von Ganzzahlen und den Katalog haben, der jede Ganzzahl mit jedem Stufenschlüsselwort verknüpft. Wie kann aus diesen Informationen ein Faktor von R rekonstruiert werden?

## Lösung

Wir simulieren einen Vektor von 20 Ganzzahlen, der die Stichproben darstellt, von denen jede einen von vier verschiedenen Werten haben kann:

```
set.seed(18)
ii <- sample(1:4, 20, replace=T)
ii
```

```
[1] 4 3 4 1 1 3 2 3 2 1 3 4 1 2 4 1 1
```

Der erste Schritt besteht darin, aus der vorherigen Reihenfolge einen Faktor zu erstellen, in dem die Ebenen oder Kategorien genau die Zahlen von 1 bis 4 sind.

```
fii <- factor(ii, levels=1:4) # it is necessary to indicate the numeric levels
fii
```

```
[1] 4 3 4 1 1 3 2 3 2 1 3 4 1 2 4 1 1
Stufen: 1 2 3 4
```

Jetzt einfach, müssen Sie den Faktor *kleiden* bereits mit dem Index - Tags erstellt:

```
levels(fii) <- c("empty", "low", "normal", "full")
fii
```

```
[1] full normal full empty leer normal niedrig normal niedrig leer
[11] normal voll leer niedrig voll leer normal leer voll leer
Levels: leer niedrig normal voll
```

Faktoren online lesen: <https://riptutorial.com/de/r/topic/1104/faktoren>

---

# Kapitel 44: Farbschemata für Grafiken

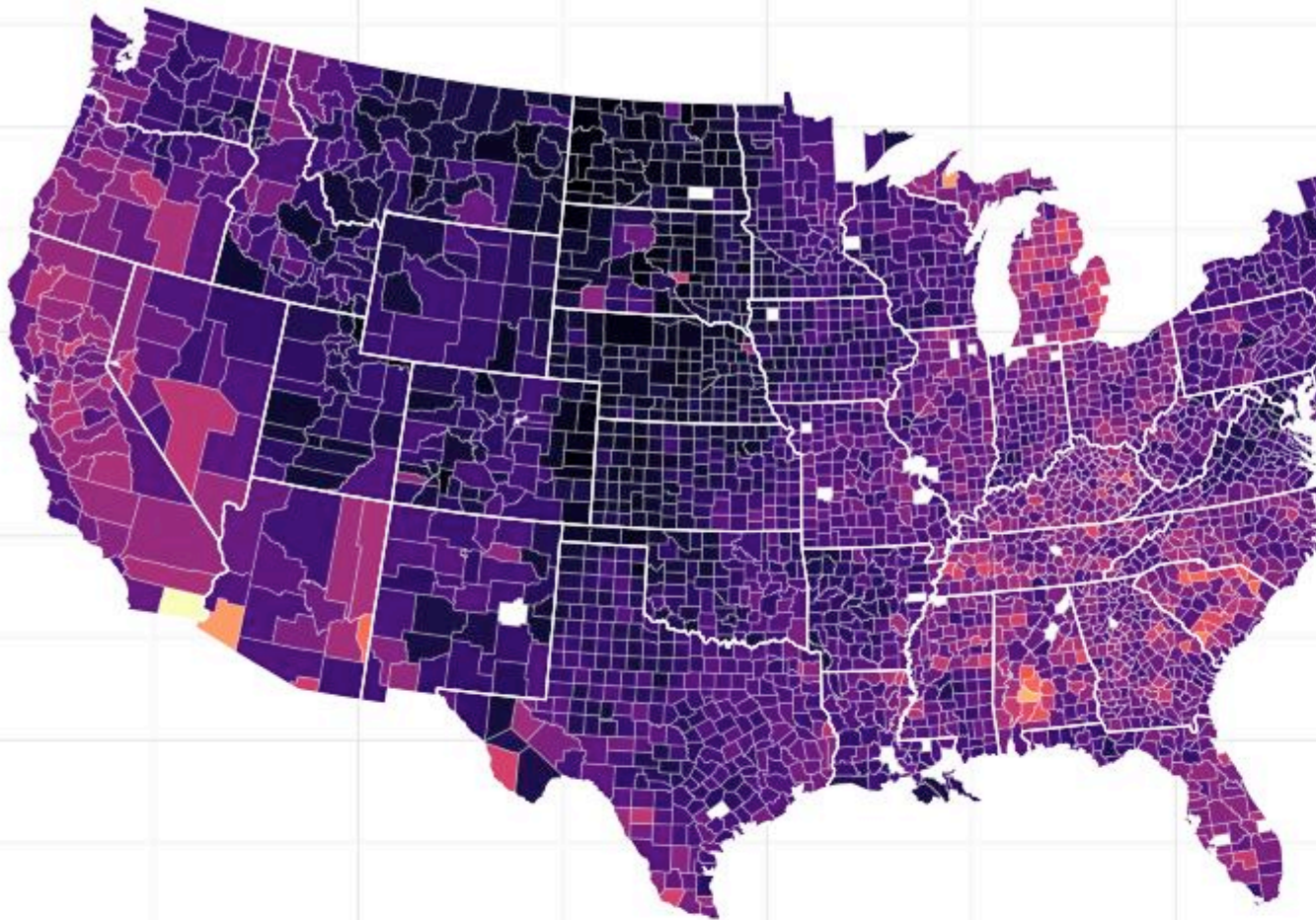
## Examples

### viridis - druck- und farblindfreundliche Paletten

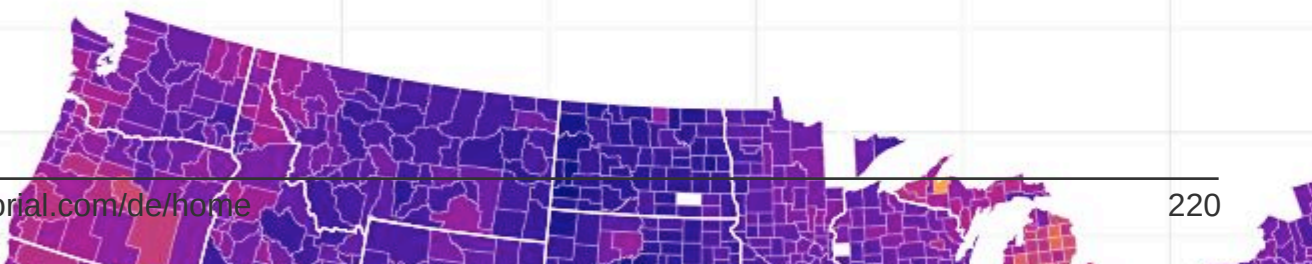
Viridis (benannt nach dem [Chromis viridis-Fisch](#) ) ist ein kürzlich [entwickeltes Farbschema für die Python-Bibliothek matplotlib](#) (die Videopräsentation über den [Link](#) erklärt, wie das Farbschema entwickelt wurde und welche Hauptvorteile es bietet). Es wird nahtlos nach `R` portiert.

Es gibt vier Varianten von Farbschemata: `magma` , `plasma` , `inferno` und `viridis` (Standard). Sie werden mit dem `option` und entsprechend als `A` , `B` , `C` und `D` codiert. Um einen Eindruck von den 4 Farbschemata zu erhalten, schauen Sie sich die Karten an:

option A aka 'magma'



option C aka 'plasma'



( [Bildquelle](#) )

---

Das Paket kann von [CRAN](#) oder [Github](#) installiert werden.

---

Die [Vignette](#) für das `viridis` Paket ist einfach genial.

---

Ein `ggplot2` Merkmal des `viridis` Farbschemas ist die Integration mit `ggplot2` . Innerhalb des Pakets sind zwei `ggplot2` Funktionen definiert: `scale_color_viridis()` und `scale_fill_viridis()` . Siehe das Beispiel unten:

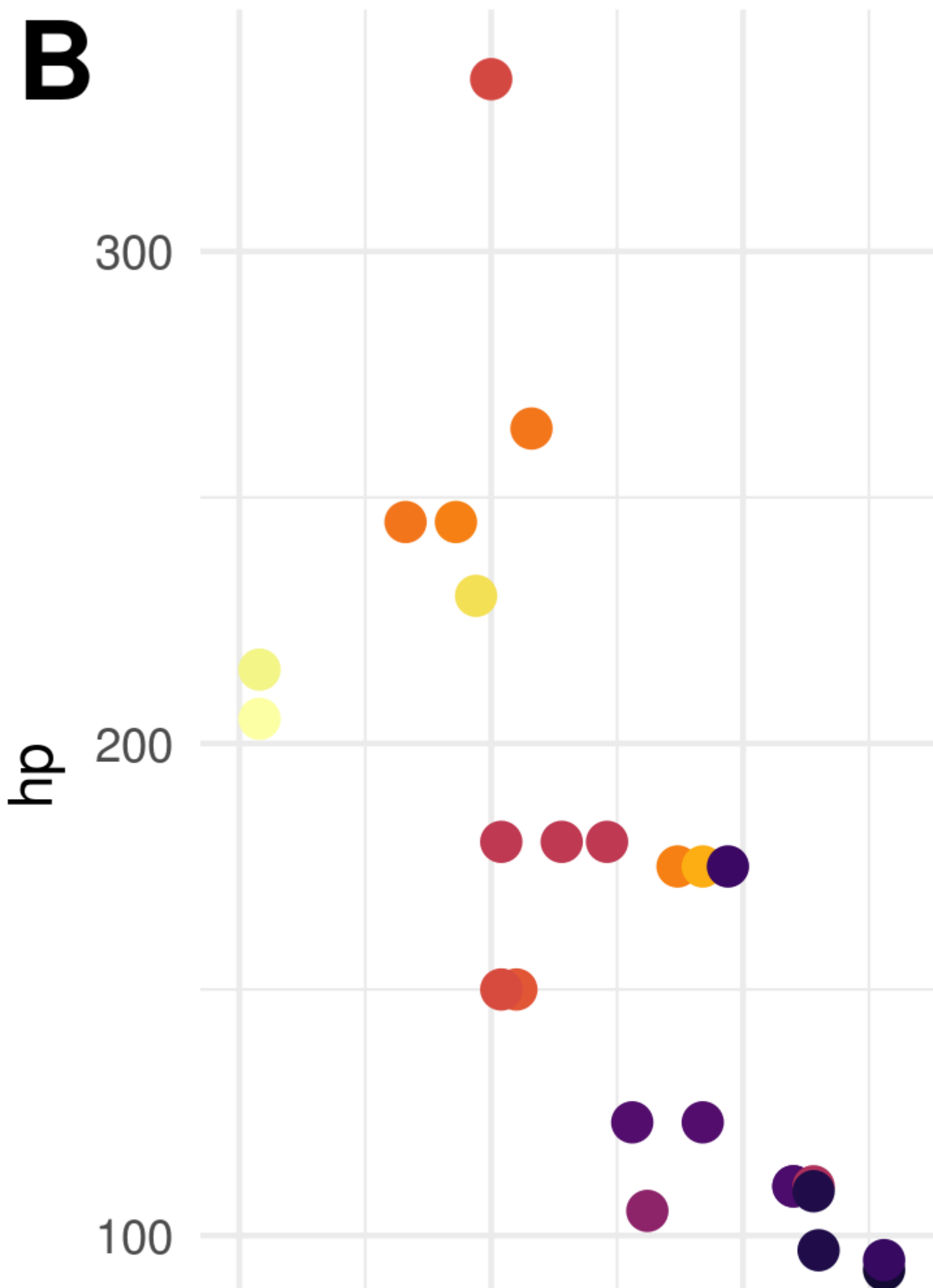
```
library(viridis)
library(ggplot2)

gg1 <- ggplot(mtcars)+
 geom_point(aes(x = mpg, y = hp, color = disp), size = 3)+
 scale_color_viridis(option = "B")+
 theme_minimal()+
 theme(legend.position = c(.8,.8))

gg2 <- ggplot(mtcars)+
 geom_violin(aes(x = factor(cyl), y = hp, fill = factor(cyl)))+
 scale_fill_viridis(discrete = T)+
 theme_minimal()+
 theme(legend.position = 'none')

library(cowplot)
output <- plot_grid(gg1,gg2, labels = c('B','D'),label_size = 20)
print(output)
```

# B



Projekt ist ein sehr beliebtes Werkzeug, um harmonisch passende Farbpaletten auszuwählen. `RColorBrewer` ist ein Port des Projekts für `R` und bietet auch farbenblindfreundliche Paletten an.

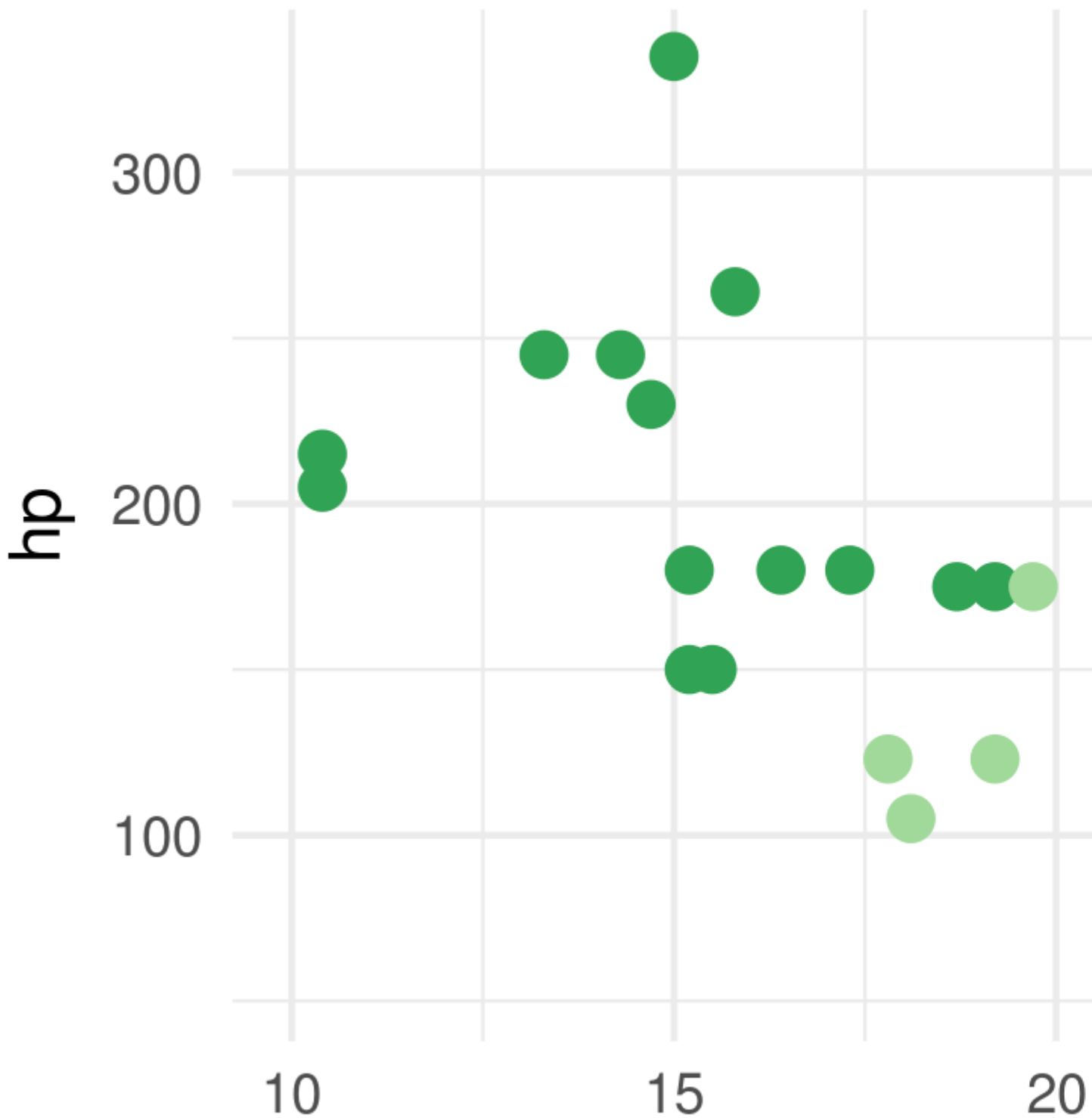
---

## Ein Anwendungsbeispiel

```
colors_vec <- brewer.pal(5, name = 'BrBG')
print(colors_vec)
[1] "#A6611A" "#DFC27D" "#F5F5F5" "#80CDC1" "#018571"
```

`RColorBrewer` erstellt Farboptionen für `ggplot2`: `scale_color_brewer` und `scale_fill_brewer`.

```
library(ggplot2)
ggplot(mtcars) +
 geom_point(aes(x = mpg, y = hp, color = factor(cyl)), size = 3) +
 scale_color_brewer(palette = 'Greens') +
 theme_minimal() +
 theme(legend.position = c(.8, .8))
```



## Eine praktische Funktion, um einen Farbvektor zu erkennen

Häufig muss die ausgewählte Farbpalette betrachtet werden. Eine elegante Lösung ist die folgende selbst definierte Funktion:

```
color_glimpse <- function(colors_string){
```

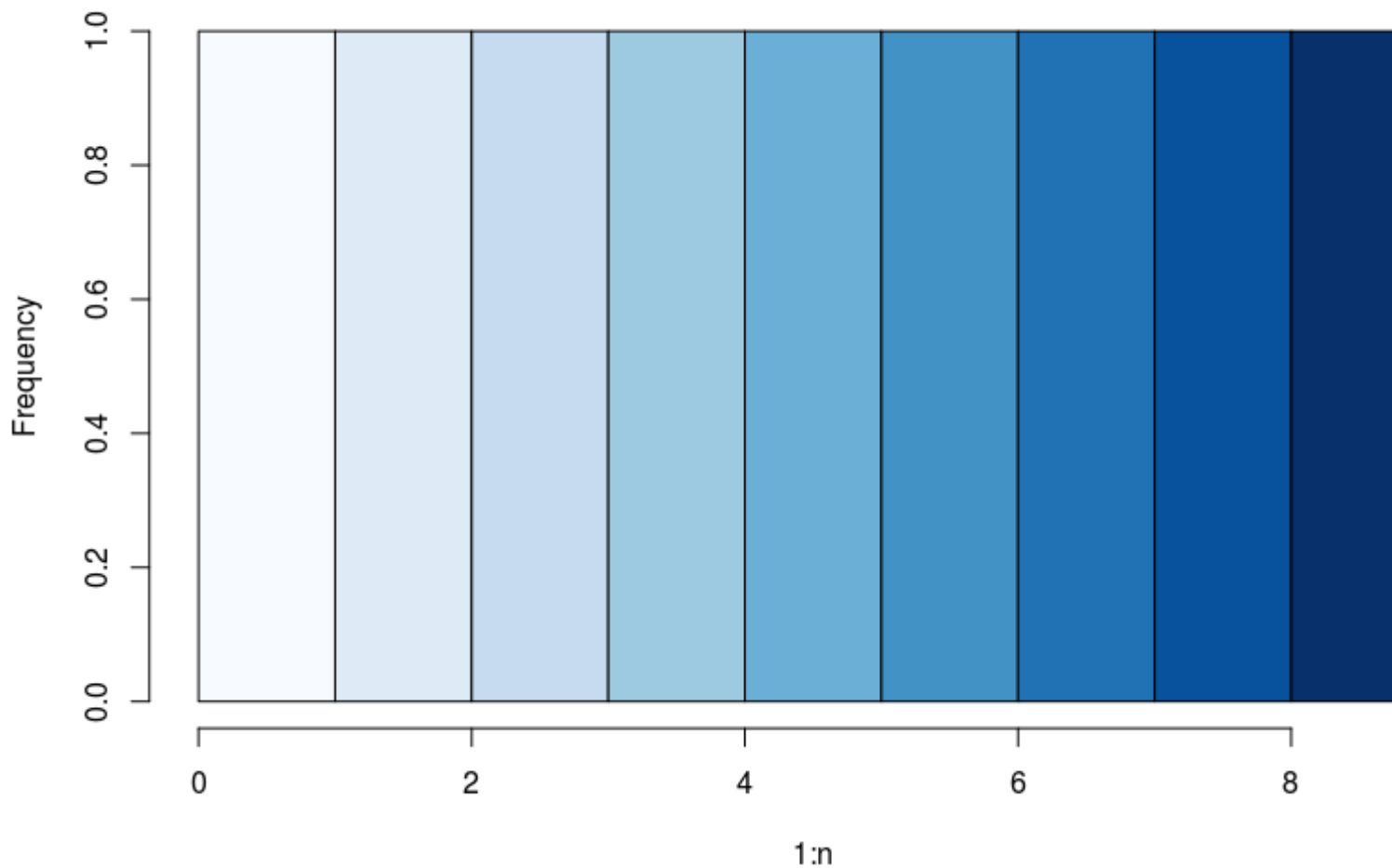


```
n <- length(colors_string)
hist(1:n,breaks=0:n,col=colors_string)
}
```

## Ein Anwendungsbeispiel

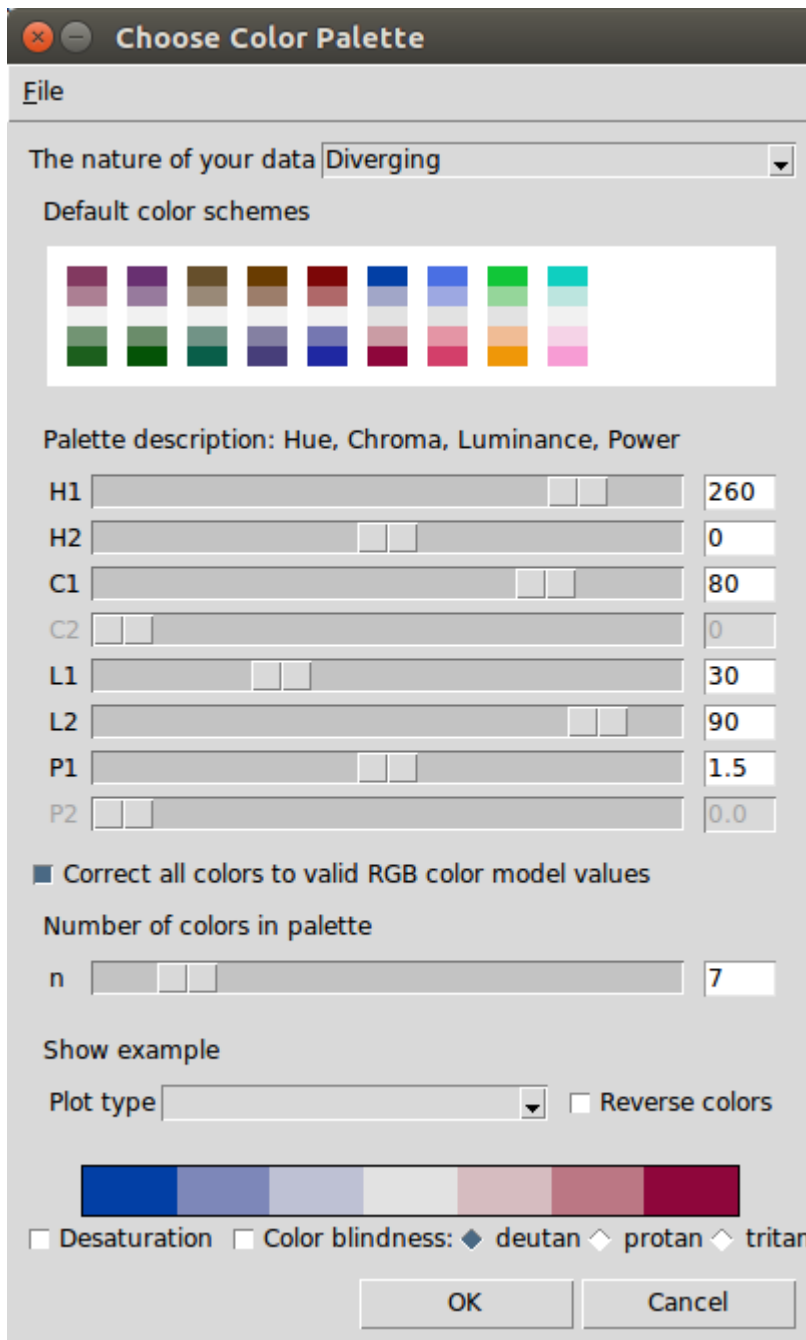
```
color_glimpse(blues9)
```

Histogram of 1:n



## Farbraum - Klicken und Ziehen der Benutzeroberfläche für Farben

Der `colorspace` bietet eine GUI zum Auswählen einer Palette. Beim Aufruf der Funktion `choose_palette()` das folgende Fenster:



Wenn die Palette ausgewählt ist, drücken Sie einfach **OK** und vergessen Sie nicht, die Ausgabe in einer Variablen zu speichern, z. B. `pal` .

```
pal <- choose_palette()
```

Die Ausgabe ist eine Funktion, die `n` (number) als Eingabe verwendet und einen Farbvektor der Länge `n` entsprechend der ausgewählten Palette erzeugt.

```
pal(10)
[1] "#023FA5" "#6371AF" "#959CC3" "#BEC1D4" "#DBDCE0" "#E0DBDC" "#D6BCC0" "#C6909A" "#AE5A6D"
"#8E063B"
```

## grundlegende R-Farbfunktionen

Funktionsfarben `colors()` listet alle Farbnamen auf, die von R erkannt werden. Es gibt [ein schönes PDF](#), in dem man diese Farben tatsächlich sehen kann.

---

`colorRampPalette` erstellt eine Funktion, die einen Satz vorgegebener Farben interpoliert, um neue Farbpaletten zu erstellen. Diese Ausgabefunktion nimmt `n` (number) als Eingabe und erzeugt einen Farbvektor der Länge `n` die Ausgangsfarben interpoliert.

```
pal <- colorRampPalette(c('white','red'))
pal(5)
[1] "#FFFFFF" "#FFBFBF" "#FF7F7F" "#FF3F3F" "#FF0000"
```

---

Jede bestimmte Farbe kann mit einer `rgb()` Funktion erzeugt werden:

```
rgb(0,1,0)
```

produziert `green` Farbe.

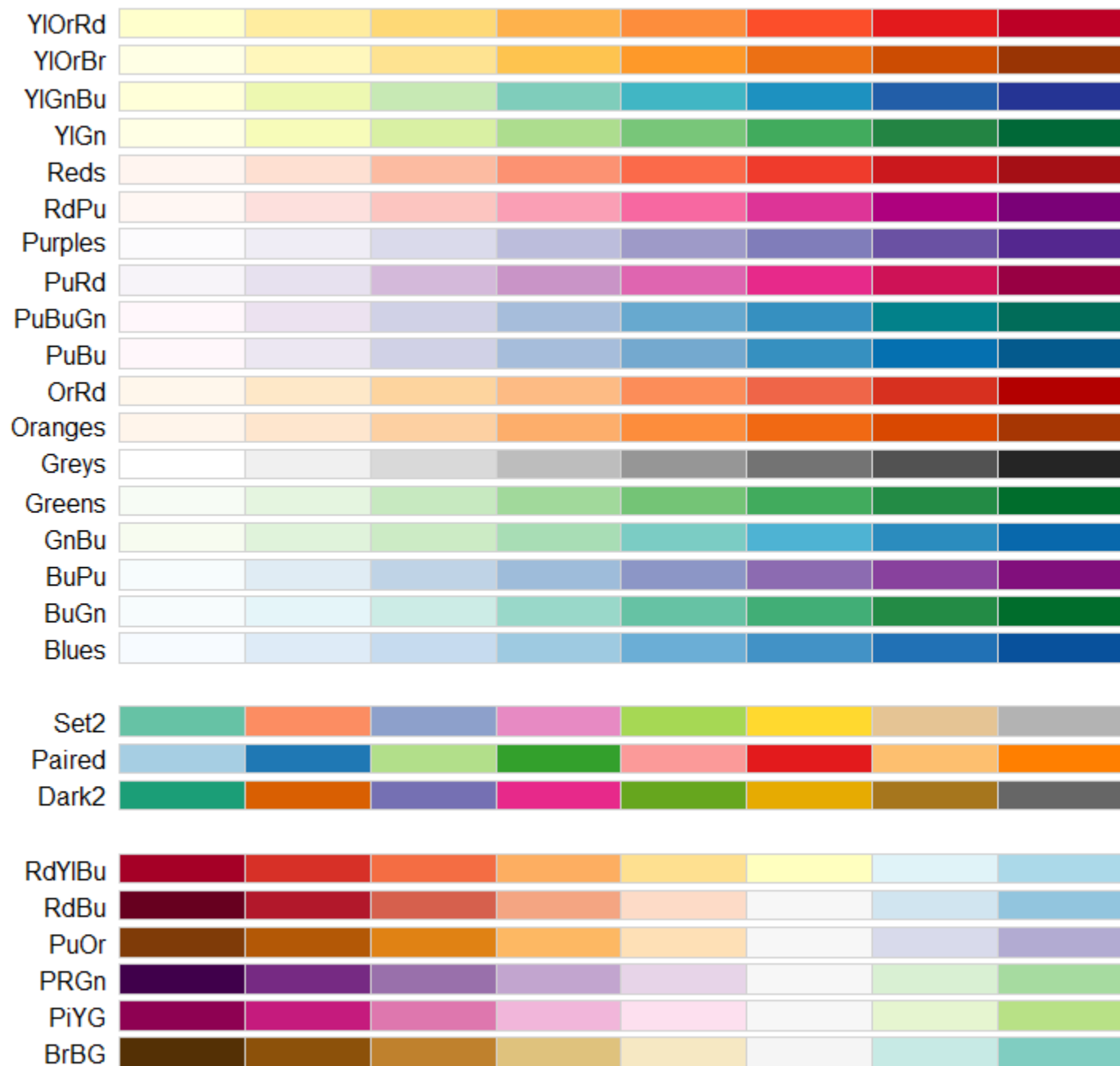
## Farbblindfreundliche Paletten

Obwohl farbblinde Menschen eine breite Palette von Farben erkennen können, kann es schwierig sein, bestimmte Farben zu unterscheiden.

---

`RColorBrewer` bietet `RColorBrewer` Paletten:

```
library(RColorBrewer)
display.brewer.all(colorblindFriendly = T)
```



Das [Color Universal Design](https://coloruniversaldesign.org/) der Universität Tokio bietet folgende Paletten an:

```
#palette using grey
```

```
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
"#CC79A7")

#palette using black
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
"#CC79A7")
```

Farbschemata für Grafiken online lesen: <https://riptutorial.com/de/r/topic/8005/farbschemata-fur-grafiken>

# Kapitel 45: Fehlende Werte

## Einführung

Wenn wir nicht wissen, welchen Wert eine Variable annimmt, sagen wir, dass ihr Wert fehlt, angegeben durch `NA`.

## Bemerkungen

Fehlende Werte werden durch das Symbol `NA` (nicht verfügbar) dargestellt. Unmögliche Werte (z. B. durch `sqrt(-1)`) werden durch das Symbol `NaN` (keine Zahl) dargestellt.

## Examples

### Untersuchen der fehlenden Daten

`anyNA` meldet, ob fehlende Werte vorhanden sind. `is.na` meldet fehlende Werte elementweise:

```
vec <- c(1, 2, 3, NA, 5)

anyNA(vec)
[1] TRUE
is.na(vec)
[1] FALSE FALSE FALSE TRUE FALSE
```

`is.na` gibt einen logischen Vektor zurück, der bei arithmetischen Operationen zu ganzzahligen Werten gezwungen wird (mit `FALSE = 0`, `TRUE = 1`). Wir können dies verwenden, um herauszufinden, wie viele Werte fehlen:

```
sum(is.na(vec))
[1] 1
```

In Erweiterung dieses Ansatzes können wir `colSums` und `is.na` in einem `is.na` verwenden, um NAs pro Spalte zu zählen:

```
colSums(is.na(airquality))
Ozone Solar.R Wind Temp Month Day
37 7 0 0 0 0
```

Das [Naniar-Paket](#) (derzeit auf Github, aber nicht auf CRAN) bietet weitere Tools zum Ermitteln fehlender Werte.

### Lesen und Schreiben von Daten mit NA-Werten

Beim Lesen von Tabellendatensätzen mit der Funktion `read.*` Sucht R automatisch nach fehlenden Werten, die wie `"NA"` aussehen. Fehlende Werte werden jedoch nicht immer von `NA`.

Manchmal gibt ein Punkt ( . ), Ein Bindestrich ( - ) oder ein Zeichenwert (zB: empty ) an, dass ein Wert NA . Mit na.strings Parameter na.strings der Funktion read.\* Kann R na.strings werden, welche Symbole / Zeichen als NA Werte behandelt werden müssen:

```
read.csv("name_of_csv_file.csv", na.strings = "-")
```

Es kann auch angegeben werden, dass mehr als ein Symbol als NA gelesen werden muss:

```
read.csv('missing.csv', na.strings = c('.', '-'))
```

In ähnlicher Weise NA kann s mit maßgeschneiderten Strings geschrieben werden , um die Verwendung von na Argument write.csv . [Andere Tools zum Lesen und Schreiben von Tabellen](#) haben ähnliche Optionen.

## Verwendung von NAs verschiedener Klassen

Das Symbol NA steht für einen logical fehlenden Wert:

```
class(NA)
#[1] "logical"
```

Dies ist praktisch, da es leicht zu anderen Atomvektortypen gezwungen werden kann und daher normalerweise die einzige NA Sie benötigen:

```
x <- c(1, NA, 1)
class(x[2])
#[1] "numeric"
```

Wenn Sie einen einzelnen NA Wert eines anderen Typs benötigen, verwenden Sie NA\_character\_ , NA\_integer\_ , NA\_real\_ oder NA\_complex\_ . Bei fehlenden Werten von ausgefallenen Klassen funktioniert das Subsetting mit NA\_integer\_ normalerweise. Zum Beispiel, um ein fehlendes Datum zu erhalten:

```
class(Sys.Date()[NA_integer_])
[1] "Date"
```

## TRUE / FALSE und / oder NA

NA ist ein logischer Typ und ein logischer Operator mit einer NA gibt NA wenn das Ergebnis mehrdeutig ist. Unten NA OR TRUE ausgewertet TRUE , da mindestens eine Seite wertet TRUE jedoch NA OR FALSE kehrt NA , weil wir nicht wissen , ob NA gewesen wäre TRUE oder FALSE

```
NA | TRUE
[1] TRUE
TRUE | TRUE is TRUE and FALSE | TRUE is also TRUE.

NA | FALSE
[1] NA
TRUE | FALSE is TRUE but FALSE | FALSE is FALSE.
```

```

NA & TRUE
[1] NA
TRUE & TRUE is TRUE but FALSE & TRUE is FALSE.

NA & FALSE
[1] FALSE
TRUE & FALSE is FALSE and FALSE & FALSE is also FALSE.

```

Diese Eigenschaften sind hilfreich, wenn Sie einen Datensatz basierend auf einigen Spalten, die `NA` enthalten, als Teilmenge festlegen möchten.

```

df <- data.frame(v1=0:9,
 v2=c(rep(1:2, each=4), NA, NA),
 v3=c(NA, letters[2:10]))

df[df$v2 == 1 & !is.na(df$v2),]
v1 v2 v3
#1 0 1 <NA>
#2 1 1 b
#3 2 1 c
#4 3 1 d

df[df$v2 == 1,]
v1 v2 v3
#1 0 1 <NA>
#2 1 1 b
#3 2 1 c
#4 3 1 d
#NA NA NA <NA>
#NA.1 NA NA <NA>

```

## Fehlende Werte auslassen oder ersetzen

# Fehlende Werte neu codieren

Fehlende Daten werden in Datensätzen regelmäßig nicht als `NA` codiert. In SPSS zum Beispiel werden fehlende Werte häufig durch den Wert `99` .

```

num.vec <- c(1, 2, 3, 99, 5)
num.vec
[1] 1 2 3 99 5

```

Es ist möglich, die `NA` direkt über das Subset zuzuweisen

```

num.vec[num.vec == 99] <- NA

```

Die bevorzugte Methode ist jedoch die Verwendung von `is.na<-` wie unten beschrieben. Die Hilfedatei (`?is.na`) besagt:

`is.na<-` kann einen sichereren Weg bieten, um den Mangel zu beheben. Es verhält sich beispielsweise für Faktoren anders.



```
is.na(num.vec) <- num.vec == 99
```

Beide Methoden kehren zurück

```
num.vec
[1] 1 2 3 NA 5
```

---

## Fehlende Werte entfernen

Fehlende Werte können auf verschiedene Weise aus einem Vektor entfernt werden:

```
num.vec[!is.na(num.vec)]
num.vec[complete.cases(num.vec)]
na.omit(num.vec)
[1] 1 2 3 5
```

---

## Fehlende Werte aus den Berechnungen ausschließen

Wenn arithmetische Funktionen für Vektoren mit fehlenden Werten verwendet werden, wird ein fehlender Wert zurückgegeben:

```
mean(num.vec) # returns: [1] NA
```

Der Parameter `na.rm` teilt der Funktion mit, die `NA` Werte von der Berechnung auszuschließen:

```
mean(num.vec, na.rm = TRUE) # returns: [1] 2.75

an alternative to using 'na.rm = TRUE':
mean(num.vec[!is.na(num.vec)]) # returns: [1] 2.75
```

Einige R-Funktionen, wie `na.action` . B. `lm` , haben einen `na.action` Parameter. Der Standardwert hierfür ist `na.omit` , aber mit `options(na.action = 'na.exclude')` das Standardverhalten von R geändert werden.

Wenn das Standardverhalten nicht geändert werden muss, aber für eine bestimmte Situation eine andere `na.action` erforderlich ist, muss der `na.action` Parameter in den Funktionsaufruf aufgenommen werden, z.

```
lm(y2 ~ y1, data = anscombe, na.action = 'na.exclude')
```

Fehlende Werte online lesen: <https://riptutorial.com/de/r/topic/3388/fehlende-werte>

# Kapitel 46: Fehlertoleranter / belastbarer Code

## Parameter

| Parameter              | Einzelheiten                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ausdruck               | Falls der "try-Teil" erfolgreich abgeschlossen wurde, gibt <code>tryCatch</code> den <b>zuletzt ausgewerteten Ausdruck zurück</b> . Der tatsächliche Wert, der zurückgegeben wird, falls alles gut <code>readLines</code> ist und keine Bedingung (dh eine <i>Warnung</i> oder ein <i>Fehler</i> ) <i>vorliegt</i> , ist der Rückgabewert von <code>readLines</code> . Beachten Sie, dass Sie nicht explizit angeben müssen, dass der Rückgabewert per <code>return</code> da der Code im "try-Teil" nicht innerhalb einer Funktionsumgebung umschlossen ist (im Gegensatz zu den Bedingungen für die Handler für Warnungen und Fehler). |
| Warnung / Fehler / etc | Stellen Sie eine Handler-Funktion für alle Bedingungen bereit, die Sie explizit behandeln möchten. AFAIU können Sie Handler für <i>jede</i> Art von Bedingung bereitstellen (nicht nur <i>Warnungen</i> und <i>Fehler</i> , sondern auch <i>benutzerdefinierte</i> Bedingungen; siehe dazu <code>simpleCondition</code> und Freunde), sofern der <b>Name der jeweiligen Handlerfunktion der Klasse der jeweiligen Bedingung entspricht</b> (siehe <i>Details</i> Teil des doc für <code>tryCatch</code> ).                                                                                                                               |
| endlich                | Hier wird alles ausgeführt, was am Ende ausgeführt werden soll, <b>unabhängig davon</b> , ob der Ausdruck im "try-Teil" erfolgreich war oder ob eine Bedingung bestand. Wenn Sie möchten, dass mehrere Ausdrücke ausgeführt werden, müssen Sie sie in geschweiften Klammern einwickeln. Andernfalls könnten Sie <code>finally = &lt;expression&gt;</code> (dh die gleiche Logik wie für "try part").                                                                                                                                                                                                                                     |

## Bemerkungen

### `tryCatch`

`tryCatch` gibt den Wert zurück, der der Ausführung von `expr` sei denn, es gibt eine Bedingung: eine Warnung oder einen Fehler. Wenn das der Fall ist , bestimmte Rückgabewerte (zB `return(NA)` oben) kann durch Zuführen einer Behandlungsfunktion für die jeweiligen Bedingungen angegeben werden (siehe Argumente `warning` und `error` in `?tryCatch` ). Dies können Funktionen sein, die bereits vorhanden sind, aber Sie können sie auch in `tryCatch` definieren (wie oben beschrieben).

### Auswirkungen der Auswahl bestimmter Rückgabewerte der Handlerfunktionen

Da wir angegeben haben, dass `NA` im Falle eines Fehlers im "try-Teil" zurückgegeben werden soll,

ist das dritte Element in `y NA`. Wenn wir `NULL` als Rückgabewert gewählt hätten, wäre die Länge von `y` nur 2 statt 3 da `lapply` die `NULL` Rückgabewerte einfach "ignorieren" / `lapply` wird. Beachten Sie auch, dass, wenn Sie keinen **expliziten** Rückgabewert über `return` angeben, die Handlerfunktionen `NULL` (dh im Falle eines *Fehlers* oder einer *Warnbedingung*).

## Warnmeldung "Unerwünscht"

Wenn das dritte Element unserer `urls` Vektor unsere Funktion trifft, erhalten wir die folgende Warnung **zusätzlich** zu der Tatsache, dass ein Fehler (tritt `readLines` klagt zuerst, dass sie die Verbindung über eine *Warnung*, bevor sie tatsächlich Fehler mit einem *Fehler* öffnen können):

```
Warning message:
 In file(con, "r") : cannot open file 'I'm no URL': No such file or directory
```

Ein *Fehler* "gewinnt" über eine *Warnung*, daher sind wir in diesem Fall nicht wirklich an der Warnung interessiert. Wir haben also in `readLines warn = FALSE readLines`, aber das scheint keine Wirkung zu haben. Eine alternative Möglichkeit, die Warnung zu unterdrücken, ist die Verwendung

```
suppressWarnings(readLines(con = url))
```

anstatt

```
readLines(con = url, warn = FALSE)
```

## Examples

### TryCatch () verwenden

Wir definieren eine robuste Version einer Funktion, die den HTML-Code von einer bestimmten URL liest. *Robust* in dem Sinne, dass wir mit Situationen umgehen wollen, in denen etwas schief geht (Fehler) oder nicht ganz so wie wir es geplant haben (Warnung). Der Oberbegriff für Fehler und Warnungen ist *Bedingung*

## Funktionsdefinition mit `tryCatch`

```
readUrl <- function(url) {
 out <- tryCatch(

#####
Try part: define the expression(s) you want to "try"
#####

{
 # Just to highlight:
 # If you want to use more than one R expression in the "try part"
 # then you'll have to use curly brackets.
 # Otherwise, just write the single expression you want to try and
```

```

 message("This is the 'try' part")
 readLines(con = url, warn = FALSE)
 },

 #####
 # Condition handler part: define how you want conditions to be handled #
 #####

 # Handler when a warning occurs:
 warning = function(cond) {
 message(paste("Reading the URL caused a warning:", url))
 message("Here's the original warning message:")
 message(cond)

 # Choose a return value when such a type of condition occurs
 return(NULL)
 },

 # Handler when an error occurs:
 error = function(cond) {
 message(paste("This seems to be an invalid URL:", url))
 message("Here's the original error message:")
 message(cond)

 # Choose a return value when such a type of condition occurs
 return(NA)
 },

 #####
 # Final part: define what should happen AFTER #
 # everything has been tried and/or handled #
 #####

 finally = {
 message(paste("Processed URL:", url))
 message("Some message at the end\n")
 }
)
return(out)
}

```

## Dinge testen

Definieren wir einen Vektor von URLs, bei dem ein Element keine gültige URL ist

```

urls <- c(
 "http://stat.ethz.ch/R-manual/R-devel/library/base/html/connections.html",
 "http://en.wikipedia.org/wiki/Xz",
 "I'm no URL"
)

```

Und übergeben Sie dies als Eingabe an die oben definierte Funktion

```

y <- lapply(urls, readUrl)
Processed URL: http://stat.ethz.ch/R-manual/R-devel/library/base/html/connections.html
Some message at the end
#

```

```
Processed URL: http://en.wikipedia.org/wiki/Xz
Some message at the end
#
URL does not seem to exist: I'm no URL
Here's the original error message:
cannot open the connection
Processed URL: I'm no URL
Some message at the end
#
Warning message:
In file(con, "r") : cannot open file 'I'm no URL': No such file or directory
```

## Untersuchung der Ausgabe

```
length(y)
[1] 3

head(y[[1]])
[1] "<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">"
[2] "<html><head><title>R: Functions to Manipulate Connections</title>"
[3] "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">"
[4] "<link rel=\"stylesheet\" type=\"text/css\" href=\"R.css\">"
[5] "</head><body>"
[6] ""

y[[3]]
[1] NA
```

Fehlertoleranter / belastbarer Code online lesen:

<https://riptutorial.com/de/r/topic/4060/fehlertoleranter---belastbarer-code>

# Kapitel 47: Formel

## Examples

### Die Grundlagen der Formel

Statistische Funktionen in R machen Gebrauch von der sogenannten Wilkinson-Rogers-Formelnotation <sup>1</sup>.

Wenn Sie Modellfunktionen wie `lm` für **lineare Regressionen ausführen**, benötigen sie eine `formula`. Diese `formula` gibt an, welche Regressionskoeffizienten geschätzt werden sollen.

```
my_formula1 <- formula(mpg ~ wt)
class(my_formula1)
gives "formula"

mod1 <- lm(my_formula1, data = mtcars)
coef(mod1)
gives (Intercept) wt
37.285126 -5.344472
```

Auf der linken Seite der `~` (LHS) wird die abhängige Variable angegeben, während die rechte Seite (RHS) die unabhängigen Variablen enthält. Technisch ist der obige `formula` überflüssig, da der Tilde-Operator eine Infix-Funktion ist, die ein Objekt mit der Formelklasse zurückgibt:

```
form <- mpg ~ wt
class(form)
#[1] "formula"
```

Der Vorteil der `formula` gegenüber `~` besteht darin, dass auch eine Umgebung zur Auswertung angegeben werden kann:

```
form_mt <- formula(mpg ~ wt, env = mtcars)
```

In diesem Fall zeigt die Ausgabe, dass ein Regressionskoeffizient für `wt` geschätzt wird, sowie (standardmäßig) ein Abschnittparameter. Der Achsenabschnitt kann durch Angabe von `0` oder `-1` in die `formula` ausgeschlossen / erzwungen werden:

```
coef(lm(mpg ~ 0 + wt, data = mtcars))
coef(lm(mpg ~ wt -1, data = mtcars))
```

Wechselwirkungen zwischen den Variablen `a` und `b` können durch Hinzufügen von `a:b` zur `formula` hinzugefügt werden:

```
coef(lm(mpg ~ wt:vs, data = mtcars))
```

Da es (aus statistischer Sicht) generell ratsam ist, keine Wechselwirkungen im Modell ohne die

Haupteffekte zu haben, würde der naive Ansatz darin bestehen, die formula auf  $a + b + a:b$ . Dies funktioniert, kann jedoch vereinfacht werden, indem  $a*b$ , wobei der Operator  $*$  eine Faktorüberkreuzung (wenn zwischen zwei Faktorspalten) oder eine Multiplikation angibt, wenn eine oder beide Spalten 'numerisch' sind:

```
coef(lm(mpg ~ wt*vs, data = mtcars))
```

Mit der  $*$ -Notation wird ein Begriff erweitert, um alle Effekte niedrigerer Ordnung einzubeziehen, z.

```
coef(lm(mpg ~ wt*vs*hp, data = mtcars))
```

gibt zusätzlich zum Intercept 7 Regressionskoeffizienten an. Eine für die Dreiwege-Interaktion, drei für die Zweiwege-Interaktion und drei für die Haupteffekte.

Wenn man zum Beispiel die Dreiwege-Interaktion ausschließen möchte, aber alle Zweiwege-Interaktionen beibehalten möchte, gibt es zwei Abkürzungen. Erstens, mit  $-$  wir können jeden bestimmten Begriff abziehen:

```
coef(lm(mpg ~ wt*vs*hp - wt:vs:hp, data = mtcars))
```

Oder wir können die  $^$ -Notation verwenden, um anzugeben, welche Interaktionsebene wir benötigen:

```
coef(lm(mpg ~ (wt + vs + hp) ^ 2, data = mtcars))
```

Diese beiden Formelspezifikationen sollten dieselbe Modellmatrix erstellen.

Zum Schluss  $.$  ist eine Abkürzung, um alle verfügbaren Variablen als Haupteffekte zu verwenden. In diesem Fall wird die `data` Argument verwendet, um die zur Verfügung stehenden Variablen zu erhalten (die nicht auf der LHS ist). Deshalb:

```
coef(lm(mpg ~ ., data = mtcars))
```

gibt Koeffizienten für den Achsenabschnitt und 10 unabhängige Variablen an. Diese Notation wird häufig in maschinellen Lernpaketen verwendet, bei denen alle Variablen zur Vorhersage oder Klassifizierung verwendet werden sollen. Beachten Sie, dass die Bedeutung von  $.$  abhängig vom Kontext (siehe zB `?update.formula` für eine andere Bedeutung).

1. GN Wilkinson und CE Rogers. *Zeitschrift der Royal Statistical Society. Serie C (Angewandte Statistik)* Vol. 3, (1973), S. 392-399

## Erstellen Sie lineare, quadratische und zweite Interaktionsbedingungen

$y \sim .$  Hier:  $.$  wird als alle Variablen mit Ausnahme von  $y$  in dem Datenrahmen interpretiert, der zur Anpassung des Modells verwendet wird. Es entspricht den linearen Kombinationen von Prädiktorvariablen. Zum Beispiel  $y \sim \text{var1} + \text{var2} + \text{var3} + \dots + \text{var15}$

`y ~ . ^ 2` gibt alle linearen (Haupteffekte) und Interaktionsterme zweiter Ordnung der Variablen im Datenrahmen aus. Es ist gleichbedeutend mit `y ~ var1 + var2 + ...+var15 + var1:var2 + var1:var3 + var1:var4...and so on`

`y ~ var1 + var2 + ...+var15 + I(var1^2) + I(var2^2) + I(var3^2)...+I(var15^2)` : Hier `I(var^2)` bezeichnet ein quadratisches Polynom einer Variablen im Datenrahmen.

`y ~ poly(var1, degree = 2) + poly(var2, degree = 2)+...poly(var15, degree = 2)`

oder

`y ~ poly(var1, var2, var3, ....var15, degree = 2)` entspricht dem obigen Ausdruck.

`poly(var1, degree = 2)` ist äquivalent zu `var1 + I(var1^2)` .

Um kubische Polynome zu erhalten, verwenden Sie in `poly()` `degree = 3` .

Bei der Verwendung von `poly` gegenüber `I(var, 2)` gibt es eine Einschränkung, die nach dem Anpassen des Modells unterschiedliche Koeffizienten erzeugt. Die angepassten Werte sind jedoch gleichwertig, da sie unterschiedliche Parametrisierungen des gleichen Modells darstellen. Es wird empfohlen, `I(var, 2)` über `poly()` zu verwenden, um den in `poly()` Zusammenfassungseffekt zu vermeiden.

Zusammenfassend erhalten Sie einen Ausdruck wie, um lineare, quadratische und zweite Interaktionsbedingungen zu erhalten

`y ~ .^2 + I(var1^2) + I(var2^2)+...I(var15^2)`

## Demo für vier Variablen:

```
old <- reformulate('y ~ x1+x2+x3+x4')
new <- reformulate(" y ~ .^2 + I(x1^2) + I(x2^2) + I(x3^2) + I(x4^2) ")
tmp <- .Call(stats::C_updateform, old, new)
terms.formula(tmp, simplify = TRUE)

~y ~ x1 + x2 + x3 + x4 + I(x1^2) + I(x2^2) + I(x3^2) + I(x4^2) +
x1:x2 + x1:x3 + x1:x4 + x2:x3 + x2:x4 + x3:x4
attr(,"variables")
list(~y, x1, x2, x3, x4, I(x1^2), I(x2^2), I(x3^2), I(x4^2))
attr(,"factors")
x1 x2 x3 x4 I(x1^2) I(x2^2) I(x3^2) I(x4^2) x1:x2 x1:x3 x1:x4 x2:x3 x2:x4 x3:x4
~y 0 0 0 0 0 0 0 0 0 0 0 0 0 0
x1 1 0 0 0 0 0 0 0 1 1 1 0 0 0
x2 0 1 0 0 0 0 0 0 1 0 0 1 1 0
x3 0 0 1 0 0 0 0 0 0 1 0 1 0 1
x4 0 0 0 1 0 0 0 0 0 0 1 0 1 1
I(x1^2) 0 0 0 0 1 0 0 0 0 0 0 0 0 0
I(x2^2) 0 0 0 0 0 1 0 0 0 0 0 0 0 0
I(x3^2) 0 0 0 0 0 0 1 0 0 0 0 0 0 0
I(x4^2) 0 0 0 0 0 0 0 1 0 0 0 0 0 0
attr(,"term.labels")
[1] "x1" "x2" "x3" "x4" "I(x1^2)" "I(x2^2)" "I(x3^2)" "I(x4^2)"
[9] "x1:x2" "x1:x3" "x1:x4" "x2:x3" "x2:x4" "x3:x4"
attr(,"order")
[1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2
attr(,"intercept")
```



```
[1] 1
attr(,"response")
[1] 1
attr(,".Environment")
<environment: R_GlobalEnv>
```

Formel online lesen: <https://riptutorial.com/de/r/topic/1061/formel>

---

# Kapitel 48: Fourier-Serie und Transformationen

## Bemerkungen

Die Fourier-Transformation zerlegt eine Funktion der Zeit (ein Signal) in die Frequenzen, aus denen sie besteht, ähnlich wie ein Musikakkord als Amplitude (oder Lautstärke) seiner konstituierenden Noten ausgedrückt werden kann. Die Fourier-Transformation einer Funktion der Zeit selbst ist eine komplexwertige Funktion der Frequenz, deren absoluter Wert den Betrag dieser in der ursprünglichen Funktion vorhandenen Frequenz darstellt und deren komplexes Argument der Phasenversatz des Basissinus in dieser Frequenz ist.

Die Fourier-Transformation wird als Frequenzdomänenendarstellung des Originalsignals bezeichnet. Der Begriff Fourier-Transformation bezieht sich sowohl auf die Frequenzbereichsdarstellung als auch auf die mathematische Operation, die die Frequenzbereichsdarstellung einer Funktion der Zeit zuordnet. Die Fourier-Transformation ist nicht auf Zeitfunktionen beschränkt, sondern um eine einheitliche Sprache zu haben, wird der Bereich der ursprünglichen Funktion allgemein als Zeitbereich bezeichnet. Für viele Funktionen von praktischem Interesse kann eine Operation definiert werden, die dies umkehrt: Die inverse Fourier-Transformation einer Frequenzbereichsrepräsentation, auch Fourier-Synthese genannt, kombiniert die Beiträge aller verschiedenen Frequenzen, um die ursprüngliche Funktion der Zeit wiederherzustellen.

Lineare Operationen, die in einer Domäne (Zeit oder Häufigkeit) ausgeführt werden, haben entsprechende Operationen in der anderen Domäne, die manchmal einfacher auszuführen sind. Der Vorgang der Differenzierung im Zeitbereich entspricht der Multiplikation mit der Frequenz, so dass einige Differenzialgleichungen im Frequenzbereich einfacher zu analysieren sind. Die Faltung im Zeitbereich entspricht auch einer gewöhnlichen Multiplikation im Frequenzbereich. Konkret bedeutet dies, dass jedes lineare zeitinvariante System, beispielsweise ein auf ein Signal angewendeter elektronischer Filter, relativ einfach als eine Operation auf Frequenzen ausgedrückt werden kann. So wird oft eine erhebliche Vereinfachung erreicht, indem Zeitfunktionen in den Frequenzbereich umgewandelt werden, die gewünschten Operationen ausgeführt werden und das Ergebnis in die Zeit zurückversetzt wird.

Die harmonische Analyse ist die systematische Untersuchung der Beziehung zwischen Frequenz- und Zeitbereichen, einschließlich der Arten von Funktionen oder Operationen, die in dem einen oder anderen "einfacher" sind, und hat tiefe Verbindungen zu fast allen Bereichen der modernen Mathematik.

Funktionen, die im Zeitbereich lokalisiert sind, weisen Fourier-Transformationen auf, die über den Frequenzbereich verteilt sind und umgekehrt. Der kritische Fall ist die Gauß-Funktion, die in der Wahrscheinlichkeitstheorie und -statistik sowie in der Untersuchung von Normalverteilungen (z. B. Diffusion), die mit geeigneten Normalisierungen unter der Fouriertransformation auftreten, von erheblicher Bedeutung ist. Joseph Fourier führte die Transformation in seine Studie der Wärmeübertragung ein, bei der Gaußsche Funktionen als Lösungen der Wärmegleichung

erscheinen.

Die Fourier-Transformation kann formal als ein unpassendes Riemann-Integral definiert werden, was sie zu einer Integraltransformation macht, obwohl diese Definition nicht für viele Anwendungen geeignet ist, die eine komplexere Integrationstheorie erfordern.

Beispielsweise verwenden viele relativ einfache Anwendungen die Dirac-Delta-Funktion, die formal wie eine Funktion behandelt werden kann, die Begründung erfordert jedoch einen mathematisch anspruchsvolleren Standpunkt. Die Fourier-Transformation kann auch auf Funktionen mehrerer Variablen im euklidischen Raum verallgemeinert werden, indem eine Funktion des dreidimensionalen Raums an eine Funktion des dreidimensionalen Impulses (oder eine Funktion des Raums und der Zeit an eine Funktion des 4-Impulses) gesendet wird.

Diese Idee macht die räumliche Fourier-Transformation in der Untersuchung von Wellen und in der Quantenmechanik sehr natürlich, wo es wichtig ist, Wellenlösungen entweder als Funktion des Raums oder des Impulses und manchmal als beides darstellen zu können. Im Allgemeinen sind Funktionen, auf die Fourier-Methoden anwendbar sind, komplexwertig und möglicherweise vektoriell. Eine weitere Verallgemeinerung ist auf Funktionen in Gruppen möglich, die neben der ursprünglichen Fourier-Transformation auf  $\mathbb{R}$  oder  $\mathbb{R}^n$  (als Gruppen unter Addition betrachtet) insbesondere die zeitdiskrete Fourier-Transformation (DTFT, group =  $\mathbb{Z}$ ), die diskrete Fourier-Transformation (DFT, Gruppe =  $\mathbb{Z} \bmod N$ ) und die Fourier-Reihe oder die zirkuläre Fourier-Transformation (Gruppe =  $S^1$ , der Einheitskreis  $\approx$  geschlossenes endliches Intervall mit identifizierten Endpunkten). Letzteres wird routinemäßig zur Abwicklung periodischer Funktionen eingesetzt. Die Fast Fourier Transformation (FFT) ist ein Algorithmus zur Berechnung der DFT.

## Examples

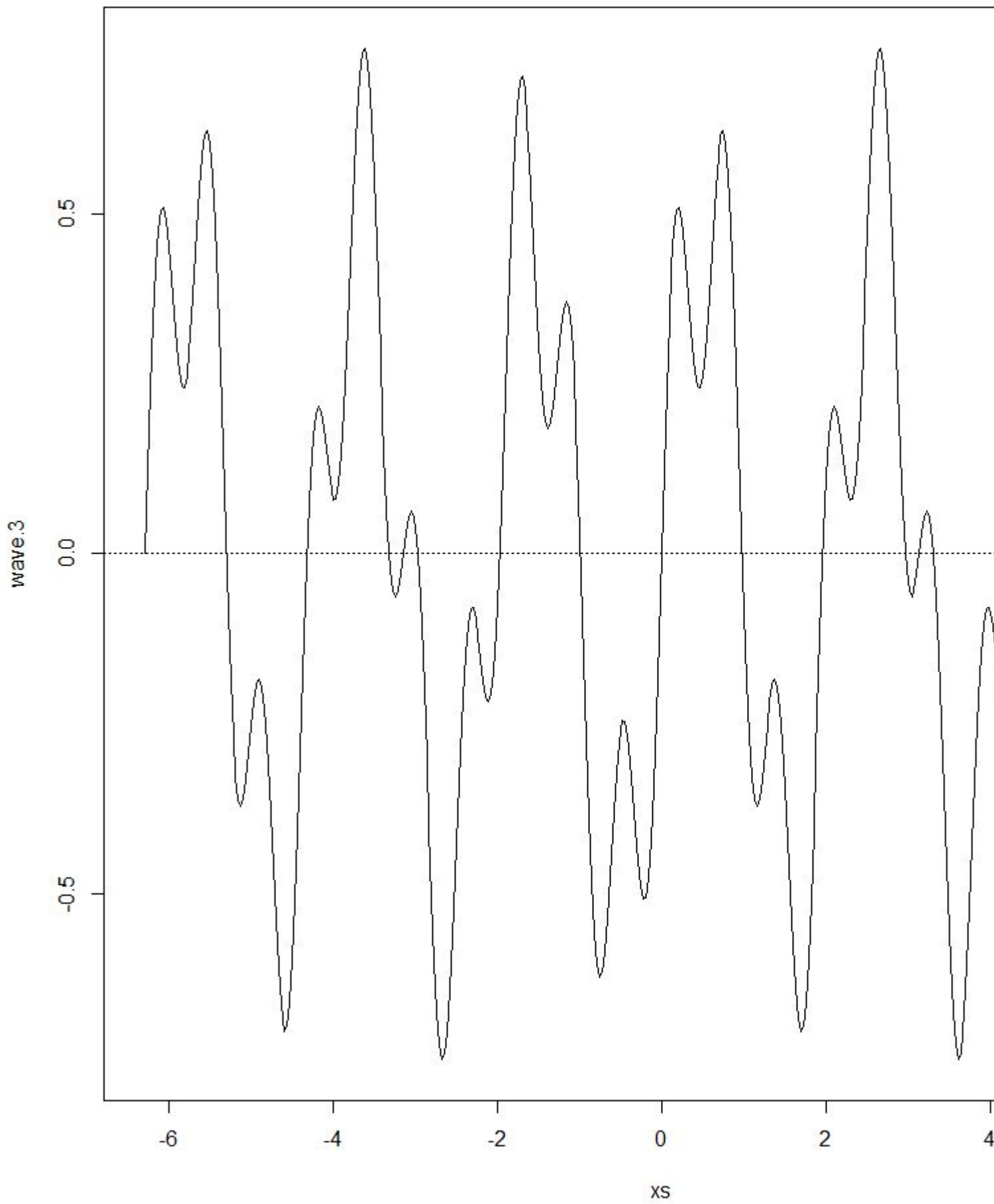
### Die Fourierreihe

Joseph Fourier zeigte, dass jede periodische Welle durch eine Summe einfacher Sinuswellen dargestellt werden kann. Diese Summe wird als Fourier-Serie bezeichnet. Die Fourier-Serie gilt nur, wenn das System linear ist. Wenn es beispielsweise einen Überlappeneffekt gibt (eine Schwelle, bei der die Ausgabe gleich bleibt, unabhängig davon, wie viel Eingabe gegeben wird), tritt ein nichtlinearer Effekt in das Bild ein, der die Sinuswelle und das Überlagerungsprinzip bricht.

```
Sine waves
xs <- seq(-2*pi,2*pi,pi/100)
wave.1 <- sin(3*xs)
wave.2 <- sin(10*xs)
par(mfrow = c(1, 2))
plot(xs, wave.1, type="l", ylim=c(-1,1)); abline(h=0, lty=3)
plot(xs, wave.2, type="l", ylim=c(-1,1)); abline(h=0, lty=3)

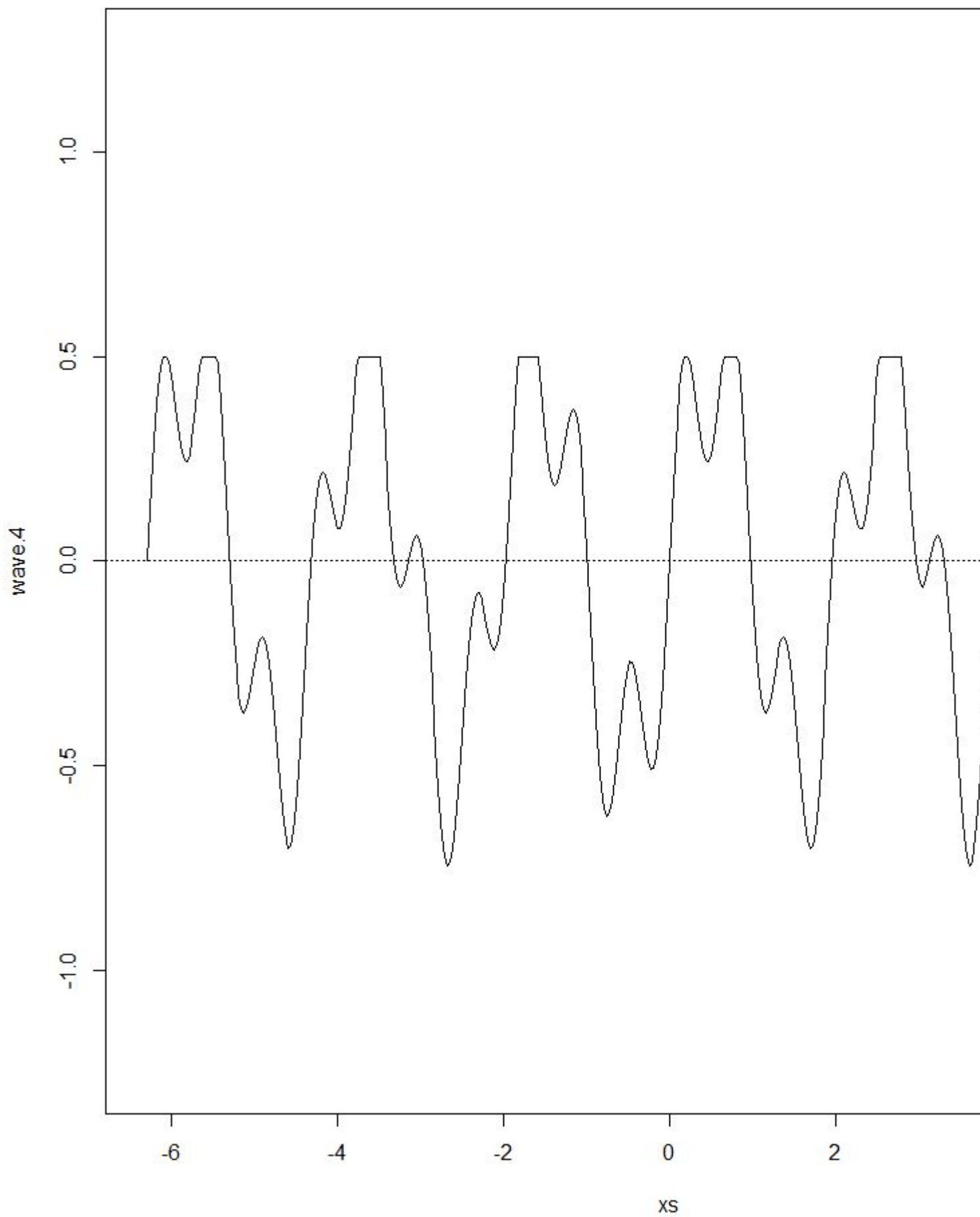
Complex Wave
wave.3 <- 0.5 * wave.1 + 0.25 * wave.2
plot(xs, wave.3, type="l"); title("Eg complex wave"); abline(h=0, lty=3)
```

### Eg complex wave



```
wave.4 <- wave.3
wave.4[wave.3>0.5] <- 0.5
plot(xs, wave.4, type="l", ylim=c(-1.25, 1.25))
title("overflowed, non-linear complex wave")
abline(h=0, lty=3)
```

### overflowed, non-linear complex wave



Die Fourier-Reihe gilt auch nur dann, wenn die Wellen periodisch sind, dh sie haben ein sich wiederholendes Muster (nicht periodische Wellen werden durch die Fourier-Transformation behandelt, siehe unten). Eine periodische Welle hat eine Frequenz  $f$  und eine Wellenlänge  $\lambda$  (eine Wellenlänge ist die Entfernung im Medium zwischen Beginn und Ende eines Zyklus,  $\lambda = v / f_0$ , wobei  $v$  die Wellengeschwindigkeit ist), die durch das sich wiederholende Muster definiert wird. Eine nichtperiodische Welle hat keine Frequenz oder Wellenlänge.

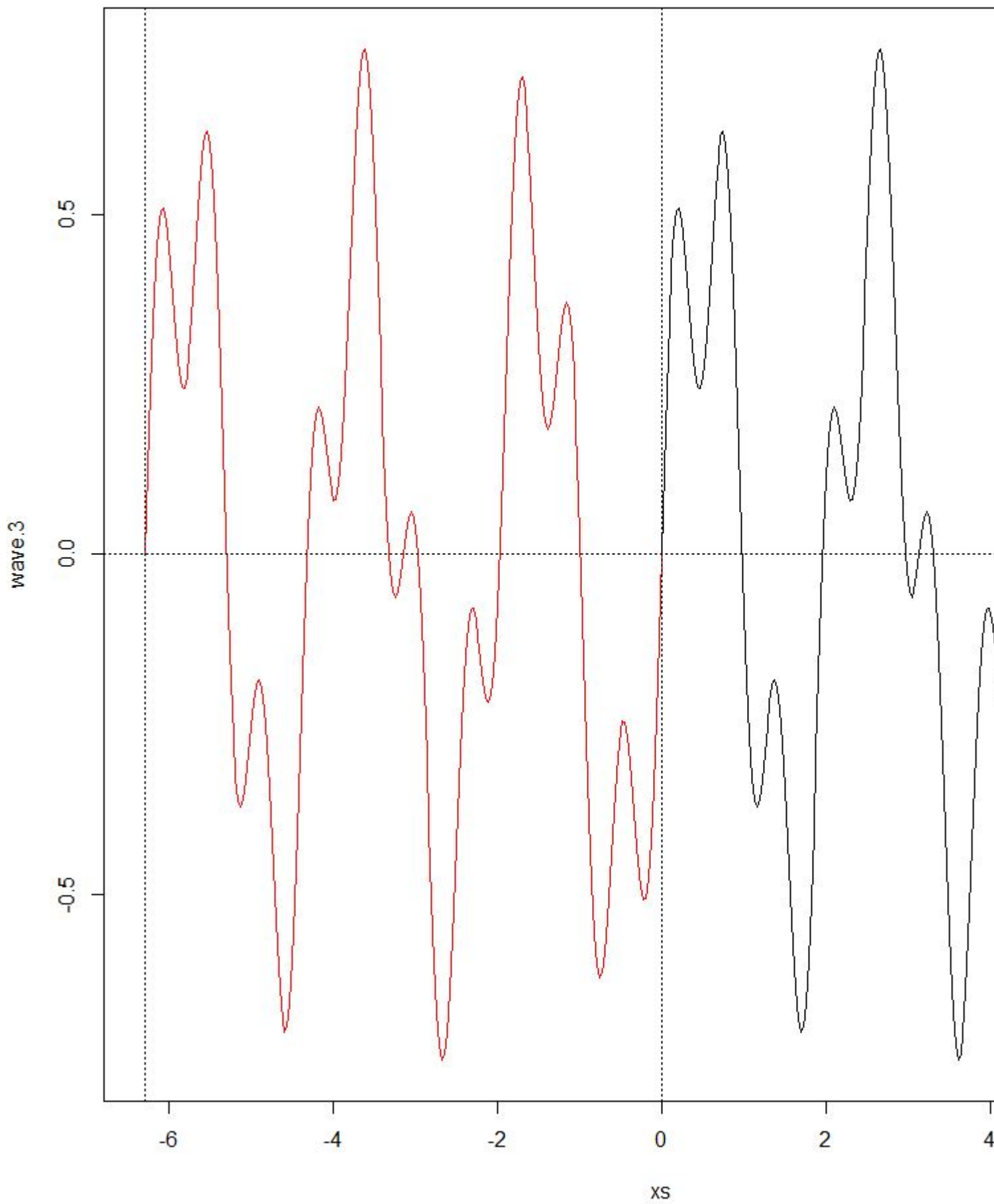
Einige Konzepte:

- Die Grundperiode  $T$  ist die Periode aller genommenen Proben, die Zeit zwischen der ersten und der letzten Probe
- Die Abtastrate,  $sr$ , ist die Anzahl von Abtastungen, die über einen Zeitraum hinweg (Akquisitionsfrequenz) genommen wurden. Zur Vereinfachung machen wir das Zeitintervall zwischen den Abtastwerten gleich. Dieses Zeitintervall wird als Abtastintervall  $si$  bezeichnet. Hierbei handelt es sich um die Grundperiodenzeit geteilt durch die Anzahl der Abtastwerte  $N$ . Also ist  $si = TN$
- Die Grundfrequenz  $f_0$  ist  $1/T$ . Die Grundfrequenz ist die Frequenz des sich wiederholenden Musters oder wie lang die Wellenlänge ist. In den vorherigen Wellen betrug die Grundfrequenz 12 & pgr ;. Die Frequenzen der Wellenkomponenten müssen ganzzahlige Vielfache der Grundfrequenz sein.  $f_0$  wird die erste Harmonische genannt, die zweite Harmonische ist  $2 * f_0$ , die dritte ist  $3 * f_0$  usw.

```
repeat.xs <- seq(-2*pi,0,pi/100)
wave.3.repeat <- 0.5*sin(3*repeat.xs) + 0.25*sin(10*repeat.xs)
plot(xs,wave.3,type="l")

title("Repeating pattern")
points(repeat.xs,wave.3.repeat,type="l",col="red");
abline(h=0,v=c(-2*pi,0),lty=3)
```

## Repeating pattern





Hier ist eine R-Funktion zum Zeichnen von Trajektorien einer Fourier-Serie:

```
plot.fourier <- function(fourier.series, f.0, ts) {
 w <- 2*pi*f.0 trajectory <- sapply(ts, function(t)
fourier.series(t,w))
 plot(ts, trajectory, type="l", xlab="time", ylab="f(t)");
 abline(h=0,lty=3)}
}
```

Fourier-Serie und Transformationen online lesen: <https://riptutorial.com/de/r/topic/4139/fourier-serie-und-transformationen>

# Kapitel 49: Funktionale Programmierung

## Examples

### Integrierte Funktionen höherer Ordnung

R hat eine Reihe von in Funktionen höherer Ordnung aufgebaut: `Map`, `Reduce`, `Filter`, `Find`, `Position`, `Negate`.

`Map` wendet eine bestimmte Funktion auf eine Liste von Werten an:

```
words <- list("this", "is", "an", "example")
Map(toupper, words)
```

`Reduce` eine binäre Funktion auf eine Liste von Werten rekursiv angewendet.

```
Reduce(`*`, 1:10)
```

`Filter` eine Prädikatsfunktion und eine Liste mit Werten filtern, wird eine gefilterte Liste zurückgegeben, die nur Werte enthält, für die die Prädikatsfunktion TRUE ist.

```
Filter(is.character, list(1, "a", 2, "b", 3, "c"))
```

`Find` gegebene Prädikatsfunktion und eine Liste von Werten gibt den ersten Wert zurück, für den die Prädikatsfunktion WAHR ist.

```
Find(is.character, list(1, "a", 2, "b", 3, "c"))
```

`Position` die eine Prädikatsfunktion und eine Liste mit Werten gegeben hat, gibt die Position des ersten Werts in der Liste zurück, für den die Prädikatsfunktion WAHR ist.

```
Position(is.character, list(1, "a", 2, "b", 3, "c"))
```

`Negate` invertiert eine Prädikatsfunktion und gibt für Werte, bei denen TRUE zurückgegeben wurde, FALSE zurück und umgekehrt.

```
is.noncharacter <- Negate(is.character)
is.noncharacter("a")
is.noncharacter(mean)
```

Funktionale Programmierung online lesen: <https://riptutorial.com/de/r/topic/5050/funktionale-programmierung>

# Kapitel 50: Funktionsauswahl in R - Entfernen überflüssiger Funktionen

## Examples

### Entfernen von Features mit einer Varianz von null oder nahezu null

Ein Feature mit einer Varianz nahe null ist ein guter Kandidat für die Entfernung.

Sie können die numerische Abweichung unterhalb Ihrer eigenen Schwelle manuell ermitteln:

```
data("GermanCredit")
variances<-apply(GermanCredit, 2, var)
variances[which(variances<=0.0025)]
```

Oder Sie können das Caret-Paket verwenden, um eine Varianz nahe Null zu finden. Ein Vorteil hierbei ist, dass die Varianz nahe Null nicht in der numerischen Berechnung der Varianz definiert wird, sondern eher als Funktion der Seltenheit:

NearZeroVar diagnostiziert Prädiktoren mit einem eindeutigen Wert (dh Null-Varianz-Prädiktoren) oder Prädiktoren mit beiden folgenden Eigenschaften: Sie haben sehr wenige eindeutige Werte im Verhältnis zur Anzahl der Abtastwerte und dem Verhältnis der Häufigkeit des häufigsten Werts zur Häufigkeit des zweithäufigsten Wertes ist groß ... "

```
library(caret)
names(GermanCredit)[nearZeroVar(GermanCredit)]
```

### Entfernen von Features mit hoher NA-Anzahl

Wenn für ein Feature weitgehend Daten fehlen, ist es ein guter Kandidat zum Entfernen:

```
library(VIM)
data(sleep)
colMeans(is.na(sleep))
```

| BodyWgt    | BrainWgt   | NonD       | Dream      | Sleep      | Span       | Gest       |
|------------|------------|------------|------------|------------|------------|------------|
| 0.00000000 | 0.00000000 | 0.22580645 | 0.19354839 | 0.06451613 | 0.06451613 | 0.06451613 |
| Pred       | Exp        | Danger     |            |            |            |            |
| 0.00000000 | 0.00000000 | 0.00000000 |            |            |            |            |

In diesem Fall möchten wir vielleicht NonD und Dream entfernen, bei denen jeweils etwa 20% fehlende Werte vorliegen (Ihr Cutoff kann variieren).

### Entfernen von stark korrelierten Features

Durch eng korrelierte Features kann Ihr Modell mehr Varianz erhalten. Durch das Entfernen eines korrelierten Paares kann dies verringert werden. Es gibt viele Möglichkeiten, Korrelationen zu erkennen. Hier ist eins:

```
library(purrr) # in order to use keep()

select correlatable vars
toCorrelate<-mtcars %>% keep(is.numeric)

calculate correlation matrix
correlationMatrix <- cor(toCorrelate)

pick only one out of each highly correlated pair's mirror image
correlationMatrix[upper.tri(correlationMatrix)]<-0

and I don't remove the highly-correlated-with-itself group
diag(correlationMatrix)<-0

find features that are highly correlated with another feature at the +- 0.85 level
apply(correlationMatrix,2, function(x) any(abs(x)>=0.85))

 mpg cyl disp hp drat wt qsec vs am gear carb
TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Ich möchte sehen, was mit MPG so stark korreliert ist, und entscheiden, was ich behalten und was ich werfen soll. Gleiches für cyl und disp. Alternativ muss ich möglicherweise einige stark korrelierte Funktionen kombinieren.

Funktionsauswahl in R - Entfernen überflüssiger Funktionen online lesen:

<https://riptutorial.com/de/r/topic/7561/funktionsauswahl-in-r---entfernen-uberflussiger-funktionen>

# Kapitel 51: Generalisierte lineare Modelle

## Examples

### Logistische Regression auf Titanic-Dataset

Die logistische Regression ist ein besonderer Fall des *verallgemeinerten linearen Modells*, das zur Modellierung dichotomer Ergebnisse verwendet wird ( *Probit*- und *Komplementär-Log-Log*-Modelle sind eng miteinander verbunden).

Der Name stammt von der verwendeten *Link-Funktion*, der *Logit*- oder *Log-Odds*- Funktion. Die Umkehrfunktion des *Logits* wird als *logistische Funktion* bezeichnet und ist gegeben durch:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Diese Funktion nimmt einen Wert zwischen  $]-\infty; +\infty[$  an und gibt einen Wert zwischen 0 und 1 zurück. dh die *logistische Funktion* nimmt einen linearen Prädiktor an und gibt eine Wahrscheinlichkeit zurück.

Logistische Regression kann mit der `glm` Funktion mit der Option `family = binomial` (Abkürzung für `family = binomial(link="logit")`); *logit* ist die Standard-Linkfunktion für die binomiale Familie).

In diesem Beispiel versuchen wir, das Schicksal der Passagiere an Bord der RMS Titanic vorherzusagen.

Lesen Sie die Daten:

```
url <- "http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.txt"
titanic <- read.csv(file = url, stringsAsFactors = FALSE)
```

Reinigen Sie die fehlenden Werte:

In diesem Fall ersetzen wir die fehlenden Werte durch eine Näherung, den Durchschnitt.

```
titanic$age[is.na(titanic$age)] <- mean(titanic$age, na.rm = TRUE)
```

Trainieren Sie das Modell:

```
titanic.train <- glm(survived ~ pclass + sex + age,
 family = binomial, data = titanic)
```

Zusammenfassung des Modells:

```
summary(titanic.train)
```

Die Ausgabe:

```
Call:
glm(formula = survived ~ pclass + sex + age, family = binomial, data = titanic)

Deviance Residuals:
 Min 1Q Median 3Q Max
-2.6452 -0.6641 -0.3679 0.6123 2.5615

Coefficients:
 Estimate Std. Error z value Pr(>|z|)
(Intercept) 3.552261 0.342188 10.381 < 2e-16 ***
pclass2nd -1.170777 0.211559 -5.534 3.13e-08 ***
pclass3rd -2.430672 0.195157 -12.455 < 2e-16 ***
sexmale -2.463377 0.154587 -15.935 < 2e-16 ***
age -0.042235 0.007415 -5.696 1.23e-08 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

 Null deviance: 1686.8 on 1312 degrees of freedom
Residual deviance: 1165.7 on 1308 degrees of freedom
AIC: 1175.7

Number of Fisher Scoring iterations: 5
```

- Das erste, was angezeigt wird, ist der Anruf. Es erinnert an das Modell und die angegebenen Optionen.
- Als Nächstes sehen wir die Abweichungs-Residuen, die ein Maß für die Modellanpassung sind. Dieser Teil der Ausgabe zeigt die Verteilung der Abweichungsrestwerte für einzelne Fälle, die im Modell verwendet werden.
- Der nächste Teil der Ausgabe zeigt die Koeffizienten, ihre Standardfehler, die Z-Statistik (manchmal auch Wald-Z-Statistik genannt) und die zugehörigen p-Werte.
  - Die qualitativen Variablen sind "dummifiziert". Eine Modalität wird als Referenz betrachtet. Die Referenzmodalität kann mit `1` in der Formel geändert werden.
  - Alle vier Prädiktoren sind mit 0,1% statistisch signifikant.
  - Die logistischen Regressionskoeffizienten geben die Änderung der logarithmischen Quoten des Ergebnisses für eine Erhöhung der Prädiktorvariablen um eine Einheit an.
  - Um die *Odds Ratio* (multiplikative Änderung der Überlebenswahrscheinlichkeit pro Einheit in einer Prädiktorvariablen) zu sehen, potenzieren Sie den Parameter.
  - Verwenden Sie `confint`, um das Konfidenzintervall (CI) des Parameters `confint`.
- Unterhalb der Koeffiziententabelle befinden sich Fit-Indizes, einschließlich der Null- und Abweichungs-Residuen und des Akaike-Informationskriteriums (AIC), die zum Vergleich der Modellleistung verwendet werden können.
  - Beim Vergleich von Modellen mit maximaler Wahrscheinlichkeit mit den gleichen Daten ist die Anpassung umso besser, je kleiner der AIC ist.
  - Ein Maß für die Modellanpassung ist die Bedeutung des Gesamtmodells. Dieser Test fragt, ob das Modell mit Prädiktoren wesentlich besser passt als ein Modell mit nur einem Intercept (dh einem Nullmodell).

## Beispiel für Quotenquoten:

```
exp(coef(titanic.train)[3])

pclass3rd
0.08797765
```

Bei diesem Modell haben die Passagiere der 3. Klasse im Vergleich zur ersten Klasse etwa ein Zehntel der Überlebenschancen.

## Beispiel eines Konfidenzintervalls für die Parameter:

```
confint(titanic.train)

Waiting for profiling to be done...
 2.5 % 97.5 %
(Intercept) 2.89486872 4.23734280
pclass2nd -1.58986065 -0.75987230
pclass3rd -2.81987935 -2.05419500
sexmale -2.77180962 -2.16528316
age -0.05695894 -0.02786211
```

## Beispiel zur Berechnung der Signifikanz des Gesamtmodells:

Die Teststatistik wird im Chi-Quadrat mit Freiheitsgraden verteilt, die den Unterschieden der Freiheitsgrade zwischen dem aktuellen und dem Nullmodell entsprechen (dh der Anzahl der Prädiktorvariablen im Modell).

```
with(titanic.train, pchisq(null.deviance - deviance, df.null - df.residual
, lower.tail = FALSE))
[1] 1.892539e-111
```

Der p-Wert liegt nahe 0 und zeigt ein stark signifikantes Modell.

Generalisierte lineare Modelle online lesen: <https://riptutorial.com/de/r/topic/2892/generalisierte-lineare-modelle>

---

# Kapitel 52: ggplot2

## Bemerkungen

ggplot2 hat eine eigene perfekte Referenz-Website <http://ggplot2.tidyverse.org/> .

In den meisten `data.frame` ist es zweckmäßiger, die Struktur oder den Inhalt der aufgezeichneten Daten (z. B. eines `data.frame` ) anzupassen, als nachträglich `data.frame` in der `data.frame` anzupassen.

RStudio veröffentlicht ein sehr hilfreiches "Data Visualization with ggplot2" Cheatsheet, das Sie [hier finden können](#) .

## Examples

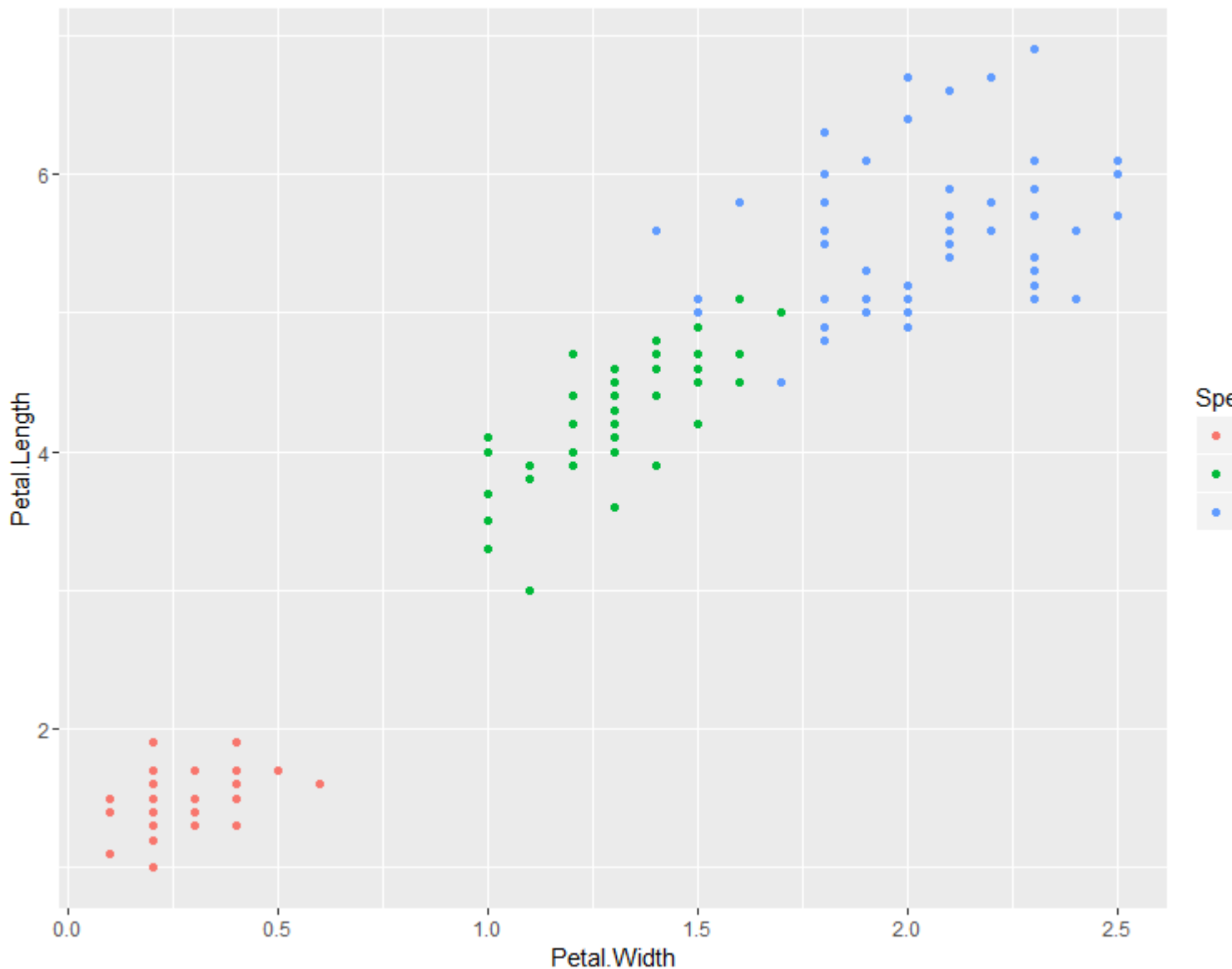
### Streudiagramme

Wir zeichnen ein einfaches Streudiagramm mit dem integrierten Iris-Datensatz wie folgt:

```
library(ggplot2)
ggplot(iris, aes(x = Petal.Width, y = Petal.Length, color = Species)) +
 geom_point()
```

Das gibt:



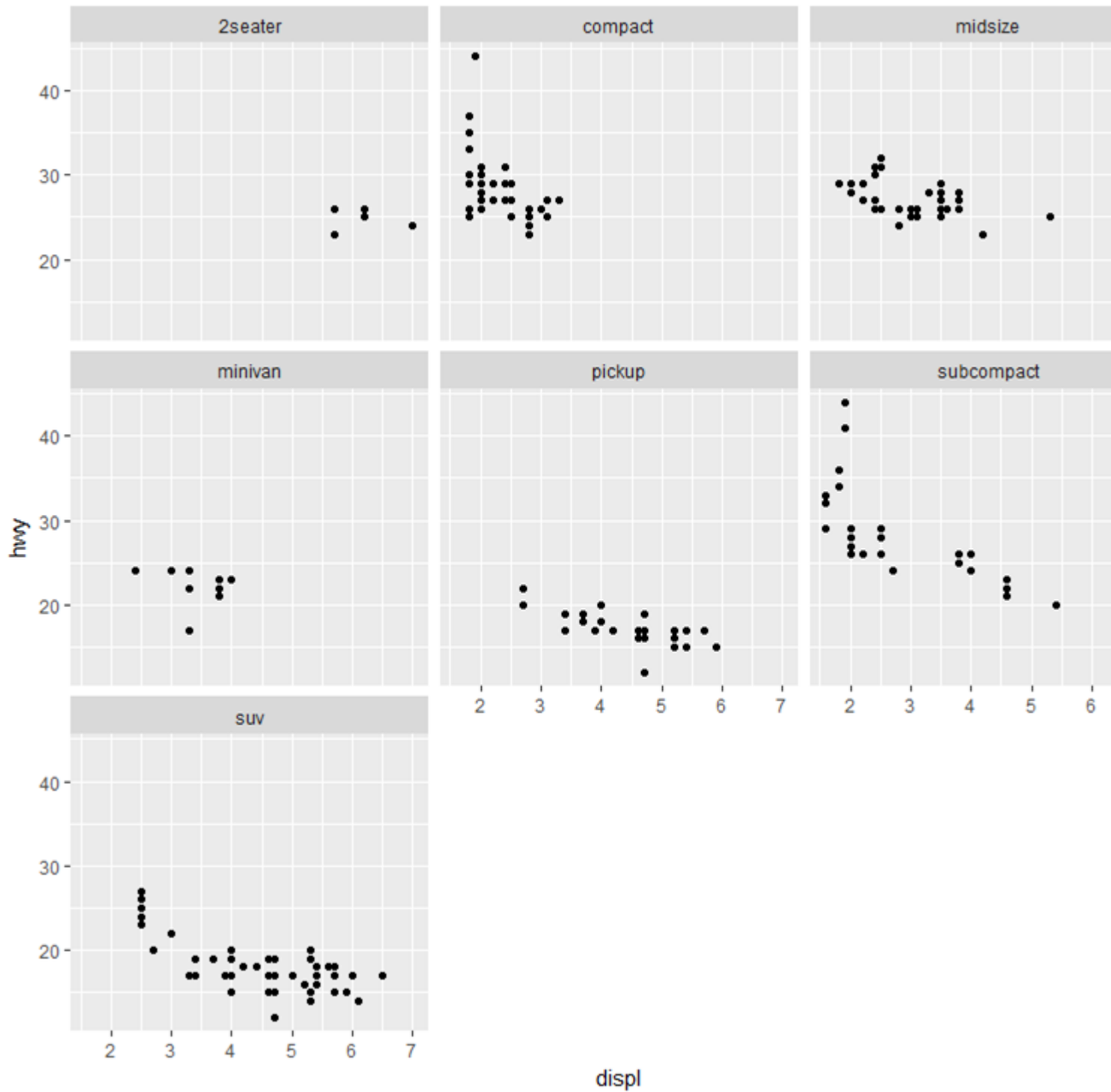


## Mehrere Plots anzeigen

Zeigen Sie mehrere Darstellungen in einem Bild mit den verschiedenen `facet`. Ein Vorteil dieser Methode besteht darin, dass alle Achsen die gleiche Skalierung über Diagramme hinweg haben, so dass sie auf einen Blick leicht zu vergleichen sind. Wir verwenden den in `ggplot2` enthaltenen `mpg` Datensatz.

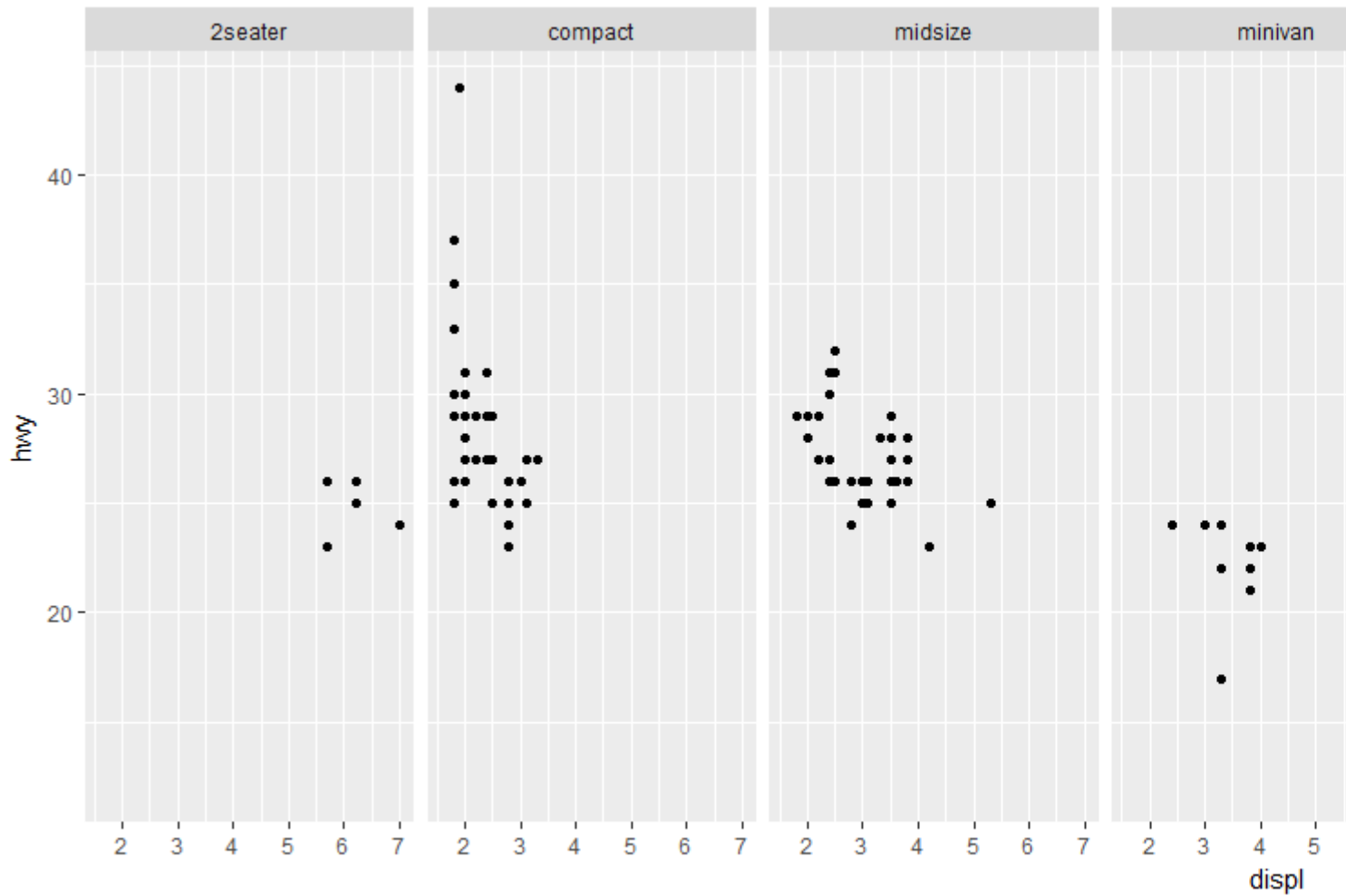
**Diagramme Zeile für Zeile umbrechen (versucht, ein quadratisches Layout zu erstellen):**

```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point() +
 facet_wrap(~class)
```



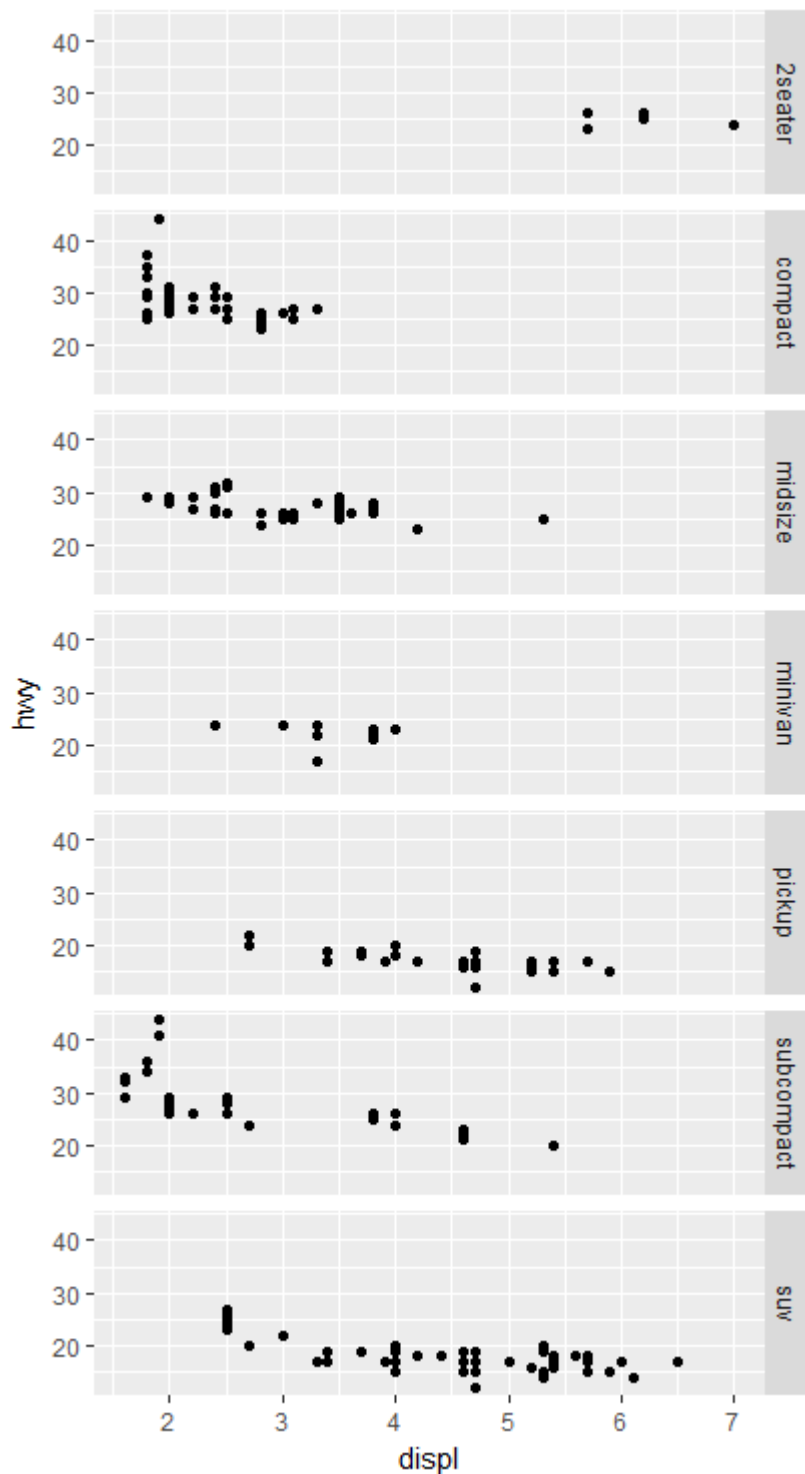
**Mehrere Diagramme in einer Zeile und in mehreren Spalten anzeigen:**

```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point() +
 facet_grid(.~class)
```



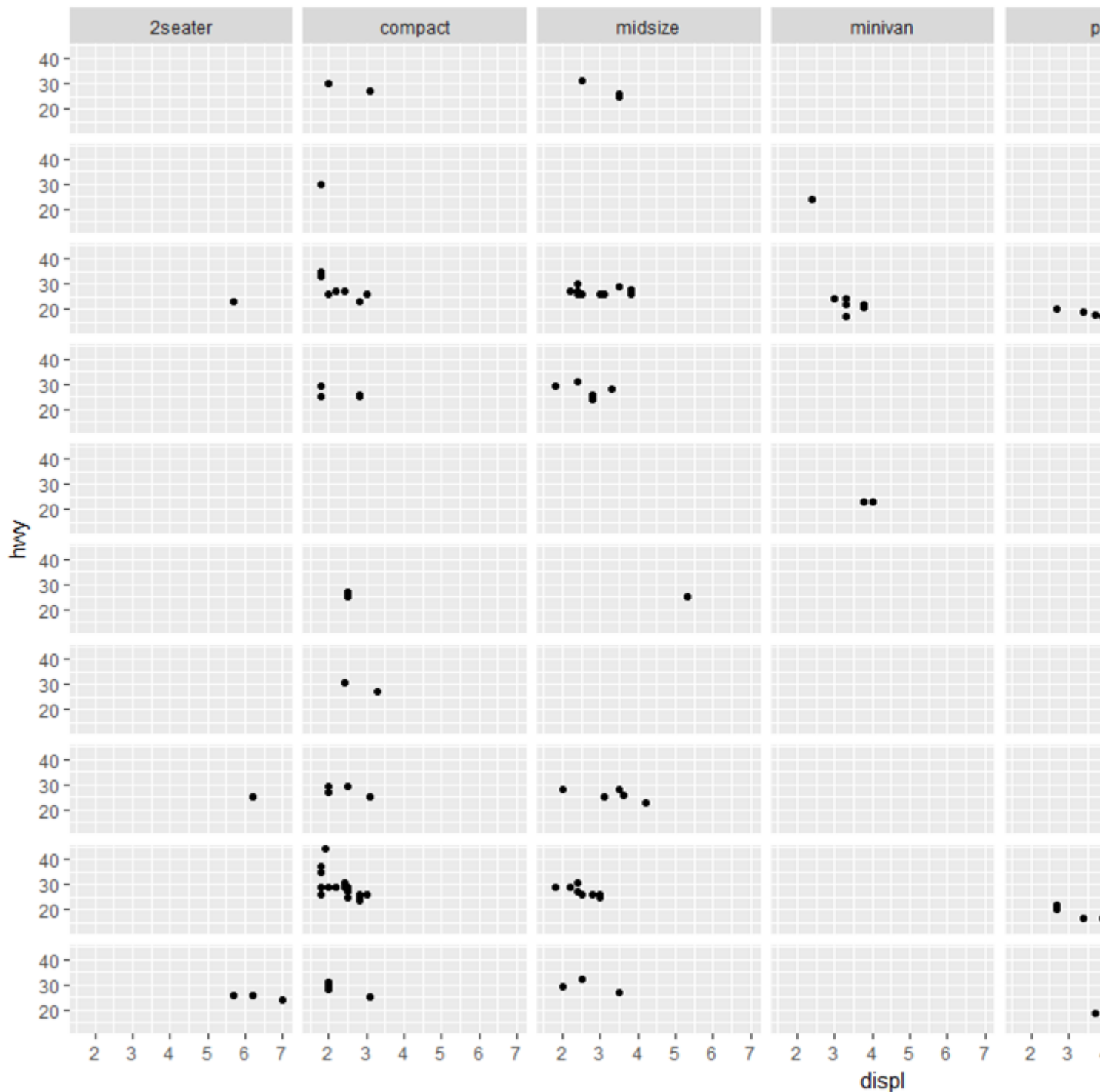
**Mehrere Diagramme in einer Spalte, in mehreren Zeilen anzeigen:**

```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point() +
 facet_grid(class~.)
```



**Zeigen Sie mehrere Diagramme in einem Raster mit zwei Variablen an:**

```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point() +
 facet_grid(trans~class) # "row" parameter, then "column" parameter
```



## Bereiten Sie Ihre Daten für das Plotten vor

`ggplot2` funktioniert am besten mit einem langen `ggplot2`. Die folgenden Beispieldaten, die die Preise für Süßigkeiten an 20 verschiedenen Tagen darstellen, werden in einem als breit bezeichneten Format dargestellt, da jede Kategorie eine Spalte enthält.

```
set.seed(47)
sweetsWide <- data.frame(date = 1:20,
 chocolate = runif(20, min = 2, max = 4),
 iceCream = runif(20, min = 0.5, max = 1),
 candy = runif(20, min = 1, max = 3))
```

```
head(sweetsWide)
date chocolate iceCream candy
1 1 3.953924 0.5890727 1.117311
2 2 2.747832 0.7783982 1.740851
3 3 3.523004 0.7578975 2.196754
4 4 3.644983 0.5667152 2.875028
5 5 3.147089 0.8446417 1.733543
6 6 3.382825 0.6900125 1.405674
```

Für die Konvertierung von `sweetsWide` in das lange Format für die Verwendung mit `ggplot2` können mehrere nützliche Funktionen von Basis R und die Pakete `reshape2`, `data.table` und `tidyr` (in chronologischer Reihenfolge) verwendet werden:

```
reshape from base R
sweetsLong <- reshape(sweetsWide, idvar = 'date', direction = 'long',
 varying = list(2:4), new.row.names = NULL, times = names(sweetsWide)[-1])

melt from 'reshape2'
library(reshape2)
sweetsLong <- melt(sweetsWide, id.vars = 'date')

melt from 'data.table'
which is an optimized & extended version of 'melt' from 'reshape2'
library(data.table)
sweetsLong <- melt(setDT(sweetsWide), id.vars = 'date')

gather from 'tidyr'
library(tidyr)
sweetsLong <- gather(sweetsWide, sweet, price, chocolate:candy)
```

Alle ergeben ein ähnliches Ergebnis:

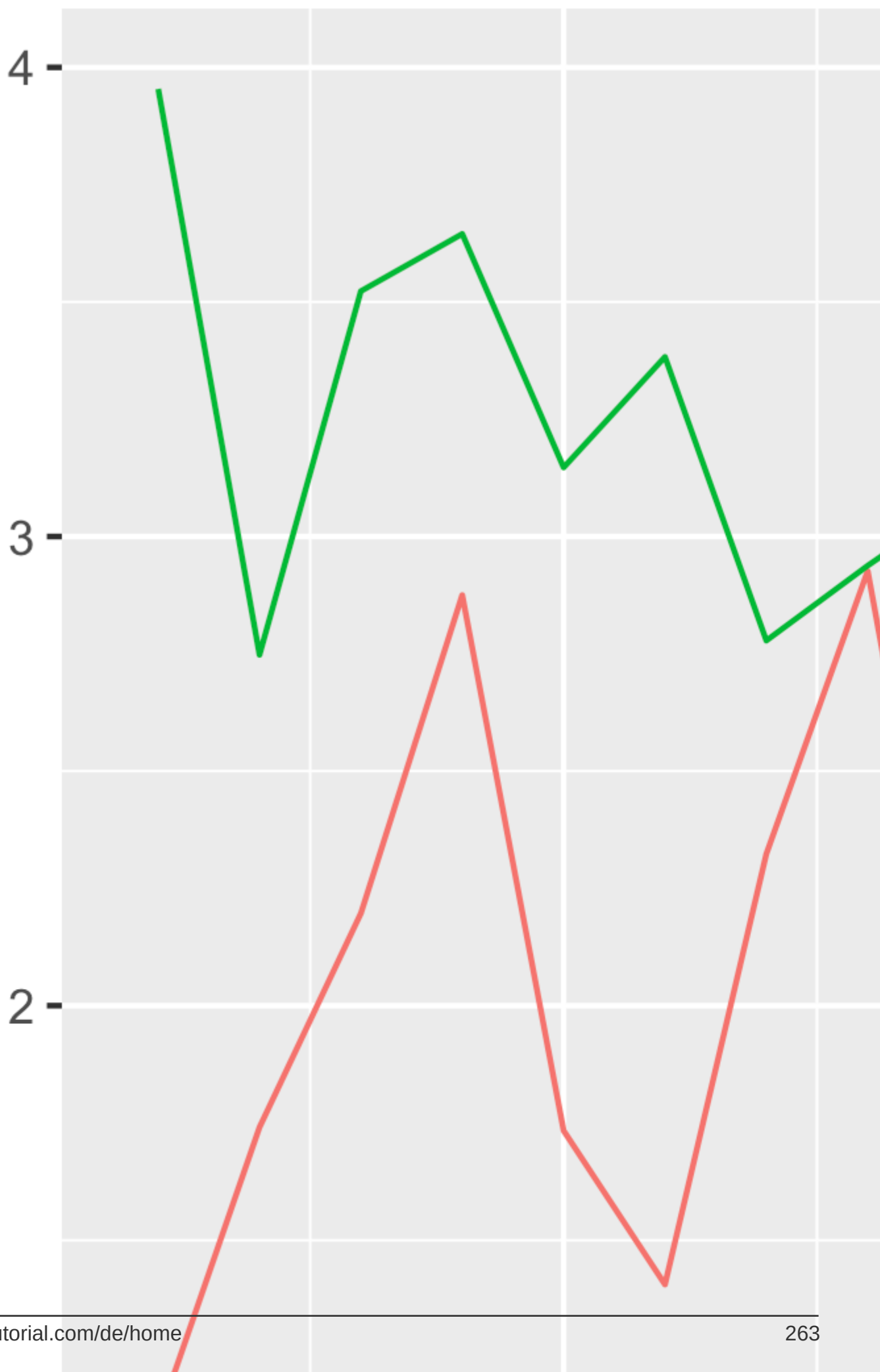
```
head(sweetsLong)
date sweet price
1 1 chocolate 3.953924
2 2 chocolate 2.747832
3 3 chocolate 3.523004
4 4 chocolate 3.644983
5 5 chocolate 3.147089
6 6 chocolate 3.382825
```

Weitere Informationen zum Konvertieren von Daten zwischen *langen* und *breiten* Formaten finden Sie unter [Umformen von Daten zwischen langen und breiten Formularen](#).

Das Ergebnis von `sweetsLong` enthält eine `sweetsLong` und eine Spalte, in der die Süßigkeit beschrieben wird. Nun ist das Plotten viel einfacher:

```
library(ggplot2)
ggplot(sweetsLong, aes(x = date, y = price, colour = sweet)) + geom_line()
```

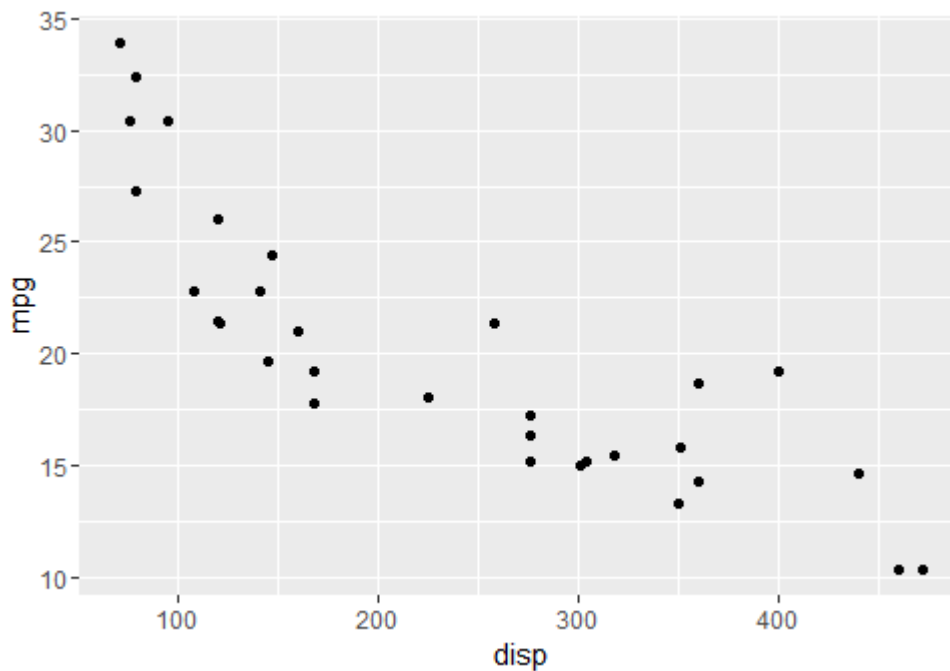
price



Funktion ähneln und versucht, Ihre Daten immer darzustellen, ohne zu viele Angaben zu machen.

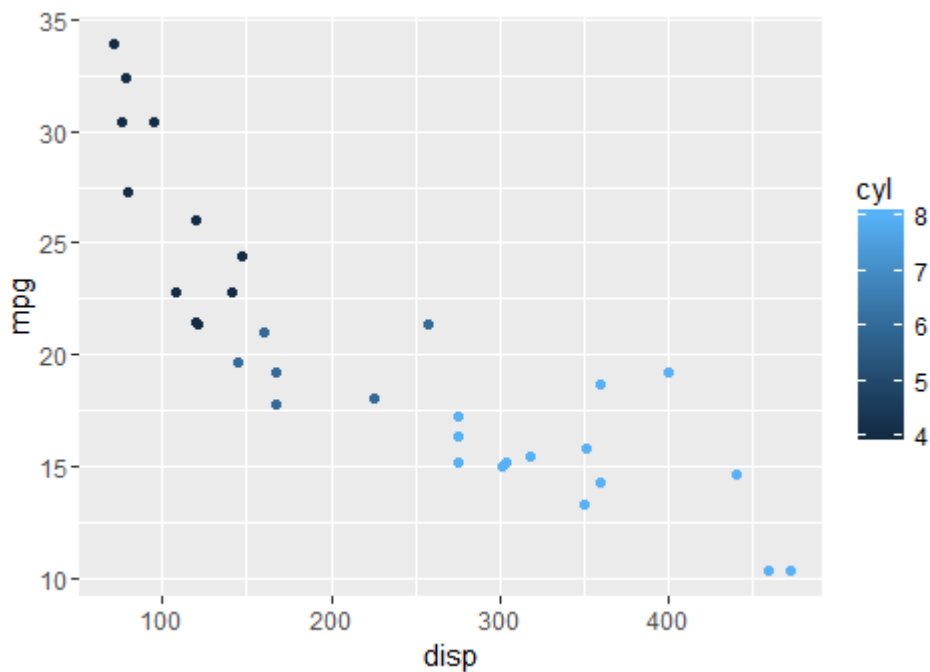
## grundlegendes qplot

```
qplot(x = disp, y = mpg, data = mtcars)
```



## Farben hinzufügen

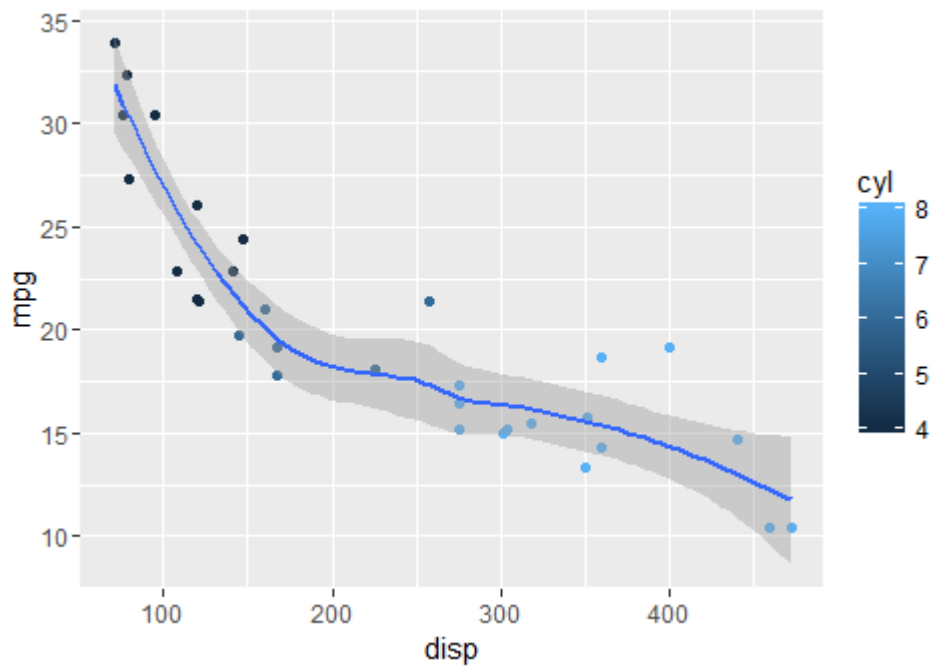
```
qplot(x = disp, y = mpg, colour = cyl, data = mtcars)
```



## ein glatter hinzufügen

```
qplot(x = disp, y = mpg, geom = c("point", "smooth"), data = mtcars)
```





ggplot2 online lesen: <https://riptutorial.com/de/r/topic/1334/ggplot2>

---

# Kapitel 53: Glänzend

## Examples

### App erstellen

Shiny ist ein von [RStudio](#) entwickeltes [R](#)-Paket, mit dem Webseiten erstellt werden können, um die Ergebnisse einer Analyse in R interaktiv anzuzeigen.

Es gibt zwei einfache Möglichkeiten, eine Shiny-App zu erstellen:

- in einer `.R` Datei oder
- in zwei Dateien: `ui.R` und `server.R`.

Eine Shiny-App besteht aus zwei Teilen:

- **ui** : Ein Benutzeroberflächenskript, das das Layout und das Erscheinungsbild der Anwendung steuert.
- **Server** : Ein Serverskript, das Code enthält, damit die Anwendung reagieren kann.

---

## Eine Datei

```
library(shiny)

Create the UI
ui <- shinyUI(fluidPage(
 # Application title
 titlePanel("Hello World!")
))

Create the server function
server <- shinyServer(function(input, output){})

Run the app
shinyApp(ui = ui, server = server)
```

---

## Zwei Dateien

### Erstellen Sie die `ui.R` Datei

```
library(shiny)

Define UI for application
shinyUI(fluidPage(
 # Application title
 titlePanel("Hello World!")
))
```

## Erstellen Sie die `server.R` Datei

```
library(shiny)

Define server logic
shinyServer(function(input, output){})
```

### Radio knopf

Sie können eine Reihe von Optionsschaltflächen erstellen, die zum Auswählen eines Elements aus einer Liste verwendet werden.

Es ist möglich, die Einstellungen zu ändern:

- ausgewählt: Der ursprünglich ausgewählte Wert (Zeichen (0) für keine Auswahl)
- Inline: horizontal oder vertikal
- Breite

Es ist auch möglich, HTML hinzuzufügen.

```
library(shiny)

ui <- fluidPage(
 radioButtons("radio",
 label = HTML('Welcome

 Your favorite color is red ?'),
 choices = list("TRUE" = 1, "FALSE" = 2),
 selected = 1,
 inline = T,
 width = "100%",
 fluidRow(column(3, textOutput("value"))))

server <- function(input, output){
 output$value <- renderPrint({
 if(input$radio == 1){return('Great !')}
 else{return("Sorry !")}}})

shinyApp(ui = ui, server = server)
```

**Welcome**

**Your favorite color is red ?**

☒ TRUE ☐ FALSE

[1] "Great !"

### Kontrollkästchen Gruppe

Erstellen Sie eine Gruppe von Kontrollkästchen, mit denen Sie mehrere Optionen unabhängig voneinander umschalten können. Der Server empfängt die Eingabe als Zeichenvektor der

ausgewählten Werte.

```
library(shiny)

ui <- fluidPage(
 checkboxGroupInput("checkboxGroup1", label = h3("This is a Checkbox group"),
 choices = list("1" = 1, "2" = 2, "3" = 3),
 selected = 1),
 fluidRow(column(3, verbatimTextOutput("text_choice")))
)

server <- function(input, output){
 output$text_choice <- renderPrint({
 return(paste0("You have chosen the choice ",input$checkboxGroup1))
 })
}

shinyApp(ui = ui, server = server)
```

## This is a Checkbox group

- ☒ 1  
☐ 2  
☐ 3

```
[1] "You have chosen the choice 1"
```

Es ist möglich, die Einstellungen zu ändern:

- Label: Titel
- Auswahlmöglichkeiten: ausgewählte Werte
- ausgewählt: Der ursprünglich ausgewählte Wert (NULL für keine Auswahl)
- Inline: horizontal oder vertikal
- Breite

Es ist auch möglich, HTML hinzuzufügen.

## Kästchen auswählen

Erstellen Sie eine Auswahlliste, mit der ein einzelnes oder mehrere Elemente aus einer Werteliste ausgewählt werden können.

```
library(shiny)

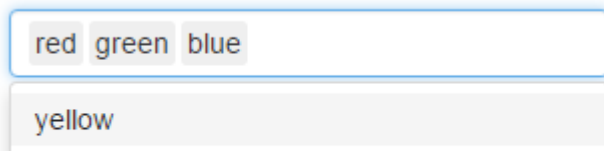
ui <- fluidPage(
 selectInput("id_selectInput",
 label = HTML('What is your favorite color ?'),
 multiple = TRUE,
 choices = list("red" = "red", "green" = "green", "blue" = "blue", "yellow" =
"yellow"),
 selected = NULL),
```

```
br(), br(),
fluidRow(column(3, textOutput("text_choice"))))

server <- function(input, output){
 output$text_choice <- renderPrint({
 return(input$id_selectInput)})
}

shinyApp(ui = ui, server = server)
```

### What is your favorite color ?



```
[1] "red" "green" "blue"
```

Es ist möglich, die Einstellungen zu ändern:

- Label: Titel
- Auswahlmöglichkeiten: ausgewählte Werte
- ausgewählt: Der ursprünglich ausgewählte Wert (NULL für keine Auswahl)
- mehrere: TRUE oder FALSE
- Breite
- Größe
- selectize: TRUE oder FALSE (für selectize.js verwenden oder nicht, Anzeige ändern)

Es ist auch möglich, HTML hinzuzufügen.

## Starten Sie eine Shiny-App

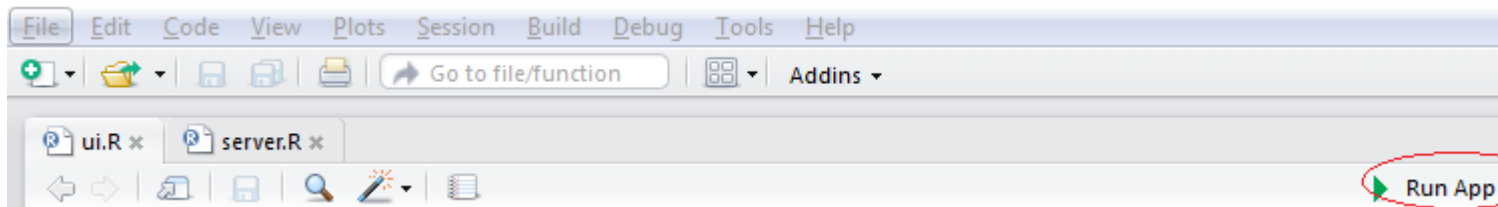
Sie können eine Anwendung auf verschiedene Arten starten, je nachdem, wie Sie Ihre App erstellen. Wenn Ihre App in zwei Dateien aufgeteilt ist, `ui.R` und `server.R` oder wenn sich Ihre gesamte App in einer Datei befindet.

## 1. Zwei Dateien App

Ihre beiden Dateien `ui.R` und `server.R` müssen sich im selben Ordner befinden. Sie können Ihre App dann starten, indem Sie in der Konsole die `shinyApp()` Funktion `shinyApp()` und den Pfad des Verzeichnisses übergeben, in dem sich die Shiny-App befindet.

```
shinyApp("path_to_the_folder_containing_the_files")
```

Sie können die App auch direkt von Rstudio aus starten, indem Sie auf die Schaltfläche **App** `ui.R` `server.R`, die beim Öffnen einer `ui.R` oder `server.R` Datei in `ui.R` `server.R` wird.



Oder Sie können `runApp()` einfach in die Konsole schreiben, wenn das Shiny App-Verzeichnis Ihr Arbeitsverzeichnis ist.

## 2. Eine Datei App

Wenn Sie Ihre in einer `R` Datei erstellen, können Sie sie auch mit der `shinyApp()` Funktion starten.

- innerhalb Ihres Codes:

```
library(shiny)

ui <- fluidPage() #Create the ui
server <- function(input, output){} #create the server

shinyApp(ui = ui, server = server) #run the App
```

- in der Konsole durch Hinzufügen eines Pfads zu einer `.R` Datei, die die Shiny-Anwendung enthält, mit dem Parameter `appFile` :

```
shinyApp(appFile="path_to_my_R_file_containig_the_app")
```

## Kontrollieren Sie Widgets

| Funktion           | Widget                                                       |
|--------------------|--------------------------------------------------------------|
| Aktionsknopf       | Aktionsknopf                                                 |
| checkboxGroupInput | Eine Gruppe von Kontrollkästchen                             |
| checkboxInput      | Ein einzelnes Kontrollkästchen                               |
| dateInput          | Ein Kalender zur Auswahl des Datums                          |
| dateRangeInput     | Ein Kalender zum Auswählen eines Datumsbereichs              |
| fileInput          | Ein Assistent zum Hochladen von Dateien                      |
| Hilfstext          | Hilfetext, der einem Eingabeformular hinzugefügt werden kann |
| numerischerEingang | Ein Feld zur Eingabe von Zahlen                              |
| Radio Knöpfe       | Eine Reihe von Optionsfeldern                                |
| selectEingabe      | Eine Box mit Auswahlmöglichkeiten                            |

| Funktion     | Widget                        |
|--------------|-------------------------------|
| sliderInput  | Eine Schiebereglerleiste      |
| submitButton | Ein Senden-Button             |
| Text Eingabe | Ein Feld zur Eingabe von Text |

```

library(shiny)

Create the UI
ui <- shinyUI(fluidPage(
 titlePanel("Basic widgets"),

 fluidRow(

 column(3,
 h3("Buttons"),
 actionButton("action", label = "Action"),
 br(),
 br(),
 submitButton("Submit")),

 column(3,
 h3("Single checkbox"),
 checkboxInput("checkbox", label = "Choice A", value = TRUE)),

 column(3,
 checkboxGroupInput("checkGroup",
 label = h3("Checkbox group"),
 choices = list("Choice 1" = 1,
 "Choice 2" = 2, "Choice 3" = 3),
 selected = 1)),

 column(3,
 dateInput("date",
 label = h3("Date input"),
 value = "2014-01-01")
),

 fluidRow(

 column(3,
 dateRangeInput("dates", label = h3("Date range"))),

 column(3,
 fileInput("file", label = h3("File input"))),

 column(3,
 h3("Help text"),
 helpText("Note: help text isn't a true widget,",
 "but it provides an easy way to add text to",
 "accompany other widgets.")),

 column(3,
 numericInput("num",
 label = h3("Numeric input"),
 value = 1)
),

),

```

```

fluidRow(

 column(3,
 radioButtons("radio", label = h3("Radio buttons"),
 choices = list("Choice 1" = 1, "Choice 2" = 2,
 "Choice 3" = 3), selected = 1)),

 column(3,
 selectInput("select", label = h3("Select box"),
 choices = list("Choice 1" = 1, "Choice 2" = 2,
 "Choice 3" = 3), selected = 1)),

 column(3,
 sliderInput("slider1", label = h3("Sliders"),
 min = 0, max = 100, value = 50),
 sliderInput("slider2", "",
 min = 0, max = 100, value = c(25, 75))
),

 column(3,
 textInput("text", label = h3("Text input"),
 value = "Enter text...")
)
)

))

Create the server function
server <- shinyServer(function(input, output){})

Run the app
shinyApp(ui = ui, server = server)

```

## Debuggen

`debug()` und `debugonce()` funktionieren im Zusammenhang mit den meisten Shiny-Debuggen nicht gut. `browser()` Anweisungen, die an kritischen Stellen eingefügt werden, können jedoch einen Einblick in die Funktionsweise Ihres Shiny-Codes (nicht) geben. Siehe auch: [Debuggen mit dem browser\(\)](#)

## Showcase-Modus

Der **Showcase-Modus** zeigt Ihre App neben dem Code an, der sie generiert, und markiert die Codezeilen in `server.R`, während sie ausgeführt werden.

Es gibt zwei Möglichkeiten, den Showcase-Modus zu aktivieren:

- Starten Sie die Shiny-App mit dem Argument `display.mode = "showcase"`, z. B.  
`runApp("MyApp", display.mode = "showcase")`.
- Erstellen Sie eine Datei mit dem Namen `DESCRIPTION` in Ihrem Shiny-App-Ordner und fügen Sie diese Zeile hinzu: `DisplayMode: Showcase`.

## Reactive Log Visualizer



**Reactive Log Visualizer** bietet ein interaktives browserbasiertes Tool zur Darstellung reaktiver Abhängigkeiten und der Ausführung in Ihrer Anwendung. Um Reactive Log Visualizer zu aktivieren, führen Sie die `options(shiny.reactlog=TRUE)` in der R-Konsole aus oder fügen Sie diese Codezeile in Ihre `server.R`-Datei ein. Um den Reactive Log Visualizer zu starten, drücken Sie bei laufender App Strg + F3 unter Windows bzw. Befehl + F3 bei Mac. Verwenden Sie die linken und rechten Pfeiltasten, um im Reactive Log Visualizer zu navigieren.

Glänzend online lesen: <https://riptutorial.com/de/r/topic/2044/glanzend>

# Kapitel 54: GPU-beschleunigtes Computing

## Bemerkungen

GPU-Computing erfordert eine "Plattform", über die die Hardware angeschlossen und verwendet werden kann. Die zwei primären einfachen Sprachen, die dies erreichen, sind CUDA und OpenCL. Ersteres erfordert die Installation des proprietären NVIDIA CUDA Toolkit und ist nur auf NVIDIA GPUs anwendbar. Letzteres ist sowohl firmenintern (zB NVIDIA, AMD, Intel) als auch hardwareunabhängig (CPU oder GPU), erfordert jedoch die Installation eines SDK (Software Development Kit). Um eine GPU über R verwenden zu können, müssen Sie zuerst eine dieser Software installieren.

Sobald das CUDA Toolkit oder ein OpenCL SDK installiert ist, können Sie ein geeignetes R-Paket installieren. Fast alle R-GPU-Pakete sind von CUDA abhängig und auf NVIDIA-GPUs beschränkt. Diese schließen ein:

1. [gputools](#)
2. [cudaBayesreg](#)
3. [HiPLARM](#)
4. [Matrix](#)

Derzeit gibt es nur zwei OpenCL-fähige Pakete

1. [OpenCL](#) - Schnittstelle von R zu OpenCL
2. [gpuR](#) - Allzweckbibliothek

**Warnung** - Die Installation kann für verschiedene Betriebssysteme mit unterschiedlichen Umgebungsvariablen und GPU-Plattformen schwierig sein.

## Examples

### gpuR gpuMatrix-Objekte

```
library(gpuR)

gpuMatrix objects
X <- gpuMatrix(rnorm(100), 10, 10)
Y <- gpuMatrix(rnorm(100), 10, 10)

transfer data to GPU when operation called
automatically copied back to CPU
Z <- X %*% Y
```

### gpuR vclMatrix-Objekte

```
library(gpuR)
```

```
vclMatrix objects
X <- vclMatrix(rnorm(100), 10, 10)
Y <- vclMatrix(rnorm(100), 10, 10)

data always on GPU
no data transfer
Z <- X %*% Y
```

GPU-beschleunigtes Computing online lesen: <https://riptutorial.com/de/r/topic/4680/gpu-beschleunigtes-computing>

---

# Kapitel 55: Hashmaps

## Examples

### Umgebungen als Hash-Maps

*Hinweis: In den folgenden Passagen werden die Begriffe **Hash-Map** und **Hash-Tabelle** austauschbar verwendet und beziehen sich auf [dasselbe Konzept](#), nämlich auf eine Datenstruktur, die eine effiziente Schlüsselsuche durch Verwendung einer internen Hash-Funktion ermöglicht.*

## Einführung

Obwohl R keine native Hashtabellenstruktur bereitstellt, kann eine ähnliche Funktionalität erreicht werden, indem die Tatsache genutzt wird, dass das `environment`, das von `new.env` (standardmäßig), Hash-Schlüsselsuchen bietet. Die folgenden beiden Anweisungen sind gleichwertig, da der `hash` Parameter standardmäßig `TRUE`:

```
H <- new.env(hash = TRUE)
H <- new.env()
```

Darüber hinaus kann angegeben werden, dass die interne Hashtabelle über den Parameter `size` mit einer bestimmten Größe vorab zugewiesen wird. Der Parameter hat den Standardwert 29. Wie alle anderen R-Objekte verwaltet auch die `environment` ihren eigenen Speicher und nimmt bei Bedarf mit der Kapazität zu. Obwohl es nicht erforderlich ist, einen nicht voreingestellten Wert für `size` anzufordern, kann dies einen geringfügigen Leistungsvorteil zur Folge haben, **wenn** das Objekt (möglicherweise) eine sehr große Anzahl von Elementen enthält. Es ist erwähnenswert, dass das Zuweisen von zusätzlichem Speicherplatz über die `size` sich nicht zu einem Objekt mit einem größeren Speicherbedarf führt:

```
object.size(new.env())
56 bytes

object.size(new.env(size = 10e4))
56 bytes
```

---

## Einfügung

Das Einfügen von Elementen kann entweder mit der für die `environment` bereitgestellten `[[<-` oder `$<-` Methode durchgeführt werden, **jedoch nicht** mit der Zuweisung einer einzelnen Klammer (`[<-`):

```
H <- new.env()
```

```
H[["key"]] <- rnorm(1)

key2 <- "xyz"
H[key2] <- data.frame(x = 1:3, y = letters[1:3])

H$another_key <- matrix(rbinom(9, 1, 0.5) > 0, nrow = 3)

H["error"] <- 42
#Error in H["error"] <- 42 :
object of type 'environment' is not subsettable
```

Wie bei anderen Facetten von R wird die erste Methode ( `object[[key]] <- value` ) im Allgemeinen der zweiten Methode ( `object[[key]] <- value` `object$key <- value` ) vorgezogen, da im ersten Fall eine Variable anstelle eines Literalwerts verwendet werden kann (zB `key2` im obigen Beispiel).

Wie bei Hash-Map-Implementierungen im Allgemeinen wird das `environment` **keine** doppelten Schlüssel speichern. Der Versuch, ein Schlüsselwertpaar für einen vorhandenen Schlüssel einzufügen, ersetzt den zuvor gespeicherten Wert:

```
H[["key3"]] <- "original value"

H[["key3"]] <- "new value"

H[["key3"]]
[1] "new value"
```

## Schlüsselsuche

Ebenso kann auf Elemente mit `[` oder `$` ] zugegriffen werden, jedoch nicht mit `[ :`

```
H[["key"]]
[1] 1.630631

H[key2] ## assuming key2 <- "xyz"
x y
1 1 a
2 2 b
3 3 c

H$another_key
[,1] [,2] [,3]
[1,] TRUE TRUE TRUE
[2,] FALSE FALSE FALSE
[3,] TRUE TRUE TRUE

H[1]
#Error in H[1] : object of type 'environment' is not subsettable
```

## Überprüfen der Hashkarte

Da es sich um eine gewöhnliche `environment` , kann die Hash-Karte mit typischen Mitteln überprüft

werden:

```
names(H)
#[1] "another_key" "xyz" "key" "key3"

ls(H)
#[1] "another_key" "key" "key3" "xyz"

str(H)
#<environment: 0x7828228>

ls.str(H)
another_key : logi [1:3, 1:3] TRUE FALSE TRUE TRUE FALSE TRUE ...
key : num 1.63
key3 : chr "new value"
xyz : 'data.frame': 3 obs. of 2 variables:
$ x: int 1 2 3
$ y: chr "a" "b" "c"
```

Elemente können mit `rm` entfernt werden:

```
rm(list = c("key", "key3"), envir = H)

ls.str(H)
another_key : logi [1:3, 1:3] TRUE FALSE TRUE TRUE FALSE TRUE ...
xyz : 'data.frame': 3 obs. of 2 variables:
$ x: int 1 2 3
$ y: chr "a" "b" "c"
```

## Flexibilität

Ein Hauptvorteil der Verwendung von `environment` als Hashtabellen besteht darin, dass praktisch alle Arten von Objekten als Wert gespeichert werden können, *selbst in anderen* `environment`:

```
H2 <- new.env()

H2[["a"]] <- LETTERS
H2[["b"]] <- as.list(x = 1:5, y = matrix(rnorm(10), 2))
H2[["c"]] <- head(mtcars, 3)
H2[["d"]] <- Sys.Date()
H2[["e"]] <- Sys.time()
H2[["f"]] <- (function() {
 H3 <- new.env()
 for (i in seq_along(names(H2))) {
 H3[[names(H2)[i]]] <- H2[[names(H2)[i]]]
 }
 H3
})()

ls.str(H2)
a : chr [1:26] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" ...
b : List of 5
$: int 1
$: int 2
$: int 3
```

```
$: int 4
$: int 5
c : 'data.frame': 3 obs. of 11 variables:
$ mpg : num 21 21 22.8
$ cyl : num 6 6 4
$ disp: num 160 160 108
$ hp : num 110 110 93
$ drat: num 3.9 3.9 3.85
$ wt : num 2.62 2.88 2.32
$ qsec: num 16.5 17 18.6
$ vs : num 0 0 1
$ am : num 1 1 1
$ gear: num 4 4 4
$ carb: num 4 4 1
d : Date[1:1], format: "2016-08-03"
e : POSIXct[1:1], format: "2016-08-03 19:25:14"
f : <environment: 0x91a7cb8>

ls.str(H2$f)
a : chr [1:26] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" ...
b : List of 5
$: int 1
$: int 2
$: int 3
$: int 4
$: int 5
c : 'data.frame': 3 obs. of 11 variables:
$ mpg : num 21 21 22.8
$ cyl : num 6 6 4
$ disp: num 160 160 108
$ hp : num 110 110 93
$ drat: num 3.9 3.9 3.85
$ wt : num 2.62 2.88 2.32
$ qsec: num 16.5 17 18.6
$ vs : num 0 0 1
$ am : num 1 1 1
$ gear: num 4 4 4
$ carb: num 4 4 1
d : Date[1:1], format: "2016-08-03"
e : POSIXct[1:1], format: "2016-08-03 19:25:14"
```

## Einschränkungen

Eine der größten Einschränkungen bei der Verwendung von `environment` als Hash-Maps besteht darin, dass im Gegensatz zu vielen Aspekten von R die Vektorisierung für das Suchen / Einfügen von Elementen nicht unterstützt wird:

```
names(H2)
#[1] "a" "b" "c" "d" "e" "f"

H2[[c("a", "b")]]
#Error in H2[[c("a", "b")]] :
wrong arguments for subsetting an environment

Keys <- c("a", "b")
H2[[Keys]]
#Error in H2[[Keys]] : wrong arguments for subsetting an environment
```

Abhängig von der Art der im Objekt gespeicherten Daten kann es möglich sein, `vapply` oder `list2env` für die gleichzeitige Zuweisung vieler Elemente zu verwenden:

```
E1 <- new.env()
invisible({
 vapply(letters, function(x) {
 E1[[x]] <- rnorm(1)
 logical(0)
 }, FUN.VALUE = logical(0))
})

all.equal(sort(names(E1)), letters)
#[1] TRUE

Keys <- letters
E2 <- list2env(
 setNames(
 as.list(rnorm(26)),
 nm = Keys),
 enviro = NULL,
 hash = TRUE
)

all.equal(sort(names(E2)), letters)
#[1] TRUE
```

Keines der obigen Punkte ist besonders knapp, kann jedoch der Verwendung einer `for` Schleife usw. vorzuziehen sein, wenn die Anzahl der Schlüsselwertpaare groß ist.

## Paket: Hash

Das [Hash-Paket](#) bietet eine Hash-Struktur in R. Allerdings ist das [Timing](#) für beide Einfügungen und Lesevorgänge ungünstig mit der Verwendung von Umgebungen als Hash. Diese Dokumentation bestätigt lediglich das Vorhandensein und enthält aus den oben genannten Gründen einen Beispiel-Timing-Code. Es gibt keinen identifizierten Fall, in dem Hash heute eine geeignete Lösung im R-Code ist.

Erwägen:

```
Generic unique string generator
unique_strings <- function(n){
 string_i <- 1
 string_len <- 1
 ans <- character(n)
 chars <- c(letters, LETTERS)
 new_strings <- function(len, pfx){
 for(i in 1:length(chars)){
 if (len == 1){
 ans[string_i] <- paste(pfx, chars[i], sep='')
 string_i <- string_i + 1
 } else {
 new_strings(len-1, pfx=paste(pfx, chars[i], sep=''))
 }
 if (string_i > n) return ()
 }
 }
}
```



```

while(string_i <= n){
 new_strings(string_len,'')
 string_len <- string_len + 1
}
sample(ans)
}

Generate timings using an environment
timingsEnv <- plyr::adply(2^(10:15),.mar=1,.fun=function(i){
 strings <- unique_strings(i)
 ht1 <- new.env(hash=TRUE)
 lapply(strings, function(s){ ht1[[s]] <- 0L})
 data.frame(
 size=c(i,i),
 seconds=c(
 system.time(for (j in 1:i) ht1[[strings[j]]]==0L)[3]),
 type = c('1_hashedEnv')
)
})

timingsHash <- plyr::adply(2^(10:15),.mar=1,.fun=function(i){
 strings <- unique_strings(i)
 ht <- hash::hash()
 lapply(strings, function(s) ht[[s]] <- 0L)
 data.frame(
 size=c(i,i),
 seconds=c(
 system.time(for (j in 1:i) ht[[strings[j]]]==0L)[3]),
 type = c('3_stringHash')
)
})

```

## Paket: listenv

Obwohl `package:listenv` eine `package:listenv` Schnittstelle für Umgebungen implementiert, ist seine Leistung im Vergleich zu Umgebungen für Hash-ähnliche Zwecke **beim Abrufen von Hashes schlecht**. Wenn die Indizes jedoch numerisch sind, kann es beim Abrufen recht schnell gehen. Sie haben jedoch andere Vorteile, z. B. Kompatibilität mit `package:future`. Das Abdecken dieses Pakets zu diesem Zweck geht über den Rahmen des aktuellen Themas hinaus. Der hier angegebene Timing-Code kann jedoch in Verbindung mit dem Beispiel für `package:hash` für Schreib-Timings verwendet werden.

```

timingsListEnv <- plyr::adply(2^(10:15),.mar=1,.fun=function(i){
 strings <- unique_strings(i)
 le <- listenv::listenv()
 lapply(strings, function(s) le[[s]] <- 0L)
 data.frame(
 size=c(i,i),
 seconds=c(
 system.time(for (k in 1:i) le[[k]]==0L)[3]),
 type = c('2_numericListEnv')
)
})

```

Hashmaps online lesen: <https://riptutorial.com/de/r/topic/5179/hashmaps>

# Kapitel 56: Heatmap und Heatmap.2

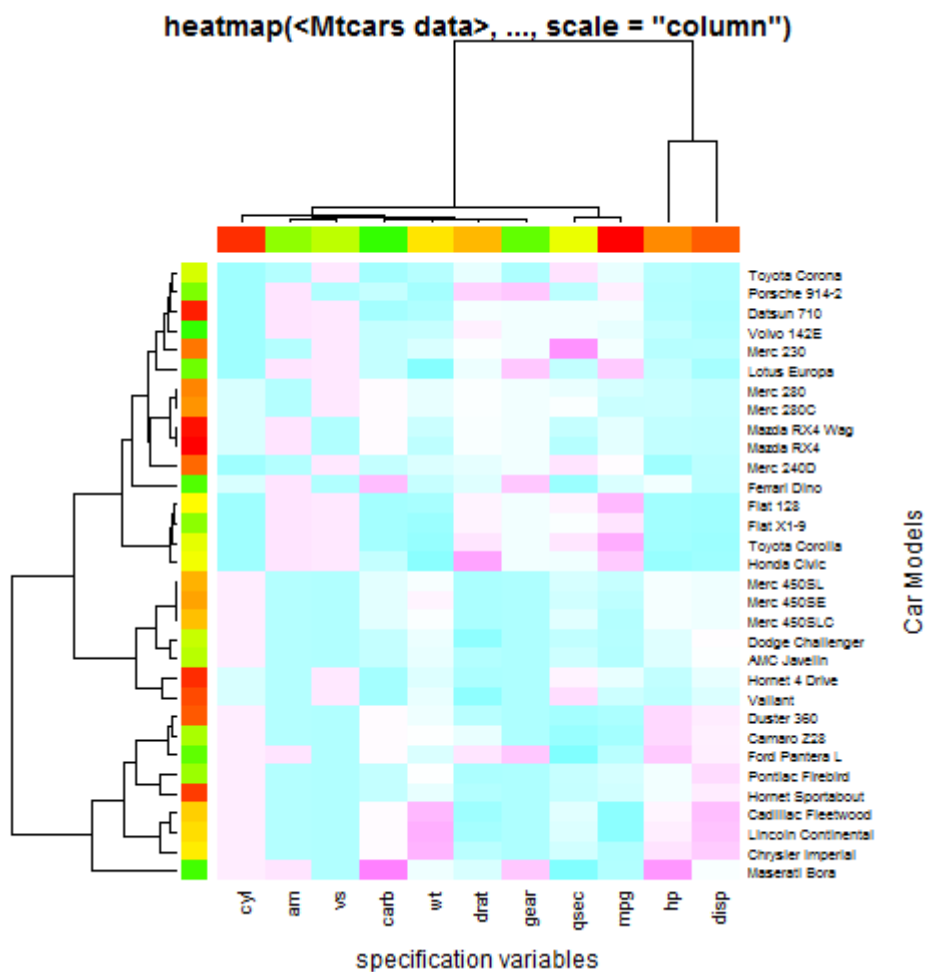
## Examples

Beispiele aus der offiziellen Dokumentation

## stats :: heatmap

### Beispiel 1 (Grundnutzung)

```
require(graphics); require(grDevices)
x <- as.matrix(mtcars)
rc <- rainbow(nrow(x), start = 0, end = .3)
cc <- rainbow(ncol(x), start = 0, end = .3)
hv <- heatmap(x, col = cm.colors(256), scale = "column",
 RowSideColors = rc, ColSideColors = cc, margins = c(5,10),
 xlab = "specification variables", ylab = "Car Models",
 main = "heatmap(<Mtcars data>, ..., scale = \"column\")")
```

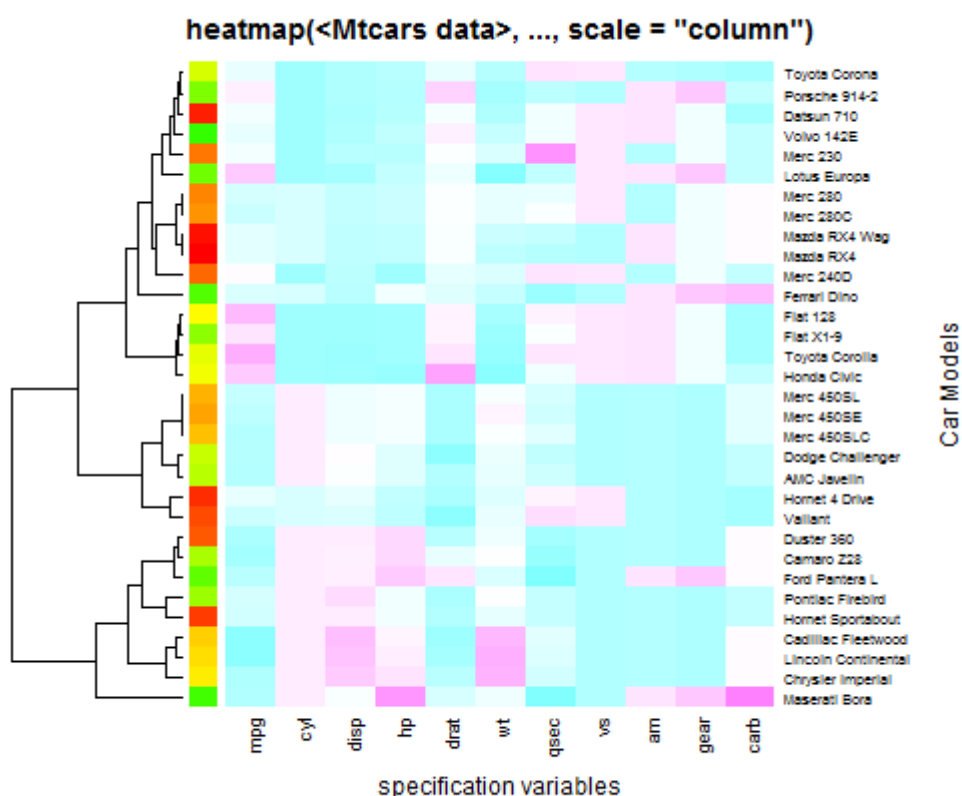


```
utils::str(hv) # the two re-ordering index vectors
List of 4
```

```
$ rowInd: int [1:32] 31 17 16 15 5 25 29 24 7 6 ...
$ colInd: int [1:11] 2 9 8 11 6 5 10 7 1 4 ...
$ Rowv : NULL
$ Colv : NULL
```

## Beispiel 2 (kein Spalten-Dendrogramm (noch Umordnen))

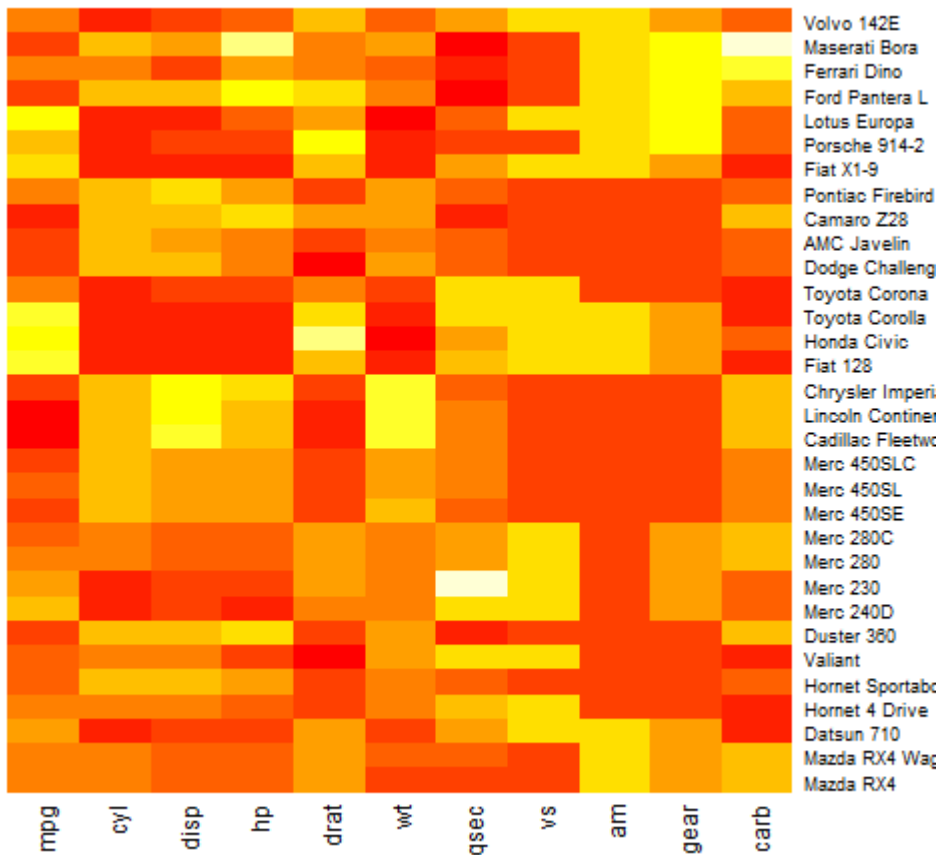
```
heatmap(x, Colv = NA, col = cm.colors(256), scale = "column",
 RowSideColors = rc, margins = c(5,10),
 xlab = "specification variables", ylab = "Car Models",
 main = "heatmap(<Mtcars data>, ..., scale = \"column\")")
```



## Beispiel 3 ("Nichts")

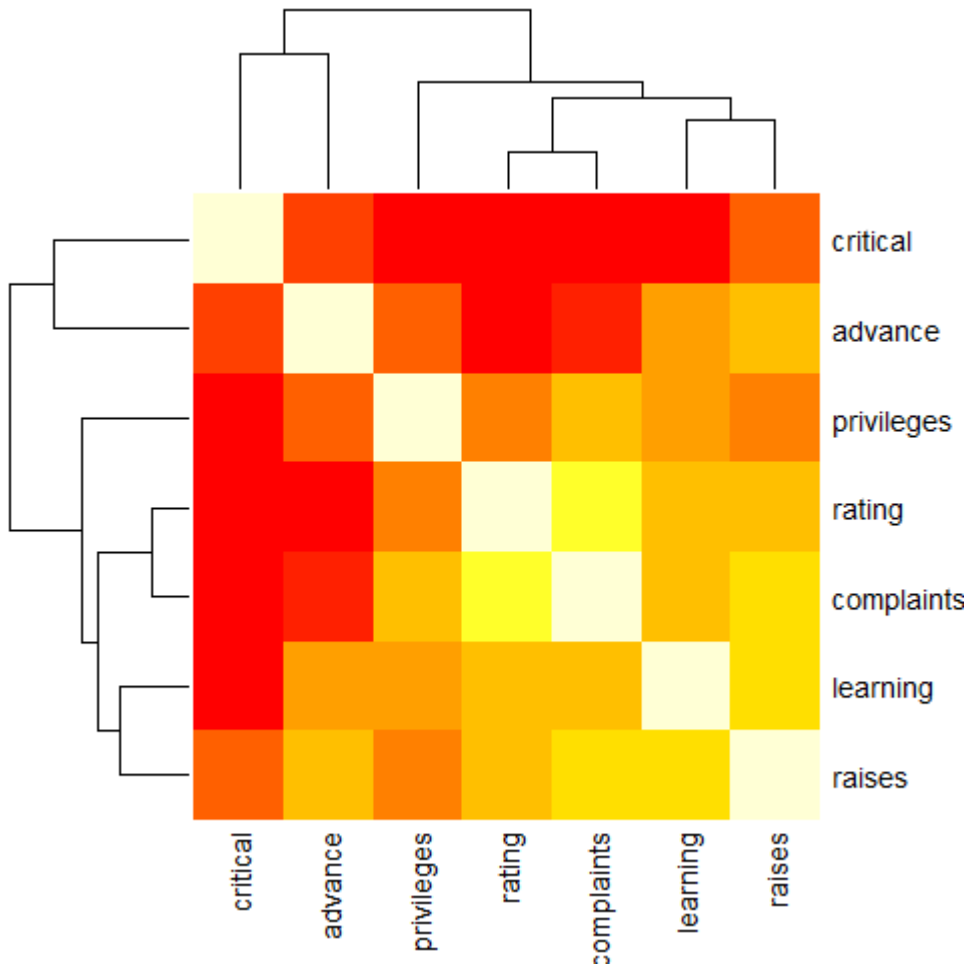
```
heatmap(x, Rowv = NA, Colv = NA, scale = "column",
 main = "heatmap(*, NA, NA) ~ image(t(x))")
```

heatmap(\*, NA, NA) ~= image(t(x))



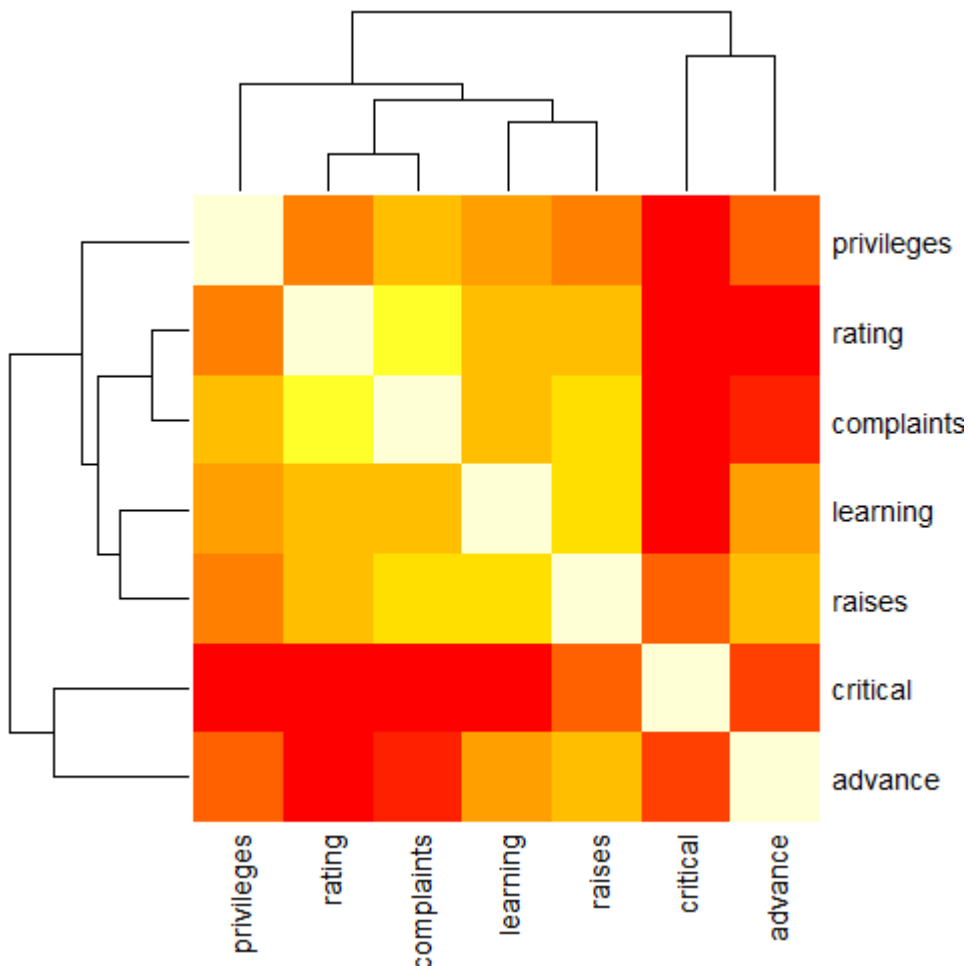
## Beispiel 4 (mit reorder ())

```
round(Ca <- cor(attendance), 2)
rating complaints privileges learning raises critical advance
rating 1.00 0.83 0.43 0.62 0.59 0.16 0.16
complaints 0.83 1.00 0.56 0.60 0.67 0.19 0.22
privileges 0.43 0.56 1.00 0.49 0.45 0.15 0.34
learning 0.62 0.60 0.49 1.00 0.64 0.12 0.53
raises 0.59 0.67 0.45 0.64 1.00 0.38 0.57
critical 0.16 0.19 0.15 0.12 0.38 1.00 0.28
advance 0.16 0.22 0.34 0.53 0.57 0.28 1.00
symnum(Ca) # simple graphic
rt cm p l rs cr a
rating 1
complaints + 1
privileges . . 1
learning , . . 1
raises . , . , 1
critical . . . 1
advance 1
attr(,"legend")
[1] 0 \ ' 0.3 \ ' 0.6 \ ' 0.8 \+ 0.9 * 0.95 \B 1
heatmap(Ca,
 symm = TRUE, margins = c(6,6))
```



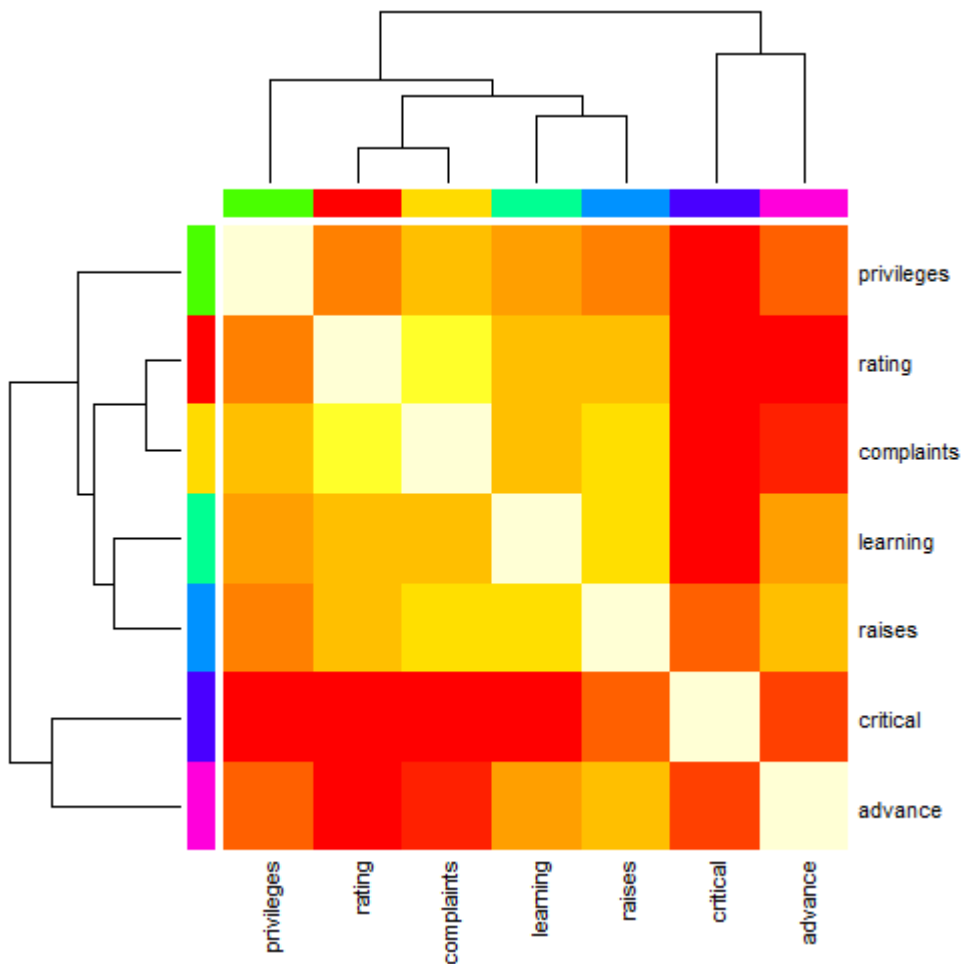
## Beispiel 5 ( *keine* Nachbestellung ())

```
heatmap(Ca, Rowv = FALSE, symm = TRUE, margins = c(6,6))
```



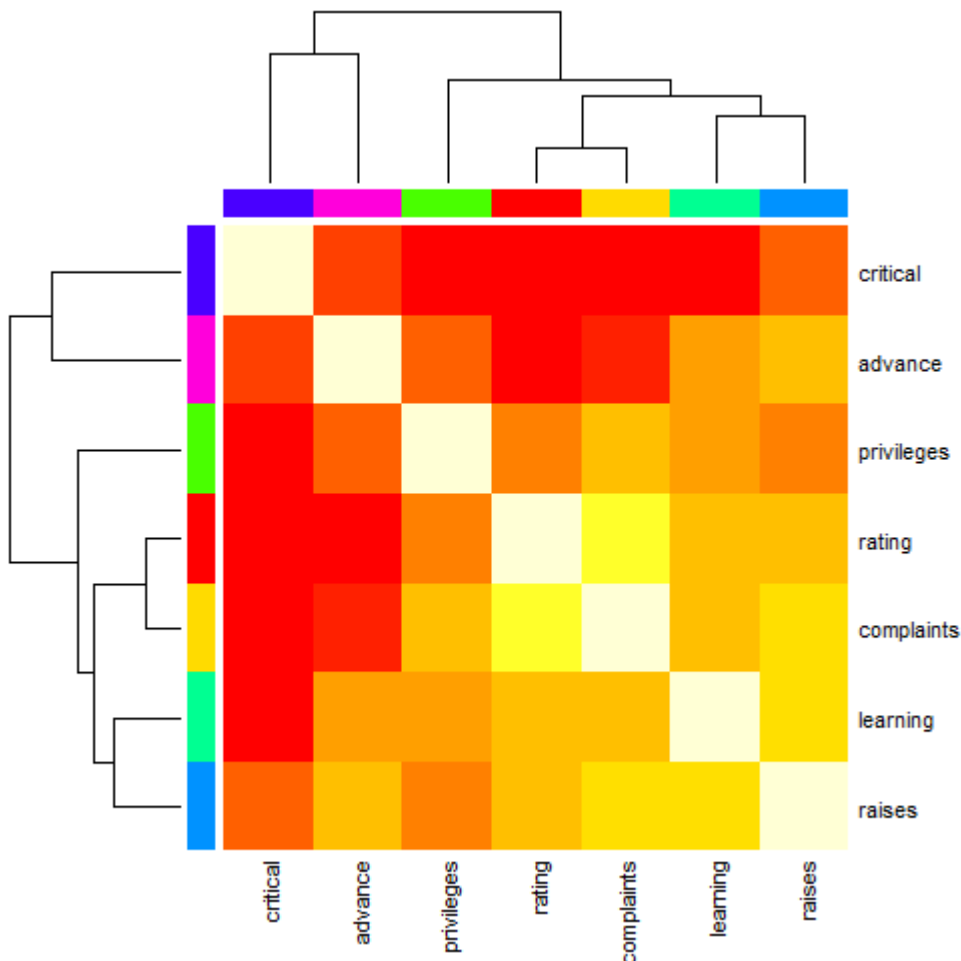
## Beispiel 6 (leicht künstlich mit Farbbalken, ohne zu bestellen)

```
cc <- rainbow(nrow(Ca))
heatmap(Ca, Rowv = FALSE, symm = TRUE, RowSideColors = cc, ColSideColors = cc,
 margins = c(6,6))
```



## Beispiel 7 (leicht künstlich mit Farbbalken, mit Bestellung)

```
heatmap(Ca, symm = TRUE, RowSideColors = cc, ColSideColors = cc,
 margins = c(6,6))
```



## Beispiel 8 (Verwenden Sie für variable Cluster eher den Abstand basierend auf cor ())

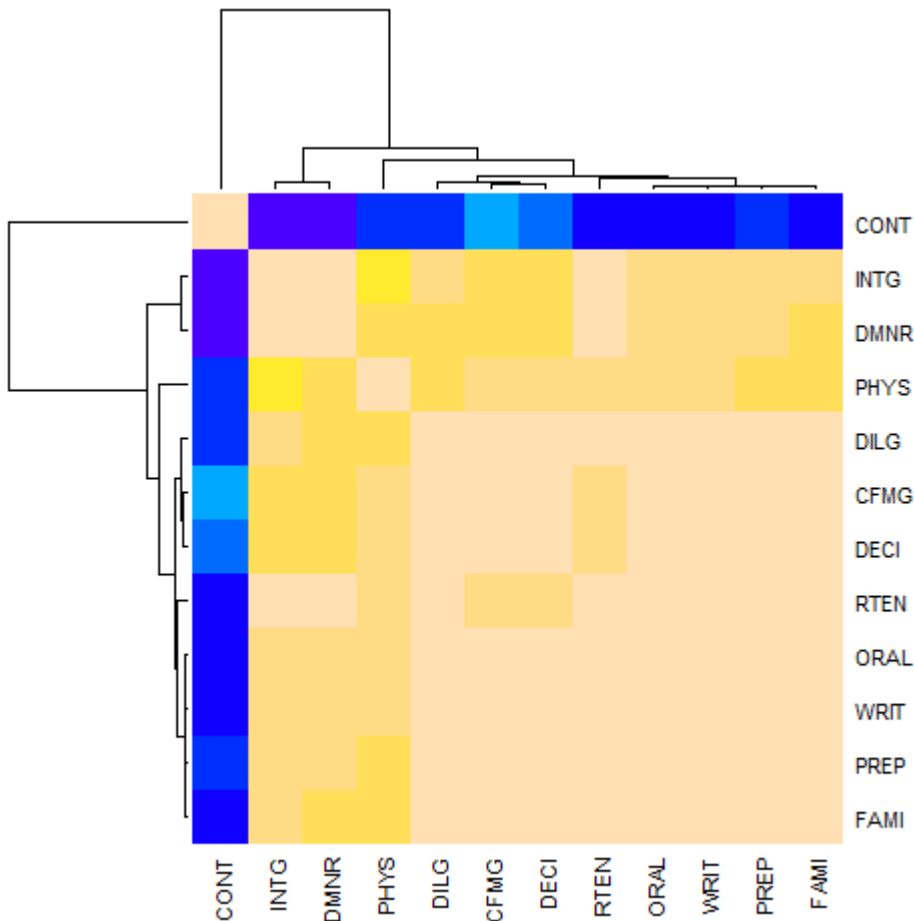
```

symnum(cU <- cor(USJudgeRatings))
CO I DM DI CF DE PR F O W PH R
CONT 1
INTG 1
DMNR B 1
DILG + + 1
CFMG + + B 1
DECI + + B B 1
PREP + + B B B 1
FAMI + + B * * B 1
ORAL * * B B * B B 1
WRIT * + B * * B B B 1
PHYS , , + + + + + + 1
RTEN * * * * * B * B B * 1
attr(,"legend")
[1] 0 ' ' 0.3 ' .' 0.6 ' ,' 0.8 '+ ' 0.9 '* ' 0.95 'B' 1

hU <- heatmap(cU, Rowv = FALSE, symm = TRUE, col = topo.colors(16),
 distfun = function(c) as.dist(1 - c), keep.dendro = TRUE)

```





```
The Correlation matrix with same reordering:
```

```
round(100 * cU[hU[[1]], hU[[2]]])
```

```
CONT INTG DMNR PHYS DILG CFMG DECI RTEN ORAL WRIT PREP FAMI
CONT 100 -13 -15 5 1 14 9 -3 -1 -4 1 -3
INTG -13 100 96 74 87 81 80 94 91 91 88 87
DMNR -15 96 100 79 84 81 80 94 91 89 86 84
PHYS 5 74 79 100 81 88 87 91 89 86 85 84
DILG 1 87 84 81 100 96 96 93 95 96 98 96
CFMG 14 81 81 88 96 100 98 93 95 94 96 94
DECI 9 80 80 87 96 98 100 92 95 95 96 94
RTEN -3 94 94 91 93 93 92 100 98 97 95 94
ORAL -1 91 91 89 95 95 95 98 100 99 98 98
WRIT -4 91 89 86 96 94 95 97 99 100 99 99
PREP 1 88 86 85 98 96 96 95 98 99 100 99
FAMI -3 87 84 84 96 94 94 94 98 99 99 100
```

```
The column dendrogram:
```

```
utils::str(hU$Colv)
```

```
--[dendrogram w/ 2 branches and 12 members at h = 1.15]
```

```
|--leaf "CONT"
```

```
`--[dendrogram w/ 2 branches and 11 members at h = 0.258]
```

```
|--[dendrogram w/ 2 branches and 2 members at h = 0.0354]
```

```
| |--leaf "INTG"
```

```
| `--leaf "DMNR"
```

```
`--[dendrogram w/ 2 branches and 9 members at h = 0.187]
```

```
|--leaf "PHYS"
```

```
`--[dendrogram w/ 2 branches and 8 members at h = 0.075]
```

```
|--[dendrogram w/ 2 branches and 3 members at h = 0.0438]
```

```
| |--leaf "DILG"
```

```
| `--[dendrogram w/ 2 branches and 2 members at h = 0.0189]
| |--leaf "CFMG"
| `--leaf "DECI"
`--[dendrogram w/ 2 branches and 5 members at h = 0.0584]
|--leaf "RTEN"
`--[dendrogram w/ 2 branches and 4 members at h = 0.0187]
|--[dendrogram w/ 2 branches and 2 members at h = 0.00657]
| |--leaf "ORAL"
| `--leaf "WRIT"
`--[dendrogram w/ 2 branches and 2 members at h = 0.0101]
|--leaf "PREP"
`--leaf "FAMI"
```

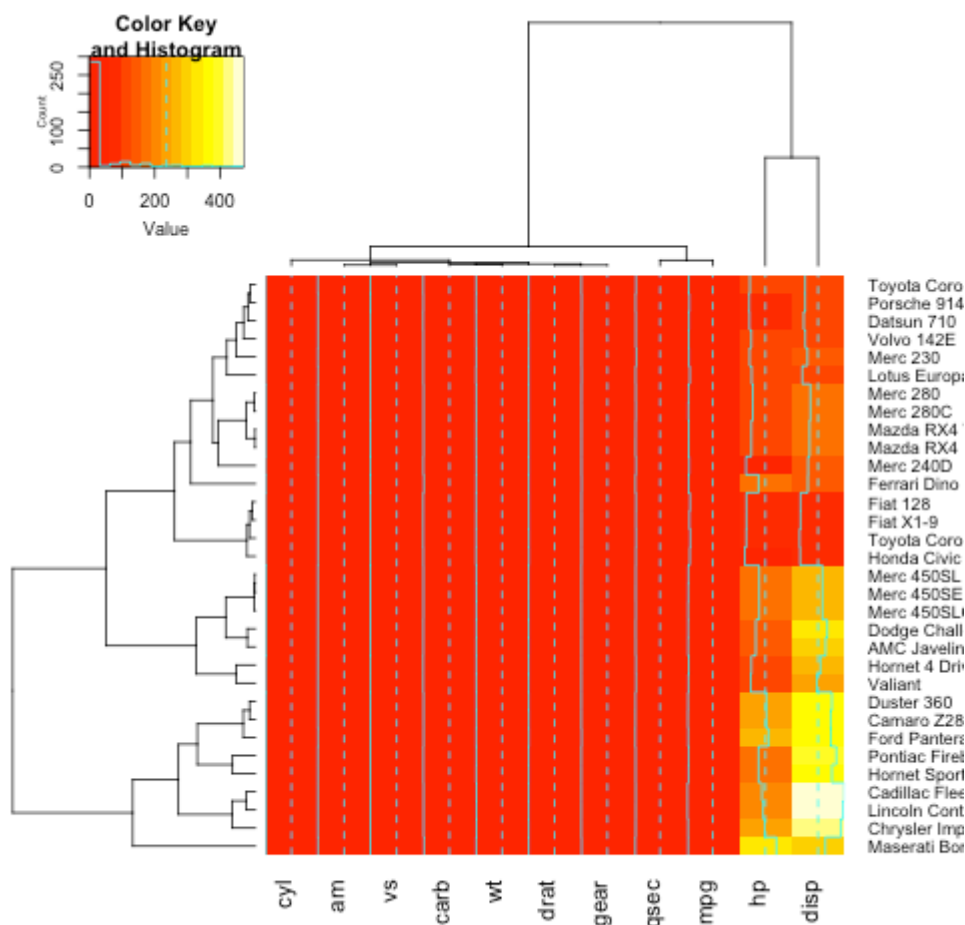
## Optimierungsparameter in heatmap.2

Gegeben:

```
x <- as.matrix(mtcars)
```

Sie können `heatmap.2` - eine neuere optimierte Version von `heatmap` , indem Sie die folgende Bibliothek laden:

```
require(gplots)
heatmap.2(x)
```



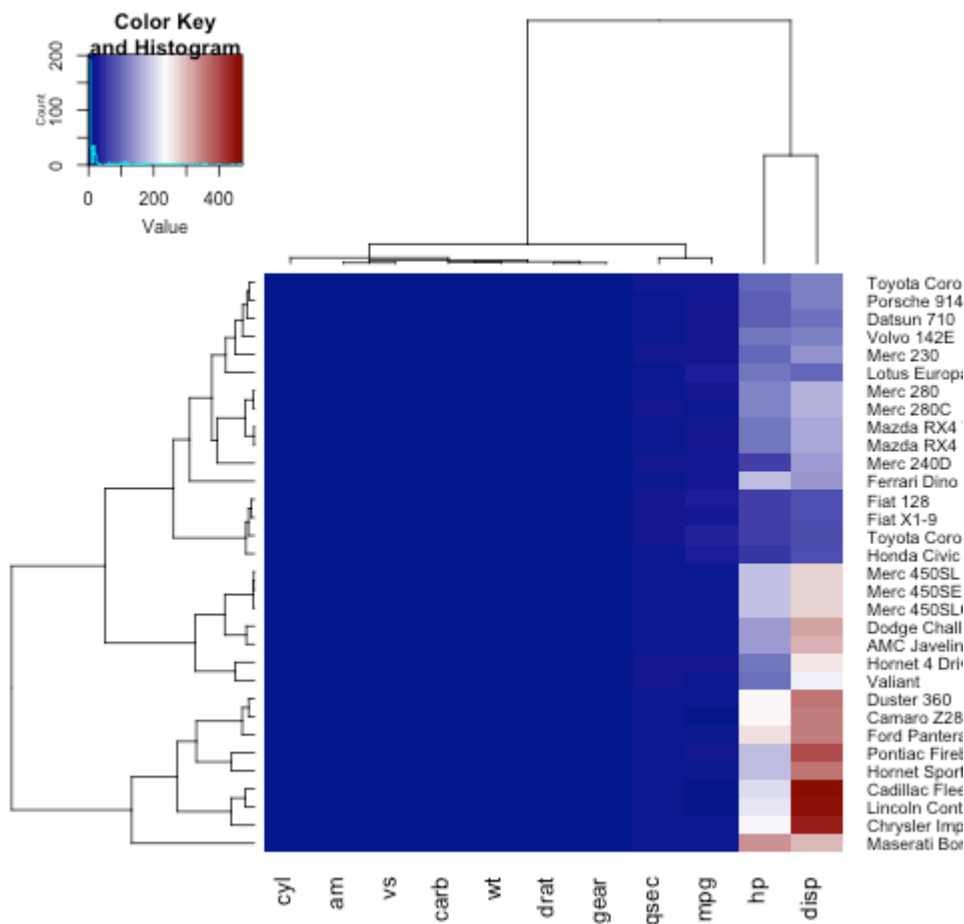
Um Ihrer Heatmap einen Titel, ein X- oder ein Y-Label hinzuzufügen, müssen Sie `main` , `xlab` und

ylab :

```
heatmap.2(x, main = "My main title: Overview of car features", xlab="Car features", ylab = "Car brands")
```

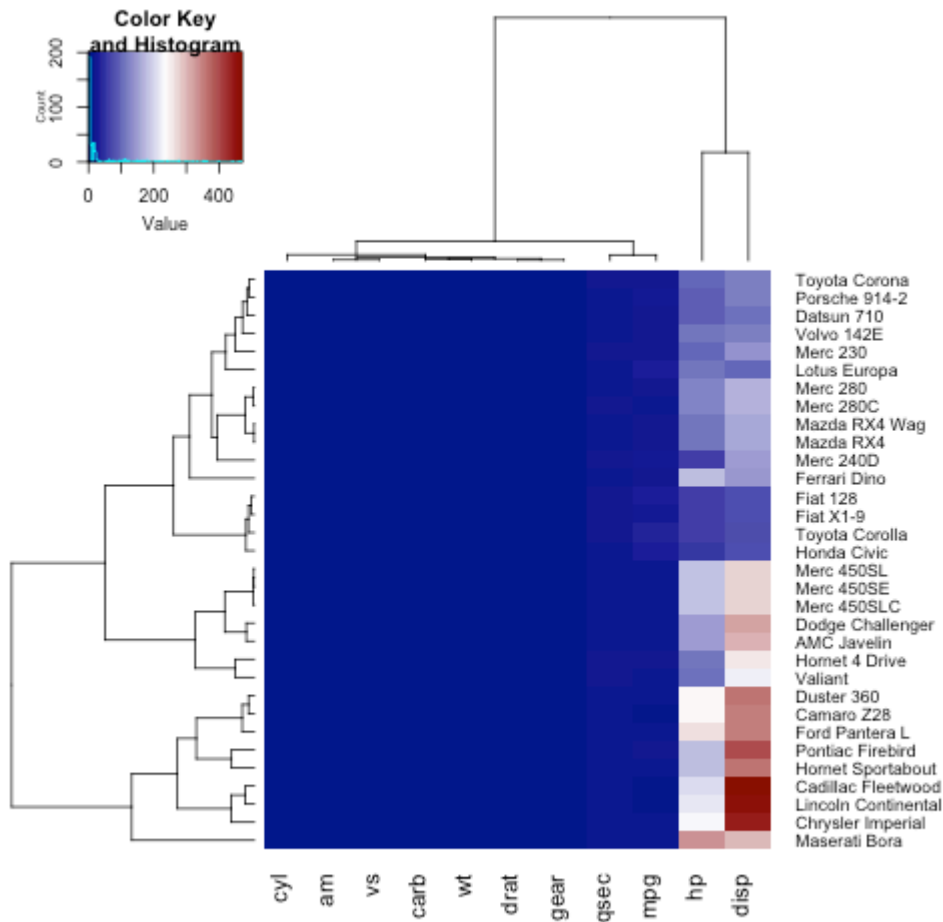
Wenn Sie eine eigene Farbpalette für Ihre Heatmap definieren möchten, können Sie den Parameter `col` mit der Funktion `colorRampPalette` :

```
heatmap.2(x, trace="none", key=TRUE, Colv=FALSE, dendrogram = "row", col = colorRampPalette(c("darkblue", "white", "darkred"))(100))
```

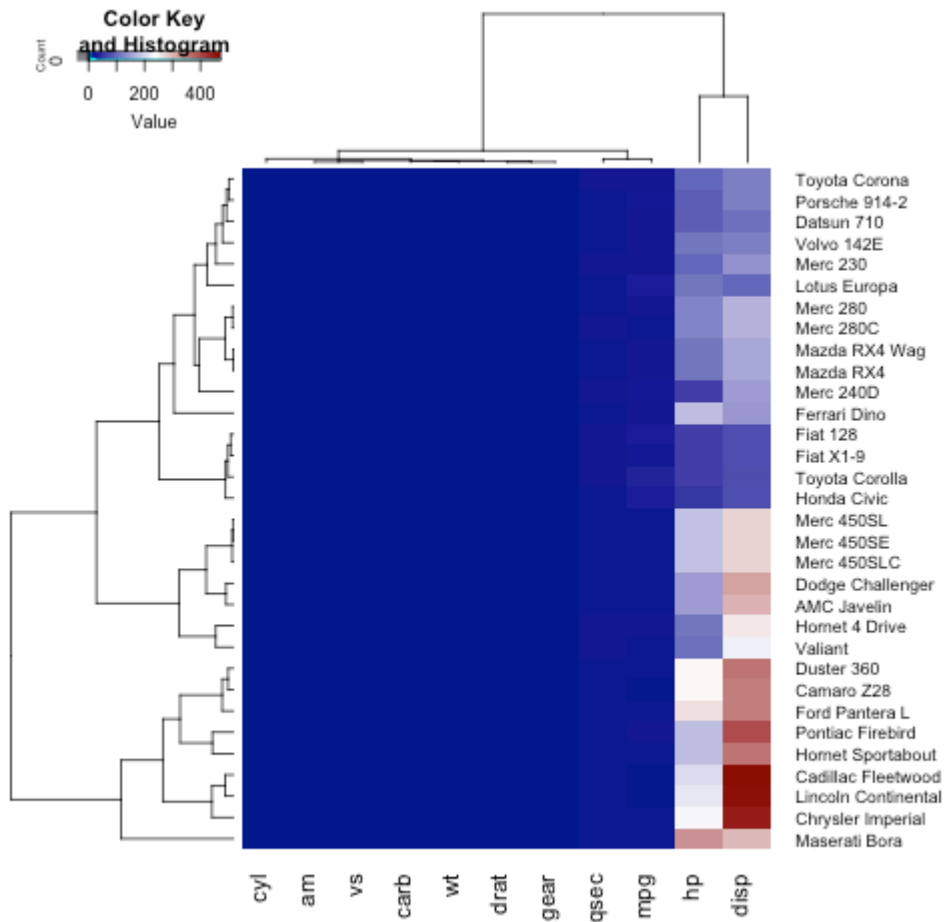


Wie Sie feststellen können, passen die Beschriftungen auf der y-Achse (die Fahrzeugnamen) nicht in die Abbildung. Um dies zu beheben, kann der Benutzer den Parameter `margins` anpassen:

```
heatmap.2(x, trace="none", key=TRUE, col = colorRampPalette(c("darkblue", "white", "darkred"))(100), margins=c(5,8))
```

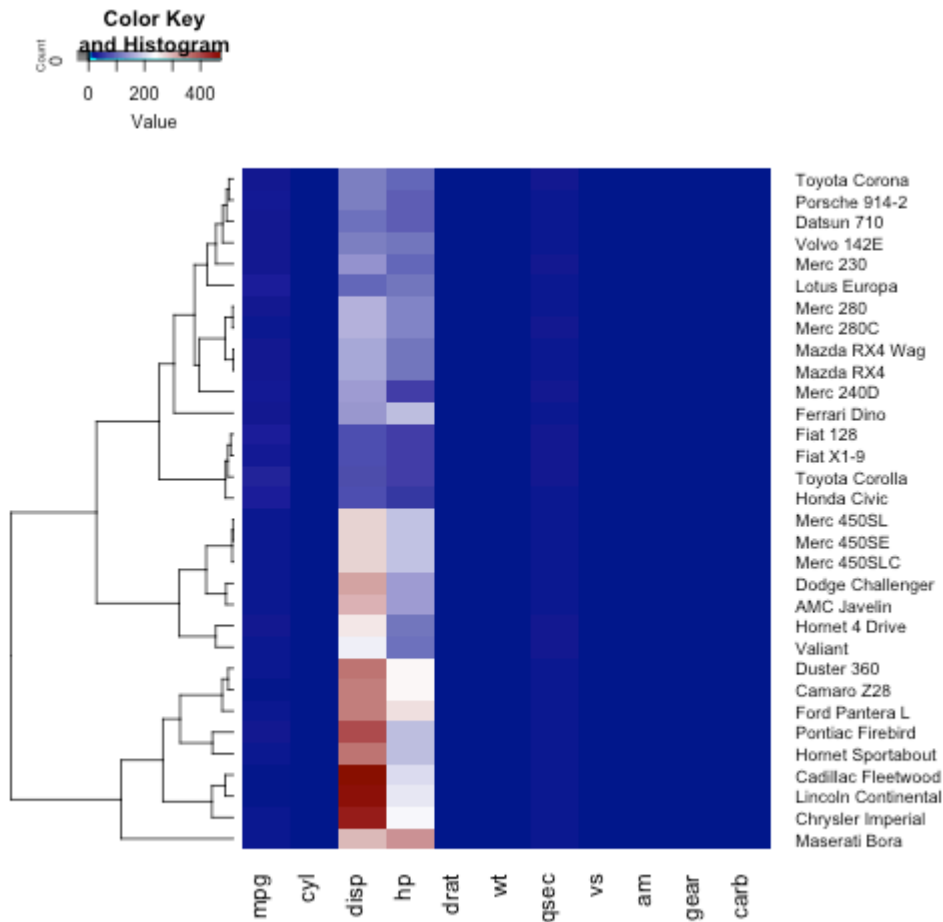


Außerdem können wir die Abmessungen jedes Abschnitts unserer Heatmap (das Schlüssel-Histogramm, die Dendogramme und die Heatmap selbst) `lhei` , indem Sie `lhei` und `lwid` :



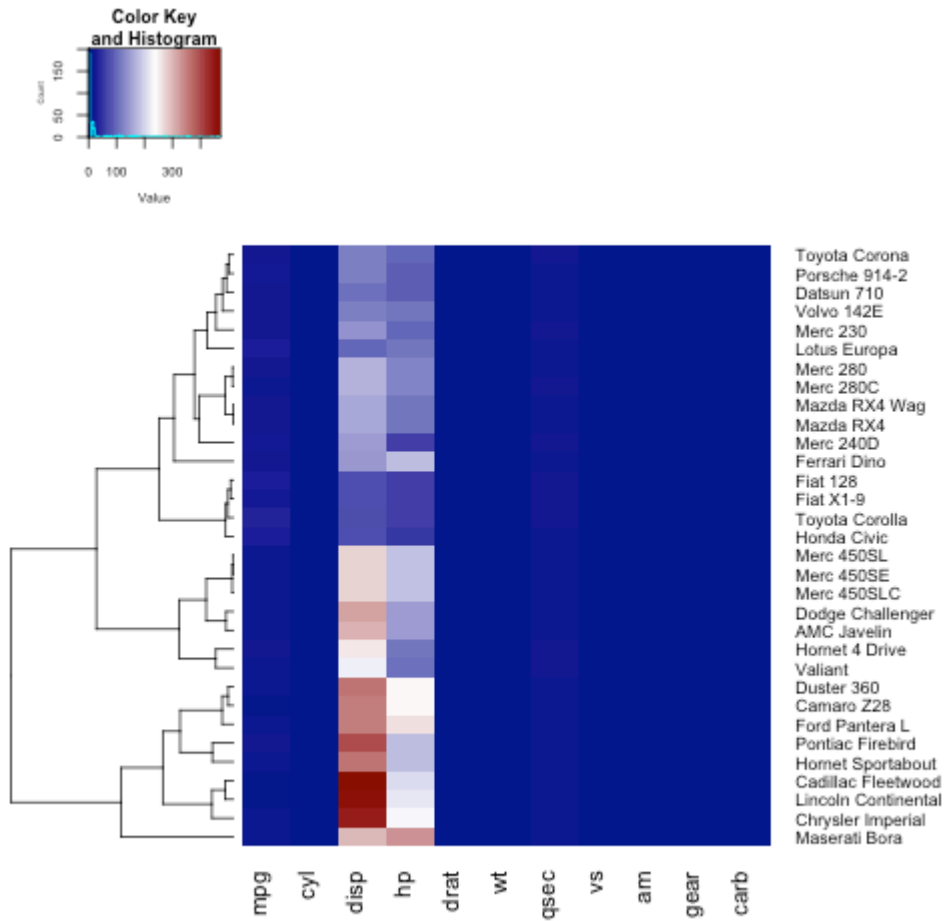
Wenn Sie nur ein Zeilen- (oder Spalten-) Dendrogramm anzeigen möchten, müssen Sie `Colv=FALSE` (oder `Rowv=FALSE`) setzen und den `dendrogram` Parameter anpassen:

```
heatmap.2(x, trace="none", key=TRUE, Colv=FALSE, dendrogram = "row", col =
colorRampPalette(c("darkblue","white","darkred"))(100), margins=c(5,8), lwid = c(5,15), lhei =
c(3,15))
```



Um die Schriftgröße des Legendentitels, der Beschriftungen und der Achse zu `cex.main`, `cex.lab`, `cex.axis` der Benutzer `cex.main`, `cex.lab`, `cex.axis` in der `par` Liste `cex.main`, `cex.lab`, `cex.axis`:

```
par(cex.main=1, cex.lab=0.7, cex.axis=0.7)
heatmap.2(x, trace="none", key=TRUE, Colv=FALSE, dendrogram = "row", col =
colorRampPalette(c("darkblue","white","darkred"))(100), margins=c(5,8), lwid = c(5,15), lhei =
c(5,15))
```



Heatmap und Heatmap.2 online lesen: <https://riptutorial.com/de/r/topic/4814/heatmap-und-heatmap-2>

# Kapitel 57: Hierarchische lineare Modellierung

## Examples

### Grundmodellanpassung

**Entschuldigung** : Da ich keinen Kanal für die Diskussion / Rückmeldung von Verbesserungsvorschlägen kenne, werde ich meine Frage hier stellen. Bitte zögern Sie nicht, einen besseren Ort dafür anzugeben! @DataTx gibt an, dass dies "völlig unklar, unvollständig ist oder schwerwiegende Formatierungsprobleme hat". Da ich keine großen Formatierungsprobleme (-:-) sehe, wäre ein wenig mehr Orientierung, was hier zur Verbesserung der Klarheit oder Vollständigkeit erwartet wird und warum das, was hier ist, nicht zu retten ist.

Die primären Pakete für das Anpassen von hierarchischen (alternativ "gemischten" oder "mehrstufigen") linearen Modellen in R sind `nlme` (älter) und `lme4` (neuer). Diese Pakete unterscheiden sich in vielerlei Hinsicht, sollten aber generell zu sehr ähnlichen, angepassten Modellen führen.

```
library(nlme)
library(lme4)
m1.nlme <- lme(Reaction~Days, random=~Days|Subject, data=sleepstudy, method="REML")
m1.lme4 <- lmer(Reaction~Days+(Days|Subject), data=sleepstudy, REML=TRUE)
all.equal(fixef(m1.nlme), fixef(m1.lme4))
[1] TRUE
```

Unterschiede zu berücksichtigen:

- Formel-Syntax ist etwas anders
- `nlme` ist (noch) etwas besser dokumentiert (z. B. Pinheiro und Bates 2000 *Mixed-Effekt-Modelle in S-PLUS*, siehe jedoch Bates et al. 2015 *Journal of Statistical Software* / `vignette("lmer", package="lme4")` für `lme4`)
- `lme4` ist schneller und ermöglicht eine einfachere Anpassung gekreuzter zufälliger Effekte
- `nlme` liefert p-Werte für lineare gemischte Modelle, `lme4` erfordert Zusatzpakete wie `lmerTest` oder `afex`
- `nlme` ermöglicht die Modellierung von Heteroskedastizität oder `nlme` (in Raum / Zeit / Phylogenie)

Die inoffizielle [GLMM-FAQ](#) enthält weitere Informationen, obwohl sie sich auf *generalisierte* lineare gemischte Modelle (GLMMs) konzentriert.

Hierarchische lineare Modellierung online lesen:

<https://riptutorial.com/de/r/topic/3460/hierarchische-lineare-modellierung>



---

# Kapitel 58: Hierarchisches Clustering mit hclust

## Einführung

Das Paket `stats` bietet die Funktion `hclust`, um hierarchisches Clustering durchzuführen.

## Bemerkungen

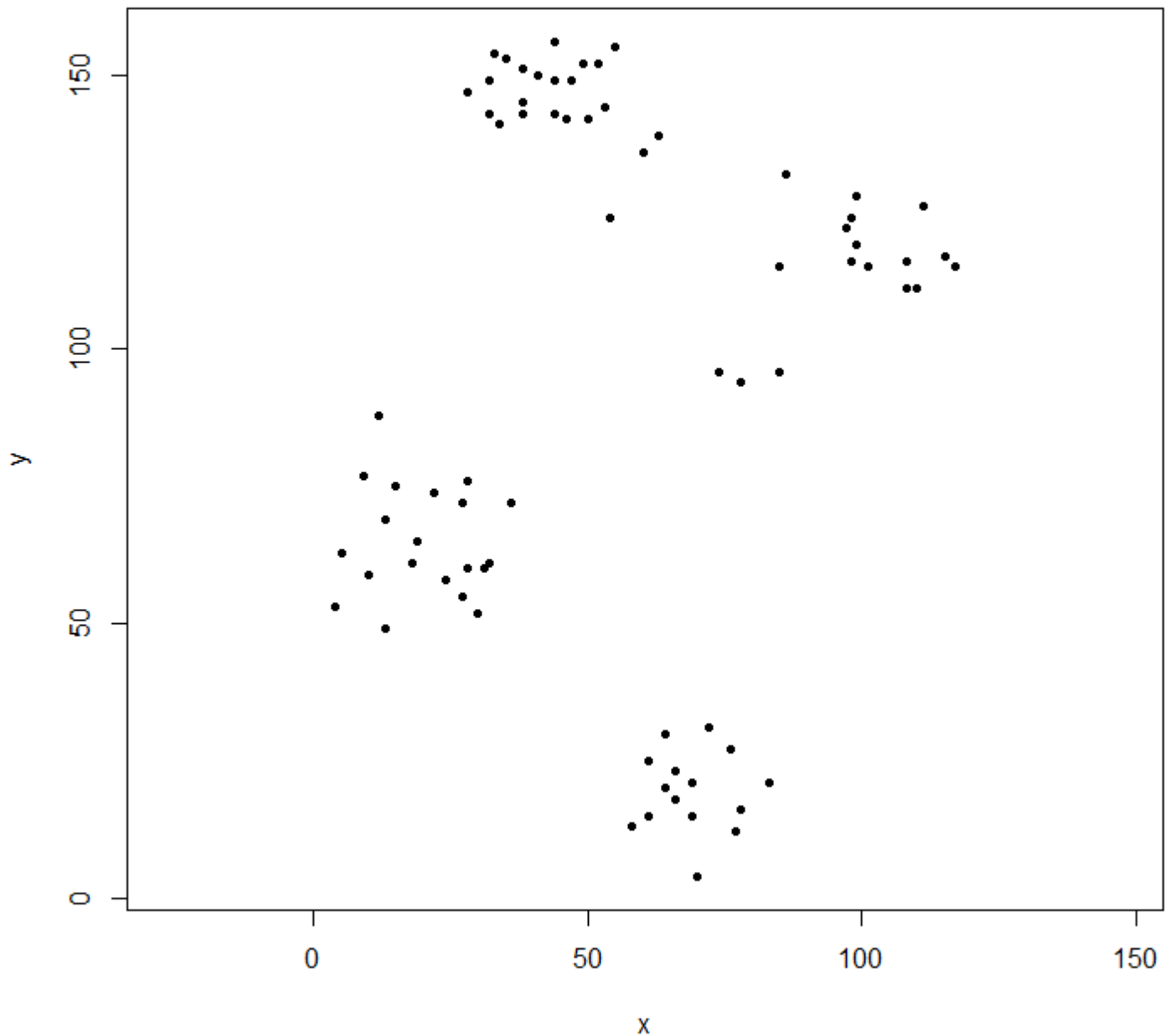
Neben `hclust` stehen auch andere Methoden zur Verfügung, siehe [CRAN- Paketansicht zum Clustering](#).

## Examples

### Beispiel 1 - Grundlegende Verwendung von hclust, Anzeige des Dendrogramms, Plotcluster

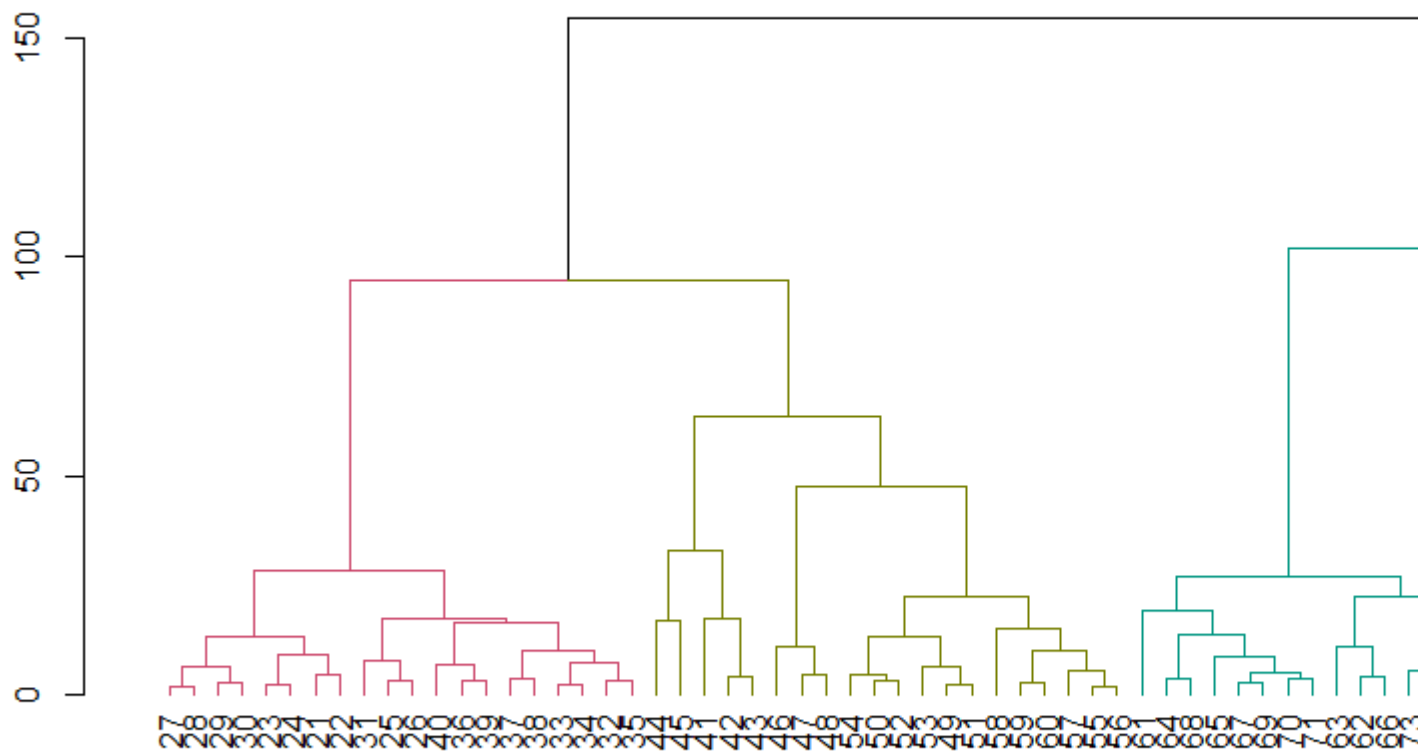
Die Clusterbibliothek enthält die Ruspini-Daten - einen Standarddatensatz zur Veranschaulichung der Clusteranalyse.

```
library(cluster) ## to get the ruspini data
plot(ruspini, asp=1, pch=20) ## take a look at the data
```



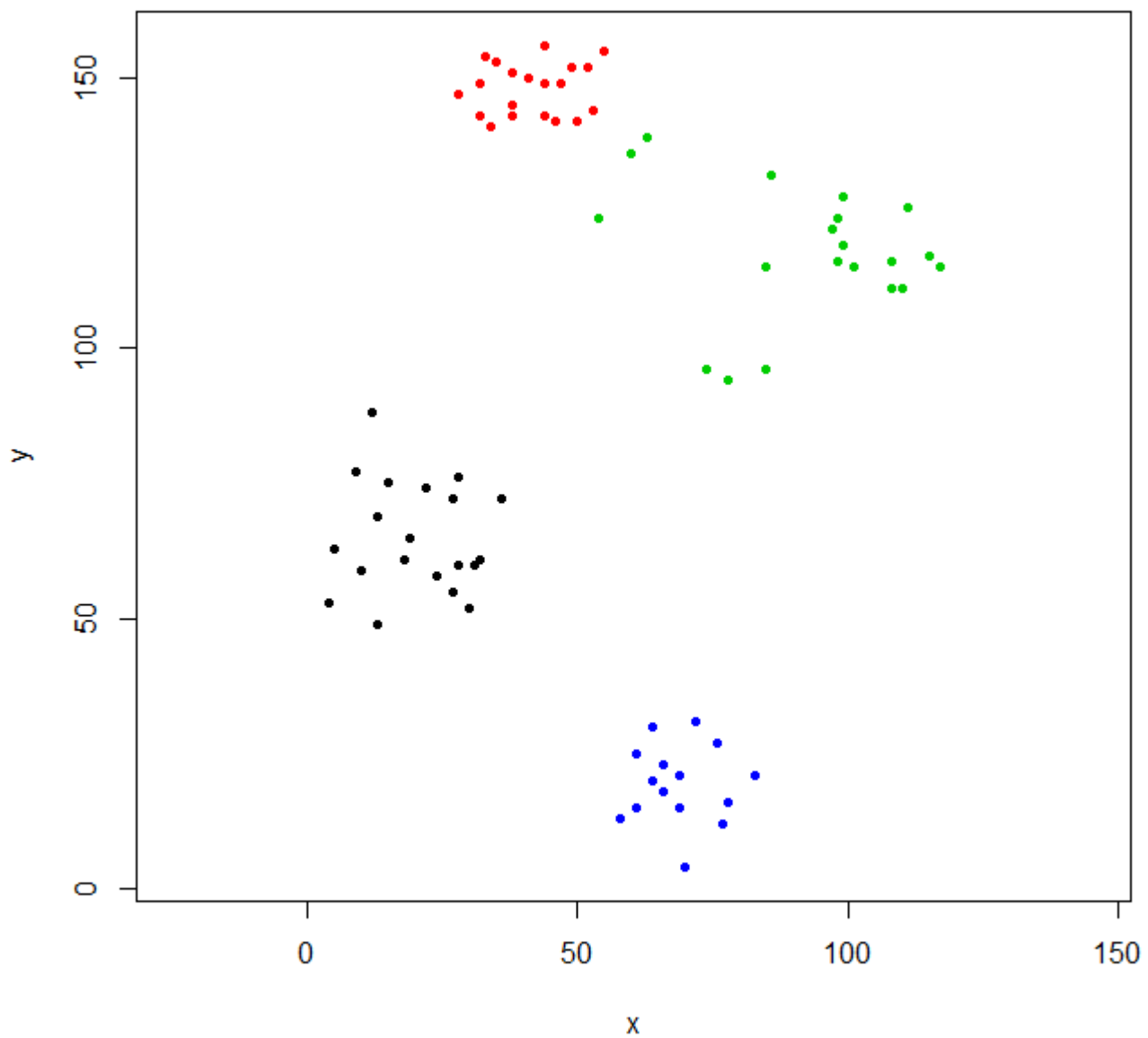
hclust erwartet eine Entfernungsmatrix, nicht die Originaldaten. Wir berechnen den Baum anhand der Standardparameter und zeigen ihn an. Mit dem Hang-Parameter werden alle Blätter des Baums entlang der Grundlinie ausgerichtet.

```
ruspini_hc_defaults <- hclust(dist(ruspini))
dend <- as.dendrogram(ruspini_hc_defaults)
if(!require(dendextend)) install.packages("dendextend"); library(dendextend)
dend <- color_branches(dend, k = 4)
plot(dend)
```



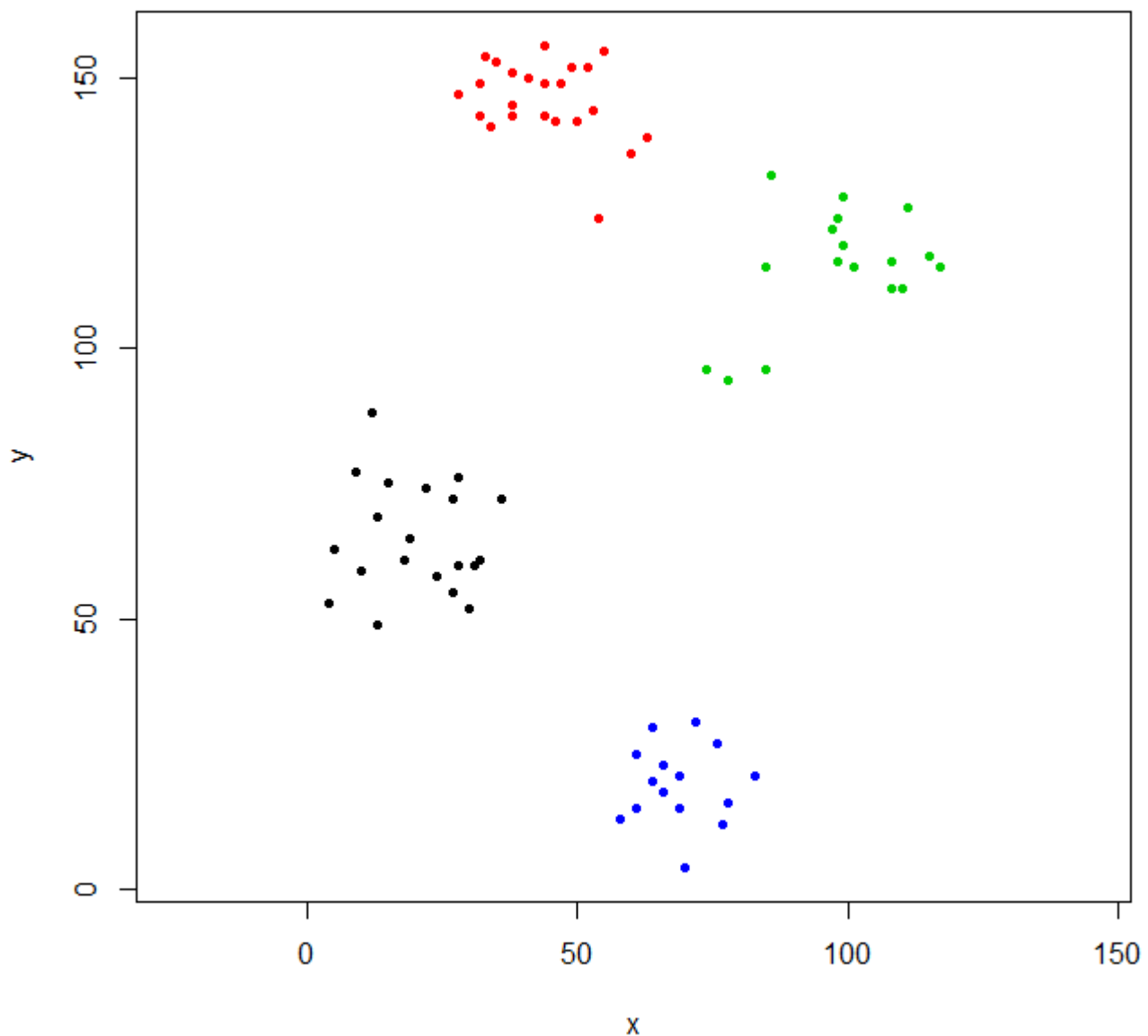
Schneiden Sie die Baumstruktur aus, um vier Cluster zu erhalten, und ordnen Sie die Daten neu zu, indem Sie die Punkte nach Cluster färben.  $k$  ist die gewünschte Anzahl von Clustern.

```
rhc_def_4 = cutree(ruspini_hc_defaults,k=4)
plot(ruspini, pch=20, asp=1, col=rhc_def_4)
```



Dieses Clustering ist etwas seltsam. Wir können ein besseres Clustering erzielen, indem wir zuerst die Daten skalieren.

```
scaled_ruspini_hc_defaults = hclust(dist(scale(ruspini)))
srhc_def_4 = cutree(scaled_ruspini_hc_defaults,4)
plot(ruspini, pch=20, asp=1, col=srhc_def_4)
```



Das Standard-Unähnlichkeitsmaß für den Vergleich von Clustern ist "complete". Sie können mit dem Methodenparameter eine andere Kennzahl angeben.

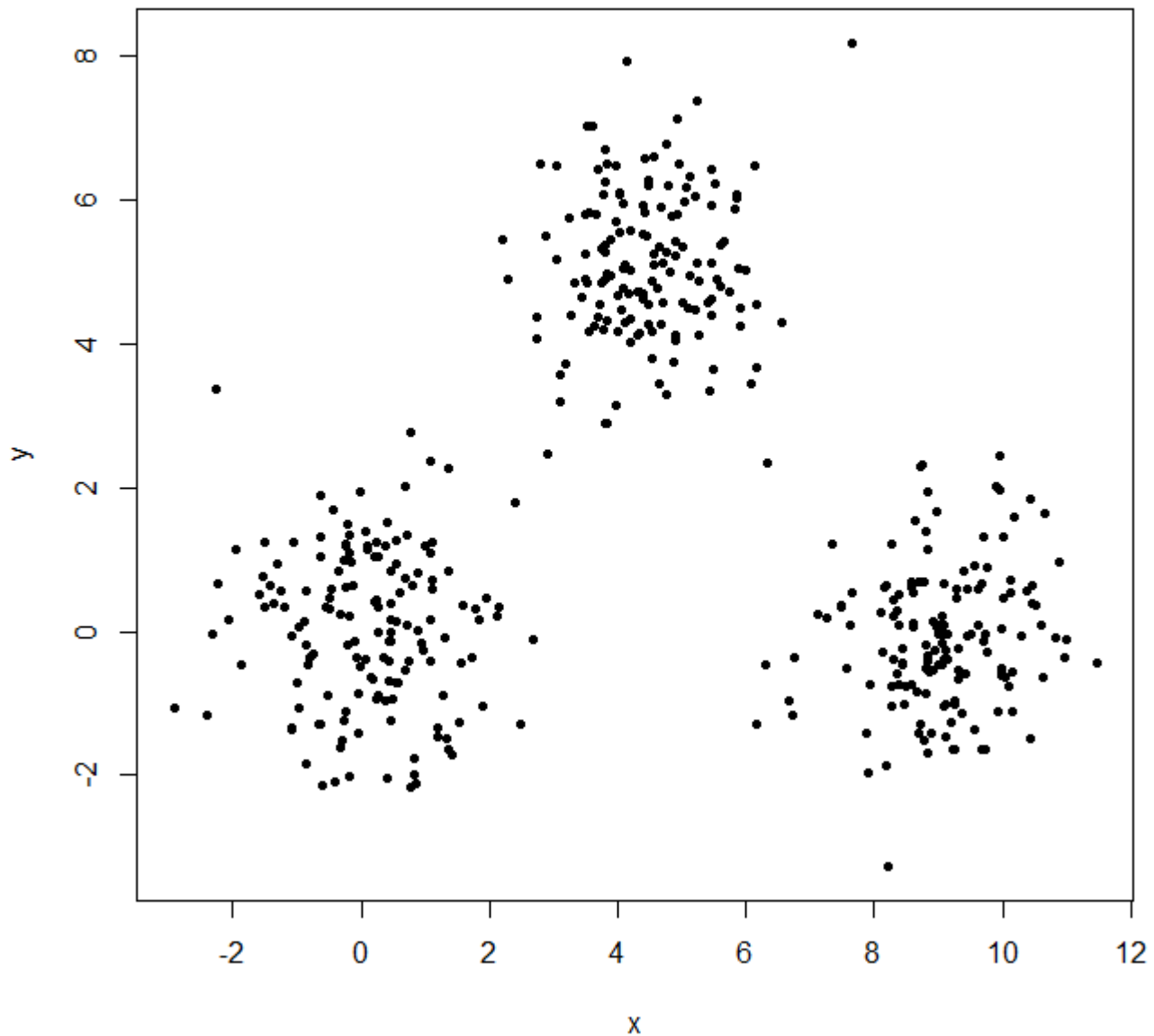
```
ruspini_hc_single = hclust(dist(ruspini), method="single")
```

## Beispiel 2 - hclust und Ausreißer

Beim hierarchischen Clustering werden Ausreißer häufig als Einpunkt-Cluster angezeigt.

Generieren Sie drei Gaußsche Verteilungen, um die Auswirkungen von Ausreißern zu veranschaulichen.

```
set.seed(656)
x = c(rnorm(150, 0, 1), rnorm(150, 9, 1), rnorm(150, 4.5, 1))
y = c(rnorm(150, 0, 1), rnorm(150, 0, 1), rnorm(150, 5, 1))
XYdf = data.frame(x, y)
plot(XYdf, pch=20)
```



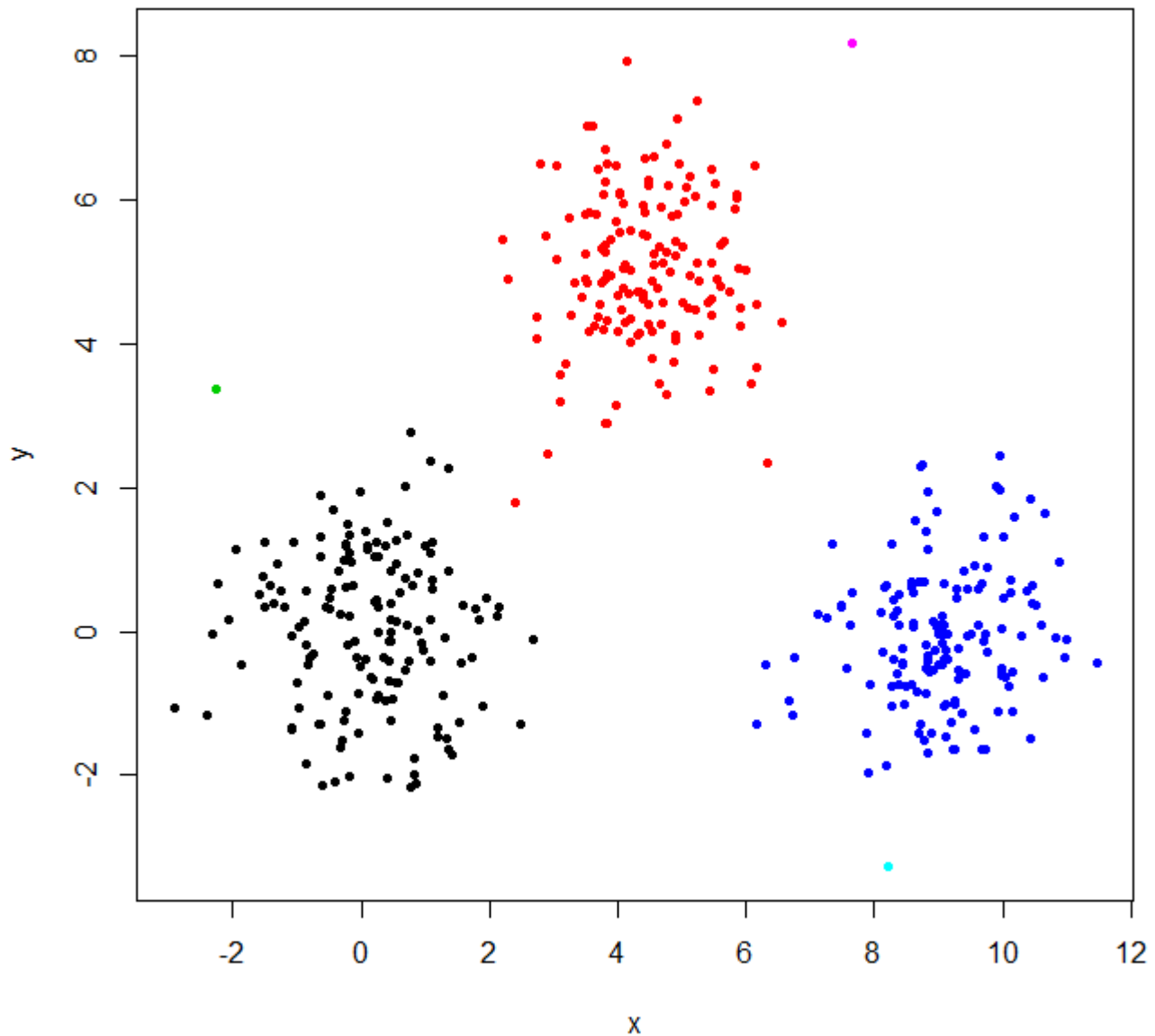
Erstellen Sie die Clusterstruktur, und teilen Sie sie in drei Cluster auf.

```
XY_sing = hclust(dist(XYdf), method="single")
XYs3 = cutree(XY_sing, k=3)
table(XYs3)
XYs3
 1 2 3
448 1 1
```

hclust hat zwei Ausreißer gefunden und alles andere in einem großen Cluster zusammengefasst. Um die "echten" Cluster zu erhalten, müssen Sie möglicherweise k höher setzen.

```
XYs6 = cutree(XY_sing, k=6)
table(XYs6)
XYs6
 1 2 3 4 5 6
```

```
148 150 1 149 1 1
plot(XYdf, pch=20, col=XYs6)
```



Dieser [StackOverflow-Beitrag](#) enthält einige Anweisungen zum Auswählen der Anzahl von Clustern. Beachten Sie jedoch dieses Verhalten beim hierarchischen Clustering.

Hierarchisches Clustering mit hclust online lesen:

<https://riptutorial.com/de/r/topic/8084/hierarchisches-clustering-mit-hclust>

# Kapitel 59: Implementieren Sie das State Machine Pattern mithilfe der S4-Klasse

## Einführung

**Finite-States-Machine**- Konzepte werden normalerweise in OOP-Sprachen (Object Oriented Programming) implementiert, z. B. in [Java](#), [basierend auf dem](#) in GOF definierten [State-Pattern](#) (verweist auf das Buch: "Design Patterns").

R bietet mehrere Mechanismen, um das OO-Paradigma zu simulieren. Wenden wir das [S4-Objektsystem](#) an, um dieses Muster zu implementieren.

## Examples

### Zeilen mit State Machine analysieren

Wenden wir das [State Machine-Muster](#) für das Analysieren von Zeilen mit dem spezifischen Muster mithilfe der S4-Klassenfunktion von R an.

#### PROBLEM ENUNCIATION

Wir müssen eine Datei analysieren, in der jede Zeile Informationen über eine Person enthält, wobei ein Trennzeichen ( ";" ) verwendet wird. Einige Angaben sind jedoch optional. Statt ein leeres Feld anzugeben, fehlt es. In jeder Zeile stehen folgende Informationen zur Verfügung: Name; [Address;]Phone . Wo die Adressinformationen optional sind, haben wir sie manchmal und manchmal nicht, zum Beispiel:

```
GREGORY BROWN; 25 NE 25TH; +1-786-987-6543
DAVID SMITH;786-123-4567
ALAN PEREZ; 25 SE 50TH; +1-786-987-5553
```

Die zweite Zeile enthält keine Adressinformationen. Daher kann die Anzahl der Trennzeichen unterschiedlich sein, wie in diesem Fall mit einem Trennzeichen und für die anderen Zeilen mit zwei Trennzeichen. Da die Anzahl der Trennzeichen variieren kann, besteht eine Möglichkeit, dieses Problem zu lösen, darin, das Vorhandensein oder Nichtvorhandensein eines bestimmten Feldes anhand seines Musters zu erkennen. In diesem Fall können wir einen [regulären Ausdruck verwenden](#), um solche Muster zu identifizieren. Zum Beispiel:

- **Name** : `"^([AZ]'?\s+)* *[AZ]+(\s+[AZ]{1,2}\.\.? +)*[AZ]+((-|\s+) [AZ]+)*$"` . Zum Beispiel: RAFAEL REAL, DAVID R. SMITH, ERNESTO PEREZ GONZALEZ, 0' CONNOR BROWN, LUIS PEREZ-MENA USW.
- **Adresse** : `"^\s[0-9]{1,4}(\s+[AZ]{1,2}[0-9]{1,2}[AZ]{1,2}|[AZ]\s[0-9]+)$"` . Zum Beispiel: 11020 LE JEUNE ROAD , 87 SW 27TH . Der Einfachheit halber haben wir hier nicht die Postleitzahl, Stadt, Bundesland, aber ich kann in dieses Feld aufgenommen werden oder zusätzliche Felder hinzufügen.

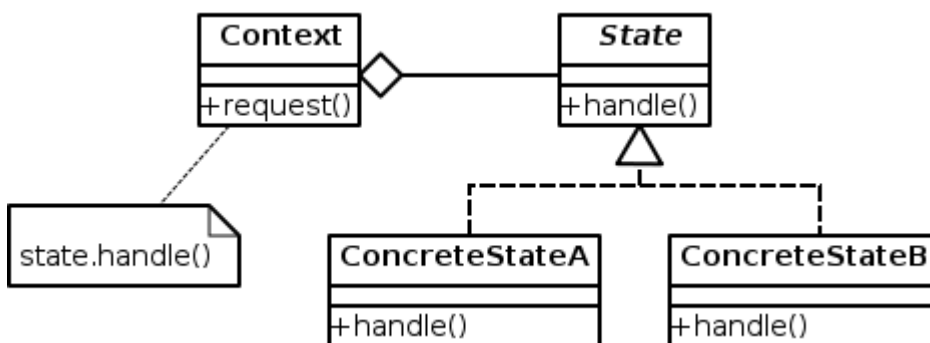


- **Telefon** : `"^\\s*(\\+1(-|\\s+))*[0-9]{3}(-|\\s+)[0-9]{3}(-|\\s+)[0-9]{4}$"` . Zum Beispiel:  
305-123-4567, 305 123 4567, +1-786-123-4567 .

### Anmerkungen :

- Ich denke über das häufigste Muster von US-Adressen und Telefonen nach. Es kann leicht erweitert werden, um allgemeinere Situationen zu berücksichtigen.
- In R hat das Zeichen `"\"` eine besondere Bedeutung für Zeichenvariablen, daher müssen wir es umgehen.
- Um das Definieren regulärer Ausdrücke zu vereinfachen, empfiehlt es sich, die folgende Webseite zu verwenden: [regex101.com](http://regex101.com) , damit Sie mit einem bestimmten Beispiel damit spielen können, bis Sie das erwartete Ergebnis für alle möglichen Kombinationen erhalten.

Die Idee ist, jedes Linienfeld anhand zuvor definierter Muster zu identifizieren. Das Zustandsmuster definiert die folgenden Entitäten (Klassen), die zusammenarbeiten, um das spezifische Verhalten zu steuern (Das Zustandsmuster ist ein Verhaltensmuster):



Beschreiben wir jedes Element unter Berücksichtigung des Kontexts unseres Problems:

- **Context** : Speichert die Kontextinformationen des Parsing-Prozesses, dh den aktuellen Status und behandelt den gesamten State Machine-Prozess. Für jeden Status wird eine Aktion ausgeführt ( `handle()` ), die jedoch vom Kontext abhängig von der für einen bestimmten Status ( `handle()` von `State` Klasse) definierten Aktionsmethode delegiert wird. Es definiert die Schnittstelle, die für Kunden von Interesse ist. Unsere `Context` Klasse kann folgendermaßen definiert werden:
  - **Attribute:** `state`
  - **Methoden:** `handle()` , ...
- **State** : Die abstrakte Klasse, die einen beliebigen Status der State Machine darstellt. Sie definiert eine Schnittstelle zum Einkapseln des Verhaltens, das einem bestimmten Zustand des Kontextes zugeordnet ist. Es kann wie folgt definiert werden:
  - **Attribute:** `name`, `pattern`
  - **Methoden:** `doAction()` , `isState` (mit `pattern` - Attribute überprüfen , ob das Eingabeargument zu diesem Zustand Mustern gehört oder nicht), ...
- **Concrete States (Statusunterklassen):** Jede Unterklasse der Klasse `State` , die ein Verhalten implementiert, das einem Zustand des `Context` . Unsere Unterklassen sind: `InitState` , `NameState` , `AddressState` , `PhoneState` . Solche Klassen implementieren nur die generische Methode unter Verwendung der spezifischen Logik für solche Zustände. Es sind keine zusätzlichen Attribute erforderlich.

**Hinweis:** Es ist eine Frage der Präferenz, wie die Methode benannt wird, die die Aktion ausführt, `handle()`, `doAction()` oder `goNext()`. Der Methodenname `doAction()` kann für beide Klassen (`State` oder `Context`) derselbe sein, den wir als `handle()` in der `Context` Klasse benannt haben, um Verwirrung zu vermeiden, wenn zwei generische Methoden mit denselben Eingabeargumenten, jedoch unterschiedlichen Klassen definiert werden.

## PERSONENKLASSE

Mit der S4-Syntax können wir eine Person-Klasse folgendermaßen definieren:

```
setClass(Class = "Person",
 slots = c(name = "character", address = "character", phone = "character")
)
```

Es ist eine gute Empfehlung, die Klassenattribute zu initialisieren. Die `setClass` Dokumentation schlägt vor, eine generische Methode zu verwenden, die als `"initialize"`, anstatt veraltete Attribute wie `prototype`, `representation`.

```
setMethod("initialize", "Person",
 definition = function(.Object, name = NA_character_,
 address = NA_character_, phone = NA_character_) {
 .Object@name <- name
 .Object@address <- address
 .Object@phone <- phone
 .Object
 }
)
```

Da die Methode `initialize` bereits eine generische Standardmethode Paket ist `methods`, müssen wir die ursprüngliche Argument Definition respektieren. Wir können die Eingabe an der Eingabeaufforderung R überprüfen:

```
> initialize
```

Es gibt die gesamte Funktionsdefinition zurück. Sie können oben sehen, wie die Funktion wie folgt definiert:

```
function (.Object, ...) {...}
```

Wenn wir `setMethod` verwenden, `setMethod` wir daher *genau* dieselbe Syntax (`.Object`) verwenden.

Eine andere existierende generische Methode ist `show`, sie entspricht `toString()` Methode von Java und es ist eine gute Idee, eine spezifische Implementierung für die Klassendomäne zu haben:

```
setMethod("show", signature = "Person",
 definition = function(object) {
 info <- sprintf("%s@name='%s', address='%s', phone='%s'",
 class(object), object@name, object@address, object@phone)
 cat(info)
 invisible(NULL)
 })
```

```
}
)
```

**Hinweis** : Wir verwenden dieselbe Konvention wie in der Standard-Java-Implementierung

`toString()` .

Angenommen, wir möchten die geparsten Informationen (eine Liste von `Person` ) in einem Dataset speichern. Dann sollten wir zuerst eine Liste von Objekten in etwas konvertieren können, das von R transformiert werden kann (z. B. das Objekt als Liste zwingen). Wir können die folgende zusätzliche Methode definieren (für weitere Details siehe den [Beitrag](#) )

```
setGeneric(name = "as.list", signature = c('x'),
 def = function(x) standardGeneric("as.list"))

Suggestion taken from here:
http://stackoverflow.com/questions/30386009/how-to-extend-as-list-in-a-canonical-way-to-s4-
objects
setMethod("as.list", signature = "Person",
 definition = function(x) {
 mapply(function(y) {
 #apply as.list if the slot is again an user-defined object
 #therefore, as.list gets applied recursively
 if (inherits(slot(x,y), "Person")) {
 as.list(slot(x,y))
 } else {
 #otherwise just return the slot
 slot(x,y)
 }
 },
 slotNames(class(x)),
 SIMPLIFY=FALSE)
 })
```

R bietet keine Zuckersyntax für OO, da die Sprache ursprünglich so konzipiert wurde, dass sie für Statistiker wertvolle Funktionen bereitstellt. Daher erfordert jede Benutzermethode zwei Teile: 1) den Definitionsteil (über `setGeneric` ) und 2) den Implementierungsteil (über `setMethod` ). Wie im obigen Beispiel.

## STAATSKLASSE

Nach der S4-Syntax definieren wir die abstrakte `State` Klasse.

```
setClass(Class = "State", slots = c(name = "character", pattern = "character"))

setMethod("initialize", "State",
 definition = function(.Object, name = NA_character_, pattern = NA_character_) {
 .Object@name <- name
 .Object@pattern <- pattern
 .Object
 })

setMethod("show", signature = "State",
 definition = function(object) {
```

```

 info <- sprintf("%s@[name='%s', pattern='%s']", class(object),
 object@name, object@pattern)
 cat(info)
 invisible(NULL)
 }
)

setGeneric(name = "isState", signature = c('obj', 'input'),
 def = function(obj, input) standardGeneric("isState"))

setGeneric(name = "doAction", signature = c('obj', 'input', 'context'),
 def = function(obj, input, context) standardGeneric("doAction"))

```

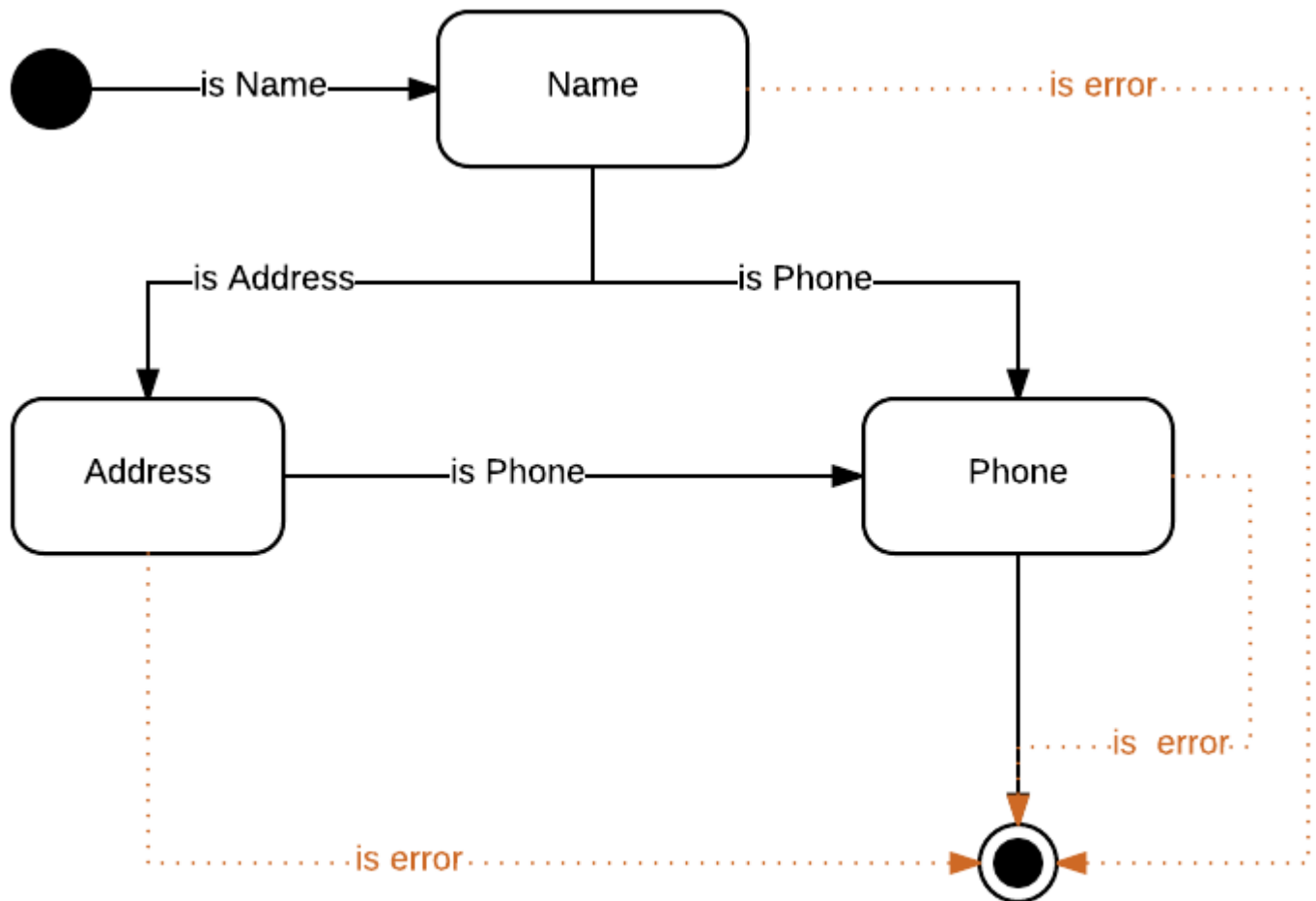
Jede Unterklasse von `State` hat einen `name` und ein `pattern`, aber auch eine Möglichkeit, um zu ermitteln, ob eine bestimmte Eingabe zu diesem Status gehört (`isState()` Methode), und die entsprechenden Aktionen für diesen Status zu `doAction()` Methode).

Um den Prozess zu verstehen, definieren wir die Übergangsmatrix für jeden Status basierend auf der erhaltenen Eingabe:

| Eingang / aktueller Status | Drin | Name    | Adresse | Telefon |
|----------------------------|------|---------|---------|---------|
| Name                       | Name |         |         |         |
| Adresse                    |      | Adresse |         |         |
| Telefon                    |      | Telefon | Telefon |         |
| Ende                       |      |         |         | Ende    |

**Hinweis:** Die Zelle `[row, col]=[i, j]` repräsentiert den Zielzustand für den aktuellen Zustand `j`, wenn sie die Eingabe `i` empfängt.

Dies bedeutet, dass unter dem Status `Name` zwei Eingaben empfangen werden können: eine `Adresse` oder eine `Telefonnummer`. Eine andere Möglichkeit, die Transaktionstabelle darzustellen, ist das folgende [UML State Machine](#)-Diagramm:



**is error:** when the input argument has an invalid pattern

Lassen Sie uns jeden einzelnen Zustand als einen Unterzustand der Klasse `State` implementieren

## STAATLICHE UNTERKLASSEN

### Init State :

Der Anfangszustand wird über folgende Klasse implementiert:

```

setClass("InitState", contains = "State")

setMethod("initialize", "InitState",
 definition = function(.Object, name = "init", pattern = NA_character_) {
 .Object@name <- name
 .Object@pattern <- pattern
 .Object
 }
)

setMethod("show", signature = "InitState",
 definition = function(object) {
 callNextMethod()
 }
)

```

```
)
```

In R bezeichnet eine Klasse eine Unterklasse einer anderen Klasse, die das Attribut `contains` und den Klassennamen der übergeordneten Klasse angibt.

Da die Unterklassen nur die generischen Methoden implementieren, ohne zusätzliche Attribute hinzuzufügen, rufen Sie die `show` Methode einfach die entsprechende Methode von der oberen Klasse auf (über die Methode `callNextMethod()` ).

Dem Anfangszustand ist kein Muster zugeordnet, er stellt lediglich den Anfang des Prozesses dar. Dann initialisieren wir die Klasse mit einem `NA` Wert.

Jetzt können Sie die generischen Methoden aus der `State` Klasse implementieren:

```
setMethod(f = "isState", signature = "InitState",
 definition = function(obj, input) {
 nameState <- new("NameState")
 result <- isState(nameState, input)
 return(result)
 }
)
```

Für diesen bestimmten Zustand (ohne `pattern` ) wird die Idee, dass der Parsing-Prozess mit dem ersten Feld nur initialisiert wird, ein `name` , andernfalls ein Fehler.

```
setMethod(f = "doAction", signature = "InitState",
 definition = function(obj, input, context) {
 nameState <- new("NameState")
 if (isState(nameState, input)) {
 person <- context@person
 person@name <- trimws(input)
 context@person <- person
 context@state <- nameState
 } else {
 msg <- sprintf("The input argument: '%s' cannot be identified", input)
 stop(msg)
 }
 return(context)
 }
)
```

Die Methode `doAction` stellt den Übergang bereit und aktualisiert den Kontext mit den extrahierten Informationen. Hier greifen wir über den `@-operator` . Stattdessen können wir `get/set` Methoden definieren, um diesen Prozess zu kapseln (wie es in den OO-Best Practices vorgeschrieben ist: encapsulation). Dies würde jedoch vier weitere Methoden pro `get-set` hinzufügen `get-set` ohne für dieses Beispiel einen Mehrwert zu schaffen.

Es ist eine gute Empfehlung in allen `doAction` Implementierungen, einen Schutz hinzuzufügen, wenn das Eingabeargument nicht richtig identifiziert wird.

## Name Staat

Hier ist die Definition dieser Klassendefinition:

```

setClass ("NameState", contains = "State")

setMethod("initialize", "NameState",
 definition=function(.Object, name="name",
 pattern = "^[A-Z]'?\\s+)* *[A-Z]+(\\s+[A-Z]{1,2}\\\\.?.? +)*[A-Z]+((-|\\s+)[A-Z]+)*$")
{
 .Object@pattern <- pattern
 .Object@name <- name
 .Object
}
)

setMethod("show", signature = "NameState",
 definition = function(object) {
 callNextMethod()
 }
)

```

Wir verwenden die Funktion `grepl` um zu überprüfen, `grepl` die Eingabe zu einem gegebenen Muster gehört.

```

setMethod(f="isState", signature="NameState",
 definition=function(obj, input) {
 result <- grepl(obj@pattern, input, perl=TRUE)
 return(result)
 }
)

```

Jetzt definieren wir die Aktion, die für einen bestimmten Zustand ausgeführt werden soll:

```

setMethod(f = "doAction", signature = "NameState",
 definition=function(obj, input, context) {
 addressState <- new("AddressState")
 phoneState <- new("PhoneState")
 person <- context@person
 if (isState(addressState, input)) {
 person@address <- trimws(input)
 context@person <- person
 context@state <- addressState
 } else if (isState(phoneState, input)) {
 person@phone <- trimws(input)
 context@person <- person
 context@state <- phoneState
 } else {
 msg <- sprintf("The input argument: '%s' cannot be identified", input)
 stop(msg)
 }
 return(context)
 }
)

```

Hier betrachten wir mögliche Übergänge: einen für den Adressstatus und den anderen für den Telefonstatus. In allen Fällen aktualisieren wir die Kontextinformationen:

- Die `person` : `address` oder `phone` mit dem Eingabeargument.
- Der `state` des Prozesses

Um den Zustand zu identifizieren, rufen Sie die Methode auf: `isState()` für einen bestimmten Zustand. Wir erstellen standardmäßige Zustände ( `addressState`, `phoneState` ) und fragen nach einer bestimmten Validierung.

Die Logik für die Implementierung der anderen Unterklassen (eine pro Status) ist sehr ähnlich.

## Adresstatus

```
setClass("AddressState", contains = "State")

setMethod("initialize", "AddressState",
 definition = function(.Object, name="address",
 pattern = "^\\s[0-9]{1,4}(\\s+[A-Z]{1,2}[0-9]{1,2}[A-Z]{1,2}|[A-Z]\\s0-9+)$") {
 .Object@pattern <- pattern
 .Object@name <- name
 .Object
 }
)

setMethod("show", signature = "AddressState",
 definition = function(object) {
 callNextMethod()
 }
)

setMethod(f="isState", signature="AddressState",
 definition=function(obj, input) {
 result <- grepl(obj@pattern, input, perl=TRUE)
 return(result)
 }
)

setMethod(f = "doAction", "AddressState",
 definition=function(obj, input, context) {
 phoneState <- new("PhoneState")
 if (isState(phoneState, input)) {
 person <- context@person
 person@phone <- trimws(input)
 context@person <- person
 context@state <- phoneState
 } else {
 msg <- sprintf("The input argument: '%s' cannot be identified", input)
 stop(msg)
 }
 return(context)
 }
)
```

## Telefonzustand

```
setClass("PhoneState", contains = "State")

setMethod("initialize", "PhoneState",
 definition = function(.Object, name = "phone",
 pattern = "^\\s*(\\+1(-\\s+))*[0-9]{3}(-\\s+)[0-9]{3}(-\\s+)[0-9]{4}$") {
 .Object@pattern <- pattern
 .Object@name <- name
 .Object
 }
)
```



```

 }
)

setMethod("show", signature = "PhoneState",
 definition = function(object) {
 callNextMethod()
 }
)

setMethod(f = "isState", signature = "PhoneState",
 definition = function(obj, input) {
 result <- grepl(obj@pattern, input, perl = TRUE)
 return(result)
 }
)

```

Hier fügen wir die Personeninformationen in die Liste der `persons` des `context` .

```

setMethod(f = "doAction", "PhoneState",
 definition = function(obj, input, context) {
 context <- addPerson(context, context@person)
 context@state <- new("InitState")
 return(context)
 }
)

```

## KONTEXTKLASSE

Nun erklären `Context` Implementierung der `Context` Klasse. Wir können es unter Berücksichtigung der folgenden Attribute definieren:

```

setClass(Class = "Context",
 slots = c(state = "State", persons = "list", person = "Person")
)

```

Woher

- `state` : Der aktuelle Status des Prozesses
- `person` : Die aktuelle Person, die die Informationen darstellt, die wir bereits aus der aktuellen Zeile analysiert haben.
- `persons` : Die Liste der analysierten Personen.

**Anmerkung** : Optional können wir einen `name` hinzufügen, um den Kontext anhand des `name` zu identifizieren, falls mit mehr als einem Parser-Typ gearbeitet wird.

```

setMethod(f="initialize", signature="Context",
 definition = function(.Object) {
 .Object@state <- new("InitState")
 .Object@persons <- list()
 .Object@person <- new("Person")
 return(.Object)
 }
)

setMethod("show", signature = "Context",

```

```

definition = function(object) {
 cat("An object of class ", class(object), "\n", sep = "")
 info <- sprintf("[state='%s', persons='%s', person='%s']", object@state,
 toString(object@persons), object@person)
 cat(info)
 invisible(NULL)
}
)

setGeneric(name = "handle", signature = c('obj', 'input', 'context'),
 def = function(obj, input, context) standardGeneric("handle"))

setGeneric(name = "addPerson", signature = c('obj', 'person'),
 def = function(obj, person) standardGeneric("addPerson"))

setGeneric(name = "parseLine", signature = c('obj', 's'),
 def = function(obj, s) standardGeneric("parseLine"))

setGeneric(name = "parseLines", signature = c('obj', 's'),
 def = function(obj, s) standardGeneric("parseLines"))

setGeneric(name = "as.df", signature = c('obj'),
 def = function(obj) standardGeneric("as.df"))

```

Mit solchen generischen Methoden kontrollieren wir das gesamte Verhalten des Analyseprozesses:

- `handle()` : `doAction()` die bestimmte `doAction()` -Methode des aktuellen `state` .
- `addPerson` : Sobald wir den `addPerson` erreicht haben, müssen wir eine `person` zur Liste der `persons` hinzufügen, die wir analysiert haben.
- `parseLine()` : Analysiert eine einzelne Zeile
- `parseLines()` : Analysiert mehrere Zeilen (ein `parseLines()` )
- `as.df()` : Extrahieren Sie die Informationen aus der `persons` in ein `as.df()` .

Lassen Sie uns nun mit den entsprechenden Implementierungen fortfahren:

`handle()` -Methode, delegiert die `doAction()` -Methode aus dem aktuellen `state` des `context` :

```

setMethod(f = "handle", signature = "Context",
 definition = function(obj, input) {
 obj <- doAction(obj@state, input, obj)
 return(obj)
 }
)

setMethod(f = "addPerson", signature = "Context",
 definition = function(obj, person) {
 obj@persons <- c(obj@persons, person)
 return(obj)
 }
)

```

Zuerst teilen wir die ursprüngliche Zeile in einem Array mit dem Trennzeichen auf, um jedes Element über die R-Funktion `strsplit()` zu identifizieren, und `strsplit()` dann jedes Element als Eingabewert für einen bestimmten Zustand. Die Methode `handle()` gibt den `context` mit den

aktualisierten Informationen ( `state` , `person` , `persons` ) erneut zurück.

```
setMethod(f = "parseLine", signature = "Context",
 definition = function(obj, s) {
 elements <- strsplit(s, ";")[[1]]
 # Adding an empty field for considering the end state.
 elements <- c(elements, "")
 n <- length(elements)
 input <- NULL
 for (i in (1:n)) {
 input <- elements[i]
 obj <- handle(obj, input)
 }
 return(obj@person)
 }
)
```

Because R kopiert das Eingabeargument. Wir müssen den Kontext ( `obj` ) zurückgeben:

```
setMethod(f = "parseLines", signature = "Context",
 definition = function(obj, s) {
 n <- length(s)
 listOfPersons <- list()
 for (i in (1:n)) {
 ipersons <- parseLine(obj, s[i])
 listOfPersons[[i]] <- ipersons
 }
 obj@persons <- listOfPersons
 return(obj)
 }
)
```

Das Attribut `persons` ist eine Liste von Exemplaren der Klasse S4 `Person` . Dieses Etwas kann nicht zu einem Standardtyp erzwungen werden, da R nicht weiß, dass es eine Instanz einer benutzerdefinierten Klasse behandelt. Die Lösung besteht darin, eine `Person` mithilfe der zuvor definierten `as.list` Methode in eine Liste zu konvertieren. Dann können wir diese Funktion auf jedes Element der Liste `persons` , über die `lapply()` Funktion. `data.frame` dann beim nächsten Aufruf von `lapply()` Funktion `data.frame` , um jedes Element der `persons.list` in einen `data.frame` zu konvertieren. Schließlich wird die Funktion `rbind()` aufgerufen, um jedes Element, das als neue Zeile des generierten `rbind()` konvertiert wurde, `rbind()` hierzu finden Sie in diesem [Beitrag](#) ).

```
Sugestion taken from this post:
http://stackoverflow.com/questions/4227223/r-list-to-data-frame
setMethod(f = "as.df", signature = "Context",
 definition = function(obj) {
 persons <- obj@persons
 persons.list <- lapply(persons, as.list)
 persons.ds <- do.call(rbind, lapply(persons.list, data.frame, stringsAsFactors = FALSE))
 return(persons.ds)
 }
)
```

## ALLES ZUSAMMENSTELLEN

Lassen Sie uns schließlich die gesamte Lösung testen. Definieren Sie die Zeilen, in denen analysiert werden soll, wo für die zweite Zeile die Adressinformationen fehlen.

```
s <- c(
 "GREGORY BROWN; 25 NE 25TH; +1-786-987-6543",
 "DAVID SMITH; 786-123-4567",
 "ALAN PEREZ; 25 SE 50TH; +1-786-987-5553"
)
```

Jetzt initialisieren wir den `context` und parsen die Zeilen:

```
context <- new("Context")
context <- parseLines(context, s)
```

Holen Sie sich schließlich den entsprechenden Datensatz und drucken Sie ihn aus:

```
df <- as.df(context)
> df
```

|   | name          | address    | phone           |
|---|---------------|------------|-----------------|
| 1 | GREGORY BROWN | 25 NE 25TH | +1-786-987-6543 |
| 2 | DAVID SMITH   | <NA>       | 786-123-4567    |
| 3 | ALAN PEREZ    | 25 SE 50TH | +1-786-987-5553 |

Testen wir jetzt die `show` Methoden:

```
> show(context@persons[[1]])
Person@[name='GREGORY BROWN', address='25 NE 25TH', phone='+1-786-987-6543']
```

Und für einige Unterzustände:

```
> show(new("PhoneState"))
PhoneState@[name='phone', pattern='^\s*(\+1(-|\s+))*[0-9]{3}(-|\s+)[0-9]{3}(-|\s+)[0-9]{4}$']
```

Testen Sie `as.list()` die `as.list()` -Methode:

```
> as.list(context@persons[[1]])
$name
[1] "GREGORY BROWN"

$address
[1] "25 NE 25TH"

$phone
[1] "+1-786-987-6543"

>
```

## FAZIT

Dieses Beispiel zeigt, wie das Zustandsmuster implementiert wird, wobei einer der verfügbaren Mechanismen von R für die Verwendung des OO-Paradigmas verwendet wird. Dennoch ist die R OO-Lösung nicht benutzerfreundlich und unterscheidet sich so sehr von anderen OOP-Sprachen.

Sie müssen Ihre Denkweise ändern, da die Syntax völlig anders ist und mehr an das Paradigma der funktionalen Programmierung erinnert. Zum Beispiel anstelle von `object.setID("A1")` wie in Java / C #, müssen Sie für R die Methode folgendermaßen aufrufen: `setID(object, "A1")`. Daher müssen Sie das Objekt immer als Eingabeargument angeben, um den Kontext der Funktion bereitzustellen. Auf dieselbe Weise gibt es kein spezielles Attribut `this` Klasse und entweder ein `"."` Notation für den Zugriff auf Methoden oder Attribute der angegebenen Klasse. Es ist eine weitere Fehleraufforderung, da das Verweisen auf eine Klasse oder Methoden über den Attributwert ( `"Person"` , `"isState"` usw.) erfolgt.

Die obige S4-Klassenlösung erfordert viel mehr Codezeilen als herkömmliche Java / C # - Sprachen, um einfache Aufgaben auszuführen. Wie auch immer, das State Pattern ist eine gute und generische Lösung für solche Probleme. Es vereinfacht den Prozess, der die Logik in einen bestimmten Zustand delegiert. Anstelle eines großen `if-else` Blocks zum Steuern aller Situationen haben wir in jeder `State` Unterklassenimplementierung kleinere `if-else` Blöcke zur Implementierung der in jedem Staat auszuführenden Aktion.

**Anhang :** [Hier können](#) Sie das gesamte Skript herunterladen.

Jeder Vorschlag ist willkommen.

Implementieren Sie das State Machine Pattern mithilfe der S4-Klasse online lesen:  
<https://riptutorial.com/de/r/topic/9126/implementieren-sie-das-state-machine-pattern-mithilfe-der-s4-klasse>

# Kapitel 60: JSON

## Examples

### JSON zu / von R-Objekten

Das `jsonlite` [Paket](#) ist ein schneller JSON-Parser und Generator, der für statistische Daten und das Web optimiert ist. Die zwei Hauptfunktionen, die zum Lesen und Schreiben von JSON verwendet werden, sind `fromJSON()` bzw. `toJSON()` und wurden für die Verwendung von `vectors`, `matrices` und `data.frames` sowie von JSON-Streams aus dem Web entwickelt.

Erstellen Sie ein JSON-Array aus einem Vektor und umgekehrt

```
library(jsonlite)

vector to JSON
toJSON(c(1,2,3))
[1,2,3]

fromJSON('[1,2,3]')
[1] 1 2 3
```

Erstellen Sie ein benanntes JSON-Array aus einer Liste und umgekehrt

```
toJSON(list(myVec = c(1,2,3)))
{"myVec":[1,2,3]}

fromJSON('{"myVec":[1,2,3]}')
$myVec
[1] 1 2 3
```

### Komplexere Listenstrukturen

```
list structures
lst <- list(a = c(1,2,3),
 b = list(letters[1:6]))

toJSON(lst)
{"a":[1,2,3],"b":[["a","b","c","d","e","f"]]}

fromJSON('{"a":[1,2,3],"b":[["a","b","c","d","e","f"]]} ')
$a
[1] 1 2 3
#
$b
[,1] [,2] [,3] [,4] [,5] [,6]
[1,] "a" "b" "c" "d" "e" "f"
```

Erstellen Sie JSON aus einem `data.frame` und umgekehrt

```
converting a data.frame to JSON
df <- data.frame(id = seq_along(1:10),
 val = letters[1:10])

toJSON(df)
#
[{"id":1,"val":"a"}, {"id":2,"val":"b"}, {"id":3,"val":"c"}, {"id":4,"val":"d"}, {"id":5,"val":"e"}, {"id":6,"val":"f"}, {"id":7,"val":"g"}, {"id":8,"val":"h"}, {"id":9,"val":"i"}, {"id":10,"val":"j"}]

reading a JSON string
fromJSON(' [{"id":1,"val":"a"}, {"id":2,"val":"b"}, {"id":3,"val":"c"}, {"id":4,"val":"d"}, {"id":5,"val":"e"}, {"id":6,"val":"f"}, {"id":7,"val":"g"}, {"id":8,"val":"h"}, {"id":9,"val":"i"}, {"id":10,"val":"j"}]')

id val
1 1 a
2 2 b
3 3 c
4 4 d
5 5 e
6 6 f
7 7 g
8 8 h
9 9 i
10 10 j
```

## Lesen Sie JSON direkt aus dem Internet

```
Reading JSON from URL
googleway_issues <- fromJSON("https://api.github.com/repos/SymbolixAU/googleway/issues")

googleway_issues$url
[1] "https://api.github.com/repos/SymbolixAU/googleway/issues/20"
[2] "https://api.github.com/repos/SymbolixAU/googleway/issues/19"
[3] "https://api.github.com/repos/SymbolixAU/googleway/issues/14"
[4] "https://api.github.com/repos/SymbolixAU/googleway/issues/11"
[5] "https://api.github.com/repos/SymbolixAU/googleway/issues/9"
[6] "https://api.github.com/repos/SymbolixAU/googleway/issues/5"
[7] "https://api.github.com/repos/SymbolixAU/googleway/issues/2"
```

JSON online lesen: <https://riptutorial.com/de/r/topic/2460/json>

# Kapitel 61: Klassen

## Einführung

Die Klasse eines Datenobjekts bestimmt, welche Funktionen seinen Inhalt verarbeiten. Das `class` ist ein Zeichenvektor, und Objekte können null, eine oder mehrere Klassen haben. Wenn kein Klassenattribut vorhanden ist, gibt es immer noch eine implizite Klasse, die vom `mode` eines Objekts bestimmt wird. Die Klasse kann mit der Funktion überprüft werden `class` und es kann durch die Menge oder modifiziert werden `class<-` Funktion. Das S3-Klassensystem wurde bereits in der Geschichte von S etabliert. Das komplexere S4-Klassensystem wurde später eingeführt

## Bemerkungen

Es gibt mehrere Funktionen, um den "Typ" eines Objekts zu prüfen. Die nützlichste solche Funktion ist die `class`, obwohl manchmal der `mode` eines Objekts untersucht werden muss. Da wir über "Typen" sprechen, könnte man denken, dass `typeof` nützlich sein würde, im Allgemeinen ist das Ergebnis des `mode` jedoch nützlicher, da für Objekte ohne explizites "class"-Attribut der Funktions-Dispatch durch die "implizite Klasse" bestimmt wird von ihrem Modus.

## Examples

### Vektoren

Die einfachste in R verfügbare Datenstruktur ist ein Vektor. Sie können mit der Funktion `c()` Vektoren aus numerischen Werten, logischen Werten und Zeichenketten erstellen. Zum Beispiel:

```
c(1, 2, 3)
[1] 1 2 3
c(TRUE, TRUE, FALSE)
[1] TRUE TRUE FALSE
c("a", "b", "c")
[1] "a" "b" "c"
```

Sie können auch mit der Funktion `c()` Verknüpfung zu Vektoren herstellen.

```
x <- c(1, 2, 5)
y <- c(3, 4, 6)
z <- c(x, y)
z
[1] 1 2 5 3 4 6
```

Eine ausführlichere Vorgehensweise zum Erstellen von Vektoren finden Sie im [Thema "Erstellen von Vektoren"](#)

## Überprüfen Sie die Klassen



Jedem Objekt in R wird eine Klasse zugewiesen. Sie können `class()` , um die Klasse des Objekts zu finden, und `str()` , um die Struktur einschließlich der darin enthaltenen Klassen anzuzeigen. Zum Beispiel:

```
class(iris)
[1] "data.frame"

str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...

class(iris$Species)
[1] "factor"
```

Wir sehen, dass Iris die Klasse `data.frame` und mit `str()` können wir die darin enthaltenen Daten untersuchen. Die Variable Species im Iris-Datenrahmen hat im Gegensatz zu den anderen Variablen, die klassennumerisch sind, einen Klassenfaktor. Die Funktion `str()` gibt auch die Länge der Variablen an und zeigt die ersten Beobachtungen, während die Funktion `class()` nur die Klasse des Objekts bereitstellt.

## Vektoren und Listen

Daten in R werden in Vektoren gespeichert. Ein typischer Vektor ist eine Folge von Werten, die alle den gleichen Speichermodus haben (z. B. Zeichenvektoren, numerische Vektoren). Weitere Informationen zu den atomaren impliziten Klassen und den entsprechenden Speichermodi finden Sie unter `?atomic`: "logical", "integer", "numeric" (synonym "double"), "complex", "character" und "raw" . Viele Klassen sind einfach ein atomarer Vektor mit einem `class` oben:

```
x <- 1826
class(x) <- "Date"
x
[1] "1975-01-01"
x <- as.Date("1970-01-01")
class(x)
[1] "Date"
is(x, "Date")
[1] TRUE
is(x, "integer")
[1] FALSE
is(x, "numeric")
[1] FALSE
mode(x)
[1] "numeric"
```

Listen sind ein spezieller Vektortyp, bei dem jedes Element alles sein kann, sogar eine andere Liste, daher der R-Ausdruck für Listen: "Rekursivvektoren":

```
mylist <- list(A = c(5,6,7,8), B = letters[1:10], CC = list(5, "Z"))
```

Listen haben zwei sehr wichtige Zwecke:

- Da Funktionen nur einen einzelnen Wert zurückgeben können, werden häufig komplizierte Ergebnisse in einer Liste zurückgegeben:

```
f <- function(x) list(xplus = x + 10, xsq = x^2)

f(7)
$xplus
[1] 17
#
$xsq
[1] 49
```

- Listen sind auch die zugrunde liegende grundlegende Klasse für **Datenrahmen**. Unter der Haube ist ein Datenrahmen eine Liste von Vektoren, die alle die gleiche Länge haben:

```
L <- list(x = 1:2, y = c("A", "B"))
DF <- data.frame(L)
DF
x y
1 1 A
2 2 B
is.list(DF)
[1] TRUE
```

Die andere Klasse rekursiver Vektoren sind R-Ausdrücke, die "Sprachobjekte" sind

Klassen online lesen: <https://riptutorial.com/de/r/topic/3563/klassen>

---

# Kapitel 62: Kombinatorik

## Examples

Aufzählen von Kombinationen einer angegebenen Länge

---

## Ohne Ersatz

Mit `combn` erscheint jeder Vektor in einer Spalte:

```
combn(LETTERS, 3)

Showing only first 10.
 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[2,] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
[3,] "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"
```

---

## Mit ersatz

Mit `expand.grid` erscheint jeder Vektor in einer Reihe:

```
expand.grid(LETTERS, LETTERS, LETTERS)
or
do.call(expand.grid, rep(list(LETTERS), 3))

Showing only first 10.
 Var1 Var2 Var3
1 A A A
2 B A A
3 C A A
4 D A A
5 E A A
6 F A A
7 G A A
8 H A A
9 I A A
10 J A A
```

Für den speziellen Fall von Paaren kann "`outer`" verwendet werden, wobei jeder Vektor in eine Zelle eingefügt wird:

```
FUN here is used as a function executed on each resulting pair.
in this case it's string concatenation.
outer(LETTERS, LETTERS, FUN=paste0)

Showing only first 10 rows and columns
 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "AA" "AB" "AC" "AD" "AE" "AF" "AG" "AH" "AI" "AJ"
```

```
[2,] "BA" "BB" "BC" "BD" "BE" "BF" "BG" "BH" "BI" "BJ"
[3,] "CA" "CB" "CC" "CD" "CE" "CF" "CG" "CH" "CI" "CJ"
[4,] "DA" "DB" "DC" "DD" "DE" "DF" "DG" "DH" "DI" "DJ"
[5,] "EA" "EB" "EC" "ED" "EE" "EF" "EG" "EH" "EI" "EJ"
[6,] "FA" "FB" "FC" "FD" "FE" "FF" "FG" "FH" "FI" "FJ"
[7,] "GA" "GB" "GC" "GD" "GE" "GF" "GG" "GH" "GI" "GJ"
[8,] "HA" "HB" "HC" "HD" "HE" "HF" "HG" "HH" "HI" "HJ"
[9,] "IA" "IB" "IC" "ID" "IE" "IF" "IG" "IH" "II" "IJ"
[10,] "JA" "JB" "JC" "JD" "JE" "JF" "JG" "JH" "JI" "JJ"
```

## Zählen von Kombinationen einer angegebenen Länge

### Ohne Ersatz

```
choose(length(LETTERS), 5)
[1] 65780
```

### Mit ersatz

```
length(letters)^5
[1] 11881376
```

Kombinatorik online lesen: <https://riptutorial.com/de/r/topic/5836/kombinatorik>

---

# Kapitel 63: Kontrollieren Sie Flussstrukturen

## Bemerkungen

Bei Schleifen handelt es sich um eine Flusssteuerungsmethode zum Wiederholen einer Aufgabe oder einer Gruppe von Aufgaben über eine Domäne. Die Kernstruktur einer for-Schleife ist

```
for ([index] in [domain]){
 [body]
}
```

Woher

1. `[index]` ist ein Name, der für jede Iteration der Schleife genau einen Wert von `[domain]` annimmt.
2. `[domain]` ist ein Vektor von Werten, über den iteriert werden soll.
3. `[body]` ist der Satz von Anweisungen, die bei jeder Iteration anzuwenden sind.

Betrachten Sie als triviales Beispiel die Verwendung einer for-Schleife, um die kumulative Summe eines Wertektors zu erhalten.

```
x <- 1:4
cumulative_sum <- 0
for (i in x){
 cumulative_sum <- cumulative_sum + x[i]
}
cumulative_sum
```

---

## Struktur von for-Schleifen optimieren

Für Schleifen kann es nützlich sein, Aufgaben zu konzeptionieren und auszuführen, die wiederholt werden sollen. Wenn sie nicht sorgfältig erstellt werden, können sie im Vergleich zu den bevorzugten Funktionen der `apply` sehr langsam ausgeführt werden. Nichtsdestotrotz gibt es eine Handvoll Elemente, die Sie in Ihre for-Schleife-Konstruktion integrieren können, um die Schleife zu optimieren. In vielen Fällen führt eine gute Konstruktion der for-Schleife zu einer Recheneffizienz, die der einer Anwendungsfunktion sehr nahe kommt.

Eine ordnungsgemäß erstellte for-Schleife baut auf der Kernstruktur auf und enthält eine Anweisung, die das Objekt deklariert, das jede Iteration der Schleife erfasst. Dieses Objekt sollte sowohl eine Klasse als auch eine Länge deklariert haben.

```
[output] <- [vector_of_length]
for ([index] in [length_safe_domain]){
 [output][index] <- [body]
}
```

Lassen Sie uns zur Veranschaulichung eine Schleife schreiben, um jeden Wert in einem numerischen Vektor zu `x_squared <- x^2` (dies ist ein triviales Beispiel, das nur zur Veranschaulichung dient. Die "richtige" Art, diese Aufgabe zu `x_squared <- x^2` wäre `x_squared <- x^2` ).

```
x <- 1:100
x_squared <- vector("numeric", length = length(x))
for (i in seq_along(x)){
 x_squared[i] <- x[i]^2
}
```

`x_squared` wieder, dass wir zuerst einen Behälter für die Ausgabe `x_squared` haben und ihm die Klasse "numerisch" mit der gleichen Länge wie `x` . Außerdem haben wir mit der Funktion `seq_along` eine "length safe domain" `seq_along` . `seq_along` erzeugt einen `seq_along` für ein Objekt, das für for-Schleifen geeignet ist. Die Verwendung `for (i in 1:length(x))` erscheint intuitiv. Wenn `x` die Länge 0 hat, versucht die Schleife, die Domäne von `1:0` zu durchlaufen, was zu einem Fehler führt (der 0. Index ist in R nicht definiert ).

Behälterobjekte und längensichere Domänen werden intern von der `apply` der Funktionen gehandhabt, und die Benutzer werden aufgefordert, den `apply` so weit wie möglich anstelle von for-Schleifen zu übernehmen. Bei richtiger Konstruktion kann eine for-Schleife jedoch gelegentlich eine größere Code-Klarheit bei minimalem Effizienzverlust bieten.

## Vektorisieren für Loops

Schleifen können häufig ein nützliches Werkzeug sein, um die Aufgaben zu konzipieren, die in jeder Iteration ausgeführt werden müssen. Wenn die Schleife vollständig entwickelt und konzipiert ist, kann es vorteilhaft sein, aus der Schleife eine Funktion zu machen.

In diesem Beispiel werden wir eine for-Schleife entwickeln, um den Mittelwert jeder Spalte im `mtcars` Dataset zu berechnen (wiederum ein triviales Beispiel, wie es mit der Funktion `colMeans` ).

```
column_mean_loop <- vector("numeric", length(mtcars))
for (k in seq_along(mtcars)){
 column_mean_loop[k] <- mean(mtcars[[k]])
}
```

Die for-Schleife kann in eine Apply-Funktion umgewandelt werden, indem der Hauptteil der Schleife als Funktion umgeschrieben wird.

```
col_mean_fn <- function(x) mean(x)
column_mean_apply <- vapply(mtcars, col_mean_fn, numeric(1))
```

Und um die Ergebnisse zu vergleichen:

```
identical(column_mean_loop,
 unname(column_mean_apply)) #* vapply added names to the elements
 #* remove them for comparison
```

Die Vorteile der vektorisierten Form sind, dass wir einige Codezeilen entfernen konnten. Die Mechanik der Bestimmung der Länge und des Typs des Ausgabeobjekts und des Iterierens über eine Länge sichere Domäne wird von der Apply-Funktion für uns behandelt. Außerdem ist die Apply-Funktion etwas schneller als die Schleife. Der Geschwindigkeitsunterschied ist in menschlicher Hinsicht oft vernachlässigbar, abhängig von der Anzahl der Iterationen und der Komplexität des Körpers.

## Examples

### Grundlegend für die Schleifenkonstruktion

In diesem Beispiel berechnen wir die quadratische Abweichung für jede Spalte in einem `mtcars`, in diesem Fall die `mtcars`.

Option A: ganzzahliger Index

```
squared_deviance <- vector("list", length(mtcars))
for (i in seq_along(mtcars)){
 squared_deviance[[i]] <- (mtcars[[i]] - mean(mtcars[[i]]))^2
}
```

`squared_deviance` ist wie erwartet eine Liste mit 11 Elementen.

```
class(squared_deviance)
length(squared_deviance)
```

Option B: Zeichenindex

```
squared_deviance <- vector("list", length(mtcars))
Squared_deviance <- setNames(squared_deviance, names(mtcars))
for (k in names(mtcars)){
 squared_deviance[[k]] <- (mtcars[[k]] - mean(mtcars[[k]]))^2
}
```

Was ist, wenn wir als Ergebnis einen `data.frame` haben wollen? Nun, es gibt viele Möglichkeiten, eine Liste in andere Objekte umzuwandeln. In diesem Fall ist es jedoch am einfachsten, die `for` Ergebnisse in einem `data.frame` zu speichern.

```
squared_deviance <- mtcars #copy the original
squared_deviance[TRUE]<-NA #replace with NA or do squared_deviance[,]<-NA
for (i in seq_along(mtcars)){
 squared_deviance[[i]] <- (mtcars[[i]] - mean(mtcars[[i]]))^2
}
dim(squared_deviance)
[1] 32 11
```

Das Ergebnis wird dasselbe Ereignis sein, obwohl wir die Zeichenoption (B) verwenden.

### Optimale Konstruktion einer for-Schleife

Um den Effekt von good for loop zu veranschaulichen, berechnen wir den Mittelwert jeder Spalte auf vier verschiedene Arten:

1. Verwendung einer schlecht optimierten Schleife
2. Verwenden Sie eine gut für for-Schleife
3. Verwendung einer `*apply` Funktionsfamilie
4. Verwenden der `colMeans` Funktion

Jede dieser Optionen wird im Code angezeigt. Ein Vergleich der Rechenzeit zum Ausführen jeder Option wird angezeigt. und schließlich wird eine Diskussion der Unterschiede gegeben.

## Schlecht für Schleifen optimiert

```
column_mean_poor <- NULL
for (i in 1:length(mtcars)){
 column_mean_poor[i] <- mean(mtcars[[i]])
}
```

## Gut für die Schleife optimiert

```
column_mean_optimal <- vector("numeric", length(mtcars))
for (i in seq_along(mtcars)){
 column_mean_optimal[i] <- mean(mtcars[[i]])
}
```

## `vapply` Funktion

```
column_mean_vapply <- vapply(mtcars, mean, numeric(1))
```

## `colMeans` Funktion

```
column_mean_colMeans <- colMeans(mtcars)
```

## Effizienzvergleich

Die Ergebnisse des Benchmarking dieser vier Ansätze sind unten dargestellt (Code wird nicht angezeigt)

```
Unit: microseconds
 expr min lq mean median uq max neval cld
 poor 240.986 262.0820 287.1125 275.8160 307.2485 442.609 100 d
 optimal 220.313 237.4455 258.8426 247.0735 280.9130 362.469 100 c
 vapply 107.042 109.7320 124.4715 113.4130 132.6695 202.473 100 a
 colMeans 155.183 161.6955 180.2067 175.0045 194.2605 259.958 100 b
```

Beachten Sie, dass die `for` Schleifen optimierte Schleife die schlecht konstruierte Schleife



überschneidet. Die schlecht konstruierte Schleife erhöht ständig die Länge des Ausgabeobjekts, und bei jeder Änderung der Länge wertet R die Klasse des Objekts neu aus.

Ein Teil dieser zusätzlichen Belastung wird durch die optimierte `for`-Schleife beseitigt, indem der Typ des Ausgabeobjekts und seine Länge vor dem Start der Schleife deklariert werden.

In diesem Beispiel verdoppelt jedoch die Verwendung einer `vapply` Funktion die Recheneffizienz, hauptsächlich weil wir R gesagt haben, dass das Ergebnis numerisch sein müsste (wenn ein Ergebnis nicht numerisch wäre, würde ein Fehler zurückgegeben werden).

Die Verwendung der `colMeans` Funktion ist etwas langsamer als die `vapply` Funktion. Dieser Unterschied ist auf einige `colMeans` in `colMeans` und hauptsächlich auf die `as.matrix` Konvertierung (da `mtcars` ein `data.frame`) zurückzuführen, die nicht in der `vapply` Funktion ausgeführt wurde.

## Die anderen Looping-Konstrukte: während und wiederholen

R stellt zwei zusätzliche Schleifenkonstrukte `while` und `repeat` bereit, die normalerweise in Situationen verwendet werden, in denen die Anzahl der erforderlichen Iterationen unbestimmt ist.

---

## Die `while` Schleife

Die allgemeine Form einer `while` Schleife lautet wie folgt:

```
while (condition) {
 ## do something
 ## in loop body
}
```

wobei der `condition` vor dem Eintritt in den Schleifenkörper bewertet wird. Wenn die `condition` den `TRUE` condition, wird der Code innerhalb des Schleifenkörpers ausgeführt, und dieser Vorgang wird wiederholt, bis die `condition` den `FALSE` (oder eine `break` Anweisung erreicht wird; siehe unten). Im Gegensatz zur `for` Schleife muss, wenn eine `while` Schleife eine Variable zum Durchführen von inkrementellen Iterationen verwendet, die Variable vorab deklariert und initialisiert und innerhalb des Schleifenkörpers aktualisiert werden. Die folgenden Schleifen führen beispielsweise dieselbe Aufgabe aus:

```
for (i in 0:4) {
 cat(i, "\n")
}
0
1
2
3
4

i <- 0
while (i < 5) {
 cat(i, "\n")
 i <- i + 1
}
```

```
0
1
2
3
4
```

In der `while` Schleife oben ist die Linie `i <- i + 1` erforderlich, um eine Endlosschleife zu verhindern.

---

Darüber hinaus ist es möglich, eine `while` Schleife mit einem Aufruf zu beenden, `break` den Loop-Rumpf zu unterbrechen:

```
iter <- 0
while (TRUE) {
 if (runif(1) < 0.25) {
 break
 } else {
 iter <- iter + 1
 }
}
iter
#[1] 4
```

In diesem Beispiel ist `condition` immer `TRUE`. Die einzige Möglichkeit, die Schleife zu beenden, ist die Aufforderung, den Body zu `break`. Beachten Sie, dass der endgültige `iter` Wert beim Ausführen dieses Beispiels vom Status Ihres PRNG abhängt und bei jeder Ausführung des Codes (im Wesentlichen) zu unterschiedlichen Ergebnissen führen sollte.

---

## Die `repeat`

Das `repeat` ist im Wesentlichen dasselbe wie `while (TRUE) { ## something }` und hat die folgende Form:

```
repeat ({
 ## do something
 ## in loop body
})
```

Die zusätzlichen `{ }` sind nicht erforderlich, aber die `()` sind. Umschreiben des vorherigen Beispiels mit `repeat`,

```
iter <- 0
repeat ({
 if (runif(1) < 0.25) {
 break
 } else {
 iter <- iter + 1
 }
})
iter
#[1] 2
```

## Mehr zur `break`

Es ist wichtig zu wissen, dass `break` *nur die unmittelbar einschließende Schleife beendet*. Das heißt, das Folgende ist eine Endlosschleife:

```
while (TRUE) {
 while (TRUE) {
 cat("inner loop\n")
 break
 }
 cat("outer loop\n")
}
```

Mit etwas Kreativität ist es jedoch möglich, innerhalb einer verschachtelten Schleife vollständig zu brechen. Betrachten Sie als Beispiel den folgenden Ausdruck, der in seinem aktuellen Zustand endlos durchlaufen wird:

```
while (TRUE) {
 cat("outer loop body\n")
 while (TRUE) {
 cat("inner loop body\n")
 x <- runif(1)
 if (x < .3) {
 break
 } else {
 cat(sprintf("x is %.5f\n", x))
 }
 }
}
```

Eine Möglichkeit ist, zu erkennen, dass, im Gegensatz zu `break`, die `return` Ausdruck die Fähigkeit **hat** Kontrolle über mehrere Ebenen der umschließenden Schleifen zurückzukehren. Da `return` jedoch nur gültig ist, wenn es innerhalb einer Funktion verwendet wird, können wir `break` nicht einfach durch `return()` ersetzen, sondern müssen den gesamten Ausdruck als anonyme Funktion einschließen:

```
(function() {
 while (TRUE) {
 cat("outer loop body\n")
 while (TRUE) {
 cat("inner loop body\n")
 x <- runif(1)
 if (x < .3) {
 return()
 } else {
 cat(sprintf("x is %.5f\n", x))
 }
 }
 }
})()
```

Alternativ können wir eine Dummy-Variable (`exit`) vor dem Ausdruck erstellen und über `<-` aus der inneren Schleife aktivieren, wenn wir fertig sind:

```
exit <- FALSE
while (TRUE) {
 cat("outer loop body\n")
 while (TRUE) {
 cat("inner loop body\n")
 x <- runif(1)
 if (x < .3) {
 exit <-<- TRUE
 break
 } else {
 cat(sprintf("x is %.5f\n", x))
 }
 }
 if (exit) break
}
```

Kontrollieren Sie Flussstrukturen online lesen: <https://riptutorial.com/de/r/topic/2201/kontrollieren-sie-flussstrukturen>

---

# Kapitel 64: Lauflängencodierung

## Bemerkungen

Ein Lauf ist eine aufeinanderfolgende Folge wiederholter Werte oder Beobachtungen. Bei wiederholten Werten beschreibt die "Lauflängencodierung" von R einen Vektor präzise in Bezug auf seine Läufe. Erwägen:

```
dat <- c(1, 2, 2, 2, 3, 1, 4, 4, 1, 1)
```

Wir haben eine Länge von 1s; dann ein 3er-Lauf von 2s; dann ein Länge-Eins-Lauf von 3s; und so weiter. Die Lauflängencodierung von R erfasst alle Längen und Werte der Läufe eines Vektors.

---

## Erweiterungen

Ein Lauf kann sich auch auf aufeinanderfolgende Beobachtungen in Tabellendaten beziehen. Obwohl R keine natürliche Art der Codierung dieser Codierungen hat, können sie mit [rleid](#) aus dem [data.table](#)-Paket (derzeit eine Sackgassenverbindung) behandelt werden .

## Examples

### Lauflängencodierung mit "rle"

Lauflängencodierung erfasst die Längen von Läufen aufeinanderfolgender Elemente in einem Vektor. Betrachten Sie einen Beispielvektor:

```
dat <- c(1, 2, 2, 2, 3, 1, 4, 4, 1, 1)
```

Die Funktion `rle` extrahiert jeden Lauf und seine Länge:

```
r <- rle(dat)
r
Run Length Encoding
lengths: int [1:6] 1 3 1 1 2 2
values : num [1:6] 1 2 3 1 4 1
```

Die Werte für jeden Lauf werden in `r$values` erfasst:

```
r$values
[1] 1 2 3 1 4 1
```

Dies zeigt, dass wir zuerst eine Reihe von Einsen gesehen haben, dann eine Reihe von Zweien, dann eine Reihe von Dreien, dann eine Reihe von Einsen und so weiter.

Die Längen jedes Laufs werden in `r$lengths` erfasst:

```
r$lengths
[1] 1 3 1 1 2 2
```

Wir sehen, dass der anfängliche Lauf von 1er Länge von 1 war, der Lauf von 2en von Länge 3 und so weiter.

## Identifizierung und Gruppierung nach Läufen in Basis R

Vielleicht möchten Sie ihre Daten nach den Läufen einer Variablen gruppieren und eine Art Analyse durchführen. Betrachten Sie das folgende einfache Dataset:

```
(dat <- data.frame(x = c(1, 1, 2, 2, 2, 1), y = 1:6))
x y
1 1 1
2 1 2
3 2 3
4 2 4
5 2 5
6 1 6
```

Die Variable `x` hat drei Läufe: einen Lauf der Länge 2 mit dem Wert 1, einen Lauf der Länge 3 mit dem Wert 2 und einen Lauf der Länge 1 mit dem Wert 1. Wir möchten vielleicht den Mittelwert der Variablen `y` in jedem der berechnen Läufe der Variablen `x` (diese Mittelwerte sind 1,5, 4 und 6).

In Basis R würden wir zuerst die Lauflängencodierung der `x` Variablen mit `rle` :

```
(r <- rle(dat$x))
Run Length Encoding
lengths: int [1:3] 2 3 1
values : num [1:3] 1 2 1
```

Der nächste Schritt besteht darin, die Laufnummer jeder Zeile unseres Datensatzes zu berechnen. Wir wissen, dass die Gesamtzahl der Läufe `length(r$lengths)` und die Länge jedes Laufs `r$lengths`, sodass wir die Laufnummer jedes unserer Läufe mit `rep` berechnen können:

```
(run.id <- rep(seq_along(r$lengths), r$lengths))
[1] 1 1 2 2 2 3
```

Jetzt können wir `tapply`, um den mittleren `y` Wert für jeden Lauf zu berechnen, `tapply` wir die Lauf-ID zusammenfassen:

```
data.frame(x=r$values, meanY=tapply(dat$y, run.id, mean))
x meanY
1 1 1.5
2 2 4.0
3 1 6.0
```

## Identifizieren und Gruppieren nach Läufen in `data.table`

Das Paket `data.table` bietet eine bequeme Möglichkeit zum Gruppieren von Läufen in Daten. Betrachten Sie die folgenden Beispieldaten:

```
library(data.table)
(DT <- data.table(x = c(1, 1, 2, 2, 2, 1), y = 1:6))
x y
1: 1 1
2: 1 2
3: 2 3
4: 2 4
5: 2 5
6: 1 6
```

Die Variable `x` hat drei Läufe: einen Lauf der Länge 2 mit dem Wert 1, einen Lauf der Länge 3 mit dem Wert 2 und einen Lauf der Länge 1 mit dem Wert 1. Wir möchten vielleicht den Mittelwert der Variablen `y` in jedem der berechnen Läufe der Variablen `x` (diese Mittelwerte sind 1,5, 4 und 6).

Die Funktion `data.table::rleid` stellt eine ID bereit, die die Lauf-ID jedes Elements eines Vektors angibt:

```
rleid(DT$x)
[1] 1 1 2 2 2 3
```

Man kann dann leicht diese Lauf-ID gruppieren und die `y` Daten zusammenfassen:

```
DT[,mean(y),by=.(x, rleid(x))]
x rleid V1
1: 1 1 1.5
2: 2 2 4.0
3: 1 3 6.0
```

## Laufgrößencodierung zum Komprimieren und Dekomprimieren von Vektoren

Lange Vektoren mit langen Läufen mit demselben Wert können erheblich komprimiert werden, indem sie in ihrer Laufgrößencodierung gespeichert werden (der Wert jedes Laufs und die Anzahl der Wiederholungen des Werts). Als Beispiel betrachten wir einen Vektor der Länge 10 Millionen mit einer großen Anzahl von Einsen und nur einer kleinen Anzahl von Nullen:

```
set.seed(144)
dat <- sample(rep(0:1, c(1, 1e5)), 1e7, replace=TRUE)
table(dat)
0 1
103 9999897
```

Das Speichern von 10 Millionen Einträgen erfordert viel Speicherplatz. Stattdessen können Sie einen Datenrahmen mit der Laufgrößencodierung dieses Vektors erstellen:

```
rle.df <- with(rle(dat), data.frame(values, lengths))
dim(rle.df)
[1] 207 2
head(rle.df)
```

```
values lengths
1 1 52818
2 0 1
3 1 219329
4 0 1
5 1 318306
6 0 1
```

Bei der Lauflängencodierung sehen wir, dass die ersten 52.818 Werte in dem Vektor Einsen sind, gefolgt von einer einzelnen 0, gefolgt von 219.329 aufeinanderfolgenden Einsen, gefolgt von einer 0 usw. Die Lauflängencodierung hat nur 207 Einträge, so dass wir nur 414 Werte statt 10 Millionen Werte speichern müssen. Da `rle.df` ein `rle.df` ist, kann er mit Standardfunktionen wie `write.csv` gespeichert werden.

Das Dekomprimieren eines Vektors bei der Lauflängencodierung kann auf zwei Arten erfolgen. Die erste Methode ist einfach zu nennen `rep`, das vorbeiziehenden `values` Element der Lauflängencodierung als erstes Argument und die `lengths` Element der Lauflängencodierung als zweites Argument:

```
decompressed <- rep(rle.df$values, rle.df$lengths)
```

Wir können bestätigen, dass unsere dekomprimierten Daten mit unseren ursprünglichen Daten identisch sind:

```
identical(decompressed, dat)
[1] TRUE
```

Die zweite Methode besteht darin, die eingebaute `inverse.rle` Funktion von R für das `rle` Objekt zu verwenden, zum Beispiel:

```
rle.obj <- rle(dat) # create a rle object here
class(rle.obj)
[1] "rle"

dat.inv <- inverse.rle(rle.obj) # apply the inverse.rle on the rle object
```

Wir können erneut bestätigen, dass dies genau die ursprünglichen `dat` :

```
identical(dat.inv, dat)
[1] TRUE
```

Lauflängencodierung online lesen: <https://riptutorial.com/de/r/topic/1133/laulangencodierung>



# Kapitel 65: Lesen und Schreiben von Tabellendaten in Klartextdateien (CSV, TSV usw.)

## Syntax

- `read.csv` (file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE, comment.char = "", ...)
- `read.csv2` (file, header = TRUE, sep = ";", quote = "\"", dec = ",", fill = TRUE, comment.char = "", ...)
- `readr::read_csv` (file, col\_names = TRUE, col\_types = NULL, locale = default\_locale(), na = c("", "NA")), comment = "", trim\_ws = TRUE, überspringen = 0, n\_max = -1 Fortschritt = interaktiv())
- `data.table::fread` (input, sep = "auto", sep2 = "auto", nrow = -1L, header = "auto", na.strings = "NA", stringsAsFactors = FALSE, verbose = getOption("datatable")\$verbose), autostart = 1L, überspringen = 0L, select = NULL, drop = NULL, colClasses = NULL, integer64 = getOption("datatable.integer64"), # default: integer64 "dec = if (sep! = \", \") \". \"else\", \", col.names, check.names = FALSE, Kodierung = \"unbekannt\", strip.white = TRUE, showProgress = getOption("datatable.showProgress"), # default: TRUE data.table = getOption("datatable.fread.datatable") # default: TRUE

## Parameter

| Parameter    | Einzelheiten                                                                                                                           |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Datei        | Name der zu lesenden CSV-Datei                                                                                                         |
| Header       | logisch: Enthält die CSV-Datei eine Kopfzeile mit Spaltennamen?                                                                        |
| sep          | Zeichen: Symbol, das die Zellen in jeder Zeile trennt                                                                                  |
| Zitat        | Zeichen: Symbol für Anführungszeichen                                                                                                  |
| dez          | Zeichen: Symbol als Dezimaltrennzeichen                                                                                                |
| füllen       | logisch: Bei TRUE werden Zeilen mit ungleicher Länge mit leeren Feldern gefüllt.                                                       |
| comment.char | Zeichen: Zeichen, das als Kommentar in der CSV-Datei verwendet wird. Zeilen, denen dieses Zeichen vorangestellt ist, werden ignoriert. |

| Parameter | Einzelheiten                                          |
|-----------|-------------------------------------------------------|
| ...       | zusätzliche Argumente, die an <code>read.table</code> |

## Bemerkungen

Beachten Sie, dass beim Exportieren in ein Nur-Text-Format ein Großteil der in den Daten codierten Informationen, wie z. B. Variablenklassen, aus Gründen der großen Portabilität geopfert wird. In Fällen, in denen eine solche Portabilität nicht erforderlich ist, kann ein Format wie [.RData](#) oder [Feather](#) sinnvoller sein.

Die Eingabe / Ausgabe für andere Dateitypen wird in mehreren anderen Themen behandelt, die alle mit [Eingabe und Ausgabe](#) verknüpft sind.

## Examples

### .Csv-Dateien importieren

## Import mit Basis R

Comma-Separated-Value-Dateien (CSVs) können mit `read.csv` importiert werden. Dabei wird `read.csv` `read.table`, aber mit `sep = ","` das Trennzeichen auf ein Komma gesetzt.

```
get the file path of a CSV included in R's utils package
csv_path <- system.file("misc", "exDIF.csv", package = "utils")

path will vary based on installation location
csv_path
[1] "/Library/Frameworks/R.framework/Resources/library/utils/misc/exDIF.csv"

df <- read.csv(csv_path)

df
Var1 Var2
1 2.70 A
2 3.14 B
3 10.00 A
4 -7.00 A
```

Die benutzerfreundliche Option `file.choose` ermöglicht das Durchsuchen der Verzeichnisse:

```
df <- read.csv(file.choose())
```

## Anmerkungen

- Im Gegensatz zu `read.table` hat `read.csv` `header = TRUE` und verwendet die erste Zeile als Spaltennamen.

- Alle diese Funktionen werden Strings konvertieren `factor` Klasse standardmäßig , es sei denn entweder `as.is = TRUE` oder `stringsAsFactors = FALSE` .
- Die `read.csv2` Variante verwendet standardmäßig `sep = ";"` und `dec = ","` für Daten aus Ländern, in denen das Komma als Dezimalpunkt und das Semikolon als Feldtrennzeichen verwendet wird.

## Importieren mit Paketen

Die `readr` Paket `read_csv` Funktion bietet viel schnellere Leistung, einen Fortschrittsbalken für große Dateien und populäreren Standardoptionen als Standard `read.csv` , einschließlich

`stringsAsFactors = FALSE` .

```
library(readr)

df <- read_csv(csv_path)

df
A tibble: 4 x 2
Var1 Var2
<dbl> <chr>
1 2.70 A
2 3.14 B
3 10.00 A
4 -7.00 A
```

## Importieren mit `data.table`

Das Paket `data.table` führt die Funktion `fread` . Während es ähnlich ist `read.table` , `fread` ist in der Regel schneller und flexibler, automatisch die Datei der Begrenzungs erraten.

```
get the file path of a CSV included in R's utils package
csv_path <- system.file("misc", "exDIF.csv", package = "utils")

path will vary based on R installation location
csv_path
[1] "/Library/Frameworks/R.framework/Resources/library/utils/misc/exDIF.csv"

dt <- fread(csv_path)

dt
Var1 Var2
1: 2.70 A
2: 3.14 B
3: 10.00 A
4: -7.00 A
```

Wo Argument `input` ist eine Zeichenfolge , die:

- der Dateiname ( zB `"filename.csv"` ),
- ein Shell-Befehl, der auf eine Datei wirkt ( z. B. `"grep 'word' filename"` ), oder
- der Eingang selbst ( zB `"input1, input2 \n A, B \n C, D"` ).

`fread` gibt ein Objekt der Klasse `data.table`, die von Klasse erbt `data.frame` mit der `data.table` der Nutzung, geeignet für die Verwendung `[]`. Um ein gewöhnliches `data.frame` zurückzugeben, setzen Sie den `data.table`-Parameter auf `FALSE`:

```
df <- fread(csv_path, data.table = FALSE)

class(df)
[1] "data.frame"

df
Var1 Var2
1 2.70 A
2 3.14 B
3 10.00 A
4 -7.00 A
```

## Anmerkungen

- `fread` hat nicht alle Optionen wie `read.table`. Ein fehlendes Argument ist `na.comment` kann zu unerwünschtem Verhalten führen, wenn die Quelldatei `#` enthält.
- `fread` verwendet nur `"` für `quote` Parameter.
- `fread` verwendet einige (5) Zeilen, um Variablentypen zu erraten.

## .Tsv-Dateien als Matrizen importieren (Basis-R)

Viele Leute verwenden `file.path` nicht, wenn Sie den Pfad zu einer Datei erstellen. Wenn Sie mit Windows-, Mac- und Linux-Computern arbeiten, empfiehlt es sich normalerweise, Pfade anstelle von `paste`.

```
FilePath <- file.path(AVariableWithFullProjectPath, "SomeSubfolder", "SomeFileName.txt.gz")

Data <- as.matrix(read.table(FilePath, header=FALSE, sep = "\t"))
```

Im Allgemeinen reicht dies für die meisten Menschen aus.

Es kommt manchmal vor, dass die Matrixabmessungen so groß sind, dass die Prozedur der Speicherzuordnung beim Einlesen der Matrix berücksichtigt werden muss, was bedeutet, dass die Matrix Zeile für Zeile eingelesen wird.

Nehmen Sie das vorherige Beispiel. In diesem Fall enthält `FilePath` eine Datei der Dimension `8970 8970` wobei 79% der Zellen Werte enthalten, die nicht Null sind.

```
system.time(expr=Data<-as.matrix(read.table(file=FilePath,header=FALSE,sep=" ")))
```

`system.time` dass 267 Sekunden zum Lesen der Datei benötigt wurden.

```
 user system elapsed
265.563 1.949 267.563
```

Ebenso kann diese Datei Zeile für Zeile gelesen werden.

```
FilePath <- "SomeFile"
connection<- gzfile(FilePath,open="r")
TableList <- list()
Counter <- 1
system.time(expr= while (length(Vector<-as.matrix(scan(file=connection, sep=" ", nlines=1,
quiet=TRUE))) > 0) {
 TableList[[Counter]]<-Vector
 Counter<-Counter+1
})
 user system elapsed
165.976 0.060 165.941
close(connection)
system.time(expr=(Data <- do.call(rbind,TableList)))
 user system elapsed
0.477 0.088 0.565
```

Es gibt auch das Paket `futile.matrix`, das eine `read.matrix` Methode implementiert. Der Code selbst enthüllt sich als das gleiche wie in Beispiel 1 beschrieben.

## .Csv-Dateien exportieren

---

# Export mit Basis R

Daten können mit `write.csv()` in eine CSV-Datei geschrieben werden:

```
write.csv(mtcars, "mtcars.csv")
```

Zu den `row.names = FALSE` angegebenen Parametern gehören `row.names = FALSE` und `na = ""`.

---

# Exportieren mit Paketen

`readr::write_csv` ist deutlich schneller als `write.csv` und schreibt keine `write.csv`.

```
library(readr)

write_csv(mtcars, "mtcars.csv")
```

## Importieren Sie mehrere CSV-Dateien

```
files = list.files(pattern="*.csv")
data_list = lapply(files, read.table, header = TRUE)
```

Dies liest jede Datei und fügt sie einer Liste hinzu. Wenn alle `data.frame` danach dieselbe Struktur haben, können sie zu einem großen `data.frame` kombiniert werden:

```
df <- do.call(rbind, data_list)
```

## Dateien mit fester Breite importieren

Feste Breite Dateien, Textdateien, in denen Spalten von keinen Zeichenbegrenzer getrennt ist, wie, oder;, sondern haben eine feste Zeichenlänge (*Breite*). Daten werden normalerweise mit Leerzeichen aufgefüllt.

Ein Beispiel:

| Column1 | Column2 | Column3           | Column4  | Column5 |
|---------|---------|-------------------|----------|---------|
| 1647    | pi      | 'important'       | 3.141596 | 28318   |
| 1731    | euler   | 'quite important' | 2.718285 | 43656   |
| 1979    | answer  | 'The Answer.'     | 42       | 42      |

Nehmen wir an, diese Datentabelle befindet sich in der lokalen Datei `constants.txt` im Arbeitsverzeichnis.

## Import mit Basis R

```
df <- read.fwf('constants.txt', widths = c(8,10,18,7,8), header = FALSE, skip = 1)
```

```
df
#> V1 V2 V3 V4 V5
#> 1 1647 pi 'important' 3.14159 6.28318
#> 2 1731 euler 'quite important' 2.71828 5.43656
#> 3 1979 answer 'The Answer.' 42 42.0000
```

Hinweis:

- `Column4Column5` müssen nicht durch ein Zeichen getrennt werden (`Column4Column5`)
- Der `widths` Parameter definiert die Breite jeder Spalte
- Nicht getrennte Header können mit `read.fwf()` nicht `read.fwf()`

## Importieren mit readr

```
library(readr)

df <- read_fwf('constants.txt',
 fwf_cols(Year = 8, Name = 10, Importance = 18, Value = 7, Doubled = 8),
 skip = 1)

df
#> # A tibble: 3 x 5
#> Year Name Importance Value Doubled
#> <int> <chr> <chr> <dbl> <dbl>
#> 1 1647 pi 'important' 3.14159 6.28318
#> 2 1731 euler 'quite important' 2.71828 5.43656
```

```
#> 3 1979 answer 'The Answer.' 42.00000 42.00000
```

Hinweis:

- Die Hilfsfunktionen `fwf_*` von `fwf_*` bieten alternative Möglichkeiten zum Angeben von Spaltenlängen, einschließlich des automatischen `fwf_empty` (`fwf_empty`).
- `readr` ist schneller als Basis R
- Spaltentitel können nicht automatisch aus der Datendatei importiert werden

Lesen und Schreiben von Tabellendaten in Klartextdateien (CSV, TSV usw.) online lesen:

<https://riptutorial.com/de/r/topic/481/lesen-und-schreiben-von-tabellendaten-in-klartextdateien--csv--tsv-usw-->

# Kapitel 66: Lineare Modelle (Regression)

## Syntax

- Im (Formel, Daten, Teilmenge, Gewichte, na.Aktion, Methode = "qr", Modell = TRUE, x = FALSCH, y = FALSE, qr = TRUE, Singular.ok = TRUE, Kontraste = NULL, Offset, .. )

## Parameter

| Parameter   | Bedeutung                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Formel      | eine Formel in der <i>Wilkinson-Rogers</i> - Notation; <code>response ~ ...</code> wo ... enthält Begriffe auf Variablen in der Umgebung entsprechen , oder in dem Datenrahmen von dem angegebenen <code>data</code>                                                                                                                                                         |
| Daten       | Datenrahmen, der die Antwort- und Prädiktorvariablen enthält                                                                                                                                                                                                                                                                                                                 |
| Teilmenge   | ein Vektor, der eine Teilmenge der zu verwendenden Beobachtungen angibt: Kann als logische Anweisung in Bezug auf die Variablen in <code>data</code> ausgedrückt <code>data</code>                                                                                                                                                                                           |
| Gewichte    | analytische Gewichte (siehe Abschnitt <i>Gewichte</i> oben)                                                                                                                                                                                                                                                                                                                  |
| na.aktion   | wie mit fehlenden ( <code>NA</code> ) Werten <code>?na.action</code> : siehe <code>?na.action</code>                                                                                                                                                                                                                                                                         |
| Methode     | wie man die Anpassung durchführt Nur Auswahlmöglichkeiten sind "qr" oder "model.frame" (letztere gibt den Modellrahmen ohne Anpassung an das Modell zurück, identisch mit der Angabe von <code>model=TRUE</code> ).                                                                                                                                                          |
| Modell-     | ob der Modellrahmen im angepassten Objekt gespeichert werden soll                                                                                                                                                                                                                                                                                                            |
| x           | ob die Modellmatrix im angepassten Objekt gespeichert werden soll                                                                                                                                                                                                                                                                                                            |
| y           | ob die Modellantwort im angepassten Objekt gespeichert werden soll                                                                                                                                                                                                                                                                                                           |
| qr          | ob die QR-Zerlegung im angepassten Objekt gespeichert werden soll                                                                                                                                                                                                                                                                                                            |
| singular.ok | ob <i>Singularanpassungen zulässig sind</i> , Modelle mit kollinearen Prädiktoren (eine Teilmenge der Koeffizienten wird in diesem Fall automatisch auf <code>NA</code>                                                                                                                                                                                                      |
| Kontraste   | eine Liste von Kontrasten, die für bestimmte Faktoren im Modell verwendet werden sollen; Siehe das Argument <code>contrasts.arg</code> von <code>?model.matrix.default</code> . Kontraste können auch mit <code>options()</code> (siehe Argumente für <code>contrasts</code> ) oder durch Zuweisen der <code>contrast</code> eines Faktors (siehe <code>?contrasts</code> ). |
| Versatz     | Wird verwendet, um eine <i>a priori</i> bekannte Komponente im Modell anzugeben. Kann auch als Teil der Formel angegeben werden. Siehe <code>?model.offset</code>                                                                                                                                                                                                            |



| Parameter | Bedeutung                                                                                                                                       |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| ...       | zusätzliche Argumente, die an untergeordnete Anpassungsfunktionen übergeben werden sollen ( <code>lm.fit()</code> oder <code>lm.wfit()</code> ) |

## Examples

### Lineare Regression im mtcars-Dataset

Der eingebaute in mtcars [Datenrahmen](#) enthält Informationen über 32 Autos, einschließlich ihres Gewichts, ihrer Kraftstoffeffizienz (in Meilen pro Gallone), Geschwindigkeit usw. (Um mehr über den Datensatz, Verwendung finden heraus `help(mtcars)` ).

Wenn wir an der Beziehung zwischen Kraftstoffeffizienz ( `mpg` ) und Gewicht ( `wt` ) interessiert sind, beginnen wir diese Variablen mit:

```
plot(mpg ~ wt, data = mtcars, col=2)
```

Die Diagramme zeigen eine (lineare) Beziehung! Wenn wir dann die lineare Regression durchführen wollen, um die Koeffizienten eines linearen Modells zu bestimmen, würden wir die Funktion `lm` verwenden:

```
fit <- lm(mpg ~ wt, data = mtcars)
```

Das `~` bedeutet hier "erklärt durch", also bedeutet die Formel `mpg ~ wt` dass wir mpg voraussagen, wie es durch wt erklärt wird. Der hilfreichste Weg zum Anzeigen der Ausgabe ist mit:

```
summary(fit)
```

Was gibt die Ausgabe:

```
Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
 Min 1Q Median 3Q Max
-4.5432 -2.3647 -0.1252 1.4096 6.8727

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.2851 1.8776 19.858 < 2e-16 ***
wt -5.3445 0.5591 -9.559 1.29e-10 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared: 0.7528, Adjusted R-squared: 0.7446
F-statistic: 91.38 on 1 and 30 DF, p-value: 1.294e-10
```

Hier finden Sie Informationen zu:

- Die geschätzte Steigung jedes Koeffizienten (  $wt$  und  $y$ -  $37.2851 + (-5.3445) * wt$  ), die auf die best-fit-Vorhersage von  $mpg$  schließen lässt, beträgt  $37.2851 + (-5.3445) * wt$
- Der p-Wert jedes Koeffizienten legt nahe, dass der Achsenabschnitt und das Gewicht wahrscheinlich nicht zufällig sind
- Gesamteinschätzungen der Anpassung wie  $R^2$  und bereinigtes  $R^2$ , die zeigen, wie viel von der Variation von  $mpg$  durch das Modell erklärt wird

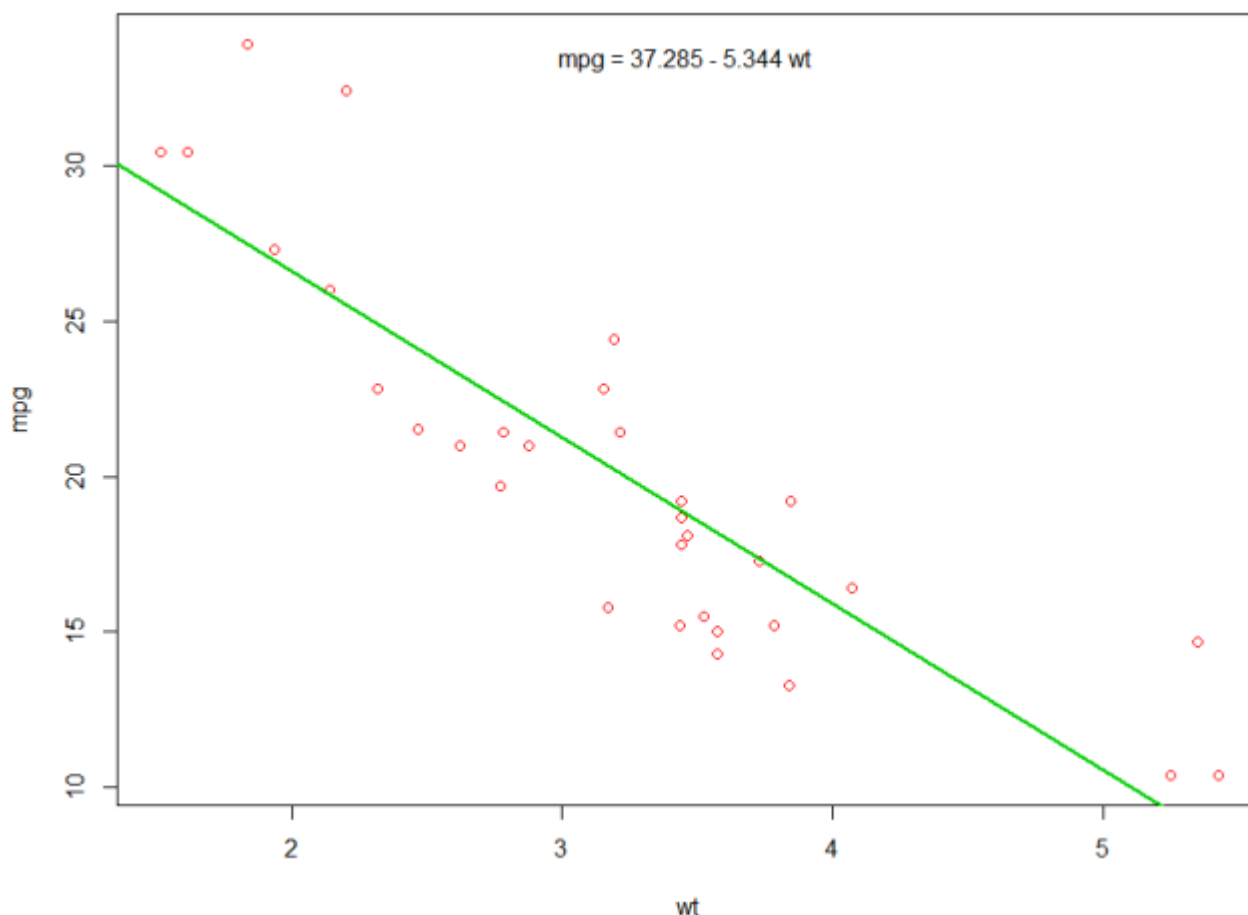
Wir könnten unserem ersten Plot eine Linie hinzufügen, um das vorhergesagte  $mpg$  :

```
abline(fit,col=3,lwd=2)
```

Es ist auch möglich, die Gleichung zu diesem Diagramm hinzuzufügen. `coef` zuerst die Koeffizienten mit `coef` . Mit `paste0` wir die Koeffizienten mit geeigneten Variablen und  $+/-$  , um die Gleichung `paste0` . Zum Schluss fügen wir es mit `mttext` zum Plot:

```
bs <- round(coef(fit), 3)
lmlab <- paste0("mpg = ", bs[1],
 ifelse(sign(bs[2])==1, " + ", " - "), abs(bs[2]), " wt ")
mttext(lmlab, 3, line=-2)
```

Das Ergebnis ist:



## Die Regression plotten (Basis)

`mtcars` dem `mtcars` Beispiel fort. Hier können Sie auf einfache Weise eine `mtcars` Darstellung Ihrer

linearen Regression erstellen, die möglicherweise für die Veröffentlichung geeignet ist.

Passen Sie zuerst das lineare Modell und an

```
fit <- lm(mpg ~ wt, data = mtcars)
```

Plotten Sie dann die beiden interessierenden Variablen und fügen Sie die Regressionslinie innerhalb der Definitionsdomäne hinzu:

```
plot(mtcars$wt,mtcars$mpg,pch=18, xlab = 'wt',ylab = 'mpg')
lines(c(min(mtcars$wt),max(mtcars$wt)),
as.numeric(predict(fit, data.frame(wt=c(min(mtcars$wt),max(mtcars$wt))))))
```

Fast dort! Der letzte Schritt ist das Hinzufügen der Regressionsgleichung, des Rechtecks sowie des Korrelationskoeffizienten. Dies geschieht mit der `vector` :

```
rp = vector('expression',3)
rp[1] = substitute(expression(italic(y) == MYOTHERVALUE3 + MYOTHERVALUE4 %*% x),
list(MYOTHERVALUE3 = format(fit$coefficients[1], digits = 2),
MYOTHERVALUE4 = format(fit$coefficients[2], digits = 2))) [2]
rp[2] = substitute(expression(italic(R)^2 == MYVALUE),
list(MYVALUE = format(summary(fit)$adj.r.squared,dig=3))) [2]
rp[3] = substitute(expression(Pearson-R == MYOTHERVALUE2),
list(MYOTHERVALUE2 = format(cor(mtcars$wt,mtcars$mpg), digits = 2))) [2]

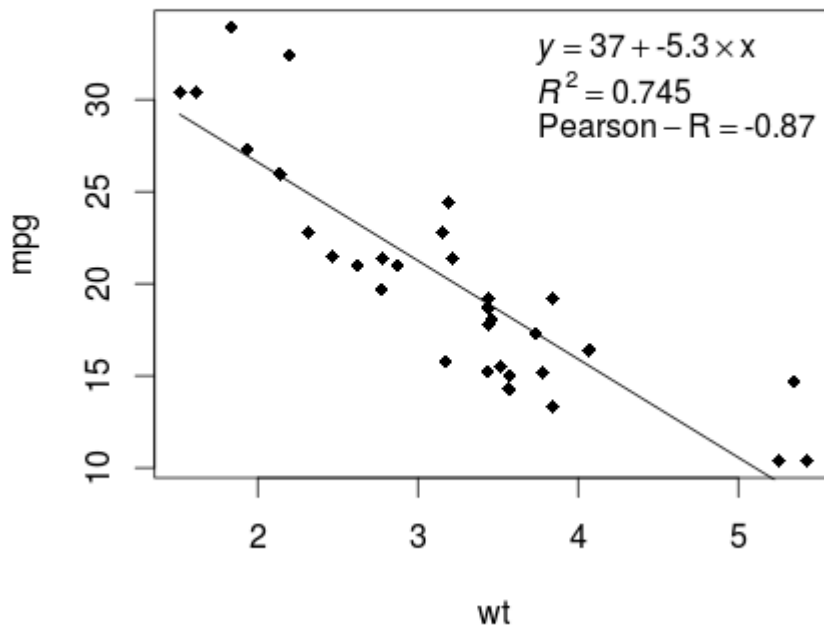
legend("topright", legend = rp, bty = 'n')
```

Beachten Sie, dass Sie durch Anpassen der Vektorfunktion beliebige andere Parameter wie RMSE hinzufügen können. Stellen Sie sich vor, Sie möchten eine Legende mit 10 Elementen. Die Vektordefinition wäre die folgende:

```
rp = vector('expression',10)
```

und Sie müssen `r[1]` .... bis `r[10]`

Hier ist die Ausgabe:



## Gewichtung

Manchmal möchten wir, dass das Modell einigen Datenpunkten oder Beispielen mehr Gewicht verleiht als anderen. Dies ist möglich, indem Sie das Gewicht für die Eingabedaten angeben, während Sie das Modell lernen. Es gibt im Allgemeinen zwei Arten von Szenarien, in denen wir in den Beispielen möglicherweise nicht einheitliche Gewichte verwenden:

- **Analytische Gewichte:** Reflektieren Sie die unterschiedlichen Genauigkeitsgrade verschiedener Beobachtungen. Wenn zum Beispiel Daten analysiert werden, bei denen jede Beobachtung die Durchschnittsergebnisse aus einem geografischen Gebiet ist, ist das Analysegewicht proportional zum Inversen der geschätzten Varianz. Nützlich beim Umgang mit Durchschnittswerten in Daten durch Angabe eines proportionalen Gewichts aufgrund der Anzahl der Beobachtungen. [Quelle](#)
- **Stichprobengewichte (inverse Wahrscheinlichkeitsgewichte - IPW):** Eine statistische Methode zur Berechnung von Statistiken, die auf eine Grundgesamtheit normiert sind, die sich von derjenigen unterscheidet, in der die Daten erfasst wurden. Studiendesigns mit einer unterschiedlichen Stichprobenpopulation und einer Zielpopulation (Zielpopulation) sind häufig in der Anwendung. Nützlich bei Daten mit fehlenden Werten. [Quelle](#)

Die Funktion `lm()` führt eine analytische Gewichtung durch. Für Stichprobengewichte wird das `survey` verwendet, um ein Vermessungsobjekt zu erstellen und `svyglm()`. Standardmäßig verwendet das `survey` Stichprobengewichte. (ANMERKUNG: `lm()` und `svyglm()` mit der Familie `gaussian()`. Werden alle die gleichen Punktschätzungen zu produzieren, weil sie sowohl für die Koeffizienten zu lösen, indem die gewichteten kleinsten Quadrate zu minimieren Sie unterscheiden sich darin, wie Standardfehler berechnet.)

## Testdaten

```
data <- structure(list(lexptot = c(9.1595012302023, 9.86330744180814,
8.92372556833205, 8.58202430280175, 10.1133857229336), progwillm = c(1L,
1L, 1L, 1L, 0L), sexhead = c(1L, 1L, 0L, 1L, 1L), agehead = c(79L,
43L, 52L, 48L, 35L), weight = c(1.04273509979248, 1.01139605045319,
1.01139605045319, 1.01139605045319, 0.76305216550827)), .Names = c("lexptot",
"progwillm", "sexhead", "agehead", "weight"), class = c("tbl_df",
"tbl", "data.frame"), row.names = c(NA, -5L))
```

## Analytische Gewichte

```
lm.analytic <- lm(lexptot ~ progwillm + sexhead + agehead,
 data = data, weight = weight)
summary(lm.analytic)
```

## Ausgabe

```
Call:
lm(formula = lexptot ~ progwillm + sexhead + agehead, data = data,
 weights = weight)

Weighted Residuals:
 1 2 3 4 5
9.249e-02 5.823e-01 0.000e+00 -6.762e-01 -1.527e-16

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.016054 1.744293 5.742 0.110
progwillm -0.781204 1.344974 -0.581 0.665
sexhead 0.306742 1.040625 0.295 0.818
agehead -0.005983 0.032024 -0.187 0.882

Residual standard error: 0.8971 on 1 degrees of freedom
Multiple R-squared: 0.467, Adjusted R-squared: -1.132
F-statistic: 0.2921 on 3 and 1 DF, p-value: 0.8386
```

## Probenahmegewichte (IPW)

```
library(survey)
data$X <- 1:nrow(data) # Create unique id

Build survey design object with unique id, ipw, and data.frame
des1 <- svydesign(id = ~X, weights = ~weight, data = data)

Run glm with survey design object
prog.lm <- svyglm(lexptot ~ progwillm + sexhead + agehead, design=des1)
```

## Ausgabe

```
Call:
svyglm(formula = lexptot ~ progwillm + sexhead + agehead, design = des1)

Survey design:
```

```
svydesign(id = ~X, weights = ~weight, data = data)

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.016054 0.183942 54.452 0.0117 *
progvillm -0.781204 0.640372 -1.220 0.4371
sexhead 0.306742 0.397089 0.772 0.5813
agehead -0.005983 0.014747 -0.406 0.7546

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.2078647)

Number of Fisher Scoring iterations: 2
```

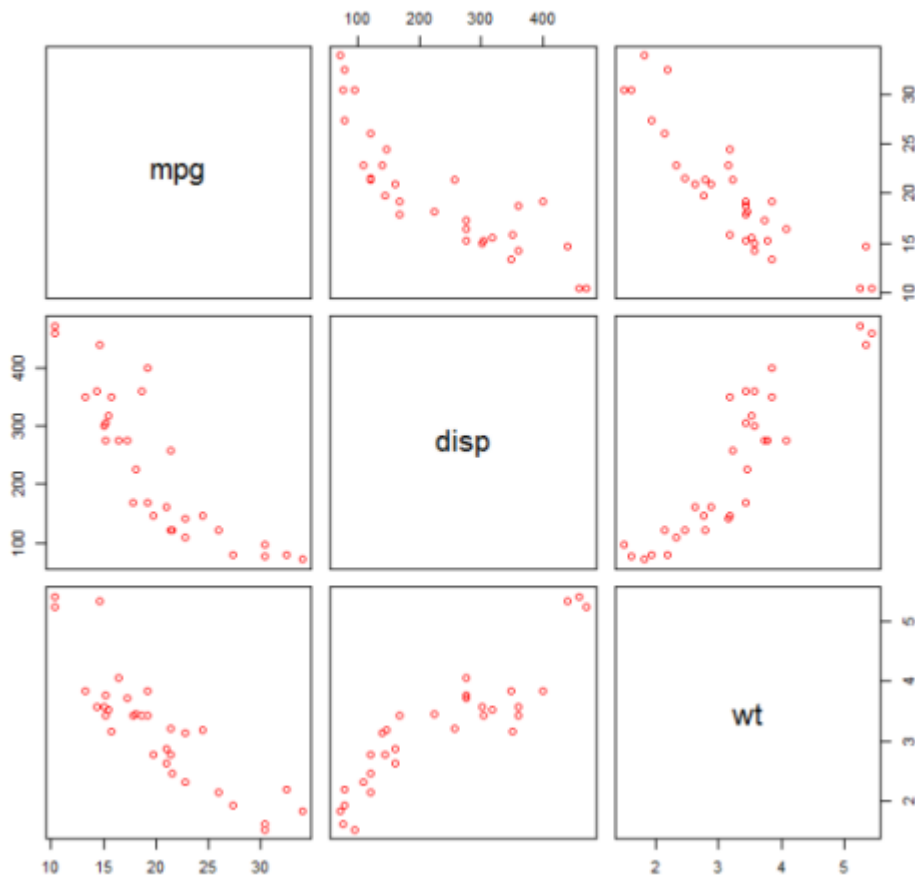
## Prüfung auf Nichtlinearität mit polynomialer Regression

Wenn Sie mit linearer Regression arbeiten, müssen wir manchmal die Nichtlinearität der Daten überprüfen. Eine Möglichkeit, dies zu tun, besteht darin, ein Polynommodell anzupassen und zu prüfen, ob es besser zu den Daten passt als ein lineares Modell. Es gibt andere Gründe, z. B. theoretische Gründe, die auf ein quadratisches oder ein Modell höherer Ordnung schließen lassen, da davon ausgegangen wird, dass die Variablenbeziehung von Natur aus polynomisch ist.

Passen Sie ein quadratisches Modell für das `mtcars` Dataset an. Für ein lineares Modell siehe [Lineare Regression im mtcars-Dataset](#).

Zuerst `disp` wir ein Streudiagramm der Variablen `mpg` (Meilen / Gallone), `disp` (Displacement (cu.in.)) und `wt` (Weight (1000 lbs)). Die Beziehung zwischen `mpg` und `disp` erscheint nicht linear.

```
plot(mtcars[,c("mpg", "disp", "wt")])
```



Eine lineare Anpassung zeigt, dass `disp` nicht signifikant ist.

```
fit0 = lm(mpg ~ wt+disp, mtcars)
summary(fit0)

Coefficients:
Estimate Std. Error t value Pr(>|t|)
#(Intercept) 34.96055 2.16454 16.151 4.91e-16 ***
#wt -3.35082 1.16413 -2.878 0.00743 **
#disp -0.01773 0.00919 -1.929 0.06362 .
#---
#Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#Residual standard error: 2.917 on 29 degrees of freedom
#Multiple R-squared: 0.7809, Adjusted R-squared: 0.7658
```

Um das Ergebnis eines quadratischen Modells zu erhalten, haben wir `I(disp^2)` hinzugefügt. Das neue Modell erscheint bei Betrachtung von  $R^2$  besser und alle Variablen sind signifikant.

```
fit1 = lm(mpg ~ wt+disp+I(disp^2), mtcars)
summary(fit1)

Coefficients:
Estimate Std. Error t value Pr(>|t|)
#(Intercept) 41.4019837 2.4266906 17.061 2.5e-16 ***
#wt -3.4179165 0.9545642 -3.581 0.001278 **
#disp -0.0823950 0.0182460 -4.516 0.000104 ***
#I(disp^2) 0.0001277 0.0000328 3.892 0.000561 ***
#---
```

```
#Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#Residual standard error: 2.391 on 28 degrees of freedom
#Multiple R-squared: 0.8578, Adjusted R-squared: 0.8426
```

Da wir drei Variablen haben, ist das angepasste Modell eine Fläche, die dargestellt wird durch:

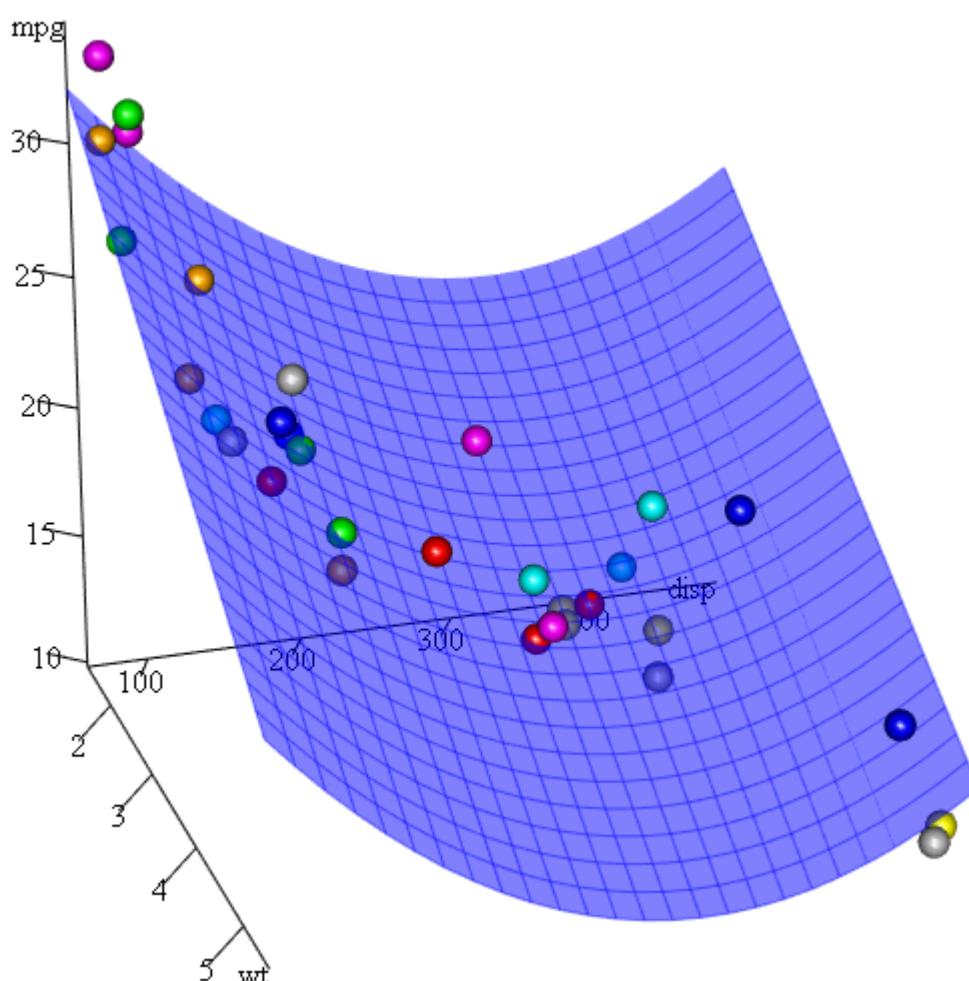
```
mpg = 41.4020 - 3.4179*wt - 0.0824*disp + 0.0001277*disp^2
```

Eine andere Möglichkeit, die Regression von Polynomen anzugeben, ist die Verwendung von `poly` mit dem Parameter `raw=TRUE`. Andernfalls werden *orthogonale Polynome* berücksichtigt (weitere Informationen finden Sie in der `help(poly)`). Wir erhalten das gleiche Ergebnis mit:

```
summary(lm(mpg ~ wt + poly(disp, 2, raw=TRUE), mtcars))
```

Was ist schließlich, wenn wir eine Darstellung der geschätzten Oberfläche anzeigen müssen? Es gibt viele Möglichkeiten, 3D Darstellungen in R erstellen. Hier verwenden wir `Fit3d` aus dem `p3d` Paket.

```
library(p3d)
Init3d(family="serif", cex = 1)
Plot3d(mpg ~ disp + wt, mtcars)
Axes3d()
Fit3d(fit1)
```

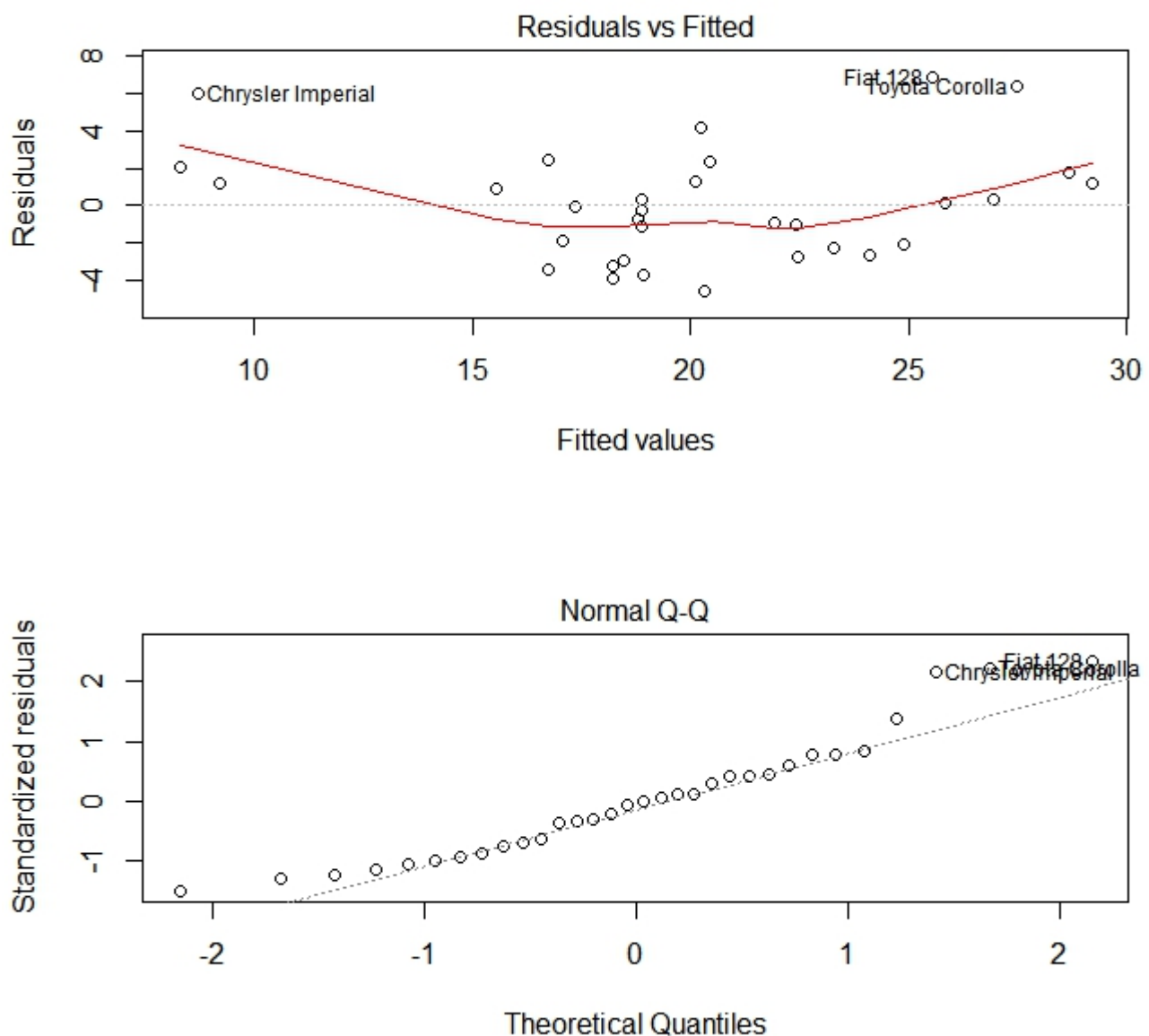




## Qualitätsprüfung

Nach dem Erstellen eines Regressionsmodells ist es wichtig, das Ergebnis zu überprüfen und zu entscheiden, ob das Modell geeignet ist und mit den vorliegenden Daten gut funktioniert. Dies kann durch Untersuchen des Residuendiagramms sowie anderer Diagnoseplots erfolgen.

```
fit the model
fit <- lm(mpg ~ wt, data = mtcars)
#
par(mfrow=c(2,1))
plot model object
plot(fit, which = 1:2)
```



Diese Diagramme überprüfen zwei Annahmen, die beim Erstellen des Modells getroffen wurden:

1. Dass der erwartete Wert der vorhergesagten Variablen (in diesem Fall `mpg`) durch eine

Linearkombination der Prädiktoren (in diesem Fall `wt`) gegeben ist. Wir erwarten, dass diese Einschätzung unvoreingenommen ist. Daher sollten die Residuen um den Mittelwert für alle Werte der Prädiktoren zentriert sein. In diesem Fall sehen wir, dass die Residuen an den Enden tendenziell positiv und in der Mitte negativ sind, was auf eine nichtlineare Beziehung zwischen den Variablen schließen lässt.

2. Die tatsächliche vorhergesagte Variable ist normalerweise um ihre Schätzung verteilt. Daher sollten die Residuen normal verteilt sein. Bei normalverteilten Daten sollten die Punkte in einem normalen QQ-Diagramm auf oder nahe der Diagonale liegen. An den Enden ist etwas Schräglauf vorhanden.

## Verwenden der Funktion "Vorhersagen"

Sobald ein Modell erstellt ist `predict` ist `predict` die Hauptfunktion zum Testen mit neuen Daten. In unserem Beispiel wird der eingebaute Datensatz von `mtcars`, um Meilen pro Gallone gegen Verschiebung zu `mtcars`:

```
my_mdl <- lm(mpg ~ disp, data=mtcars)
my_mdl

Call:
lm(formula = mpg ~ disp, data = mtcars)

Coefficients:
(Intercept) disp
 29.59985 -0.04122
```

Wenn ich eine neue Datenquelle mit Verschiebung hätte, könnte ich die geschätzten Meilen pro Gallone sehen.

```
set.seed(1234)
newdata <- sample(mtcars$disp, 5)
newdata
[1] 258.0 71.1 75.7 145.0 400.0

newdf <- data.frame(disp=newdata)
predict(my_mdl, newdf)
 1 2 3 4 5
18.96635 26.66946 26.47987 23.62366 13.11381
```

Der wichtigste Teil des Prozesses ist das Erstellen eines neuen Datenrahmens mit den gleichen Spaltennamen wie die Originaldaten. In diesem Fall hatten die ursprünglichen Daten eine Spalte mit der Bezeichnung `disp`. Ich war sicher, die neuen Daten denselben Namen zu nennen.

## Vorsicht

Schauen wir uns ein paar häufige Fallstricke an:

1. kein `data.frame` im neuen Objekt verwenden:

```
predict(my_mdl, newdata)
Error in eval(predvars, data, env) :
```

```
numeric 'envir' arg not of length one
```

## 2. In neuen Datenrahmen nicht dieselben Namen verwenden:

```
newdf2 <- data.frame(newdata)
predict(my_mdl, newdf2)
Error in eval(expr, envir, enclos) : object 'disp' not found
```

## Richtigkeit

Um die Genauigkeit der Vorhersage zu überprüfen, benötigen Sie die tatsächlichen y-Werte der neuen Daten. In diesem Beispiel benötigt `newdf` eine Spalte für 'mpg' und 'disp'.

```
newdf <- data.frame(mpg=mtcars$mpg[1:10], disp=mtcars$disp[1:10])
mpg disp
1 21.0 160.0
2 21.0 160.0
3 22.8 108.0
4 21.4 258.0
5 18.7 360.0
6 18.1 225.0
7 14.3 360.0
8 24.4 146.7
9 22.8 140.8
10 19.2 167.6

p <- predict(my_mdl, newdf)

#root mean square error
sqrt(mean((p - newdf$mpg)^2, na.rm=TRUE))
[1] 2.325148
```

Lineare Modelle (Regression) online lesen: <https://riptutorial.com/de/r/topic/801/lineare-modelle--regression->

# Kapitel 67: Listen

## Examples

### Schnelle Einführung in die Listen

Im Allgemeinen sind die meisten Objekte, mit denen Sie als Benutzer interagieren würden, tendenziell ein Vektor. zB numerischer Vektor, logischer Vektor. Diese Objekte können nur einen einzigen Variablentyp aufnehmen (ein numerischer Vektor kann nur Zahlen enthalten).

Eine Liste kann jede Typvariable darin speichern und sie zu einem generischen Objekt machen, das jeden Typ von Variablen speichern kann, die wir benötigen.

### Beispiel für das Initialisieren einer Liste

```
exampleList1 <- list('a', 'b')
exampleList2 <- list(1, 2)
exampleList3 <- list('a', 1, 2)
```

Um die in der Liste definierten Daten zu verstehen, können wir die str-Funktion verwenden.

```
str(exampleList1)
str(exampleList2)
str(exampleList3)
```

Bei der Teilauswahl von Listen unterscheidet man zwischen dem Extrahieren eines Slice der Liste, dh dem Abrufen einer Liste mit einer Teilmenge der Elemente in der ursprünglichen Liste und dem Extrahieren eines einzelnen Elements. Wenn Sie den Operator `[`, der üblicherweise für Vektoren verwendet wird, wird eine neue Liste erstellt.

```
Returns List
exampleList3[1]
exampleList3[1:2]
```

Um ein einzelnes Element zu erhalten, verwenden Sie stattdessen `[[`.

```
Returns Character
exampleList3[[1]]
```

Listeneinträge können benannt werden:

```
exampleList4 <- list(
 num = 1:3,
 numeric = 0.5,
 char = c('a', 'b')
)
```

Auf die Einträge in benannten Listen kann über ihren Namen statt über ihren Index zugegriffen

werden.

```
exampleList4[['char']]
```

Alternativ kann mit dem Operator `$` auf benannte Elemente zugegriffen werden.

```
exampleList4$num
```

Dies hat den Vorteil, dass es schneller zu tippen ist und möglicherweise einfacher zu lesen ist, aber es ist wichtig, dass Sie sich einer möglichen Fallstricke bewusst sind. Der Operator `$` verwendet partielle Übereinstimmung, um übereinstimmende Listenelemente zu identifizieren, und kann zu unerwarteten Ergebnissen führen.

```
exampleList5 <- exampleList4[2:3]

exampleList4$num
c(1, 2, 3)

exampleList5$num
0.5

exampleList5[['num']]
NULL
```

Listen können besonders nützlich sein, da sie Objekte unterschiedlicher Länge und verschiedener Klassen speichern können.

```
Numeric vector
exampleVector1 <- c(12, 13, 14)
Character vector
exampleVector2 <- c("a", "b", "c", "d", "e", "f")
Matrix
exampleMatrix1 <- matrix(rnorm(4), ncol = 2, nrow = 2)
List
exampleList3 <- list('a', 1, 2)

exampleList6 <- list(
 num = exampleVector1,
 char = exampleVector2,
 mat = exampleMatrix1,
 list = exampleList3
)
exampleList6
#$num
#[1] 12 13 14
#
#$char
#[1] "a" "b" "c" "d" "e" "f"
#
#$mat
[,1] [,2]
#[1,] 0.5013050 -1.88801542
#[2,] 0.4295266 0.09751379
#
#$list
```

```
#$list[[1]]
#[1] "a"

#$list[[2]]
#[1] 1

#$list[[3]]
#[1] 2
```

## Einführung in die Listen

Listen ermöglichen Benutzern das Speichern mehrerer Elemente (wie Vektoren und Matrizen) unter einem einzigen Objekt. Sie können die `list` verwenden, um eine Liste zu erstellen:

```
l1 <- list(c(1, 2, 3), c("a", "b", "c"))
l1
[[1]]
[1] 1 2 3

[[2]]
[1] "a" "b" "c"
```

Beachten Sie, dass die Vektoren, die die obige Liste bilden, unterschiedliche Klassen sind. Listen ermöglichen Benutzern das Gruppieren von Elementen verschiedener Klassen. Jedes Element in einer Liste kann auch einen Namen haben. Auf Listennamen wird von der `names` zugegriffen und in derselben Weise Zeilen- und Spaltennamen in einer Matrix zugewiesen.

```
names(l1)
NULL
names(l1) <- c("vector1", "vector2")
l1
$vector1
[1] 1 2 3

$vector2
[1] "a" "b" "c"
```

Es ist oft einfacher und sicherer, die Listennamen beim Erstellen des Listenobjekts anzugeben.

```
l2 <- list(vec = c(1, 3, 5, 7, 9),
 mat = matrix(data = c(1, 2, 3), nrow = 3))
l2
$vec
[1] 1 3 5 7 9

$mat
[,1]
[1,] 1
[2,] 2
[3,] 3
names(l2)
[1] "vec" "mat"
```

Über der Liste befinden sich zwei Elemente, "vec" und "mat", ein Vektor und eine Matrix.

## Gründe für die Verwendung von Listen

Für den durchschnittlichen R-Benutzer kann die Listenstruktur als eine der komplizierteren zu manipulierenden Datenstrukturen erscheinen. Es gibt keine Garantie dafür, dass alle Elemente des gleichen Typs sind. Es gibt keine garantierte Struktur dafür, wie kompliziert / nicht kompliziert die Liste wäre (Ein Element in einer Liste könnte eine Liste sein)

Einer der Hauptgründe, warum Listen verwendet werden, um Parameter zwischen Funktionen zu übergeben.

```
Function example which returns a single element numeric vector
exampleFunction1 <- function(num1, num2){
 result <- num1 + num2
 return(result)
}

Using example function 1
exampleFunction1(1, 2)

Function example which returns a simple numeric vector
exampleFunction2 <- function(num1, num2, multiplier){
 tempResult1 <- num1 + num2
 tempResult2 <- tempResult1 * multiplier
 result <- c(tempResult1, tempResult2)
 return(result)
}

Using example function 2
exampleFunction2(1, 2, 4)
```

In dem obigen Beispiel handelt es sich bei den zurückgegebenen Ergebnissen nur um einfache numerische Vektoren. Es gibt keine Probleme, über solche einfachen Vektoren hinwegzugehen.

Beachten Sie an dieser Stelle, dass R-Funktionen im Allgemeinen nur jeweils 1 Ergebnis zurückgeben (Sie können if-Bedingungen verwenden, um andere Ergebnisse zurückzugeben). Wenn Sie jedoch eine Funktion erstellen möchten, die einen Parametersatz übernimmt und verschiedene Arten von Ergebnissen zurückgibt, z. B. einen numerischen Vektor (Einstellungswert) und einen Datenrahmen (aus der Berechnung), müssen Sie alle diese Ergebnisse in einer Liste speichern bevor Sie es zurückschicken.

```
We will be using mtcars dataset here
Function which returns a result that is supposed to contain multiple type of results
This can be solved by putting the results into a list
exampleFunction3 <- function(dataframe, removeColumn, sumColumn){
 resultDataFrame <- dataframe[, -removeColumn]
 resultSum <- sum(dataframe[, sumColumn])
 resultList <- list(resultDataFrame, resultSum)
 return(resultList)
}

Using example function 3
exampleResult <- exampleFunction3(mtcars, 2, 4)
exampleResult[[1]]
exampleResult[[2]]
```

## Konvertieren Sie eine Liste in einen Vektor, wobei leere Listenelemente leer bleiben

Wenn Sie eine Liste in ein Vektor- oder data.frame-Objekt konvertieren möchten, werden normalerweise leere Elemente gelöscht.

Dies kann problematisch sein, wenn eine Liste mit einer gewünschten Länge mit einigen leeren Werten erstellt wird (z. B. wird eine Liste mit n Elementen erstellt, die einer mxn-Matrix, data.frame oder data.table hinzugefügt wird). Es ist möglich, eine Liste verlustlos in einen Vektor zu konvertieren, wobei leere Elemente erhalten bleiben:

```
res <- list(character(0), c("Luzhuang", "Laisu", "Peihui"), character(0),
 c("Anjiangping", "Xinzhai", "Yongfeng"), character(0), character(0),
 c("Puji", "Gaotun", "Banjingcun"), character(0), character(0),
 character(0))
res
```

```
[[1]]
character(0)

[[2]]
[1] "Luzhuang" "Laisu" "Peihui"

[[3]]
character(0)

[[4]]
[1] "Anjiangping" "Xinzhai" "Yongfeng"

[[5]]
character(0)

[[6]]
character(0)

[[7]]
[1] "Puji" "Gaotun" "Banjingcun"

[[8]]
character(0)

[[9]]
character(0)

[[10]]
character(0)
```

```
res <- sapply(res, function(s) if (length(s) == 0) NA_character_ else paste(s, collapse = "
"))
res
```

```
[1] NA "Luzhuang Laisu Peihui" NA
"Anjiangping Xinzhai Yongfeng" NA
```



|        |                          |    |
|--------|--------------------------|----|
| [6] NA | "Puji Gaotun Banjingcun" | NA |
| NA     | NA                       |    |

## Serialisierung: Verwenden von Listen zur Weitergabe von Informationen

Es gibt Fälle, in denen Daten unterschiedlichen Typs zusammengestellt werden müssen. In Azure ML müssen beispielsweise Informationen aus einem R-Skriptmodul ausschließlich über Datenrahmen an ein anderes übergeben werden. Angenommen, wir haben einen Datenrahmen und eine Nummer:

```
> df
 name height team fun_index title age desc Y
1 Andrea 195 Lazio 97 6 33 eccellente 1
2 Paja 165 Fiorentina 87 6 31 deciso 1
3 Roro 190 Lazio 65 6 28 strano 0
4 Gioele 70 Lazio 100 0 2 simpatico 1
5 Cacio 170 Juventus 81 3 33 duro 0
6 Edola 171 Lazio 72 5 32 svampito 1
7 Salami 175 Inter 75 3 30 doppiopasso 1
8 Braugo 180 Inter 79 5 32 gjn 0
9 Benna 158 Juventus 80 6 28 esaurito 0
10 Riggio 182 Lazio 92 5 31 certezza 1
11 Giordano 185 Roma 79 5 29 buono 1

> number <- "42"
```

Wir können auf diese Informationen zugreifen:

```
> paste(df$name[4], "is a", df3$team[4], "supporter.")
[1] "Gioele is a Lazio supporter."
> paste("The answer to THE question is", number)
[1] "The answer to THE question is 42"
```

Um verschiedene Datentypen in einem Datenrahmen zu speichern, müssen wir das Listenobjekt und die Serialisierung verwenden. Insbesondere müssen wir die Daten in eine generische Liste einfügen und dann die Liste in einen bestimmten Datenrahmen einfügen:

```
l <- list(df, number)
dataframe_container <- data.frame(out2 = as.integer(serialize(l, connection=NULL)))
```

Nachdem wir die Informationen im Datenrahmen gespeichert haben, müssen wir sie deserialisieren, um sie verwenden zu können:

```
#----- unserialize -----+
unser_obj <- unserialize(as.raw(dataframe_container$out2))
#----- taking back the elements-----+
df_mod <- unser_obj[1][[1]]
number_mod <- unser_obj[2][[1]]
```

Dann können wir überprüfen, ob die Daten korrekt übertragen werden:

```
> paste(df_mod$name[4], "is a", df_mod$team[4], "supporter.")
[1] "Gioele is a Lazio supporter."
> paste("The answer to THE question is", number_mod)
[1] "The answer to THE question is 42"
```

Listen online lesen: <https://riptutorial.com/de/r/topic/1365/listen>

# Kapitel 68: Lösen von ODEs in R

## Syntax

- ode (y, times, func, parms, method, ...)

## Parameter

| Parameter | Einzelheiten                                                                                   |
|-----------|------------------------------------------------------------------------------------------------|
| y         | (benannter) numerischer Vektor: die ursprünglichen (Zustands-) Werte für das ODE-System        |
| mal       | Zeitfolge, für die Ausgabe gewünscht wird; Der erste Wert der Zeiten muss die Anfangszeit sein |
| func      | Name der Funktion, die die Werte der Ableitungen im ODE-System berechnet                       |
| Parms     | (benannter) numerischer Vektor: an func übergebene Parameter                                   |
| Methode   | Der zu verwendende Integrator ist standardmäßig: lsoda                                         |

## Bemerkungen

Beachten Sie, dass die Änderungsrate in derselben Reihenfolge wie die Angabe der Zustandsvariablen zurückgegeben werden muss. Im Beispiel "Das Lorenz-Modell" bedeutet dies, dass in der Funktion "Lorenz" der Befehl steht

```
return(list(c(dX, dY, dZ)))
```

hat die gleiche Reihenfolge wie die Definition der Zustandsvariablen

```
yini <- c(X = 1, Y = 1, Z = 1)
```

## Examples

### Das Lorenz-Modell

Das Lorenz-Modell beschreibt die Dynamik der drei Zustandsvariablen X, Y und Z. Die Modellgleichungen sind:

$$\frac{dX}{dt} = a \cdot X + Y \cdot Z$$

$$\frac{dY}{dt} = b \cdot (Y - Z)$$

$$\frac{dZ}{dT} = -X \cdot Y + c \cdot Y - Z$$

Die Anfangsbedingungen sind:

$$X(0) = Y(0) = Z(0) = 1$$

und a, b und c sind drei Parameter mit

$$a = -8/3$$

$$b = -10$$

$$c = 28$$

```
library(deSolve)

Define R-function

Lorenz <- function (t, y, parms) {
 with(as.list(c(y, parms)), {
 dX <- a * X + Y * Z
 dY <- b * (Y - Z)
 dZ <- -X * Y + c * Y - Z

 return(list(c(dX, dY, dZ)))
 })
}

Define parameters and variables

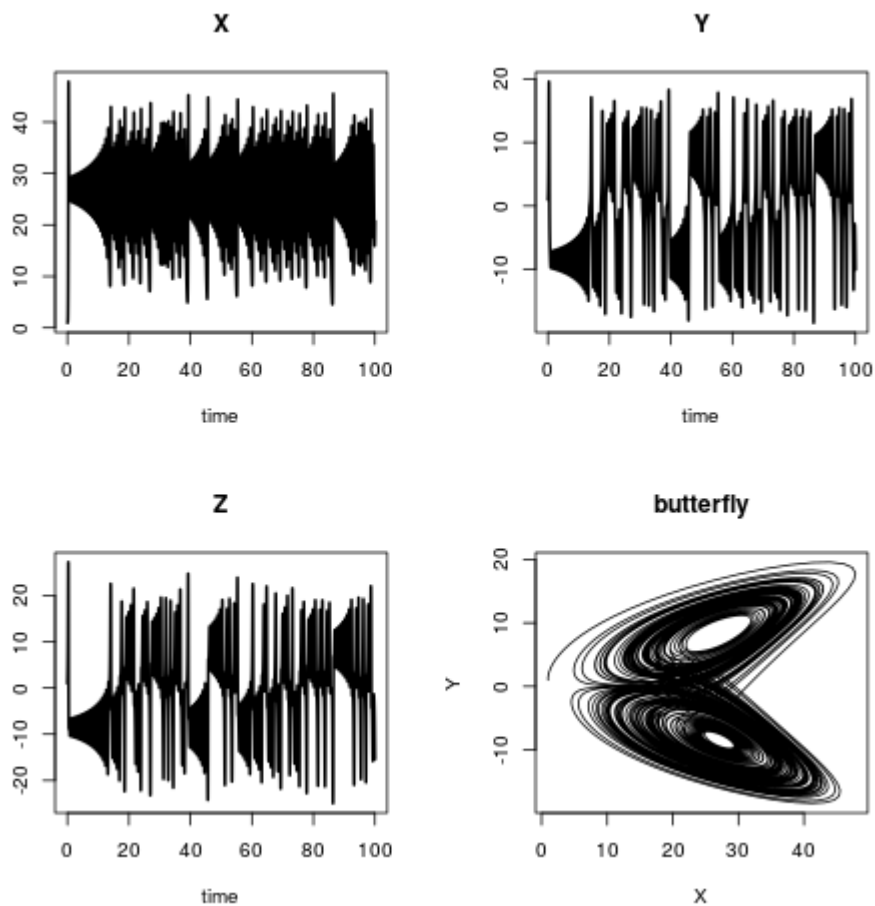
parms <- c(a = -8/3, b = -10, c = 28)
yini <- c(X = 1, Y = 1, Z = 1)
times <- seq(from = 0, to = 100, by = 0.01)

Solve the ODEs

out <- ode(y = yini, times = times, func = Lorenz, parms = parms)

Plot the results

plot(out, lwd = 2)
plot(out[, "X"], out[, "Y"],
 type = "l", xlab = "X",
 ylab = "Y", main = "butterfly")
```



## Lotka-Volterra oder: Beute gegen Raubtier

```
library(deSolve)

Define R-function

LV <- function(t, y, parms) {
 with(as.list(c(y, parms)), {

 dP <- rG * P * (1 - P/K) - rI * P * C
 dC <- rI * P * C * AE - rM * C

 return(list(c(dP, dC), sum = C+P))
 })
}

Define parameters and variables

parms <- c(rI = 0.2, rG = 1.0, rM = 0.2, AE = 0.5, K = 10)
yini <- c(P = 1, C = 2)
times <- seq(from = 0, to = 200, by = 1)

Solve the ODEs

```

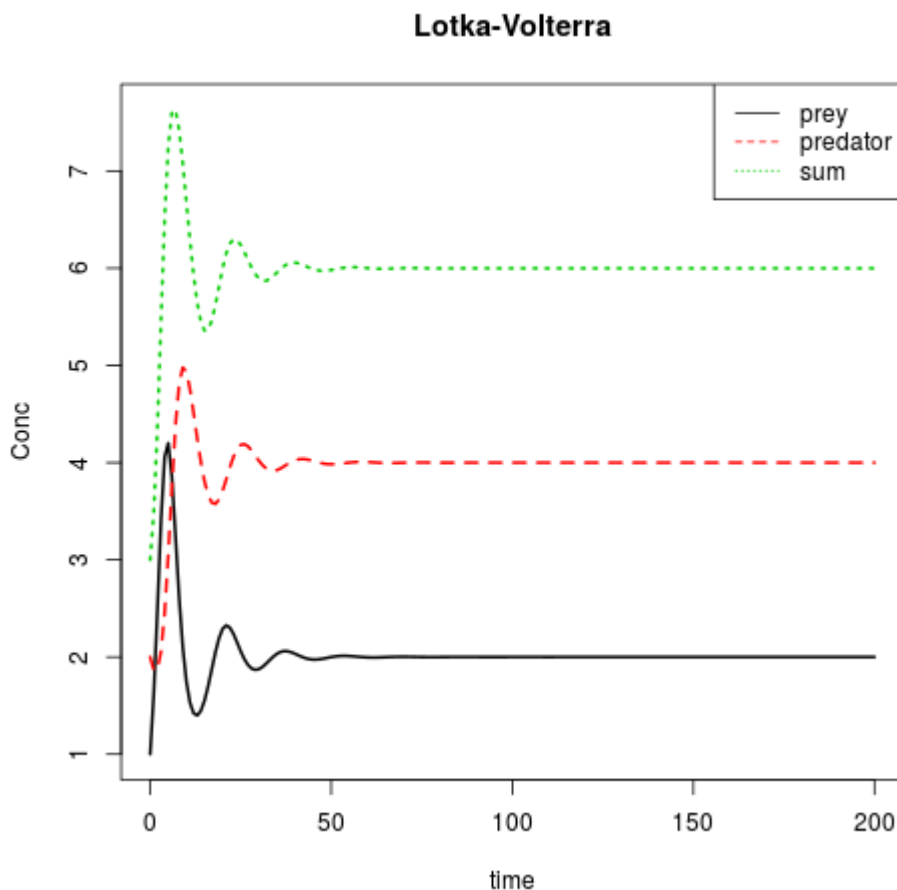
```

out <- ode(y = yini, times = times, func = LV, parms = parms)

Plot the results

matplot(out[,1], out[,2:4], type = "l", xlab = "time", ylab = "Conc",
 main = "Lotka-Volterra", lwd = 2)
legend("topright", c("prey", "predator", "sum"), col = 1:3, lty = 1:3)

```



## ODEs in kompilierten Sprachen - Definition in R

```

library(deSolve)

Define parameters and variables

eps <- 0.01;
M <- 10
k <- M * eps^2/2
L <- 1
L0 <- 0.5
r <- 0.1
w <- 10
g <- 1

```

```

parameter <- c(eps = eps, M = M, k = k, L = L, L0 = L0, r = r, w = w, g = g)

yini <- c(xl = 0, yl = L0, xr = L, yr = L0,
 ul = -L0/L, vl = 0,
 ur = -L0/L, vr = 0,
 lam1 = 0, lam2 = 0)

times <- seq(from = 0, to = 3, by = 0.01)

Define R-function

caraxis_R <- function(t, y, parms) {
 with(as.list(c(y, parms)), {

 yb <- r * sin(w * t)
 xb <- sqrt(L * L - yb * yb)
 Ll <- sqrt(xl^2 + yl^2)
 Lr <- sqrt((xr - xb)^2 + (yr - yb)^2)

 dxl <- ul; dyl <- vl; dxr <- ur; dyr <- vr

 dul <- (L0-Ll) * xl/Ll + 2 * lam2 * (xl-xr) + lam1*xb
 dvl <- (L0-Ll) * yl/Ll + 2 * lam2 * (yl-yr) + lam1*yb - k * g

 dur <- (L0-Lr) * (xr-xb)/Lr - 2 * lam2 * (xl-xr)
 dvr <- (L0-Lr) * (yr-yb)/Lr - 2 * lam2 * (yl-yr) - k * g

 c1 <- xb * xl + yb * yl
 c2 <- (xl - xr)^2 + (yl - yr)^2 - L * L

 return(list(c(dxl, dyl, dxr, dyr, dul, dvl, dur, dvr, c1, c2)))
 })
}

```

## ODEs in kompilierten Sprachen - Definition in C

```

sink("caraxis_C.c")
cat("
/* suitable names for parameters and state variables */

#include <R.h>
#include <math.h>
static double parms[8];

#define eps parms[0]
#define m parms[1]
#define k parms[2]
#define L parms[3]
#define L0 parms[4]
#define r parms[5]
#define w parms[6]
#define g parms[7]

/*-----
 initialising the parameter common block

```

```

*/
void init_C(void (* daeparms)(int *, double *)) {
 int N = 8;
 daeparms(&N, parms);
}
/* Compartments */

#define xl y[0]
#define yl y[1]
#define xr y[2]
#define yr y[3]
#define lam1 y[8]
#define lam2 y[9]

/*-----
the residual function

*/
void caraxis_C (int *neq, double *t, double *y, double *ydot,
 double *yout, int* ip)
{
 double yb, xb, Lr, Ll;

 yb = r * sin(w * *t) ;
 xb = sqrt(L * L - yb * yb);
 Ll = sqrt(xl * xl + yl * yl) ;
 Lr = sqrt((xr-xb)*(xr-xb) + (yr-yb)*(yr-yb));

 ydot[0] = y[4];
 ydot[1] = y[5];
 ydot[2] = y[6];
 ydot[3] = y[7];

 ydot[4] = (L0-Ll) * xl/Ll + lam1*xb + 2*lam2*(xl-xr) ;
 ydot[5] = (L0-Ll) * yl/Ll + lam1*yb + 2*lam2*(yl-yr) - k*g;
 ydot[6] = (L0-Lr) * (xr-xb)/Lr - 2*lam2*(xl-xr) ;
 ydot[7] = (L0-Lr) * (yr-yb)/Lr - 2*lam2*(yl-yr) - k*g ;

 ydot[8] = xb * xl + yb * yl;
 ydot[9] = (xl-xr) * (xl-xr) + (yl-yr) * (yl-yr) - L*L;
}
", fill = TRUE)
sink()
system("R CMD SHLIB caraxis_C.c")
dyn.load(paste("caraxis_C", .Platform$dynlib.ext, sep = ""))
dllname_C <- dyn.load(paste("caraxis_C", .Platform$dynlib.ext, sep = ""))[[1]]

```

## ODEs in kompilierten Sprachen - Definition in Fortran

```

sink("caraxis_fortran.f")
cat("
c-----
c Initialiser for parameter common block
c-----
 subroutine init_fortran(daeparms)

 external daeparms
 integer, parameter :: N = 8

```



```

double precision parms(N)
common /myparms/parms

call daeparms(N, parms)
return
end

c-----
c rate of change
c-----

subroutine caraxis_fortran(neq, t, y, ydot, out, ip)
implicit none
integer neq, IP(*)
double precision t, y(neq), ydot(neq), out(*)
double precision eps, M, k, L, L0, r, w, g
common /myparms/ eps, M, k, L, L0, r, w, g

double precision xl, yl, xr, yr, ul, vl, ur, vr, lam1, lam2
double precision yb, xb, Ll, Lr, dxl, dyl, dxr, dyr
double precision dul, dvl, dur, dvr, c1, c2

c expand state variables
xl = y(1)
yl = y(2)
xr = y(3)
yr = y(4)
ul = y(5)
vl = y(6)
ur = y(7)
vr = y(8)
lam1 = y(9)
lam2 = y(10)

yb = r * sin(w * t)
xb = sqrt(L * L - yb * yb)
Ll = sqrt(xl**2 + yl**2)
Lr = sqrt((xr - xb)**2 + (yr - yb)**2)

dxl = ul
dyl = vl
dxr = ur
dyr = vr

dul = (L0-Ll) * xl/Ll + 2 * lam2 * (xl-xr) + lam1*xb
dvl = (L0-Ll) * yl/Ll + 2 * lam2 * (yl-yr) + lam1*yb - k*g
dur = (L0-Lr) * (xr-xb)/Lr - 2 * lam2 * (xl-xr)
dvr = (L0-Lr) * (yr-yb)/Lr - 2 * lam2 * (yl-yr) - k*g

c1 = xb * xl + yb * yl
c2 = (xl - xr)**2 + (yl - yr)**2 - L * L

c function values in ydot
ydot(1) = dxl
ydot(2) = dyl
ydot(3) = dxr
ydot(4) = dyr
ydot(5) = dul
ydot(6) = dvl
ydot(7) = dur
ydot(8) = dvr
ydot(9) = c1

```

```

 ydot(10) = c2
 return
 end
", fill = TRUE)

sink()
system("R CMD SHLIB caraxis_fortran.f")
dyn.load(paste("caraxis_fortran", .Platform$dynlib.ext, sep = ""))
dllname_fortran <- dyn.load(paste("caraxis_fortran", .Platform$dynlib.ext, sep = ""))[[1]]

```

## ODEs in kompilierten Sprachen - ein Benchmark-Test

Wenn Sie den Code in den drei Beispielen zuvor kompiliert und geladen haben (ODEs in kompilierten Sprachen - Definition in R, ODEs in kompilierten Sprachen - Definition in C und ODEs in kompilierten Sprachen - Definition in Fortran), können Sie einen Benchmark-Test durchführen.

```

library(microbenchmark)

R <- function(){
 out <- ode(y = yini, times = times, func = caraxis_R,
 parms = parameter)
}

C <- function(){
 out <- ode(y = yini, times = times, func = "caraxis_C",
 initfunc = "init_C", parms = parameter,
 dllname = dllname_C)
}

fortran <- function(){
 out <- ode(y = yini, times = times, func = "caraxis_fortran",
 initfunc = "init_fortran", parms = parameter,
 dllname = dllname_fortran)
}

```

Prüfen Sie, ob die Ergebnisse gleich sind:

```

all.equal(tail(R()), tail(fortran()))
all.equal(R()[,2], fortran()[,2])
all.equal(R()[,2], C()[,2])

```

Machen Sie einen Benchmark (Hinweis: An Ihrer Maschine sind die Zeiten natürlich unterschiedlich):

```

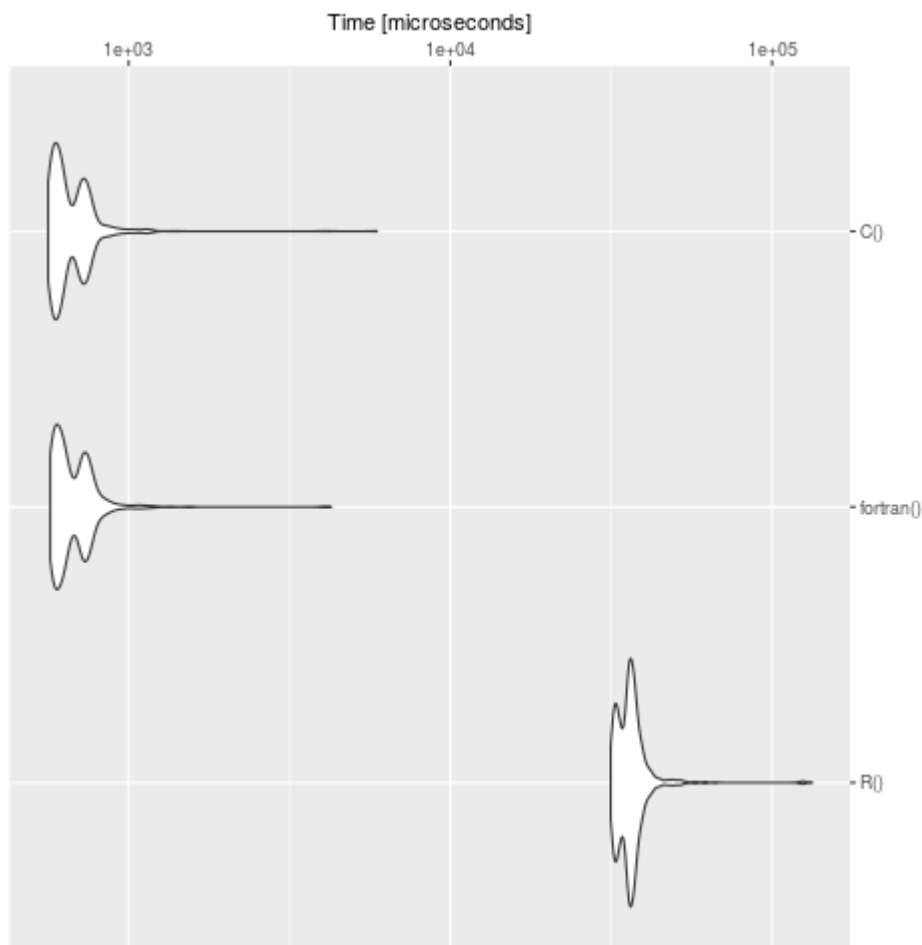
bench <- microbenchmark::microbenchmark(
 R(),
 fortran(),
 C(),
 times = 1000
)

summary(bench)

```

| expr      | min | lq | mean | median | uq | max | neval | cld |
|-----------|-----|----|------|--------|----|-----|-------|-----|
| R()       |     |    |      |        |    |     |       |     |
| fortran() |     |    |      |        |    |     |       |     |
| C()       |     |    |      |        |    |     |       |     |

|           |           |           |            |            |            |            |      |   |
|-----------|-----------|-----------|------------|------------|------------|------------|------|---|
| R()       | 31508.928 | 33651.541 | 36747.8733 | 36062.2475 | 37546.8025 | 132996.564 | 1000 | b |
| fortran() | 570.674   | 596.700   | 686.1084   | 637.4605   | 730.1775   | 4256.555   | 1000 | a |
| C()       | 562.163   | 590.377   | 673.6124   | 625.0700   | 723.8460   | 5914.347   | 1000 | a |



Wir sehen deutlich, dass R im Gegensatz zu der Definition in C und Fortran langsam ist. Für große Modelle lohnt es sich, das Problem in eine kompilierte Sprache zu übersetzen. Das Paket `cOde` ist eine Möglichkeit, ODEs von R nach C zu übersetzen.

Lösen von ODEs in R online lesen: <https://riptutorial.com/de/r/topic/7448/losen-von-odes-in-r>

---

# Kapitel 69: Lubridat

## Syntax

- `ymd_hms (... , quiet = FALSE, tz = "UTC"), locale = Sys.getlocale ("LC_TIME"))`
- `jetzt (tzone = "")`
- `Intervall (Start, Ende, Tzone = Attr (Start, "Tzone"))`
- `Dauer (num = NULL, Einheiten = "Sekunden", ...)`
- `Periode (num = NULL, Einheiten = "Sekunde", ...)`

## Bemerkungen

So installieren Sie das Paket von CRAN:

```
install.packages("lubridate")
```

So installieren Sie die Entwicklungsversion von Github:

```
library(devtools)
dev mode allows testing of development packages in a sandbox, without interfering
with the other packages you have installed.
dev_mode(on=T)
install_github("hadley/lubridate")
dev_mode(on=F)
```

So erhalten Sie Vignetten für das Lubridat-Paket:

```
vignette("lubridate")
```

Um Hilfe zu erhalten über eine Funktion `foo` :

```
help(foo) # help about function foo
?foo # same thing

Example
help("is.period")
?is.period
```

Um Beispiele für eine Funktion zu erhalten, `foo` :

```
example("foo")

Example
example("interval")
```

## Examples

## Analysieren von Datumsangaben und Datumsangaben von Zeichenfolgen mit Lubridat

Das `lubridate` Paket bietet praktische Funktionen zum Formatieren von datei- und datetime-Objekten aus Zeichenketten. Die Funktionen sind Permutationen von

| Brief           | Zu analysierendes Element | Basis-R-Äquivalent |
|-----------------|---------------------------|--------------------|
| y               | Jahr                      | %y , %Y            |
| m (mit y und d) | Monat                     | %m , %b , %h , %B  |
| d               | Tag                       | %d , %e            |
| h               | Stunde                    | %H , %I%p          |
| m (mit h und s) | Minute                    | %M                 |
| s               | Sekunden                  | %S                 |

zB `ymd()` zum Parsen eines Datums mit dem Jahr gefolgt vom Monat gefolgt vom Tag, zB "2016-07-22" oder `ymd_hms()` zum Parsen einer Datumszeit in der Reihenfolge Jahr, Monat, Tag, Stunden, Minuten, Sekunden, zB "2016-07-22 13:04:47".

Die Funktionen können die meisten Trennzeichen (wie / , - und Whitespace) ohne zusätzliche Argumente erkennen. Sie arbeiten auch mit inkonsistenten Trennzeichen.

---

## Termine

Die Date-Funktionen geben ein Objekt der Klasse `Date`.

```
library(lubridate)

mdy(c(' 07/02/2016 ', '7 / 03 / 2016', ' 7 / 4 / 16 '))
[1] "2016-07-02" "2016-07-03" "2016-07-04"

ymd(c("20160724", "2016/07/23", "2016-07-25")) # inconsistent separators
[1] "2016-07-24" "2016-07-23" "2016-07-25"
```

---

## Datumsangaben

### Dienstprogrammfunktionen

Datenzeiten können mit `ymd_hms` Varianten einschließlich `ymd_hm` und `ymd_h`. Alle datetime-

Funktionen können ein `tz` Zeitzoneargument akzeptieren, das dem von `as.POSIXct` oder `strptime`, jedoch standardmäßig `"UTC"` anstelle der lokalen Zeitzone verwendet.

Die `datetime`-Funktionen geben ein Objekt der Klasse `POSIXct`.

```
x <- c("20160724 130102", "2016/07/23 14:02:01", "2016-07-25 15:03:00")
ymd_hms(x, tz="EST")
[1] "2016-07-24 13:01:02 EST" "2016-07-23 14:02:01 EST"
[3] "2016-07-25 15:03:00 EST"

ymd_hms(x)
[1] "2016-07-24 13:01:02 UTC" "2016-07-23 14:02:01 UTC"
[3] "2016-07-25 15:03:00 UTC"
```

## Parser-Funktionen

`lubridate` enthält außerdem drei Funktionen zum Analysieren von Datumsangaben mit einer Formatierungszeichenfolge wie `as.POSIXct` oder `strptime`:

| Funktion                      | Ausgabeklasse                                                                            | Formatierungszeichenfolgen werden akzeptiert                                                                                                                                                                                                              |
|-------------------------------|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parse_date_time</code>  | <code>POSIXct</code>                                                                     | Flexibel. Akzeptiert <code>strptime</code> mit <code>%</code> oder <code>lubridate</code> <code>datetime</code> Funktionsnamenstil, zB <code>"ymd hms"</code> . Akzeptiert einen Ordnungsvektor für heterogene Daten und schätzt, ob dies angemessen ist. |
| <code>parse_date_time2</code> | Default <code>POSIXct</code> ;<br>Wenn <code>lt = TRUE</code> ,<br><code>POSIXlt</code>  | Streng Akzeptiert nur <code>strptime</code> Token (mit oder ohne <code>%</code> ) aus einer begrenzten Menge.                                                                                                                                             |
| <code>fast_strptime</code>    | Default <code>POSIXlt</code> ;<br>wenn <code>lt = FALSE</code> ,<br><code>POSIXct</code> | Streng Akzeptiert nur <code>%</code> separiert <code>strptime</code> Token mit Trennzeichen ( <code>-</code> , <code>/</code> , <code>:</code> , etc.) aus einer begrenzten Menge.                                                                        |

```
x <- c('2016-07-22 13:04:47', '07/22/2016 1:04:47 pm')

parse_date_time(x, orders = c('mdy lmsp', 'ymd hms'))
[1] "2016-07-22 13:04:47 UTC" "2016-07-22 13:04:47 UTC"

x <- c('2016-07-22 13:04:47', '2016-07-22 14:47:58')

parse_date_time2(x, orders = 'Ymd HMS')
[1] "2016-07-22 13:04:47 UTC" "2016-07-22 14:47:58 UTC"

fast_strptime(x, format = '%Y-%m-%d %H:%M:%S')
[1] "2016-07-22 13:04:47 UTC" "2016-07-22 14:47:58 UTC"
```

`parse_date_time2` und `fast_strptime` verwenden aus `parse_date_time2` `fast_strptime` einen schnellen C-Parser.

?`parse_date_time` zum Formatieren von Tokens finden Sie unter ?`parse_date_time`.

## Analysieren von Datum und Uhrzeit in Lubridate

Lubridate bietet `ymd()` Funktionsserien zum Analysieren von Zeichenketten in Datumsangaben. Die Buchstaben y, m und d entsprechen den Elementen year, month und day eines Datums / einer Uhrzeit.

```
mdy("07-21-2016") # Returns Date

[1] "2016-07-21"

mdy("07-21-2016", tz = "UTC") # Returns a vector of class POSIXt

"2016-07-21 UTC"

dmy("21-07-2016") # Returns Date

[1] "2016-07-21"

dmy(c("21.07.2016", "22.07.2016")) # Returns vector of class Date

[1] "2016-07-21" "2016-07-22"
```

## Datum und Uhrzeit in Lubridate bearbeiten

```
date <- now()
date
"2016-07-22 03:42:35 IST"

year(date)
2016

minute(date)
42

wday(date, label = T, abbr = T)
[1] Fri
Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat

day(date) <- 31
"2016-07-31 03:42:35 IST"

If an element is set to a larger value than it supports, the difference
will roll over into the next higher element
day(date) <- 32
"2016-08-01 03:42:35 IST"
```

## Momente

Ein Moment ist ein bestimmter Zeitpunkt. Jedes Datums- / Uhrzeitobjekt, das sich auf einen Zeitpunkt bezieht, wird als Moment erkannt. Um zu testen, ob ein Objekt ein Moment ist, verwenden Sie `is.instant`.

```
library(lubridate)
```

```

today_start <- dmy_hms("22.07.2016 12:00:00", tz = "IST") # default tz="UTC"
today_start
[1] "2016-07-22 12:00:00 IST"
is.instant(today_start)
[1] TRUE

now_dt <- ymd_hms(now(), tz="IST")
now_dt
[1] "2016-07-22 13:53:09 IST"
is.instant(now_dt)
[1] TRUE

is.instant("helloworld")
[1] FALSE
is.instant(60)
[1] FALSE

```

## Intervalle, Dauern und Perioden

**Intervalle** sind die einfachste Möglichkeit, Zeiträume in Lubridate aufzuzeichnen. Ein Intervall ist eine Zeitspanne, die zwischen zwei bestimmten **Zeitpunkten** liegt .

```

create interval by subtracting two instants
today_start <- ymd_hms("2016-07-22 12:00:00", tz="IST")
today_start
[1] "2016-07-22 12:00:00 IST"
today_end <- ymd_hms("2016-07-22 23:59:59", tz="IST")
today_end
[1] "2016-07-22 23:59:59 IST"
span <- today_end - today_start
span
Time difference of 11.99972 hours
as.interval(span, today_start)
[1] 2016-07-22 12:00:00 IST--2016-07-22 23:59:59 IST

create interval using interval() function
span <- interval(today_start, today_end)
[1] 2016-07-22 12:00:00 IST--2016-07-22 23:59:59 IST

```

**Die Dauer** misst die genaue Zeit zwischen zwei Zeitpunkten.

```

duration(60, "seconds")
[1] "60s"

duration(2, "minutes")
[1] "120s (~2 minutes)"

```

**Hinweis:** Einheiten, die größer als Wochen sind, werden aufgrund ihrer Variabilität nicht verwendet.

Dauer kann mit Hilfe von `dseconds` , `dminutes` und anderen `dminutes` Dauer erstellt werden. Führen Sie `?quick_durations` für eine vollständige Liste aus.

```

dseconds(60)
[1] "60s"

```



```

dhours(2)
[1] "7200s (~2 hours)"

dyears(1)
[1] "31536000s (~365 days)"

```

Dauer kann abgezogen und zu Zeitpunkten addiert werden, um neue Zeitpunkte zu erhalten.

```

today_start + dhours(5)
[1] "2016-07-22 17:00:00 IST"

today_start + dhours(5) + dminutes(30) + dseconds(15)
[1] "2016-07-22 17:30:15 IST"

```

Dauer kann aus Intervallen erstellt werden.

```

as.duration(span)
[1] "43199s (~12 hours)"

```

**Perioden** messen die Änderung der Uhrzeit zwischen zwei Zeitpunkten.

Perioden können mit der `period` sowie anderen Hilfsfunktionen wie `seconds`, `hours` usw. erstellt werden. Um eine vollständige Liste der Periodenhilfsfunktionen zu erhalten, `?quick_periods` Run `?quick_periods`.

```

period(1, "hour")
[1] "1H 0M 0S"

hours(1)
[1] "1H 0M 0S"

period(6, "months")
[1] "6m 0d 0H 0M 0S"

months(6)
[1] "6m 0d 0H 0M 0S"

years(1)
[1] "1y 0m 0d 0H 0M 0S"

```

**is.period Funktion** `is.period` kann verwendet werden, um zu prüfen, ob ein Objekt eine Periode ist.

```

is.period(years(1))
[1] TRUE

is.period(dyears(1))
[1] FALSE

```

## Rundungsdaten

```

now_dt <- ymd_hms(now(), tz="IST")

```

```
now_dt
[1] "2016-07-22 13:53:09 IST"
```

`round_date()` ein Date-Time-Objekt und rundet es auf den nächsten ganzzahligen Wert der angegebenen Zeiteinheit.

```
round_date(now_dt, "minute")
[1] "2016-07-22 13:53:00 IST"

round_date(now_dt, "hour")
[1] "2016-07-22 14:00:00 IST"

round_date(now_dt, "year")
[1] "2017-01-01 IST"
```

`floor_date()` ein Date-Time-Objekt und rundet es auf den nächsten ganzzahligen Wert der angegebenen Zeiteinheit ab.

```
floor_date(now_dt, "minute")
[1] "2016-07-22 13:53:00 IST"

floor_date(now_dt, "hour")
[1] "2016-07-22 13:00:00 IST"

floor_date(now_dt, "year")
[1] "2016-01-01 IST"
```

`ceiling_date()` ein Date-Time-Objekt und rundet es auf den nächsten ganzzahligen Wert der angegebenen Zeiteinheit.

```
ceiling_date(now_dt, "minute")
[1] "2016-07-22 13:54:00 IST"

ceiling_date(now_dt, "hour")
[1] "2016-07-22 14:00:00 IST"

ceiling_date(now_dt, "year")
[1] "2017-01-01 IST"
```

## Unterschied zwischen Zeitraum und Dauer

Im Gegensatz zu Dauern können Perioden verwendet werden, um die Uhrzeiten genau zu modellieren, ohne zu wissen, wann Ereignisse wie Schaltsekunden, Schalttage und DST-Änderungen auftreten.

```
start_2012 <- ymd_hms("2012-01-01 12:00:00")
[1] "2012-01-01 12:00:00 UTC"

period() considers leap year calculations.
start_2012 + period(1, "years")
[1] "2013-01-01 12:00:00 UTC"

Here duration() doesn't consider leap year calculations.
```

```
start_2012 + duration(1)
[1] "2012-12-31 12:00:00 UTC"
```

## Zeitzone

`with_tz` gibt ein Datum und eine Uhrzeit zurück, wie es in einer anderen Zeitzone erscheinen würde.

```
nyc_time <- now("America/New_York")
nyc_time
[1] "2016-07-22 05:49:08 EDT"

corresponding Europe/Moscow time
with_tz(nyc_time, tzone = "Europe/Moscow")
[1] "2016-07-22 12:49:08 MSK"
```

`force_tz` gibt das Datum und die Uhrzeit zurück, das dieselbe Uhrzeit wie x in der neuen Zeitzone hat.

```
nyc_time <- now("America/New_York")
nyc_time
[1] "2016-07-22 05:49:08 EDT"

force_tz(nyc_time, tzone = "Europe/Moscow") # only timezone changes
[1] "2016-07-22 05:49:08 MSK"
```

Lubridat online lesen: <https://riptutorial.com/de/r/topic/2496/lubridat>

# Kapitel 70: Maschinelles lernen

## Examples

### Ein zufälliges Gesamtstrukturmodell erstellen

Ein Beispiel für Algorithmen für maschinelles Lernen ist der Random Forest-Algorithmus (Breiman, L. (2001). Random Forests. *Machine Learning* 45 (5) , S. 5-32). Dieser Algorithmus ist in R gemäß der ursprünglichen Fortran-Implementierung von Breiman im `randomForest` Paket implementiert.

Random Forest-Klassifiziererobjekte können in R erstellt werden, indem die Klassenvariable als `factor` vorbereitet `factor` , der bereits in der `iris` Datei sichtbar ist. Daher können wir leicht einen Zufallswald erstellen, indem wir:

```
library(randomForest)

rf <- randomForest(x = iris[, 1:4],
 y = iris$Species,
 ntree = 500,
 do.trace = 100)

rf

Call:
randomForest(x = iris[, 1:4], y = iris$Species, ntree = 500, do.trace = 100)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 2
#
OOB estimate of error rate: 4%
Confusion matrix:
setosa versicolor virginica class.error
setosa 50 0 0 0.00
versicolor 0 47 3 0.06
virginica 0 3 47 0.06
```

| Parameter | Beschreibung                                                                                                                                                               |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x         | ein Datenrahmen, der die beschreibenden Variablen der Klassen enthält                                                                                                      |
| y         | die Klassen der einzelnen Observationen. Wenn dieser Vektor ein <code>factor</code> , wird ein Klassifizierungsmodell erstellt, wenn kein Regressionsmodell erstellt wird. |
| Baum      | Die Anzahl der einzelnen CART-Bäume                                                                                                                                        |
| do.trace  | Bei jedem <i>i-ten</i> Schritt werden die sofort eintreffenden Fehler für jede Klasse zurückgegeben                                                                        |

Maschinelles lernen online lesen: <https://riptutorial.com/de/r/topic/8326/maschinelles-lernen>

# Kapitel 71: Matrizen

## Einführung

Matrizen speichern Daten

## Examples

### Matrizen erstellen

Unter der Haube ist eine Matrix eine besondere Art von Vektor mit zwei Dimensionen. Wie ein Vektor kann eine Matrix nur eine Datenklasse haben. Sie können Matrizen mit der `matrix` wie unten gezeigt erstellen.

```
matrix(data = 1:6, nrow = 2, ncol = 3)
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
```

Wie Sie sehen, erhalten Sie eine Matrix aus allen Zahlen von 1 bis 6 mit zwei Zeilen und drei Spalten. Die `data` - Parameter nehmen einen Vektor von Werten, `nrow` gibt die Anzahl der Zeilen in der Matrix und `ncol` gibt die Anzahl der Spalten. In der Regel wird die Matrix mit einer Spalte gefüllt. Das Standardverhalten kann mit dem `byrow` -Parameter wie `byrow` geändert werden:

```
matrix(data = 1:6, nrow = 2, ncol = 3, byrow = TRUE)
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6
```

Matrizen müssen nicht numerisch sein - jeder Vektor kann in eine Matrix umgewandelt werden. Zum Beispiel:

```
matrix(data = c(TRUE, TRUE, TRUE, FALSE, FALSE, FALSE), nrow = 3, ncol = 2)
[,1] [,2]
[1,] TRUE FALSE
[2,] TRUE FALSE
[3,] TRUE FALSE
matrix(data = c("a", "b", "c", "d", "e", "f"), nrow = 3, ncol = 2)
[,1] [,2]
[1,] "a" "d"
[2,] "b" "e"
[3,] "c" "f"
```

Wie Vektoren können Matrizen als Variablen gespeichert und später aufgerufen werden. Die Zeilen und Spalten einer Matrix können Namen haben. Sie können diese mit den Funktionen `rownames` und `colnames`. Wie unten gezeigt, haben die Zeilen und Spalten zunächst keine Namen, was mit `NULL` bezeichnet wird. Sie können ihnen jedoch Werte zuweisen.

```
mat1 <- matrix(data = 1:6, nrow = 2, ncol = 3, byrow = TRUE)
rownames(mat1)
NULL
colnames(mat1)
NULL
rownames(mat1) <- c("Row 1", "Row 2")
colnames(mat1) <- c("Col 1", "Col 2", "Col 3")
mat1
Col 1 Col 2 Col 3
Row 1 1 2 3
Row 2 4 5 6
```

Es ist wichtig anzumerken, dass Matrizen ähnlich wie Vektoren nur einen Datentyp haben können. Wenn Sie versuchen, eine Matrix mit mehreren Datentypen anzugeben, werden die Daten in die übergeordnete Datenklasse umgewandelt.

Die `class`, `is` und `as` Funktionen kann verwendet werden, um Datenstrukturen auf dieselbe Weise zu prüfen und zu zwingen, wie sie für die Vektoren in Klasse 1 verwendet wurden.

```
class(mat1)
[1] "matrix"
is.matrix(mat1)
[1] TRUE
as.vector(mat1)
[1] 1 4 2 5 3 6
```

Matrizen online lesen: <https://riptutorial.com/de/r/topic/9019/matrizen>

---

# Kapitel 72: Meta: Dokumentationsrichtlinien

## Bemerkungen

Besuchen Sie den [R-Chat](#) , um die Bearbeitung der R-Tag-Dokumente zu diskutieren.

## Examples

### Gute Beispiele machen

Die meisten Anleitungen zum [Erstellen guter Beispiele](#) für Fragen und Antworten werden in die Dokumentation übernommen.

- Machen Sie es minimal und kommen Sie auf den Punkt. Komplikationen und Abschweifungen sind kontraproduktiv.
- Fügen Sie sowohl Arbeitscode als auch Prosa zum Erläutern hinzu. Keines reicht alleine aus.
- Verlassen Sie sich nicht auf externe Datenquellen. Generieren Sie Daten oder verwenden Sie nach Möglichkeit die Datensatzbibliothek:

```
library(help = "datasets")
```

Es gibt einige zusätzliche Überlegungen im Zusammenhang mit Dokumenten:

- `?data.frame` Sie sich auf eingebaute Dokumente wie `?data.frame` wenn dies relevant ist. Die SO-Dokumente sind kein Versuch, die integrierten Dokumente zu ersetzen. Es ist wichtig sicherzustellen, dass neue R-Benutzer wissen, dass die integrierten Dokumente vorhanden sind, und wie sie zu finden sind.
- Verschieben Sie den Inhalt, der für mehrere Beispiele gilt, in den Abschnitt "Anmerkungen".

## Stil

---

## Eingabeaufforderungen

Wenn Sie möchten, dass Ihr Code kopierfähig ist, entfernen Sie am Anfang jeder neuen Zeile Eingabeaufforderungen wie `R>` , `>` oder `+` . Einige Autoren von Texten ziehen es vor, das Kopieren nicht einfach zu machen, und das ist in Ordnung.

---

## Konsolenausgabe

Die Konsolenausgabe sollte klar vom Code unterschieden werden. Gängige Ansätze sind:

- Einfügen von Eingabeaufforderungen bei der Eingabe (wie bei Verwendung der Konsole zu sehen).
- Kommentieren Sie alle Ausgaben aus, wobei # oder ## jeder Zeile stehen.
- Drucken Sie in der vorliegenden Form und vertrauen Sie der führenden [1] , um die Ausgabe von der Eingabe abzuheben.
- Fügen Sie zwischen Code und Konsolenausgabe eine Leerzeile hinzu.

---

## Zuordnung

= und <- sind gut für die Zuordnung von R-Objekten. Verwenden Sie Leerzeichen entsprechend, um das Schreiben von Code zu vermeiden, der schwer zu analysieren ist, z. B. `x<-1` (mehrdeutig zwischen `x <- 1` und `x < -1`).

---

## Code-Kommentare

Stellen Sie sicher, dass Sie den Zweck und die Funktion des Codes selbst erklären. Es gibt keine feste Regel, ob diese Erklärung in Prosa oder Code-Kommentaren enthalten sein sollte. Prosa ist möglicherweise lesbarer und erlaubt längere Erklärungen, aber Code-Kommentare erleichtern das Einfügen von Text. Denken Sie an beide Optionen.

---

## Abschnitte

Viele Beispiele sind kurz genug, um keine Abschnitte zu benötigen. Wenn Sie sie jedoch verwenden, beginnen Sie mit [H1](#) .

Meta: Dokumentationsrichtlinien online lesen: <https://riptutorial.com/de/r/topic/5410/meta--dokumentationsrichtlinien>



---

# Kapitel 73: Musterabgleich und Ersatz

## Einführung

Dieses Thema behandelt übereinstimmende Zeichenfolgenmuster sowie das Extrahieren oder Ersetzen dieser Muster. Einzelheiten zum Definieren komplizierter Muster finden Sie unter [Reguläre Ausdrücke](#) .

## Syntax

- `grep ("query", "subject", optional_args)`
- `grepl ("Abfrage", "Betreff", optional_args)`
- `gsub ("(group1) (group2)", "\\ group #", "subject")`

## Bemerkungen

---

## Unterschiede zu anderen Sprachen

Entkam [regex](#) Symbole (wie `\1` ) werden muss ein zweites Mal (wie entwertet werden `\\1` ), nicht nur in dem `pattern` Argumente, sondern auch im `replacement` zu `sub` und `gsub` .

Standardmäßig handelt es sich bei dem Muster für alle Befehle (`grep`, `sub`, `regexpr`) nicht um Perl-kompatible reguläre Ausdrücke (PCRE). Einige Dinge wie Lookarounds werden nicht unterstützt. Jede Funktion akzeptiert jedoch ein `perl=TRUE` Argument, um sie zu aktivieren. Weitere Informationen finden Sie im [Abschnitt R reguläre Ausdrücke](#) .

---

## Spezialisierte Pakete

- [stringi](#)
- `stringr`

## Examples

### Substitutionen machen

```
example data
test_sentences <- c("The quick brown fox quickly", "jumps over the lazy dog")
```

Lassen Sie uns den braunen Fuchs rot machen:

```
sub("brown","red", test_sentences)
#[1] "The quick red fox quickly" "jumps over the lazy dog"
```

Nun lassen Sie uns das machen "fast" Fuchs Akt "fastly" . Das wird es nicht tun:

```
sub("quick", "fast", test_sentences)
#[1] "The fast red fox quickly" "jumps over the lazy dog"
```

sub macht nur den ersten verfügbaren Ersatz, wir brauchen gsub für den [globalen Austausch](#) :

```
gsub("quick", "fast", test_sentences)
#[1] "The fast red fox fastly" "jumps over the lazy dog"
```

Weitere Beispiele finden Sie [unter Ersetzen von Zeichenfolgen](#) .

## Übereinstimmungen finden

```
example data
test_sentences <- c("The quick brown fox", "jumps over the lazy dog")
```

# Gibt es eine Übereinstimmung?

grepl() wird verwendet, um zu prüfen, ob ein Wort oder ein regulärer Ausdruck in einem String oder Zeichenvektor vorhanden ist. Die Funktion gibt einen TRUE / FALSE (oder "Boolean") - Vektor zurück.

Beachten Sie, dass wir jeden String auf das Wort "fox" prüfen können und im Gegenzug einen booleschen Vektor erhalten.

```
grepl("fox", test_sentences)
#[1] TRUE FALSE
```

# Orte abgleichen

grep übernimmt eine Zeichenkette und einen regulären Ausdruck. Es gibt einen numerischen Vektor von Indizes zurück. Dadurch wird der Satz zurückgegeben, der das Wort "Fuchs" enthält.

```
grep("fox", test_sentences)
#[1] 1
```

# Übereinstimmende Werte

So wählen Sie Sätze aus, die einem Muster entsprechen:

```
each of the following lines does the job:
test_sentences[grep("fox", test_sentences)]
test_sentences[grepl("fox", test_sentences)]
grep("fox", test_sentences, value = TRUE)
[1] "The quick brown fox"
```

## Einzelheiten

Da es sich bei dem "fox" -Muster nicht um einen regulären Ausdruck, sondern um ein Wort handelt, können Sie die Leistung verbessern (entweder mit `grep` oder `grepl`), indem Sie `fixed = TRUE` `grepl` .

```
grep("fox", test_sentences, fixed = TRUE)
#[1] 1
```

Um Sätze auszuwählen, *die keinem* Muster entsprechen, können Sie `grep` mit `invert = TRUE` . oder folgen Sie den [Subsetting](#)-Regeln mit `-grep(...)` oder `!grepl(...)` .

Sowohl in `grepl(pattern, x)` als auch in `grep(pattern, x)` ist der `x` Parameter [vektoriert](#) , der `pattern` Parameter jedoch nicht. Daher können Sie diese nicht direkt verwenden, um `pattern[1]` mit `x[1]` , `pattern[2]` mit `x[2]` usw. abzugleichen.

## Zusammenfassung der Spiele

Nach dem Ausführen des Befehls `grepl` möchten Sie möglicherweise einen Überblick darüber erhalten, wie viele Übereinstimmungen `TRUE` oder `FALSE` . Dies ist z. B. bei großen Datensätzen hilfreich. Führen Sie dazu den `summary` Befehl aus:

```
example data
test_sentences <- c("The quick brown fox", "jumps over the lazy dog")

find matches
matches <- grepl("fox", test_sentences)

overview
summary(matches)
```

### Single und Global Match.

Wenn Sie mit regulären Ausdrücken arbeiten, ist ein Modifikator für PCRE `g` für globale Übereinstimmung.

In R haben Matching- und Replacement-Funktionen zwei Versionen: First Match und Global Match:

- `sub(pattern, replacement, text)` ersetzt das erste Vorkommen von Muster durch Ersetzen im Text

- `gsub(pattern, replacement, text)` wird dasselbe tun wie `sub`, jedoch für jedes Auftreten eines Musters
- `regexpr(pattern, text)` gibt die Position der Übereinstimmung für die erste Instanz des Musters zurück
- `gregexpr(pattern, text)` gibt alle Übereinstimmungen zurück.

Einige zufällige Daten:

```
set.seed(123)
teststring <- paste0(sample(letters, 20), collapse="")

teststring
#[1] "htjuwakqxyzpgrsbncvyo"
```

Mal sehen, wie das funktioniert, wenn wir Vokale durch etwas anderes ersetzen wollen:

```
sub("[aeiou]", " ** HERE WAS A VOWEL** ", teststring)
#[1] "htj ** HERE WAS A VOWEL** wakqxyzpgrsbncvyo"

gsub("[aeiou]", " ** HERE WAS A VOWEL** ", teststring)
#[1] "htj ** HERE WAS A VOWEL** w ** HERE WAS A VOWEL** kqxyzpgrsbncv ** HERE WAS A VOWEL** **
HERE WAS A VOWEL** "
```

Nun wollen wir sehen, wie wir einen Konsonanten finden können, dem unmittelbar ein oder mehrere Vokale folgen:

```
regexpr("^[aeiou][aeiou]+", teststring)
#[1] 3
#attr(,"match.length")
#[1] 2
#attr(,"useBytes")
#[1] TRUE
```

Wir haben eine Übereinstimmung auf Position 3 der Zeichenfolge der Länge 2, dh: `ju`

Nun, wenn wir alle Spiele bekommen wollen:

```
gregexpr("^[aeiou][aeiou]+", teststring)
[[1]]
#[1] 3 5 19
#attr(,"match.length")
#[1] 2 2 2
#attr(,"useBytes")
#[1] TRUE
```

Das alles ist wirklich großartig, aber dies gibt nur Spielpositionen des Spiels und es ist nicht so einfach, das zu finden, was übereinstimmt, und hier kommt das `regmatches` es ist nur der Zweck, die übereinstimmende Zeichenfolge aus `regexpr` zu extrahieren, aber es hat eine andere Syntax.

Speichern Sie unsere Übereinstimmungen in einer Variablen und extrahieren Sie sie aus dem

ursprünglichen String:

```
matches <- gregexpr("[^aeiou][aeiou]+",teststring)
regmatches(teststring,matches)
#[[1]]
#[1] "ju" "wa" "yo"
```

Dies mag seltsam klingen, wenn es keine Abkürzung gibt, aber dies erlaubt das Extrahieren aus einer anderen Saite durch die Übereinstimmungen unserer ersten (denken Sie daran, zwei lange Vektoren miteinander zu vergleichen, bei denen Sie wissen, dass es ein gemeinsames Muster für die erste, aber nicht für die zweite, dies erlaubt eine einfacher Vergleich):

```
teststring2 <- "this is another string to match against"
regmatches(teststring2,matches)
#[[1]]
#[1] "is" " i" "ri"
```

Achtung: Standardmäßig ist das Muster kein Perl-kompatibler regulärer Ausdruck. Einige Dinge wie Lookarounds werden nicht unterstützt, aber jede hier dargestellte Funktion ermöglicht das Argument `perl=TRUE` , um sie zu aktivieren.

## Finden Sie Übereinstimmungen in großen Datensätzen

Bei großen Datenmengen `grepl("fox", test_sentences)` der Aufruf von `grepl("fox", test_sentences)` nicht gut. Big Data-Sets sind beispielsweise gecrawlte Websites oder Millionen von Tweets usw.

Die erste Beschleunigung ist die Verwendung der Option `perl = TRUE` . Noch schneller ist die Option `fixed = TRUE` . Ein vollständiges Beispiel wäre:

```
example data
test_sentences <- c("The quick brown fox", "jumps over the lazy dog")

grepl("fox", test_sentences, perl = TRUE)
#[1] TRUE FALSE
```

Bei Text Mining wird häufig ein Korpus verwendet. Ein Korpus kann nicht direkt mit `grepl` . Betrachten Sie daher diese Funktion:

```
searchCorpus <- function(corpus, pattern) {
 return(tm_index(corpus, FUN = function(x) {
 grepl(pattern, x, ignore.case = TRUE, perl = TRUE)
 }))
}
```

Musterabgleich und Ersatz online lesen: <https://riptutorial.com/de/r/topic/1123/musterabgleich-und-ersatz>

# Kapitel 74: Netzwerkanalyse mit dem igraph-Paket

## Examples

### Einfache gerichtete und nicht gerichtete Netzwerkgrafik

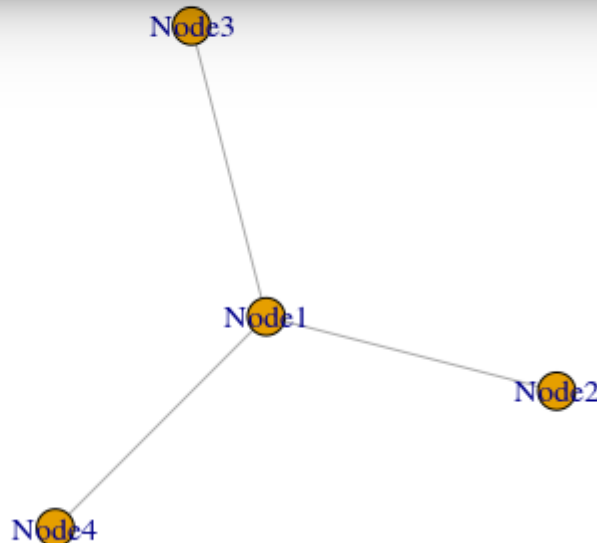
Das igraph-Paket für R ist ein wunderbares Werkzeug, mit dem sowohl reale als auch virtuelle Netzwerke auf einfache Weise modelliert werden können. In diesem Beispiel soll veranschaulicht werden, wie zwei einfache Netzwerkdiagramme mit dem Paket igraph in R v.3.2.3 erstellt werden.

#### Nicht gerichtetes Netzwerk

Das Netzwerk wird mit diesem Code erstellt:

```
g<-graph.formula(Node1-Node2, Node1-Node3, Node4-Node1)
plot(g)
```

```
> g<-graph.formula(Node1-Node2, Node1-Node3, Node4-Node1)
> plot(g)
>
```

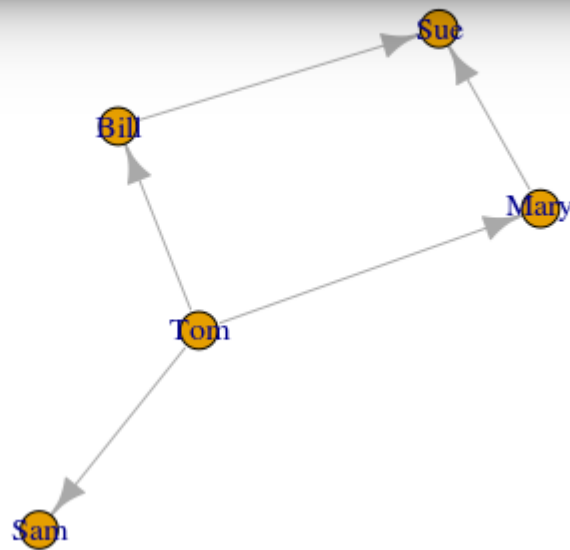


#### Gerichtetes Netzwerk

```
dg<-graph.formula(Tom->Mary, Tom->Bill, Tom->Sam, Sue->Mary, Bill->Sue)
plot(dg)
```

Dieser Code erzeugt dann ein Netzwerk mit Pfeilen:

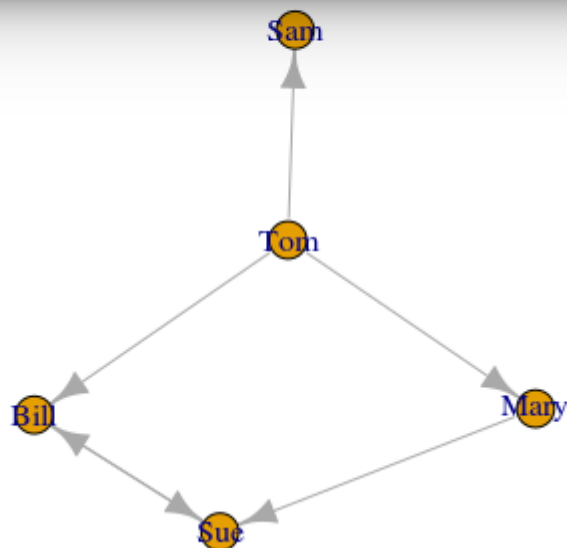
```
> dg<-graph.formula(Tom->Mary, Tom->Bill, Tom->Sam, Sue->Mary, Bill->Sue)
> plot(dg)
>
```



Codebeispiel für das Erstellen eines doppelseitigen Pfeils:

```
dg<-graph.formula(Tom->Mary, Tom->Bill, Tom->Sam, Sue<->Mary, Bill<->Sue)
plot(dg)
```

```
> dg<-graph.formula(Tom->Mary, Tom->Bill, Tom->Sam, Sue<->Mary, Bill<->Sue)
> plot(dg)
>
```



Netzwerkanalyse mit dem igraph-Paket online lesen:

<https://riptutorial.com/de/r/topic/4851/netzwerkanalyse-mit-dem-igraph-paket>

# Kapitel 75: Nichtstandardmäßige Bewertung und Standardbewertung

## Einführung

Dplyr und viele moderne Bibliotheken in R verwenden Nicht-Standard-Bewertung (NSE) für die interaktive Programmierung und Standard-Bewertung (SE) für die Programmierung [1](#).

Zum Beispiel kann das `summarise()` Funktion Verwendung nicht-Standardauswertung sondern verlässt sich auf die `summarise_()` die Standardauswertung verwendet.

Die Lazyeval-Bibliothek macht es einfach, Standardauswertungsfunktionen in NSE-Funktionen umzuwandeln.

## Examples

### Beispiele mit Standard-Dplyr-Verben

Bei der interaktiven Programmierung sollten NSE-Funktionen verwendet werden. Wenn Sie jedoch neue Funktionen in einem neuen Paket entwickeln, sollten Sie die SE-Version verwenden.

Laden Sie Dplyr und Lazyeval:

```
library(dplyr)
library(lazyeval)
```

## Filterung

### NSE-Version

```
filter(mtcars, cyl == 8)
filter(mtcars, cyl < 6)
filter(mtcars, cyl < 6 & vs == 1)
```

*SE-Version (zur Verwendung bei der Programmierung von Funktionen in einem neuen Paket)*

```
filter_(mtcars, .dots = list(~ cyl == 8))
filter_(mtcars, .dots = list(~ cyl < 6))
filter_(mtcars, .dots = list(~ cyl < 6, ~ vs == 1))
```

## Zusammenfassen

### NSE-Version

```
summarise(mtcars, mean(displ))
summarise(mtcars, mean_displ = mean(displ))
```



## SE-Version

```
summarise_(mtcars, .dots = lazyeval::interp(~ mean(x), x = quote(displ)))
summarise_(mtcars, .dots = setNames(list(lazyeval::interp(~ mean(x), x = quote(displ))),
"mean_displ"))
summarise_(mtcars, .dots = list("mean_displ" = lazyeval::interp(~ mean(x), x = quote(displ))))
```

## Mutieren

### NSE-Version

```
mutate(mtcars, displ_l = displ / 61.0237)
```

### SE-Version

```
mutate_(
 .data = mtcars,
 .dots = list(
 "displ_l" = lazyeval::interp(
 ~ x / 61.0237, x = quote(displ)
)
)
)
```

Nichtstandardmäßige Bewertung und Standardbewertung online lesen:

<https://riptutorial.com/de/r/topic/9365/nichtstandardma-ige-bewertung-und-standardbewertung>

---

# Kapitel 76: Numerische Klassen und Speichermodi

## Examples

### Numerisch

Numerisch stellt Ganzzahlen und Verdopplungen dar und ist der Standardmodus, der Zahlenvektoren zugewiesen wird. Die Funktion `is.numeric()` prüft, ob ein Vektor numerisch ist. Es ist wichtig zu beachten, dass `is.numeric()` und Doubles zwar `is.numeric()`, die Funktion `as.numeric()` versucht jedoch immer, in den Typ `double` zu konvertieren.

```
x <- 12.3
y <- 12L

#confirm types
typeof(x)
[1] "double"
typeof(y)
[1] "integer"

confirm both numeric
is.numeric(x)
[1] TRUE
is.numeric(y)
[1] TRUE

logical to numeric
as.numeric(TRUE)
[1] 1

While TRUE == 1, it is a double and not an integer
is.integer(as.numeric(TRUE))
[1] FALSE
```

**Doubles** sind der numerische Standardwert von R. Sie sind Vektoren mit doppelter Genauigkeit, was bedeutet, dass sie für jeden Wert im Vektor 8 Byte Speicher benötigen. R hat keinen Datentyp mit einfacher Genauigkeit, daher werden alle reellen Zahlen im Format mit doppelter Genauigkeit gespeichert.

```
is.double(1)
TRUE
is.double(1.0)
TRUE
is.double(1L)
FALSE
```

**Ganzzahlen** sind ganze Zahlen, die ohne gebrochene Komponente geschrieben werden können. Ganzzahlen werden durch eine Zahl mit einem L dargestellt. Jede Zahl ohne ein L danach wird als

## Doppelzahl betrachtet.

```
typeof(1)
[1] "double"
class(1)
[1] "numeric"
typeof(1L)
[1] "integer"
class(1L)
[1] "integer"
```

Obwohl in den meisten Fällen die Verwendung einer ganzen oder doppelten Zahl keine Rolle spielt, verbraucht das Ersetzen von doppelten Zahlen durch ganze Zahlen manchmal weniger Speicher und Betriebszeit. Ein Doppelvektor verwendet 8 Bytes pro Element, während ein Ganzzahlvektor nur 4 Bytes pro Element verwendet. Wenn die Größe der Vektoren zunimmt, kann die Verwendung geeigneter Typen Prozesse erheblich beschleunigen.

```
test speed on lots of arithmetic
microbenchmark(
 for(i in 1:100000){
 2L * i
 10L + i
 },

 for(i in 1:100000){
 2.0 * i
 10.0 + i
 }
)
Unit: milliseconds
```

|                      | expr   | min       | lq       | mean     | median   | uq       |          |
|----------------------|--------|-----------|----------|----------|----------|----------|----------|
| max neval            |        |           |          |          |          |          |          |
| for (i in 1:1e+05) { | 2L * i | 10L + i } | 40.74775 | 42.34747 | 50.70543 | 42.99120 | 65.46864 |
| 94.11804             | 100    |           |          |          |          |          |          |
| for (i in 1:1e+05) { | 2 * i  | 10 + i }  | 41.07807 | 42.38358 | 53.52588 | 44.26364 | 65.84971 |
| 83.00456             | 100    |           |          |          |          |          |          |

Numerische Klassen und Speichermodi online lesen:

<https://riptutorial.com/de/r/topic/9018/numerische-klassen-und-speichermodi>

# Kapitel 77: Objektorientierte Programmierung in R

## Einführung

Diese Dokumentationsseite beschreibt die vier Objektsysteme in R und deren Gemeinsamkeiten und Unterschiede. Ausführlichere Informationen zu jedem einzelnen System finden Sie auf einer eigenen Themenseite.

Die vier Systeme sind: S3, S4, Referenzklassen und S6.

## Examples

### S3

Das S3-Objektsystem ist ein sehr einfaches OO-System in R.

Jedes Objekt hat eine S3-Klasse. Es kann mit der Funktionsklasse `class` .

```
> class(3)
[1] "numeric"
```

Es kann auch mit der Funktion eingestellt wird `class` :

```
> bicycle <- 2
> class(bicycle) <- 'vehicle'
> class(bicycle)
[1] "vehicle"
```

Es kann auch mit der Funktion `attr` :

```
> velocipede <- 2
> attr(velocipede, 'class') <- 'vehicle'
> class(velocipede)
[1] "vehicle"
```

Ein Objekt kann viele Klassen haben:

```
> class(x = bicycle) <- c('human-powered vehicle', class(x = bicycle))
> class(x = bicycle)
[1] "human-powered vehicle" "vehicle"
```

Bei Verwendung einer generischen Funktion verwendet R das erste Element der Klasse, für das ein generisches Element verfügbar ist.

Zum Beispiel:

```
> summary.vehicle <- function(object, ...) {
+ message('this is a vehicle')
+ }
> summary(object = my_bike)
this is a vehicle
```

Aber wenn wir jetzt eine `summary.bicycle` definieren, `summary.bicycle` :

```
> summary.bicycle <- function(object, ...) {
+ message('this is a bicycle')
+ }
> summary(object = my_bike)
this is a bicycle
```

Objektorientierte Programmierung in R online lesen:

<https://riptutorial.com/de/r/topic/9723/objektorientierte-programmierung-in-r>

---

# Kapitel 78: Operationen einstellen

## Bemerkungen

Ein Set enthält nur eine Kopie jedes einzelnen Elements. Im Gegensatz zu anderen Programmiersprachen verfügt Base R nicht über einen dedizierten Datentyp für Sätze. Stattdessen behandelt R einen Vektor wie eine Menge, indem nur seine verschiedenen Elemente genommen werden. Dies gilt für die Mengenoperatoren `setdiff`, `intersect`, `union`, `setequal` und `%in%`. Für `v %in% s` wird nur `s` als Menge behandelt, nicht jedoch der Vektor `v`.

Für einen True Set-Datentyp in R bietet das Rcpp-Paket [einige Optionen](#).

## Examples

Legen Sie Operatoren für Vektorpaare fest

---

## Sets vergleichen

In R kann ein Vektor doppelte Elemente enthalten:

```
v = "A"
w = c("A", "A")
```

Ein Satz enthält jedoch nur eine Kopie jedes Elements. R behandelt einen Vektor wie eine Menge, indem nur seine unterschiedlichen Elemente genommen werden, sodass die beiden obigen Vektoren als gleich betrachtet werden

```
setequal(v, w)
TRUE
```

---

## Sets kombinieren

Die Schlüsselfunktionen haben natürliche Namen:

```
x = c(1, 2, 3)
y = c(2, 4)

union(x, y)
1 2 3 4

intersect(x, y)
2

setdiff(x, y)
1 3
```

Diese sind alle auf derselben Seite (`?union` dokumentiert).

## Legen Sie die Mitgliedschaft für Vektoren fest

Der Operator `%in%` vergleicht einen Vektor mit einer Menge.

```
v = "A"
w = c("A", "A")

w %in% v
TRUE TRUE

v %in% w
TRUE
```

Jedes Element auf der linken Seite wird einzeln behandelt und auf Mitgliedschaft in der Menge geprüft, die dem Vektor auf der rechten Seite zugeordnet ist (bestehend aus allen seinen verschiedenen Elementen).

Im Gegensatz zu Gleichheitstests gibt `%in%` immer `TRUE` oder `FALSE` :

```
c(1, NA) %in% c(1, 2, 3, 4)
TRUE FALSE
```

Die Dokumentation liegt bei `?`%in%``.

## Kartesische oder "Kreuz" -Produkte von Vektoren

Um jeden Vektor der Form (x, y) zu finden, in dem x vom Vektor X und y von Y gezeichnet wird, verwenden wir `expand.grid` :

```
X = c(1, 1, 2)
Y = c(4, 5)

expand.grid(X, Y)

Var1 Var2
1 1 4
2 1 4
3 2 4
4 1 5
5 1 5
6 2 5
```

Das Ergebnis ist ein `data.frame` mit einer Spalte für jeden übergebenen Vektor. Oft wollen wir das kartesische Produkt von Mengen nehmen, anstatt ein "Gitter" von Vektoren zu erweitern. Wir können `unique`, `lapply` und `do.call` :

```
m = do.call(expand.grid, lapply(list(X, Y), unique))

Var1 Var2
1 1 4
```

```
2 2 4
3 1 5
4 2 5
```

## Funktionen auf Kombinationen anwenden

Wenn Sie dann auf jede resultierende Kombination  $f(x,y)$  eine Funktion anwenden möchten, kann diese als weitere Spalte hinzugefügt werden:

```
m$p = with(m, Var1*Var2)
Var1 Var2 p
1 1 4 4
2 2 4 8
3 1 5 5
4 2 5 10
```

Dieser Ansatz funktioniert für so viele Vektoren, wie wir brauchen, aber im speziellen Fall von zwei ist es manchmal besser, das Ergebnis in einer Matrix zu haben, die mit `outer` erreicht werden kann:

```
uX = unique(X)
uY = unique(Y)

outer(setNames(uX, uX), setNames(uY, uY), `*`)

4 5
1 4 5
2 8 10
```

Zu verwandten Konzepten und Werkzeugen siehe das Thema Kombinatorik.

## Erstellen Sie eindeutige Elemente, und legen Sie doppelte Elemente aus einem Vektor ab

`unique` legt Duplikate ab, sodass jedes Element im Ergebnis eindeutig ist (nur einmal erscheint):

```
x = c(2, 1, 1, 2, 1)

unique(x)
2 1
```

Die Werte werden in der Reihenfolge zurückgegeben, in der sie zuerst angezeigt wurden.

`duplicated` Tags für jedes duplizierte Element:

```
duplicated(x)
FALSE FALSE TRUE TRUE TRUE
```

`anyDuplicated(x) > 0L` schnell überprüfen, ob ein Vektor Duplikate enthält.



## Messsatzüberdeckungen / Venn-Diagramme für Vektoren

Um zu zählen, wie viele Elemente von zwei Sätzen sich überlappen, könnte man eine benutzerdefinierte Funktion schreiben:

```
xtab_set <- function(A, B){
 both <- union(A, B)
 inA <- both %in% A
 inB <- both %in% B
 return(table(inA, inB))
}
```

```
A = 1:20
B = 10:30
```

```
xtab_set(A, B)
```

```
inB
inA FALSE TRUE
FALSE 0 10
TRUE 9 11
```

Ein Venn-Diagramm, das von verschiedenen Paketen angeboten wird, kann verwendet werden, um Überlappungszählungen über mehrere Sätze hinweg zu visualisieren.

Operationen einstellen online lesen: <https://riptutorial.com/de/r/topic/1383/operationen-einstellen>

# Kapitel 79: ordentlich

## Examples

### Tbl\_df erstellen

A `tbl_df` (ausgesprochen *Tibble diff*) ist eine Variation eines [Datenrahmens](#), der oft in tidyverse Paketen verwendet wird. Es ist im [tibble](#)-Paket implementiert.

Verwenden Sie die Funktion `as_data_frame`, um einen Datenrahmen in eine `Tbl_df`-Datei umzuwandeln:

```
library(tibble)
mtcars_tbl <- as_data_frame(mtcars)
```

Einer der bemerkenswertesten Unterschiede zwischen `data.frames` und `tbl_dfs` ist, wie sie gedruckt werden:

```
A tibble: 32 x 11
 mpg cyl disp hp drat wt qsec vs am gear carb
* <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4
2 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4
3 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1
4 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1
5 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2
6 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1
7 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4
8 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2
9 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2
10 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4
... with 22 more rows
```

- Die gedruckte Ausgabe enthält eine Zusammenfassung der Abmessungen der Tabelle ( 32 x 11 ).
- Es beinhaltet den Typ jeder Spalte ( `dbl` )
- Es gibt eine begrenzte Anzahl von Zeilen aus. (Um dies zu ändern, verwenden Sie `options(tibble.print_max = [number])` ).

Viele Funktionen im `dplyr`-Paket funktionieren natürlich mit `tbl_dfs`, wie beispielsweise `group_by()`.

### Tidyverse: eine Übersicht

## Was ist `tidyverse` ?

`tidyverse` ist der schnelle und elegante Weg, um aus Basic R ein verbessertes Werkzeug zu machen, das von Hadley / Rstudio neu gestaltet wurde. Die Entwicklung aller in `tidyverse`

enthaltenen `tidyverse` folgt den Grundregeln des [Manifestes](#) von [The tidy tools](#) . Aber lassen Sie die Autoren zuerst ihr Meisterwerk beschreiben:

Tidyverse besteht aus einer Reihe von Paketen, die in Harmonie arbeiten, da sie gemeinsame Datenrepräsentationen und API-Design verwenden. Das Paket `tidyverse` ist so konzipiert, dass es einfach ist, Core-Pakete mit einem einzigen Befehl von `tidyverse` zu installieren und zu laden.

Der beste Ort, um mehr über die Pakete in Ordnung zu erfahren, und wie sie zusammenpassen, ist R for Data Science. Erwarten Sie in den kommenden Monaten mehr über die Ordnung in Ordnung, während ich an verbesserten Paket-Websites arbeite, die Zitierweise vereinfachen und ein gemeinsames Zuhause für Diskussionen über die Datenanalyse mit der Ordnung bereitstellen.

( [Quelle](#) )

---

## Wie benutze ich es?

Nur bei den normalen `R` Paketen müssen Sie das Paket installieren und laden.

```
install.packages("tidyverse")
library("tidyverse")
```

Der Unterschied besteht darin, dass mit einem einzigen Befehl mehrere Dutzend Pakete installiert / geladen werden. Als Bonus kann man sich darauf verlassen, dass alle installierten / geladenen Pakete kompatible Versionen haben.

---

## Was sind das für Pakete?

Die allgemein bekannten und weit verbreiteten Pakete:

- [ggplot2](#) : erweiterte Datenvisualisierung [SO\\_doc](#)
- [dplyr](#) : schneller ( [Rcpp](#) ) und kohärenter Ansatz zur Datenmanipulation [SO\\_doc](#)
- [tidyr](#) : Werkzeuge zur Datenaufbereitung von [SO\\_doc](#)
- [readr](#) : für den Datenimport.
- [purrr](#) : lässt Ihre reinen Funktionen schnurren, indem Sie die funktionalen Programmierwerkzeuge von R mit wichtigen Funktionen aus anderen Sprachen im Stil der JS-Pakete `underscore.js`, `lodash` und `lazy.js`.
- [tibble](#) : eine moderne umgestaltung von [datenrahmen](#) .
- [magrittr](#) : [piping](#) , um Code lesbarer zu machen [SO\\_doc](#)

Pakete zur Bearbeitung bestimmter Datenformate:

- [hms](#) : einfach mal ablesen
- [stringr](#) : [Bietet](#) einen zusammenhängenden Satz von Funktionen, der das Arbeiten mit Strings so einfach wie möglich macht

- [lubridate](#) : erweiterte Datums- / Zeitmanipulationen [SO\\_doc](#)
- [forcats](#) : Fortgeschrittene Arbeit mit [Faktoren](#) .

#### Datenimport:

- [DBI](#) : Definiert eine gemeinsame Schnittstelle zwischen R und Datenbankverwaltungssystemen (DBMS).
- [Port](#) : [SPSS](#)- , [SAS](#)- und [Stata](#)-Dateien [SO\\_doc](#) können problemlos importiert werden
- [httr](#) : Das Ziel von [httr](#) ist es, einen Wrapper für das [curl](#)-Paket bereitzustellen, der an die Anforderungen moderner Web-APIs angepasst ist
- [jsonlite](#) : ein schneller JSON-Parser und Generator, der für statistische Daten und das Web optimiert ist
- [readxl](#) : [read.xls](#)- und [.xlsx](#)-Dateien, ohne dass die Abhängigkeitspakete [SO\\_doc](#) erforderlich sind
- [rvest](#) : [rvest](#) hilft Ihnen, Informationen von den Webseiten [SO\\_doc](#) abzurufen
- [xml2](#) : für XML

#### Und modellieren:

- [modelr](#) : bietet Funktionen, mit denen Sie beim Modellieren elegante Pipelines erstellen können
- [Besen](#) : Einfach die Modelle in ordentliche Daten extrahieren

#### Schließlich `tidyverse` die Verwendung von:

- [knitr](#) : Die erstaunliche universelle Programmier-Engine für allgemeine Kenntnisse mit leichtgewichtigen APIs, die Benutzern die volle Kontrolle über die Ausgabe ohne umfangreiche Codierungsarbeit ermöglichen. [SO\\_docs](#): [eins](#) , [zwei](#)
- [rmarkdown](#) : Rstudio-Paket für reproduzierbare Programmierung. [SO\\_docs](#): [eins](#) , [zwei](#) , [drei](#) , [vier](#)

ordentlich online lesen: <https://riptutorial.com/de/r/topic/1395/ordentlich>

# Kapitel 80: Pakete installieren

## Syntax

- `install.packages` (pkgs, lib, repos, method, destdir, abhängigkeiten, ...)

## Parameter

| Parameter      | Einzelheiten                                                                                                                                                                                                                    |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkgs           | Zeichenvektor der Namen von Paketen. Wenn <code>repos = NULL</code> , ein Zeichenvektor der Dateipfade.                                                                                                                         |
| lib            | Zeichenvektor, der den Bibliotheksverzeichnis angibt, wo die Pakete installiert werden sollen.                                                                                                                                  |
| Repos          | Zeichenvektor, die Basis-URL (s) der zu verwendenden Repositorys, kann aus lokalen Dateien <code>NULL</code>                                                                                                                    |
| Methode        | Download-Methode                                                                                                                                                                                                                |
| destdir        | Verzeichnis, in dem heruntergeladene Pakete gespeichert werden                                                                                                                                                                  |
| Abhängigkeiten | logisch, um anzugeben, ob auch deinstallierte Pakete installiert werden sollen, bei denen diese Pakete von / link / import / suggest (und so weiter, rekursiv) abhängen. Wird nicht verwendet, wenn <code>repos = NULL</code> . |
| ...            | Argumente, die an 'download.file' oder an die Funktionen für binäre Installationen unter OS X und Windows übergeben werden sollen.                                                                                              |

## Bemerkungen

## Verwandte Dokumente

- [Pakete prüfen](#)

## Examples

Laden Sie Pakete aus Repositorys herunter und installieren Sie sie

Pakete sind Sammlungen von R-Funktionen, Daten und kompiliertem Code in einem [genau definierten Format](#). Öffentliche (und private) Repositorys werden zum Hosten von Sammlungen von R-Paketen verwendet. Die größte Sammlung von R-Paketen ist bei CRAN erhältlich.

---

# Verwendung von CRAN

Ein Paket kann mit folgendem Code von [CRAN](#) installiert werden:

```
install.packages("dplyr")
```

Dabei wird "dplyr" als Zeichenvektor bezeichnet.

Sie können mehrere Pakete auf einmal installieren, indem Sie die Kombinationsfunktion `c()` und eine Reihe von Zeichenvektoren mit Paketnamen übergeben:

```
install.packages(c("dplyr", "tidyr", "ggplot2"))
```

In einigen Fällen kann `install.packages` je nach Wert von `getOption("repos")` zur `getOption("repos")` eines CRAN-Spiegels `getOption("repos")` oder fehlschlagen. Um dies zu verhindern, geben Sie als `repos` Argument einen [CRAN-Mirror](#) an:

```
install.packages("dplyr", repos = "https://cloud.r-project.org/")
```

Mit dem Argument `repos` ist es auch möglich, andere Repositorys zu installieren. Um vollständige Informationen zu allen verfügbaren Optionen zu erhalten, führen Sie `?install.packages`.

Die meisten Pakete erfordern Funktionen, die in anderen Paketen implementiert wurden (zB das Paket `data.table`). Um ein Paket (oder mehr Pakete) mit allen Paketen zu installieren, die von diesem bestimmten Paket verwendet werden, die Argumente `dependencies` gesetzt werden sollten (`TRUE`):

```
install.packages("data.table", dependencies = TRUE)
```

---

# Verwendung von Bioconductor

[Bioconductor](#) enthält eine umfangreiche Sammlung von Paketen im Zusammenhang mit Bioinformatik. Sie bieten ein eigenes Paketmanagement, das auf der `biocLite` Funktion `biocLite`:

```
Try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
biocLite()
```

Standardmäßig werden einige Pakete installiert, die die am häufigsten verwendeten Funktionen bereitstellen. Bestimmte Pakete können durch Übergabe eines Vektors von Paketnamen installiert werden. So installieren Sie beispielsweise `RImmPort` von Bioconductor:

```
source("https://bioconductor.org/biocLite.R")
biocLite("RImmPort")
```

## Paket aus lokaler Quelle installieren

So installieren Sie das Paket aus der lokalen Quelldatei:

```
install.packages(path_to_source, repos = NULL, type="source")

install.packages("~/Downloads/dplyr-master.zip", repos=NULL, type="source")
```

`path_to_source` ist hier der absolute Pfad der lokalen Quelldatei.

Ein weiterer Befehl, der ein Fenster zum Auswählen heruntergeladener ZIP- oder tar.gz-Quelldateien öffnet, ist:

```
install.packages(file.choose(), repos=NULL)
```

---

Eine andere Möglichkeit ist die Verwendung des *GUI-basierten RStudio* :

**Schritt 1:** Gehen Sie zu **Tools** .

**Schritt 2:** Installieren Sie Pakete .

**Schritt 3:** Legen Sie in der *Installation* als **Paketarchivdatei fest (.zip; .tar.gz)**.

**Schritt 4:** Dann *Durchsuchen* Sie Ihre Paketdatei (zB `crayon_1.3.1.zip`) und *nach einiger Zeit* finden (*nachdem es den **Paketpfad** zeigt **und** den **Namen** im Package Archive Registerkarte **Datei***)

---

Eine andere Möglichkeit, ein R-Paket aus lokaler Quelle zu installieren, ist die Verwendung der Funktion `install_local()` aus dem Paket `devtools`.

```
library(devtools)
install_local("~/Downloads/dplyr-master.zip")
```

## Installiere Pakete von GitHub

Um Pakete direkt über GitHub zu installieren, verwenden Sie das Paket `devtools` :

```
library(devtools)
install_github("authorName/repositoryName")
```

So installieren Sie `ggplot2` von github:

```
devtools::install_github("tidyverse/ggplot2")
```

Der obige Befehl wird die Version von installieren `ggplot2` , die mit dem *Master* - Zweig entspricht. Um von einem anderen Zweig eines Repositorys zu installieren, geben Sie den Namen des Zweigs mit dem Argument `ref` an. Mit dem folgenden Befehl wird beispielsweise der Zweig

dev\_general des googleway Pakets installiert.

```
devtools::install_github("SymbolixAU/googleway", ref = "dev_general")
```

Eine andere Option ist die Verwendung des ghitr Pakets. Es bietet eine einfache Alternative für die Installation von Paketen von github:

```
install.packages("ghitr")
ghitr::install_github("google/CausalImpact")
```

Um ein Paket zu installieren, das sich in einem **privaten** Repository auf Github befindet, generieren Sie ein **persönliches Zugriffstoken** unter <http://www.github.com/settings/tokens/> . Folge diesen Schritten:

1. 

```
install.packages(c("curl", "httr"))
```
2. 

```
config = httr::config(ssl_verifypeer = FALSE)
```
3. 

```
install.packages("RCurl")
options(RCurlOptions = c(getOption("RCurlOptions"), ssl.verifypeer = FALSE,
ssl.verifyhost = FALSE))
```
4. 

```
getOption("RCurlOptions")
```

Sie sollten Folgendes sehen:

```
ssl.verifypeer ssl.verifyhost
FALSE FALSE
```

5. 

```
library(httr)
set_config(config(ssl_verifypeer = 0L))
```

Dies verhindert den allgemeinen Fehler: "Peer-Zertifikat kann nicht mit angegebenen CA-Zertifikaten authentifiziert werden"

6. Verwenden Sie schließlich den folgenden Befehl, um Ihr Paket nahtlos zu installieren

```
install_github("username/package_name", auth_token="abc")
```

Setzen Sie alternativ eine Umgebungsvariable GITHUB\_PAT mit

```
Sys.setenv(GITHUB_PAT = "access_token")
devtools::install_github("organisation/package_name")
```

Die in Github generierte PAT ist nur einmal sichtbar, dh, wenn sie ursprünglich erstellt wurde, ist es .Rprofile , das Token in .Rprofile zu speichern. Dies ist auch hilfreich, wenn die Organisation über viele private Repositories verfügt.



## Verwenden eines CLI-Paketmanagers - grundlegende Pacman-Verwendung

`pacman` ist ein einfacher Paketmanager für R.

`pacman` ermöglicht es einem Benutzer, alle gewünschten Pakete mit einem einzigen Befehl, `p_load`, kompakt zu laden und fehlende Pakete (und deren Abhängigkeiten) zu `p_load.pacman` fordert den Benutzer nicht auf, einen Paketnamen in Anführungszeichen zu setzen. Grundlegende Verwendung ist wie folgt:

```
p_load(data.table, dplyr, ggplot2)
```

Das einzige Paket eine erfordern `library`, `require`, oder `install.packages` Erklärung mit diesem Ansatz ist `pacman` selbst:

```
library(pacman)
p_load(data.table, dplyr, ggplot2)
```

oder gleich gültig:

```
pacman::p_load(data.table, dplyr, ggplot2)
```

`pacman` spart nicht nur Zeit, da weniger Code für die Verwaltung von Paketen erforderlich ist, sondern auch die Erstellung von reproduzierbarem Code, indem alle erforderlichen Pakete installiert werden, sofern diese noch nicht installiert sind.

Da Sie möglicherweise nicht sicher sind, ob `pacman` in der Bibliothek eines Benutzers installiert ist, der Ihren Code verwendet (oder für sich selbst bei zukünftiger Verwendung Ihres eigenen Codes), `pacman` es sich, eine bedingte Anweisung zur Installation von `pacman` falls dies nicht bereits der `pacman` ist geladen:

```
if(!(require(pacman)) install.packages("pacman"))
pacman::p_load(data.table, dplyr, ggplot2)
```

## Installieren Sie die lokale Entwicklungsversion eines Pakets

Bei der Entwicklung eines R-Pakets muss häufig die neueste Version des Pakets installiert werden. Dies kann erreicht werden, indem zuerst eine Quellverteilung des Pakets erstellt wird (in der Befehlszeile).

```
R CMD build my_package
```

und dann [in R installieren](#). Alle laufenden R-Sitzungen mit einer vorherigen Version des Pakets müssen neu geladen werden.

```
unloadNamespace("my_package")
library(my_package)
```

Ein praktischerer Ansatz verwendet das Paket `devtools` , um den Prozess zu vereinfachen. In einer R-Sitzung, wobei das Arbeitsverzeichnis auf das Paketverzeichnis gesetzt ist

```
devtools::install()
```

wird das Paket erstellen, installieren und neu laden.

Pakete installieren online lesen: <https://riptutorial.com/de/r/topic/1719/pakete-installieren>

---

# Kapitel 81: Pakete mit devtools erstellen

## Einführung

Dieses Thema behandelt die Erstellung von R-Paketen von Grund auf mit dem Paket devtools.

## Bemerkungen

1. [Offizielles R-Handbuch zum Erstellen von Paketen](#)
2. [Referenzhandbuch für roxygen2](#)
3. [devtools Referenzhandbuch](#)

## Examples

### Pakete erstellen und verteilen

Dies ist eine *kompakte Anleitung* zum schnellen Erstellen eines R-Pakets aus Ihrem Code. Umfassende Dokumentationen werden, wenn verfügbar, verlinkt und sollten gelesen werden, wenn Sie die Situation besser verstehen möchten. Siehe *Anmerkungen* für weitere Ressourcen.

Das Verzeichnis, in dem Ihr Code steht, wird als `./`, und alle Befehle sollen von einer R-Eingabeaufforderung in diesem Ordner aus ausgeführt werden.

---

## Erstellung der Dokumentation

Die Dokumentation für Ihren Code muss in einem Format vorliegen, das LaTeX sehr ähnlich ist.

Wir werden jedoch ein Tool namens `roxygen` verwenden, um den Prozess zu vereinfachen:

```
install.packages("devtools")
library("devtools")
install.packages("roxygen2")
library("roxygen2")
```

Die vollständige Manpage für roxygen finden Sie [hier](#). Es ist dem *Sauerstoff* sehr ähnlich.

Hier ein praktisches Beispiel zur Dokumentation einer Funktion mit *roxygen* :

```
#' Increment a variable.
#'
#' Note that the behavior of this function
#' is undefined if `x` is not of class `numeric`.
#'
#' @export
#' @author another guy
```

```

#' @name Increment Function
#' @title increment
#'
#' @param x Variable to increment
#' @return `x` incremented of 1
#'
#' @seealso `other_function`
#'
#' @examples
#' increment(3)
#' > 4
increment <- function(x) {
 return (x+1)
}

```

Und [hier wird das Ergebnis sein](#) .

Es wird außerdem empfohlen, eine Vignette zu erstellen (siehe Thema *Erstellen von Vignetten* ). Dies ist eine vollständige Anleitung zu Ihrem Paket.

## Bau des Paketskeletts

Unter der Annahme, dass Ihr Code beispielsweise in die Dateien `./script1.R` und `./script2.R` , starten Sie den folgenden Befehl, um den Dateibaum Ihres Pakets zu erstellen:

```
package.skeleton(name="MyPackage", code_files=c("script1.R","script2.R"))
```

Löschen Sie dann alle Dateien in `./MyPackage/man/` . Sie müssen nun die Dokumentation zusammenstellen:

```
roxygenize("MyPackage")
```

Sie sollten auch ein Referenzhandbuch aus Ihrer Dokumentation mit `R CMD Rd2pdf MyPackage` über eine in `./` gestartete *Eingabeaufforderung* `R CMD Rd2pdf MyPackage` .

## Edition der Paketeigenschaften

### 1. Beschreibung des Pakets

`./MyPackage/DESCRIPTION` Sie `./MyPackage/DESCRIPTION` Ihren Anforderungen entsprechend. Die Felder `Package` , `Version` , `License` , `Description` , `Title` , `Author` und `Maintainer` sind Pflichtfelder, die anderen sind optional.

Wenn Ihr Paket von anderen Paketen abhängig ist, geben Sie diese in einem Feld mit dem Namen `Depends` (`R-Version <3.2.0`) oder `Imports` (`R-Version > 3.2.0`) an.

## 2. Optionale Ordner

Nachdem Sie den Skeleton-Build gestartet haben, hatte `./MyPackage/` nur noch `R/` und `man/` Unterordner. Es kann jedoch einige andere haben:

- `data/` : Hier können Sie die Daten platzieren, die Ihre Bibliothek benötigt und die kein Code sind. Es muss als Datensatz mit der Erweiterung `.RData` werden und Sie können es zur Laufzeit mit `data()` und `load()`
- `tests/` : Alle Codedateien in diesem Ordner werden zur Installationszeit ausgeführt. Wenn ein Fehler auftritt, schlägt die Installation fehl.
- `src/` : für C / C ++ / Fortran-Quelldateien, die Sie benötigen (mithilfe von `Rcpp` ...).
- `exec/` : für andere ausführbare Dateien.
- `misc/` : für fast alles andere.

---

## Fertigstellung und Build

Sie können `./MyPackage/Read-and-delete-me` löschen.

Das Paket kann jetzt installiert werden.

Sie können es mit `devtools::install("MyPackage")` .

Um Ihr Paket als Quell-Tarball zu erstellen, müssen Sie den folgenden Befehl über eine *Eingabeaufforderung* in `./` : `R CMD build MyPackage`

---

## Verteilung Ihres Pakets

### Durch Github

Erstellen Sie einfach ein neues Repository mit dem Namen *MyPackage* und laden Sie alles in `MyPackage/` in den Master-Zweig `MyPackage/` . Hier ist [ein Beispiel](#) .

Dann kann jeder Ihr Paket von github mit devtools installieren:

```
install_package("MyPackage", "your_github_username")
```

### Durch CRAN

Ihr Paket muss der [CRAN-Repository-Richtlinie entsprechen](#) . Einschließlich, aber nicht beschränkt auf: Ihr Paket muss plattformübergreifend sein (mit Ausnahme einiger sehr spezieller Fälle). Es sollte den `R CMD check` Prüfungstest bestehen.

Hier ist das [Einreichungsformular](#) . Sie müssen das Quell-Tarball hochladen.

## Vignetten erstellen

Eine Vignette ist eine ausführliche Anleitung für Ihr Paket. Die Funktionsdokumentation ist großartig, wenn Sie den Namen der gewünschten Funktion kennen, ansonsten aber nichts. Eine Vignette ist wie ein Buchkapitel oder eine wissenschaftliche Arbeit: Sie kann das Problem beschreiben, das Ihr Paket lösen soll, und dem Leser dann zeigen, wie er es lösen kann.

Vignetten werden vollständig in Markdown erstellt.

## Bedarf

- Rmarkdown: `install.packages("rmarkdown")`
- [Pandoc](#)

## Vignetten-Erstellung

```
devtools::use_vignette("MyVignette", "MyPackage")
```

Sie können Ihre Vignette jetzt unter `./vignettes/MyVignette.Rmd` bearbeiten.

Der Text in Ihrer Vignette wird als [Markdown](#) formatiert.

Der einzige Zusatz zum ursprünglichen Markdown ist ein Tag, das R-Code übernimmt, ausführt, die Ausgabe erfasst und in formatiertes Markdown übersetzt:

```
```${r}
# Add two numbers together
add <- function(a, b) a + b
add(10, 20)
```
```

Wird angezeigt als:

```
Add two numbers together
add <- function(a, b) a + b
add(10, 20)
[1] 30
```

Daher müssen alle Pakete, die Sie in Ihren Vignetten verwenden, als Abhängigkeiten in

`./DESCRIPTION` .

**Pakete mit devtools erstellen online lesen:** <https://riptutorial.com/de/r/topic/10884/pakete-mit-devtools-erstellen>

---

# Kapitel 82: Pakete prüfen

## Einführung

Pakete bauen auf Basis R auf. In diesem Dokument wird erläutert, wie installierte Pakete und ihre Funktionalität geprüft werden. Verwandte Dokumente: [Pakete installieren](#)

## Bemerkungen

Das Comprehensive R Archive Network (CRAN) ist das primäre [Paket-Repository](#) .

## Examples

### Paketinformationen anzeigen

So rufen Sie Informationen zum dplyr-Paket und seinen Funktionsbeschreibungen ab:

```
help(package = "dplyr")
```

Sie müssen das Paket nicht erst laden.

### Zeigen Sie die integrierten Datensätze des Pakets an

Um eingebaute Datensätze aus dem Paket dplyr anzuzeigen

```
data(package = "dplyr")
```

Sie müssen das Paket nicht erst laden.

### Listen Sie die exportierten Funktionen eines Pakets auf

Um die Liste der Funktionen im Paket dplyr zu erhalten, müssen Sie das Paket zunächst laden:

```
library(dplyr)
ls("package:dplyr")
```

### Paketversion anzeigen

Bedingungen: Das Paket sollte mindestens installiert sein. Wenn nicht in der aktuellen Sitzung geladen, kein Problem.

```
Checking package version which was installed at past or
installed currently but not loaded in the current session

packageVersion("seqinr")
```

```
[1] '3.3.3'
packageVersion("RWeka")
[1] '0.4.29'
```

## Geladene Pakete in der aktuellen Sitzung anzeigen

So überprüfen Sie die Liste der geladenen Pakete

```
search()
```

ODER

```
(.packages())
```

Pakete prüfen online lesen: <https://riptutorial.com/de/r/topic/7408/pakete-prufen>



# Kapitel 83: Parallelverarbeitung

## Bemerkungen

Für die Parallelisierung auf Remote-Computern müssen Bibliotheken auf jedem Computer heruntergeladen werden. `package::function()` Aufrufe bevorzugen. Mehrere Pakete verfügen über eine eingebaute Parallelisierung, einschließlich `caret`, `pls` und `plyr`.

[Microsoft R Open](#) (Revolution R) verwendet auch Multithread-BLAS / LAPACK-Bibliotheken, die viele allgemeine Funktionen an sich parallelisieren.

## Examples

### Parallelverarbeitung mit foreach-Paket

Mit dem `foreach` Paket wird die Leistung der parallelen Verarbeitung erweitert. Bevor Sie jedoch Multi-Core-CPU's verwenden möchten, müssen Sie einen Multi-Core-Cluster zuweisen. Das `doSNOW` Paket ist eine Möglichkeit.

Eine einfache Anwendung der `foreach`-Schleife besteht darin, die Summe der Quadratwurzel und des Quadrats aller Zahlen von 1 bis 100000 zu berechnen.

```
library(foreach)
library(doSNOW)

cl <- makeCluster(5, type = "SOCK")
registerDoSNOW(cl)

f <- foreach(i = 1:100000, .combine = c, .inorder = F) %dopar% {
 k <- i ** 2 + sqrt(i)
 k
}
```

Die Struktur der Ausgabe von `foreach` wird durch das `.combine` Argument gesteuert. Die Standardausgabestruktur ist eine Liste. Im obigen Code wird stattdessen `c` verwendet, um einen Vektor zurückzugeben. Es ist zu beachten, dass eine Berechnungsfunktion (oder ein Operator) wie `+` auch verwendet werden kann, um eine Berechnung durchzuführen und ein weiterverarbeitetes Objekt zurückzugeben.

Es ist wichtig zu erwähnen, dass das Ergebnis jeder `foreach`-Schleife der letzte Aufruf ist. In diesem Beispiel wird also `k` zum Ergebnis addiert.

| Parameter                 | Einzelheiten                                                                                                                                                                                           |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.kombinieren</code> | Funktion kombinieren. Legt fest, wie die Ergebnisse der Schleife kombiniert werden. Mögliche Werte sind <code>c</code> , <code>cbind</code> , <code>rbind</code> , <code>+</code> , <code>*</code> ... |
| <code>.in Ordnung</code>  | Wenn der Wert <code>TRUE</code> das Ergebnis gemäß der Reihenfolge der Iteration                                                                                                                       |

| Parameter            | Einzelheiten                                                                                                                                                                                                                                  |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | vairable (hier <code>i</code> ) geordnet. Bei <code>FALSE</code> das Ergebnis nicht bestellt. Dies kann positive Auswirkungen auf die Rechenzeit haben.                                                                                       |
| <code>.Pakete</code> | Für Funktionen, die von einem Paket mit Ausnahme von <code>base</code> bereitgestellt werden, wie beispielsweise <code>mass</code> , <code>randomForest</code> oder sonst, müssen Sie diese Pakete mit <code>c("mass", "randomForest")</code> |

## Parallelverarbeitung mit Parallelpaket

Das Basispaket `parallel` ermöglicht die parallele Berechnung durch Gabelung, Sockets und Erzeugung von Zufallszahlen.

Ermitteln Sie die Anzahl der auf dem localhost vorhandenen Kerne:

```
parallel::detectCores(all.tests = FALSE, logical = TRUE)
```

Erstellen Sie einen Cluster der Kerne auf dem localhost:

```
parallelCluster <- parallel::makeCluster(parallel::detectCores())
```

Zunächst muss eine für die Parallelisierung geeignete Funktion erstellt werden. Betrachten Sie das `mtcars` Dataset. Eine Regression bei `mpg` könnte durch Erstellen eines separaten Regressionsmodells für jede `cyl` Ebene verbessert werden.

```
data <- mtcars
yfactor <- 'cyl'
zlevels <- sort(unique(data[[yfactor]]))
datay <- data[,1]
dataz <- data[,2]
datax <- data[,3:11]

fitmodel <- function(zlevel, datax, datay, dataz) {
 glm.fit(x = datax[dataz == zlevel,], y = datay[dataz == zlevel])
}
```

Erstellen Sie eine Funktion, die alle möglichen Iterationen von `zlevels` . Dies ist immer noch in Serie, aber es ist ein wichtiger Schritt, da es den genauen Prozess bestimmt, der parallelisiert wird.

```
fitmodel <- function(zlevel, datax, datay, dataz) {
 glm.fit(x = datax[dataz == zlevel,], y = datay[dataz == zlevel])
}

for (zlevel in zlevels) {
 print("*****")
 print(zlevel)
 print(fitmodel(zlevel, datax, datay, dataz))
}
```

Curry diese Funktion:

```
worker <- function(zlevel) {
 fitmodel(zlevel,datax, datay, dataz)
}
```

Paralleles Computing mit `parallel` kann nicht auf die globale Umgebung zugreifen. Glücklicherweise erstellt jede Funktion eine lokale Umgebung, auf die `parallel` kann. Die Erstellung einer Wrapper-Funktion ermöglicht die Parallelisierung. Die anzuwendende Funktion muss auch innerhalb der Umgebung platziert werden.

```
wrapper <- function(datax, datay, dataz) {
 # force evaluation of all paramters not supplied by parallelization apply
 force(datax)
 force(datay)
 force(dataz)
 # these variables are now in an enviroment accessible by parallel function

 # function to be applied also in the environment
 fitmodel <- function(zlevel, datax, datay, dataz) {
 glm.fit(x = datax[dataz == zlevel,], y = datay[dataz == zlevel])
 }

 # calling in this environment iterating over single parameter zlevel
 worker <- function(zlevel) {
 fitmodel(zlevel,datax, datay, dataz)
 }
 return(worker)
}
```

Erstellen Sie nun einen Cluster und führen Sie die Wrapper-Funktion aus.

```
parallelcluster <- parallel::makeCluster(parallel::detectCores())
models <- parallel::parLapply(parallelcluster,zlevels,
 wrapper(datax, datay, dataz))
```

Stoppen Sie den Cluster immer, wenn Sie fertig sind.

```
parallel::stopCluster(parallelcluster)
```

Das `parallel` - Paket enthält die gesamte `apply()` Familie, mit dem Präfix `par` .

## Zufallszahlengenerierung

Ein Hauptproblem bei der Parallelisierung ist die Verwendung von RNG als Saatgut. Zufallszahlen nach Anzahl werden durch die Anzahl der Operationen vom Beginn der Sitzung oder vom letzten `set.seed()` . Da parallele Prozesse aus derselben Funktion stammen, kann derselbe Samen verwendet werden, was möglicherweise zu identischen Ergebnissen führt! Anrufe werden seriell auf den verschiedenen Kernen ausgeführt, bieten jedoch keinen Vorteil.

Ein Satz von Startwerten muss generiert und an jeden parallelen Prozess gesendet werden. Dies geschieht in einigen Paketen automatisch ( `parallel` , `snow` usw.), muss aber in anderen explizit

angesprochen werden.

```
s <- seed
for (i in 1:numofcores) {
 s <- nextRNGStream(s)
 # send s to worker i as .Random.seed
}
```

Samen können auch auf Reproduzierbarkeit eingestellt werden.

```
clusterSetRNGStream(cl = parallelcluster, iseed)
```

## mcparallelDo

Das Paket `mcparallelDo` ermöglicht die asynchrone Bewertung von R-Code unter Unix-ähnlichen Betriebssystemen (z. B. Linux und MacOSX). Die zugrunde liegende Philosophie des Pakets orientiert sich an den Erfordernissen der explorativen Datenanalyse und nicht an der Codierung. Berücksichtigen Sie für die Codierung der Asynchronität das `future` Paket.

## Beispiel

Daten erstellen

```
data(ToothGrowth)
```

Löst `mcparallelDo` aus, um eine Analyse an einer Gabel durchzuführen

```
mcparallelDo({glm(len ~ supp * dose, data=ToothGrowth)},"interactionPredictorModel")
```

Andere Dinge tun, z

```
binaryPredictorModel <- glm(len ~ supp, data=ToothGrowth)
gaussianPredictorModel <- glm(len ~ dose, data=ToothGrowth)
```

Das Ergebnis von `mcparallelDo` wird in Ihrer `targetEnvironment` zurückgegeben, z. B. `.GlobalEnv`, wenn eine Nachricht vollständig ist (standardmäßig)

```
summary(interactionPredictorModel)
```

## Andere Beispiele

```
Example of not returning a value until we return to the top level
for (i in 1:10) {
 if (i == 1) {
 mcparallelDo({2+2}, targetValue = "output")
 }
}
```

```
 if (exists("output")) print(i)
 }

Example of getting a value without returning to the top level
for (i in 1:10) {
 if (i == 1) {
 mcpipelineDo({2+2}, targetValue = "output")
 }
 mcpipelineDoCheck()
 if (exists("output")) print(i)
}
```

Parallelverarbeitung online lesen: <https://riptutorial.com/de/r/topic/1677/parallelverarbeitung>

# Kapitel 84: Pivot und Unpivot mit data.table

## Syntax

- Schmelze `melt(DT, id.vars=c(...), variable.name="CategoryLabel", value.name="Value")`
- Cast mit `dcast(DT, LHS ~ RHS, value.var="Value", fun.aggregate=sum)`

## Parameter

| Parameter      | Einzelheiten                                                                                                                                                                                         |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id.vars        | Sagen Sie der <code>melt</code> welche Säulen sie behalten sollen                                                                                                                                    |
| Variablennamen | tell <code>melt</code> , was die Spalte mit der Kategorie Etiketten aufrufen                                                                                                                         |
| value.name     | Sagen Sie der <code>melt</code> wie die Spalte aufgerufen werden soll, deren Werte mit Kategoriebeschriftungen verknüpft sind                                                                        |
| value.var      | <code>dcast</code> sagen, wo die Werte zu finden sind, die in Spalten umgewandelt werden sollen                                                                                                      |
| Formel         | <code>dcast</code> mitteilen, welche Spalten beibehalten werden sollen, um eine eindeutige Datensatzkennung (LHS) zu bilden, und welche <code>dcast</code> die Kategorienbezeichnungen (RHS) enthält |
| fun.aggregate  | Geben Sie die zu verwendende Funktion an, wenn der Castingvorgang in jeder Zelle eine Liste von Werten generiert                                                                                     |

## Bemerkungen

Ein Großteil dessen, was zur Konditionierung von Daten zum Erstellen von Modellen oder Visualisierungen benötigt wird, kann mit `data.table` . Im Vergleich zu anderen Optionen bietet `data.table` Vorteile in `data.table` auf Geschwindigkeit und Flexibilität.

## Examples

### Tabellendaten mit data.table - I

Konvertieren Sie von Breitform in Langform

Laden Sie `data USArrests` aus `datasets` .

```
data("USArrests")
head(USArrests)
```

|            | Murder | Assault | UrbanPop | Rape |
|------------|--------|---------|----------|------|
| Alabama    | 13.2   | 236     | 58       | 21.2 |
| Alaska     | 10.0   | 263     | 48       | 44.5 |
| Arizona    | 8.1    | 294     | 80       | 31.0 |
| Arkansas   | 8.8    | 190     | 50       | 19.5 |
| California | 9.0    | 276     | 91       | 40.6 |
| Colorado   | 7.9    | 204     | 78       | 38.7 |

Verwenden Sie `?USArrests`, um mehr zu erfahren. Konvertieren Sie zuerst in `data.table`. Die Zustandsnamen sind `data.frame` im ursprünglichen `data.frame`.

```
library(data.table)
DT <- as.data.table(USArrests, keep.rownames=TRUE)
```

Dies sind Daten in der breiten Form. Für jede Variable gibt es eine Spalte. Die Daten können auch ohne Informationsverlust in langer Form gespeichert werden. Das lange Formular enthält eine Spalte, in der die Variablennamen gespeichert sind. Dann hat es eine weitere Spalte für die Variablenwerte. Die lange Form von USA- `USArrests` sieht so aus.

|      | State         | Crime  | Rate |
|------|---------------|--------|------|
| 1:   | Alabama       | Murder | 13.2 |
| 2:   | Alaska        | Murder | 10.0 |
| 3:   | Arizona       | Murder | 8.1  |
| 4:   | Arkansas      | Murder | 8.8  |
| 5:   | California    | Murder | 9.0  |
| ---  |               |        |      |
| 196: | Virginia      | Rape   | 20.7 |
| 197: | Washington    | Rape   | 26.2 |
| 198: | West Virginia | Rape   | 9.3  |
| 199: | Wisconsin     | Rape   | 10.8 |
| 200: | Wyoming       | Rape   | 15.6 |

Wir verwenden die `melt`, um von Breitform zu Langform zu wechseln.

```
DTm <- melt(DT)
names(DTm) <- c("State", "Crime", "Rate")
```

Standardmäßig behandelt `melt` alle Spalten mit numerischen Daten als Variablen mit Werten. In `USArrests` repräsentiert die Variable `UrbanPop` den prozentualen Anteil der städtischen Bevölkerung eines Staates. Es unterscheidet sich von den anderen Variablen, `Murder`, `Assault` und `Rape`, die pro 100.000 Menschen berichteten Gewaltverbrechen sind. Angenommen, wir möchten die `UrbanPop` Spalte beibehalten. Dies erreichen wir, indem wir `id.vars` wie folgt `id.vars`.

```
DTmu <- melt(DT, id.vars=c("rn", "UrbanPop"),
 variable.name='Crime', value.name = "Rate")
names(DTmu)[1] <- "State"
```

Beachten Sie, dass wir die Namen der Spalte angegeben haben, die Kategorienamen (`Murder`, `Assault` usw.) mit `variable.name` und die Spalte, die die Werte mit `value.name`. Unsere Daten sehen so aus.

|    | State      | UrbanPop | Crime  | Rate |
|----|------------|----------|--------|------|
| 1: | Alabama    | 58       | Murder | 13.2 |
| 2: | Alaska     | 48       | Murder | 10.0 |
| 3: | Arizona    | 80       | Murder | 8.1  |
| 4: | Arkansas   | 50       | Murder | 8.8  |
| 5: | California | 91       | Murder | 9.0  |

Das Erstellen von Zusammenfassungen mit Split-Apply-Combine-Stil ist ein Kinderspiel. Zum Beispiel, um Gewaltverbrechen nach Staat zusammenzufassen?

```
DTmu[, .(ViolentCrime = sum(Rate)), by=State]
```

Das gibt:

|    | State      | ViolentCrime |
|----|------------|--------------|
| 1: | Alabama    | 270.4        |
| 2: | Alaska     | 317.5        |
| 3: | Arizona    | 333.1        |
| 4: | Arkansas   | 218.3        |
| 5: | California | 325.6        |
| 6: | Colorado   | 250.6        |

## Tabellendaten mit data.table - II pivotieren und aufheben

Konvertieren Sie von Langform in Breitform

Um Daten aus dem vorherigen Beispiel `dcast` , verwenden Sie `dcast` so.

```
DTc <- dcast(DTmu, State + UrbanPop ~ Crime)
```

Dies gibt die Daten in der ursprünglichen breiten Form an.

|    | State      | UrbanPop | Murder | Assault | Rape |
|----|------------|----------|--------|---------|------|
| 1: | Alabama    | 58       | 13.2   | 236     | 21.2 |
| 2: | Alaska     | 48       | 10.0   | 263     | 44.5 |
| 3: | Arizona    | 80       | 8.1    | 294     | 31.0 |
| 4: | Arkansas   | 50       | 8.8    | 190     | 19.5 |
| 5: | California | 91       | 9.0    | 276     | 40.6 |

Hier wird die Formelnotation verwendet, um die Spalten anzugeben, die eine eindeutige Datensatzkennung (LHS) bilden, und die Spalte, die Kategoriebeschriftungen für neue Spaltennamen (RHS) enthält. Welche Spalte für die numerischen Werte? Standardmäßig verwendet `dcast` die erste Spalte mit numerischen Werten, die bei der Angabe der Formel übrig bleiben. Um dies explizit zu machen, verwenden Sie den Parameter `value.var` mit dem Spaltennamen.

Wenn der Vorgang eine Liste von Werten in jeder Zelle erzeugt, `dcast` stellt eine `fun.aggregate` Methode , die Situation zu handhaben . Angenommen, ich interessiere mich für Staaten mit einer ähnlichen städtischen Bevölkerung, wenn die Kriminalitätsrate untersucht wird. Ich füge eine Spalte `Decile` mit berechneten Informationen hinzu.



```
DTmu[, Decile := cut(UrbanPop, quantile(UrbanPop, probs = seq(0, 1, by=0.1)))]
levels(DTmu$Decile) <- paste0(1:10, "D")
```

Wenn Sie `Decile ~ Crime` werden jetzt mehrere Werte pro Zelle erzeugt. Ich kann `fun.aggregate` um festzustellen, wie mit diesen `fun.aggregate` wird. Sowohl Text- als auch Zahlenwerte können auf diese Weise verarbeitet werden.

```
dcast(DTmu, Decile ~ Crime, value.var="Rate", fun.aggregate=sum)
```

Das gibt:

```
dcast(DTmu, Decile ~ Crime, value.var="Rate", fun.aggregate=mean)
```

Das gibt:

|    | State      | UrbanPop | Crime  | Rate | Decile |
|----|------------|----------|--------|------|--------|
| 1: | Alabama    | 58       | Murder | 13.2 | 4D     |
| 2: | Alaska     | 48       | Murder | 10.0 | 2D     |
| 3: | Arizona    | 80       | Murder | 8.1  | 8D     |
| 4: | Arkansas   | 50       | Murder | 8.8  | 2D     |
| 5: | California | 91       | Murder | 9.0  | 10D    |

In jedem Dezil der städtischen Bevölkerung gibt es mehrere Staaten. Verwenden Sie `fun.aggregate` um anzugeben, wie diese behandelt werden sollen.

```
dcast(DTmu, Decile ~ Crime, value.var="Rate", fun.aggregate=sum)
```

Dies summiert sich über die Daten für ähnliche Zustände, was folgendes ergibt.

|    | Decile | Murder | Assault | Rape  |
|----|--------|--------|---------|-------|
| 1: | 1D     | 39.4   | 808     | 62.6  |
| 2: | 2D     | 35.3   | 815     | 94.3  |
| 3: | 3D     | 22.6   | 451     | 67.7  |
| 4: | 4D     | 54.9   | 898     | 106.0 |
| 5: | 5D     | 42.4   | 758     | 107.6 |

Pivot und Unpivot mit `data.table` online lesen: <https://riptutorial.com/de/r/topic/6934/pivot-und-unpivot-mit-data-table>

# Kapitel 85: R in LaTeX mit Knitr

## Syntax

1. `<< interner Code-Chunk-Name, Optionen ... >> =`  
    `# R-Code hier`  
    `@`
2. `\ Sexpr {#R-Code hier}`
3. `<< read-external-R-file >> =`  
    `read_chunk ('r-file.R')`  
    `@`  
    `<< externer Code-Chunk-Name, Optionen ... >> =`  
    `@`

## Parameter

| Möglichkeit      | Einzelheiten                                                                                                       |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| Echo             | (TRUE / FALSE) - Gibt an, ob R-Quellcode in die Ausgabedatei aufgenommen werden soll                               |
| Botschaft        | (TRUE / FALSE) - Gibt an, ob Nachrichten aus der R-Quellenausführung in die Ausgabedatei aufgenommen werden sollen |
| Warnung          | (TRUE / FALSE) - Gibt an, ob Warnungen aus der R-Quellenausführung in die Ausgabedatei aufgenommen werden sollen   |
| Error            | (TRUE / FALSE) - Gibt an, ob Fehler bei der Ausführung der R-Quelle in die Ausgabedatei aufgenommen werden sollen  |
| Zwischenspeicher | (TRUE / FALSE) - Gibt an, ob die Ergebnisse der R-Quellenausführung zwischengespeichert werden sollen              |
| fig.breite       | (numerisch) - Breite der Darstellung, die durch die Ausführung der R-Quelle generiert wurde                        |
| fig.höhe         | (numerisch) - Höhe des Plots, der durch die Ausführung der R-Quelle generiert wird                                 |

## Bemerkungen

Knitr ist ein Werkzeug, mit dem natürliche Sprache (in Form von LaTeX) und Quellcode (in Form von R) miteinander verwoben werden können. Im Allgemeinen wird das Konzept des Durchdringens von natürlicher Sprache und Quellcode als gebildete [Programmierung bezeichnet](#).

Da Knitr-Dateien eine Mischung aus LaTeX (traditionell in .tex-Dateien enthalten) und R (traditionell in .R-Dateien) enthalten, ist eine neue Dateierweiterung namens R noweb (.Rnw) erforderlich. .Rnw-Dateien enthalten eine Mischung aus LaTeX- und R-Code.

Knitr ermöglicht die Erstellung statistischer Berichte im PDF-Format und ist ein Schlüsselwerkzeug für [reproduzierbare Forschung](#).

Das Kompilieren von .Rnw-Dateien in eine PDF-Datei erfolgt in zwei Schritten. Zunächst müssen wir wissen, wie der R-Code ausgeführt und die Ausgabe in einem Format erfasst wird, das ein LaTeX-Compiler verstehen kann (ein Vorgang, der als "Knitting" bezeichnet wird). Wir machen dies mit dem knitr-Paket. Der Befehl dafür wird unten angezeigt, vorausgesetzt, Sie haben [das Knitr-Paket installiert](#):

```
Rscript -e "library(knitr); knit('r-noweb-file.Rnw')
```

Dadurch wird eine normale .tex-Datei (in diesem Beispiel als r-noweb.tex bezeichnet) generiert, die dann in eine PDF-Datei umgewandelt werden kann:

```
pdflatex r-noweb-file.tex
```

## Examples

### R in Latex mit Knitr und Code Externalisierung

Knitr ist ein R-Paket, mit dem wir R-Code mit LaTeX-Code vermischen können. Eine Möglichkeit, dies zu erreichen, sind externe Codeabschnitte. Externe Codeabschnitte ermöglichen es uns, R-Skripts in einer R-Entwicklungsumgebung zu entwickeln / testen und die Ergebnisse in einen Bericht aufzunehmen. Es ist eine mächtige Organisationstechnik. Dieser Ansatz wird im Folgenden gezeigt.

```
r-noweb-file.Rnw
\documentclass{article}

<<echo=FALSE,cache=FALSE>>=
knitr::opts_chunk$set(echo=FALSE, cache=TRUE)
knitr::read_chunk('r-file.R')
@

\begin{document}
This is an Rnw file (R noweb). It contains a combination of LaTeX and R.

One we have called the read_chunk command above we can reference sections of code in the r-
file.R script.

<<Chunk1>>=
@
\end{document}
```

Bei diesem Ansatz bewahren wir unseren Code in einer separaten R-Datei auf (siehe unten).

```
r-file.R
note the specific comment style of a single pound sign followed by four dashes

---- Chunk1 ----

print("This is R Code in an external file")

x <- seq(1:10)
y <- rev(seq(1:10))
plot(x,y)
```

## R in Latex mit Knitr und Inline Code Chunks

Knitr ist ein R-Paket, mit dem wir R-Code mit LaTeX-Code vermischen können. Eine Möglichkeit, dies zu erreichen, sind Inline-Code-Blöcke. Dieser Ansatz wird im Folgenden gezeigt.

```
r-noweb-file.Rnw
\documentclass{article}
\begin{document}
This is an Rnw file (R noweb). It contains a combination of LaTeX and R.

<<my-label>>=
print("This is an R Code Chunk")
x <- seq(1:10)
@

Above is an internal code chunk.
We can access data created in any code chunk inline with our LaTeX code like this.
The length of array x is \Sexpr{length(x)}.

\end{document}
```

## R in LaTeX mit Knitr und Internal Code Chunks

Knitr ist ein R-Paket, mit dem wir R-Code mit LaTeX-Code vermischen können. Eine Möglichkeit, dies zu erreichen, sind interne Codeabschnitte. Dieser Ansatz wird im Folgenden gezeigt.

```
r-noweb-file.Rnw
\documentclass{article}
\begin{document}
This is an Rnw file (R noweb). It contains a combination of LaTeX and R.

<<code-chunk-label>>=
print("This is an R Code Chunk")
x <- seq(1:10)
y <- seq(1:10)
plot(x,y) # Brownian motion
@

\end{document}
```

R in LaTeX mit Knitr online lesen: <https://riptutorial.com/de/r/topic/4334/r-in-latex-mit-knitr>

---

# Kapitel 86: R Markdown-Notizbücher (von RStudio)

## Einführung

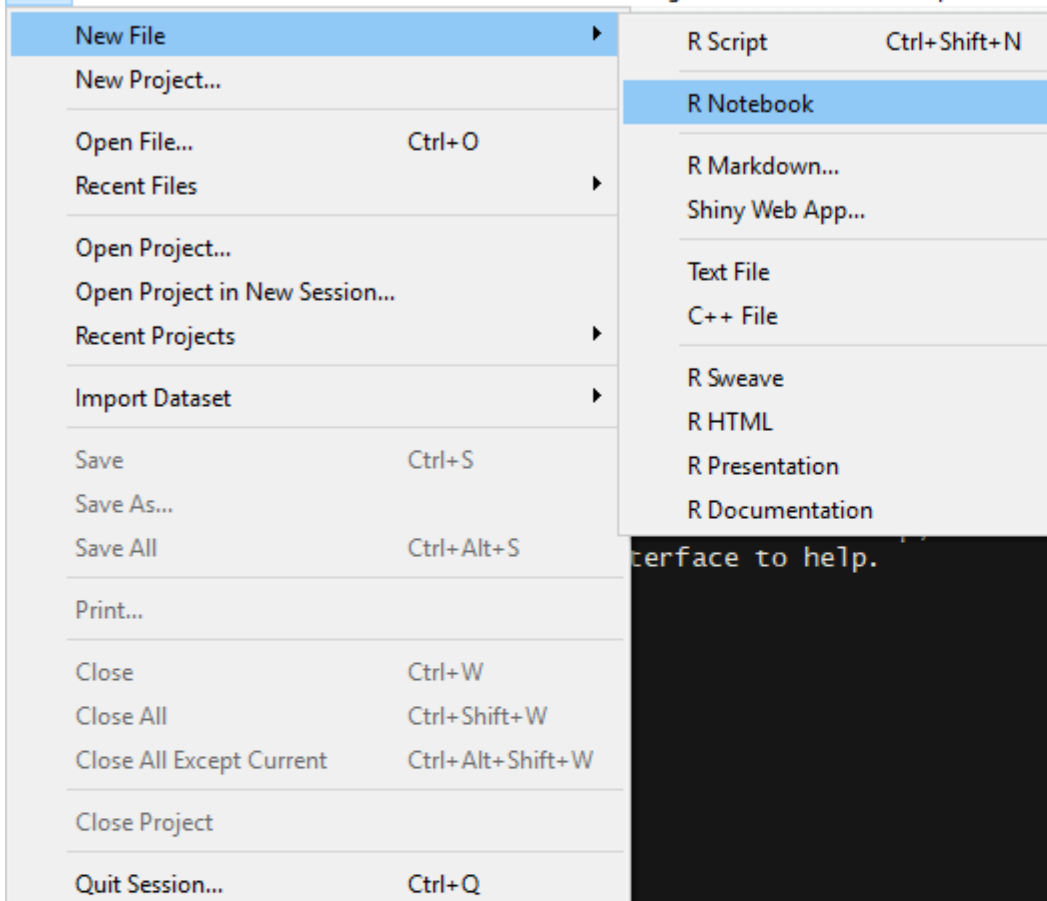
Ein R-Notebook ist ein R-Markdown-Dokument mit Abschnitten, die unabhängig und interaktiv ausgeführt werden können, wobei die Ausgabe unmittelbar unter der Eingabe sichtbar ist. Sie ähneln R Markdown-Dokumenten, mit Ausnahme der Ergebnisse, die im R Notebook-Erstellungs- / Bearbeitungsmodus und nicht in der gerenderten Ausgabe angezeigt werden. **Hinweis:** R Notebooks sind eine neue Funktion von RStudio und sind nur in Version 1.0 oder höher von RStudio verfügbar.

## Examples

### Ein Notizbuch erstellen

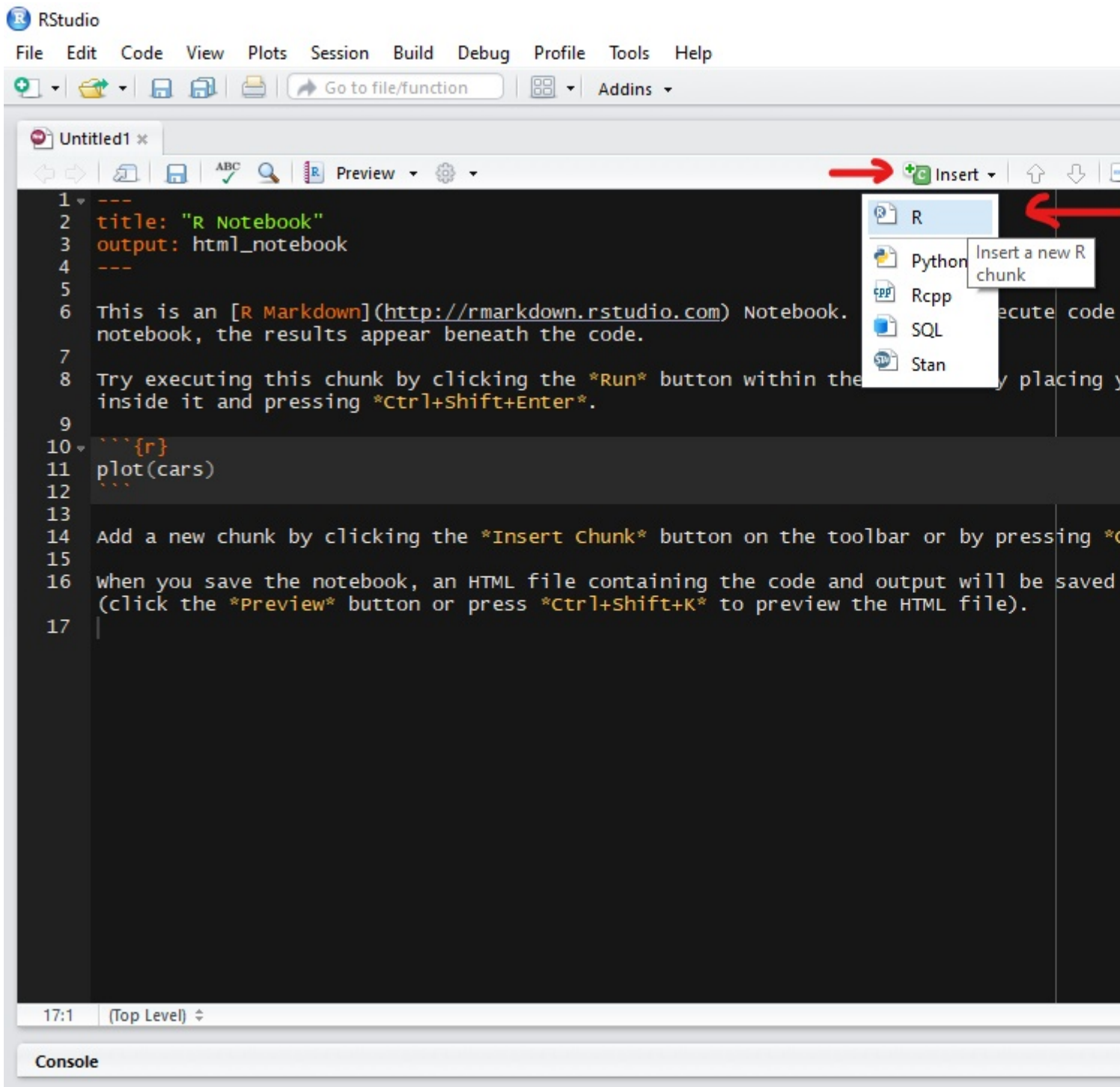
Sie können ein neues Notizbuch in RStudio mit dem Menübefehl Datei -> Neue Datei -> R Notizbuch erstellen

Wenn die Option für R Notebook nicht angezeigt wird, müssen Sie Ihre Version von RStudio aktualisieren. Für die Installation von RStudio folgen Sie [dieser Anleitung](#)



## Brocken einfügen

Chunks sind Code-Teile, die interaktiv ausgeführt werden können. Um einen neuen Block einzufügen, klicken Sie auf die Schaltfläche **Einfügen** in der Symbolleiste des Notebooks und wählen Sie die gewünschte Code-Plattform aus (R in diesem Fall, da wir R-Code schreiben möchten). Alternativ können Sie Tastenkombinationen verwenden, um einen neuen Block einzufügen. **Strg + Alt + I (OS X: Cmd + Wahl + I)**.



## Chunk-Code ausführen

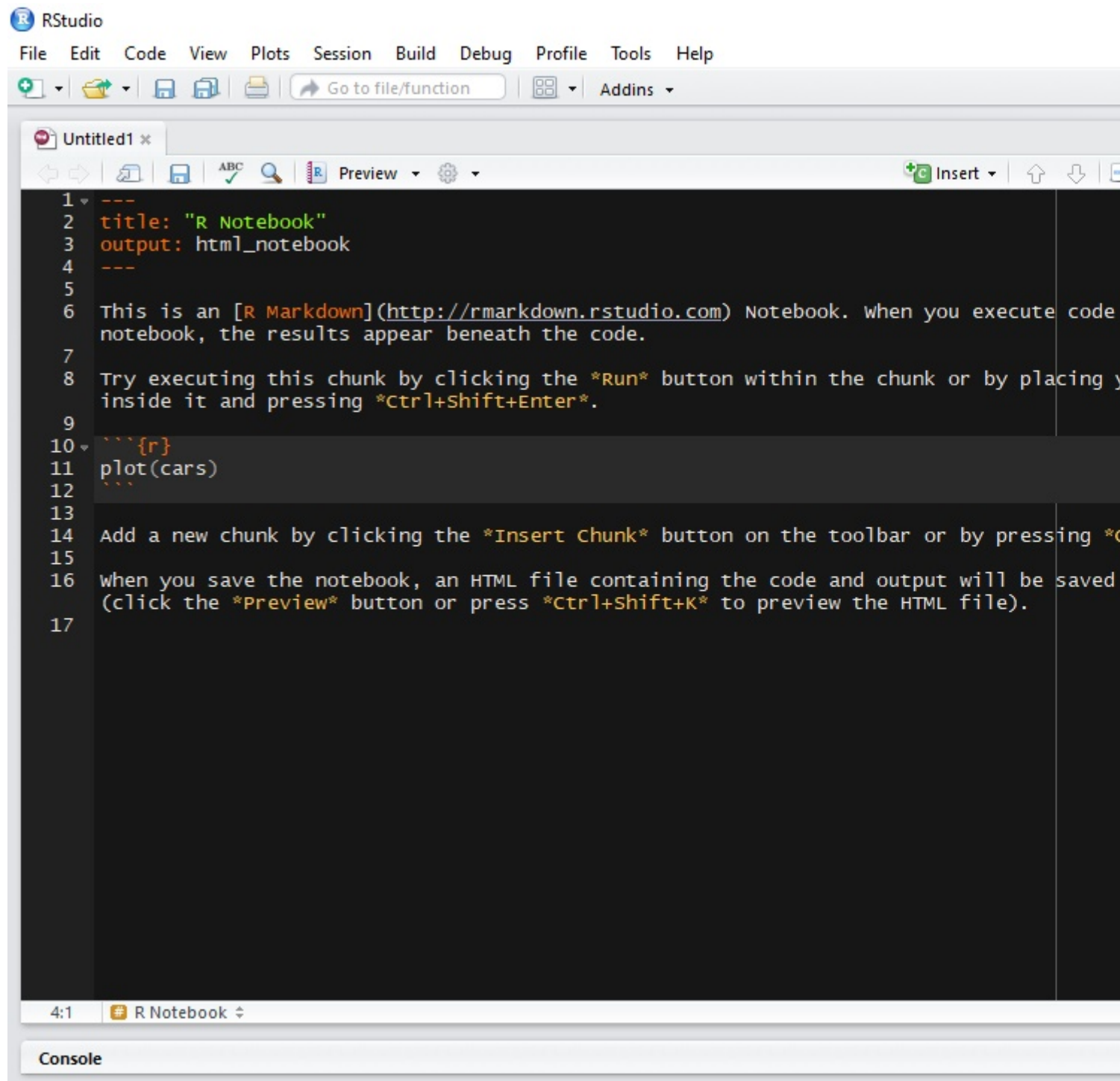
Sie können den aktuellen Chunk ausführen, indem Sie auf **Aktuellen Chunk ausführen (grüne Wiedergabeschaltfläche)** auf der rechten Seite des Chunks klicken. Alternativ können Sie die Tastenkombination **Strg + Umschalttaste + Eingabetaste** verwenden (OS X: **Befehlstaste + Umschalttaste + Eingabetaste**).

Die Ausgabe aller Zeilen im Block wird unter dem Block angezeigt.

## Code in Chunks aufteilen

Da ein Chunk seine Ausgabe unterhalb des Chunks erzeugt, ist es oft hilfreich, wenn mehrere Codezeilen in einem einzigen Chunk erstellt werden, der Mehrfachausgaben erzeugt, so dass jeder Chunk eine Ausgabe erzeugt.

Wählen Sie dazu den Code aus, den Sie in einen neuen Block teilen möchten, und drücken Sie **Strg + Alt + I** (OS X: **Cmd + Wahltaste + I**).



## Ausführungsfortschritt

Wenn Sie Code in einem Notizbuch ausführen, wird in der Zwischenablage ein Indikator angezeigt, der den Fortschritt der Ausführung anzeigt. Codezeilen, die an R gesendet wurden, sind mit dunkelgrün markiert. Zeilen, die noch nicht an R gesendet wurden, sind mit hellem Grün



markiert.

## Mehrere Chunks ausführen

Das Ausführen oder erneute Ausführen einzelner Abschnitte durch Drücken von "Ausführen" für alle in einem Dokument vorhandenen Abschnitte kann schmerzhaft sein. Wir können **Alle** ausführen aus dem Menü Einfügen in der Symbolleiste verwenden, um alle im Notizbuch vorhandenen Blöcke auszuführen. Tastenkombination ist **Strg + Alt + R (OS X: Befehlstaste + Wahltaste + R)**

Es gibt auch die Option **Restart R und Run All Chunks** (verfügbar im Menü Ausführen in der Editor-Symbolleiste), mit der Sie vor dem Ausführen aller Chunks eine neue R-Sitzung erhalten.

Wir haben auch Optionen wie " **Alle Chunks oben** ausführen" und " **Alle Chunks unten** ausführen", um Chunks oberhalb oder unterhalb eines ausgewählten **Chunks** auszuführen.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Untitled1\* x

Preview

```
14 data("iris")
15 head(iris,5)
16
```

|   | Sepal.Length<br><dbl> | Sepal.Width<br><dbl> | Petal.Length<br><dbl> | Petal.Width<br><dbl> |
|---|-----------------------|----------------------|-----------------------|----------------------|
| 1 | 5.1                   | 3.5                  | 1.4                   | 0.2                  |
| 2 | 4.9                   | 3.0                  | 1.4                   | 0.2                  |
| 3 | 4.7                   | 3.2                  | 1.3                   | 0.2                  |
| 4 | 4.6                   | 3.1                  | 1.5                   | 0.2                  |
| 5 | 5.0                   | 3.6                  | 1.4                   | 0.2                  |

5 rows

```
17
18 Divide Iris data to x (contain the all features) and y (only the classes)
19 {r}
20 x <- subset(iris, select=-species)
21 y <- iris$species
22
23
24 Create SVM Model and show summary
25 {r}
26 svm_model <- svm(x,y)
27
28 summary(svm_model)
29
30
31 Run Prediction
32 {r}
33 pred <- predict(svm_model,x)
34
35
36 you can time taken by using system.time
```

74:1 (Top Level) ↕

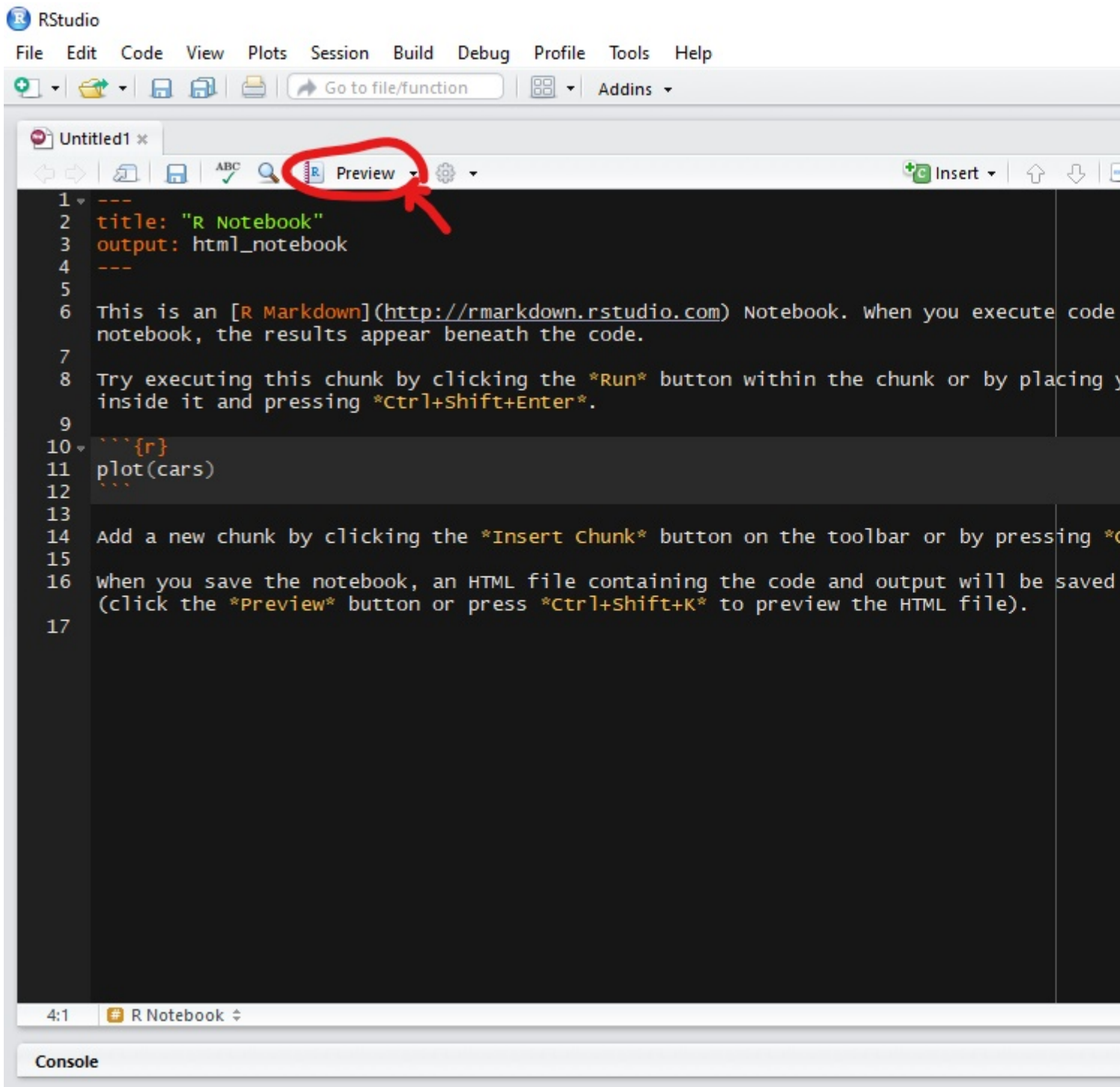
Run All:

Console

## Vorschau Ausgabe

Bevor die endgültige Version eines Notizbuchs gerendert wird, können Sie eine Vorschau der Ausgabe anzeigen. Klicken Sie in der Symbolleiste auf die Schaltfläche **Vorschau** und wählen Sie das gewünschte Ausgabeformat.

Sie können die Art der Ausgabe ändern, indem Sie die Ausgabeoptionen als "pdf\_document" oder "html\_notebook" verwenden.



## Speichern und teilen

Beim `.Rmd` einer `.nb.html` eines Notizbuchs wird eine `.nb.html` Datei erstellt. Diese Datei ist eine in sich geschlossene HTML-Datei, die sowohl eine gerenderte Kopie des Notizbuchs mit allen aktuellen Chunk-Ausgaben (geeignet für die Anzeige auf einer Website) als auch eine Kopie des Notizbuchs `.Rmd` selbst enthält.

Weitere Informationen finden Sie unter [RStudio docs](#)

R Markdown-Notizbücher (von RStudio) online lesen: <https://riptutorial.com/de/r/topic/10728/r-markdown-notizbuecher-von-rstudio->

---

# Kapitel 87: Randomisierung

## Einführung

Die Sprache R wird üblicherweise für die statistische Analyse verwendet. Daher enthält es eine Reihe robuster Optionen für die Randomisierung. Spezifische Informationen zur Probennahme aus Wahrscheinlichkeitsverteilungen finden Sie in der Dokumentation der [Verteilungsfunktionen](#).

## Bemerkungen

Benutzer, die aus anderen Programmiersprachen stammen, können durch das Fehlen einer `rand` verwirrt werden, die dem entspricht, was sie zuvor erlebt haben. Die grundlegende Erzeugung von Zufallszahlen erfolgt mit der `r*`-Familie von Funktionen für jede Verteilung (siehe den Link oben). Zufällige Zahlen, die einheitlich aus einem Bereich gezogen werden, können mit `runif` für "Random Uniform" generiert werden. Da dies auch verdächtig nach "run if" aussieht, ist es für neue R-Benutzer oft schwer herauszufinden.

## Examples

### Zufällige Draws und Permutationen

Mit dem `sample` können Sie klassische Wahrscheinlichkeitsprobleme simulieren, z. B. aus einer Urne mit und ohne Ersatz zeichnen oder zufällige Permutationen erstellen.

Beachten Sie, dass in diesem Beispiel `set.seed` wird, um sicherzustellen, dass der Beispielcode reproduzierbar ist. `sample` funktioniert jedoch ohne explizites Aufrufen von `set.seed`.

---

## Zufällige Permutation

In der einfachsten Form erstellt `sample` eine zufällige Permutation eines Ganzzahlvektors. Dies kann erreicht werden mit:

```
set.seed(1251)
sample(x = 10)

[1] 7 1 4 8 6 3 10 5 2 9
```

Wenn keine anderen Argumente angegeben werden, gibt `sample` eine zufällige Permutation des Vektors von 1 bis `x`. Dies kann nützlich sein, wenn Sie versuchen, die Reihenfolge der Zeilen in einem Datenrahmen zufällig zu bestimmen. Dies ist eine übliche Aufgabe, wenn Randomisierungstabellen für Studien erstellt werden oder wenn eine zufällige Teilmenge von Zeilen für die Analyse ausgewählt wird.

```
library(datasets)
```

```
set.seed(1171)
iris_rand <- iris[sample(x = 1:nrow(iris)),]

> head(iris)
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa

> head(iris_rand)
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
145 6.7 3.3 5.7 2.5 virginica
5 5.0 3.6 1.4 0.2 setosa
85 5.4 3.0 4.5 1.5 versicolor
137 6.3 3.4 5.6 2.4 virginica
128 6.1 3.0 4.9 1.8 virginica
105 6.5 3.0 5.8 2.2 virginica
```

## Zieht ohne Ersatz

Anhand eines `sample` können wir auch das Zeichnen eines Sets mit und ohne Ersatz simulieren. Um ein Muster ohne Ersatz (StandardEinstellung) zu erhalten, müssen Sie dem Muster einen Satz und die Anzahl der Ziehungen zur Verfügung stellen. Die Menge, aus der gezogen werden soll, wird als Vektor angegeben.

```
set.seed(7043)
sample(x = LETTERS, size = 7)

[1] "S" "P" "J" "F" "Z" "G" "R"
```

Wenn das Argument zu `size` der Länge des Arguments in `x`, erstellen Sie eine zufällige Permutation. Beachten Sie außerdem, dass Sie keine Größe angeben können, die größer als die Länge von `x` wenn Sie die Probenahme ohne Ersatz durchführen.

```
set.seed(7305)
sample(x = letters, size = 26)

[1] "x" "z" "y" "i" "k" "f" "d" "s" "g" "v" "j" "o" "e" "c" "m" "n" "h" "u" "a" "b" "l" "r"
"w" "t" "q" "p"

sample(x = letters, size = 30)
Error in sample.int(length(x), size, replace, prob) :
 cannot take a sample larger than the population when 'replace = FALSE'
```

Dies bringt uns zum Zeichnen mit Ersatz.

## Zieht mit Ersatz

Um zufällig schöpft aus einem Set mit Ersatz, verwenden Sie die `replace` Argument zu `sample`. `replace` ist standardmäßig `FALSE`. Die Einstellung auf `TRUE` bedeutet, dass jedes Element der Gruppe, aus der gezogen wird, mehr als einmal im Endergebnis erscheinen kann.

```
set.seed(5062)
sample(x = c("A", "B", "C", "D"), size = 8, replace = TRUE)

[1] "D" "C" "D" "B" "A" "A" "A" "A"
```

## Ändern der Draw-Wahrscheinlichkeiten

Bei der Verwendung von `sample` wird standardmäßig davon ausgegangen, dass die Wahrscheinlichkeit der Auswahl jedes Elements gleich ist. Betrachten Sie es als ein grundlegendes "Urnenproblem". Der folgende Code entspricht dem 20-maligen Zeichnen eines farbigen Marmors aus einer Urne, dem Schreiben der Farbe und dem anschließenden Einsetzen des Marmors in die Urne. Die Urne enthält einen roten, einen blauen und einen grünen Marmor. Dies bedeutet, dass die Wahrscheinlichkeit, dass jede Farbe gezeichnet wird,  $1/3$  beträgt.

```
set.seed(6472)
sample(x = c("Red", "Blue", "Green"),
 size = 20,
 replace = TRUE)
```

Angenommen, wir wollten stattdessen dieselbe Aufgabe ausführen, aber unsere Urne enthält 2 rote Murmeln, 1 blauen Marmor und 1 grünen Marmor. Eine Option wäre, das von uns gesendete Argument in `x` zu ändern, um ein zusätzliches `Red` hinzuzufügen. Allerdings ist eine bessere Wahl ist, die verwenden `prob` Argument zu `sample`.

Das `prob` Argument akzeptiert einen Vektor mit der Wahrscheinlichkeit, jedes Element zu zeichnen. In unserem obigen Beispiel wäre die Wahrscheinlichkeit des Zeichnens eines roten Marmors  $1/2$ , während die Wahrscheinlichkeit des Zeichnens eines blauen oder grünen Marmors  $1/4$  beträgt.

```
set.seed(28432)
sample(x = c("Red", "Blue", "Green"),
 size = 20,
 replace = TRUE,
 prob = c(0.50, 0.25, 0.25))
```

`prob` muss das zu `prob` gegebene Argument nicht zu 1 summiert werden. R wandelt die angegebenen Argumente immer in Wahrscheinlichkeiten um, die sich zu 1 summieren. Betrachten wir beispielsweise unser voriges Beispiel von 2 Rot, 1 Blau und 1 Grün. Mit diesen Zahlen können Sie dieselben Ergebnisse wie bei unserem vorherigen Code erzielen:

```
set.seed(28432)
frac_prob_example <- sample(x = c("Red", "Blue", "Green"),
 size = 200,
 replace = TRUE,
 prob = c(0.50, 0.25, 0.25))
```

```

set.seed(28432)
numeric_prob_example <- sample(x = c("Red", "Blue", "Green"),
 size = 200,
 replace = TRUE,
 prob = c(2,1,1))

> identical(frac_prob_example, numeric_prob_example)
[1] TRUE

```

Die Hauptbeschränkung besteht darin, dass Sie nicht alle Wahrscheinlichkeiten auf Null setzen können und keine von ihnen kleiner als Null sein kann.

Sie können auch nutzen `prob` wenn `replace` eingestellt ist `FALSE`. In dieser Situation geben die Anteile der `prob` Werte für die übrigen Elemente nach dem Zeichnen jedes Elements die Wahrscheinlichkeit für die nächste Ziehung an. In dieser Situation müssen Sie über genügend Nicht-Null-Wahrscheinlichkeiten verfügen, um die `size` des von Ihnen gezeichneten Beispiels zu erreichen. Zum Beispiel:

```

set.seed(21741)
sample(x = c("Red", "Blue", "Green"),
 size = 2,
 replace = FALSE,
 prob = c(0.8, 0.19, 0.01))

```

In diesem Beispiel wird in der ersten Ziehung (als erstes Element) Rot gezeichnet. Es gab eine 80% ige Chance, dass Rot gezogen wurde, eine 19% ige Chance, dass Blau gezogen wurde, und eine 1% ige Chance, dass Grün gezogen wurde.

Bei der nächsten Ziehung ist Rot nicht mehr in der Urne. Die Summe der Wahrscheinlichkeiten unter den verbleibenden Elementen beträgt 20% (19% für Blau und 1% für Grün). Bei dieser Auslosung besteht eine Chance von 95%, dass der Gegenstand blau ist (19/20), und eine Chance von 5% ist grün (1/20).

## Samen setzen

Mit der Funktion `set.seed` wird der Zufallsstartwert für alle Randomisierungsfunktionen festgelegt. Wenn Sie R verwenden, um eine Randomisierung zu erstellen, die Sie reproduzieren möchten, sollten `set.seed` zunächst `set.seed`.

```

set.seed(1643)
samp1 <- sample(x = 1:5, size = 200, replace = TRUE)

set.seed(1643)
samp2 <- sample(x = 1:5, size = 200, replace = TRUE)

> identical(x = samp1, y = samp2)
[1] TRUE

```

Beachten Sie, dass die parallele Verarbeitung eine spezielle Behandlung des Zufalls-Samens erfordert, die an anderer Stelle beschrieben wird.

Randomisierung online lesen: <https://riptutorial.com/de/r/topic/9574/randomisierung>



# Kapitel 88: Raster- und Bildanalyse

## Einführung

Siehe auch [E / A für Rasterbilder](#)

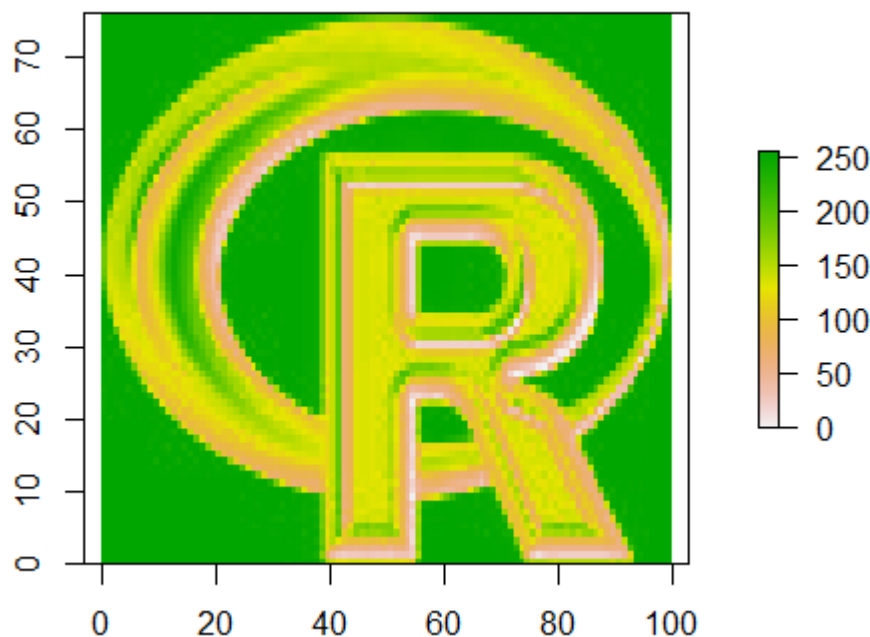
## Examples

### Berechnung der GLCM-Textur

Die Textur der Grauegel- [Co-Vorkommen-Matrix](#) (Haralick et al. 1973) ist eine leistungsstarke Bildfunktion für die Bildanalyse. Das `glcm` Paket bietet eine einfach zu verwendende Funktion zum Berechnen solcher Textfunktionen für `RasterLayer` Objekte in R.

```
library(glcm)
library(raster)

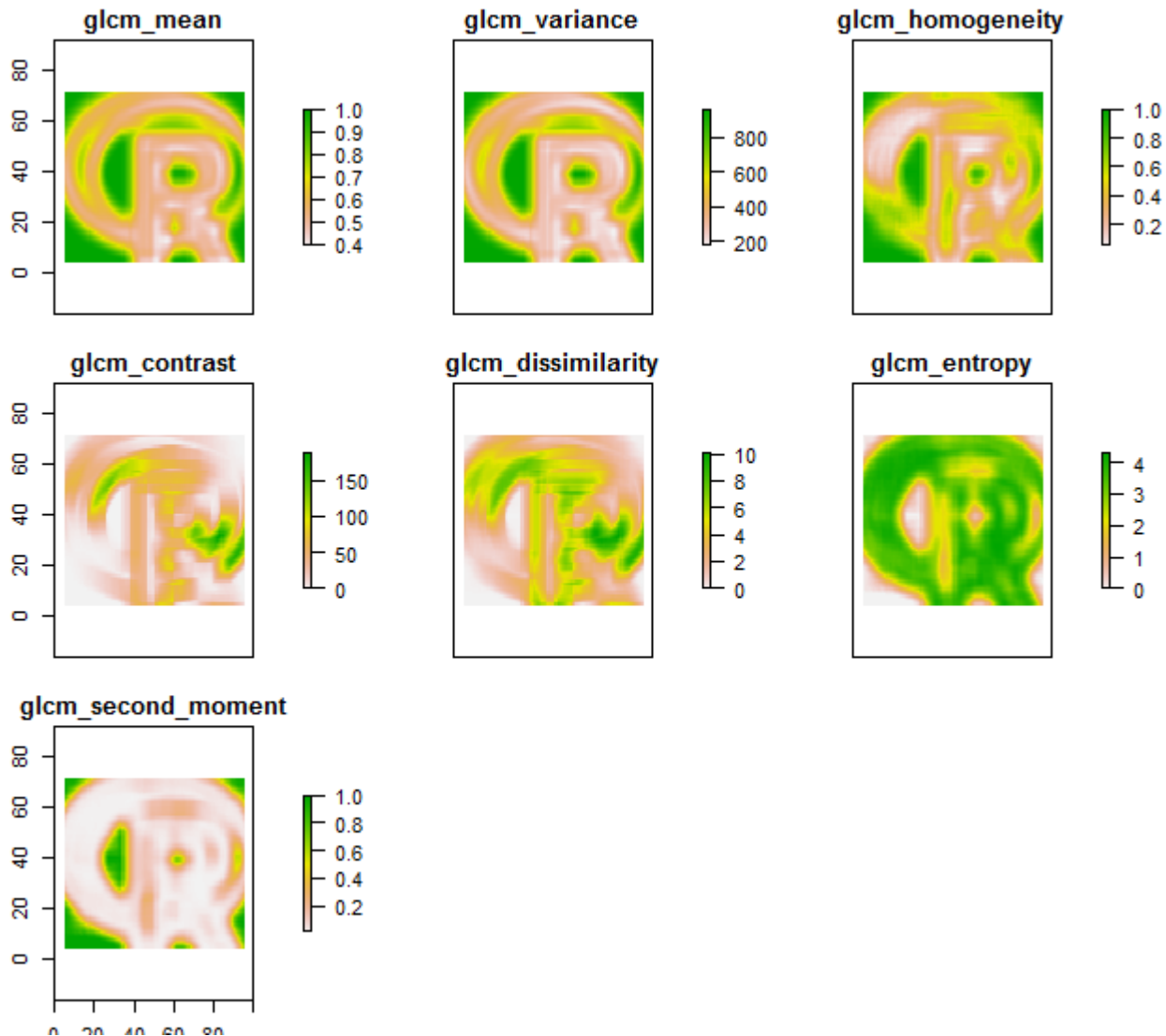
r <- raster("C:/Program Files/R/R-3.2.3/doc/html/logo.jpg")
plot(r)
```



### Berechnung von GLCM-Texturen in eine Richtung

```
rglcm <- glcm(r,
 window = c(9,9),
 shift = c(1,1),
 statistics = c("mean", "variance", "homogeneity", "contrast",
 "dissimilarity", "entropy", "second_moment")
)
```

```
plot(rglcm)
```

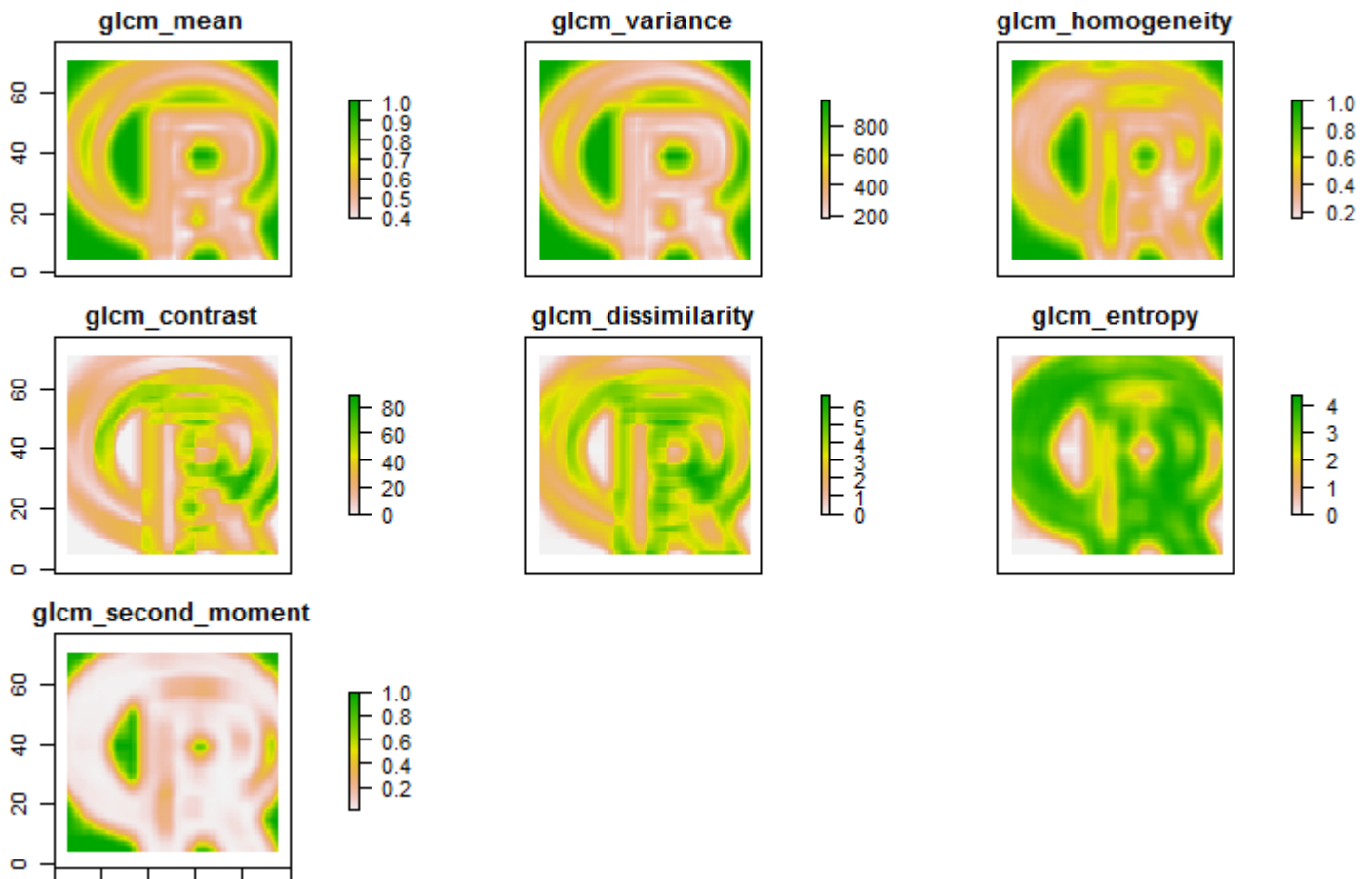


## Berechnung rotationsinvariante Texturmerkmale

Die Strukturmerkmale können auch in allen 4 Richtungen (0 °, 45 °, 90 ° und 135 °) berechnet und dann zu einer rotationsinvarianten Textur kombiniert werden. Der Schlüssel dazu ist der `shift` Parameter:

```
rglcm1 <- glcm(r,
 window = c(9,9),
 shift=list(c(0,1), c(1,1), c(1,0), c(1,-1)),
 statistics = c("mean", "variance", "homogeneity", "contrast",
 "dissimilarity", "entropy", "second_moment")
)

plot(rglcm1)
```

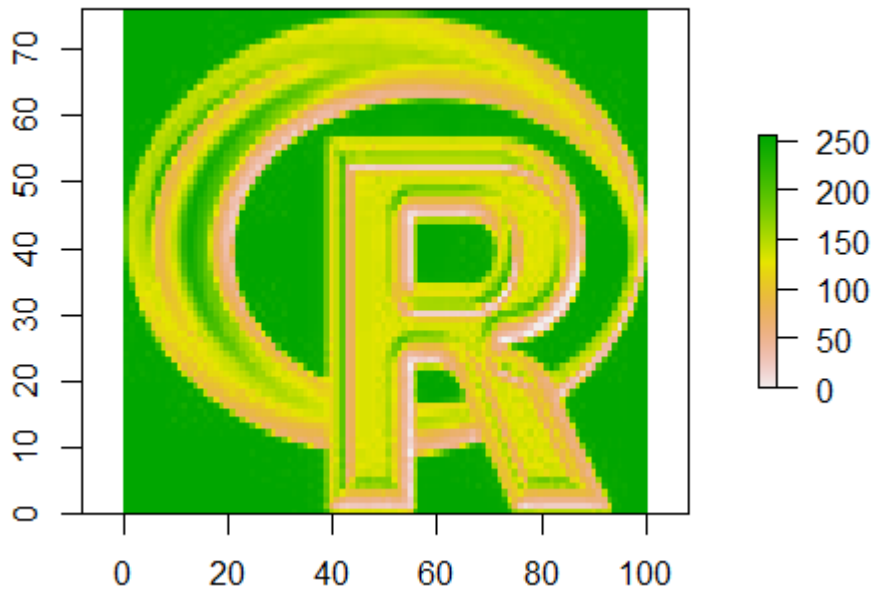


## Mathematische Morphologien

Das Paket `mmand` bietet Funktionen zur Berechnung mathematischer Morphologien für n-dimensionale Arrays. Mit einer kleinen Problemumgehung können diese auch für Rasterbilder berechnet werden.

```
library(raster)
library(mmand)

r <- raster("C:/Program Files/R/R-3.2.3/doc/html/logo.jpg")
plot(r)
```



Zunächst muss ein Kern (Moving Window) mit einer Größe (zB 9x9) und einem Formtyp (zB `disc`, `box` oder `diamond`) eingestellt werden

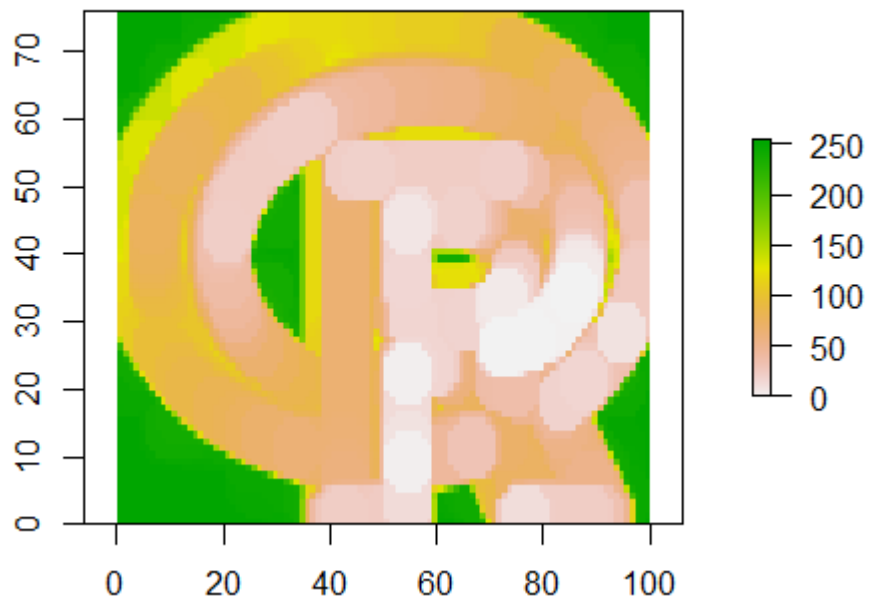
```
sk <- shapeKernel(c(9,9), type="disc")
```

Danach muss die Rasterebene in ein Array umgewandelt werden, das als Eingabe für die Funktion `erode()` verwendet wird.

```
rArr <- as.array(r, transpose = TRUE)
rErode <- erode(rArr, sk)
rErode <- setValues(r, as.vector(aperm(rErode)))
```

Neben `erode()` können auch die morphologischen Funktionen `dilate()`, `opening()` und `closing()` angewendet werden.

```
plot(rErode)
```



Raster- und Bildanalyse online lesen: <https://riptutorial.com/de/r/topic/3726/raster--und-bildanalyse>

# Kapitel 89: räumliche Analyse

## Examples

### Erstellen Sie räumliche Punkte aus dem XY-Datensatz

Wenn es um geographische Daten geht, ist R ein leistungsfähiges Werkzeug für Datenhandhabung, -analyse und -visualisierung.

Oft sind räumliche Daten als XY-Koordinatendatensatz in Tabellenform verfügbar. Dieses Beispiel zeigt, wie Sie einen Geodatensatz aus einem XY-Datensatz erstellen.

Die Pakete `rgdal` und `sp` bieten leistungsstarke Funktionen. Räumliche Daten in R können als `Spatial*DataFrame` gespeichert werden (wobei `* Points`, `Lines` oder `Polygons`).

In diesem Beispiel werden Daten verwendet, die bei [OpenGeocode](#) heruntergeladen werden können.

Zunächst muss das Arbeitsverzeichnis auf den Ordner der heruntergeladenen CSV-Datei eingestellt werden. Außerdem muss das Paket `rgdal` geladen werden.

```
setwd("D:/GeocodeExample/")
library(rgdal)
```

Anschließend wird die CSV-Datei, in der Städte und ihre geographischen Koordinaten `data.frame` als `data.frame` in R `data.frame`

```
xy <- read.csv("worldcities.csv", stringsAsFactors = FALSE)
```

Häufig ist es hilfreich, einen Blick auf die Daten und ihre Struktur (z. B. Spaltennamen, Datentypen usw.) zu erhalten.

```
head(xy)
str(xy)
```

Dies zeigt, dass die Breiten- und Längengradspalten als Zeichenwerte interpretiert werden, da sie Einträge wie "-33.532" enthalten. Für die später verwendete Funktion `SpatialPointsDataFrame()` die den Geodatensatz erstellt, müssen die Koordinatenwerte jedoch vom Datentyp `numeric`. Daher müssen die beiden Spalten konvertiert werden.

```
xy$latitude <- as.numeric(xy$latitude)
xy$longitude <- as.numeric(xy$longitude)
```

Einige der Werte können nicht in numerische Daten konvertiert werden. Daher werden `NA` Werte erstellt. Sie müssen entfernt werden.

```
xy <- xy[!is.na(xy$longitude),]
```

Schließlich kann der XY-Datensatz in einen räumlichen Datensatz umgewandelt werden. Dies erfordert die Koordinaten und die Spezifikation des Koordinaten-Aufrischungssystems (CRS), in dem die Koordinaten gespeichert sind.

```
xySPoints <- SpatialPointsDataFrame(coords = c(xy[,c("longitude", "latitude")]),
proj4string = CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"),
data = xy
)
```

Mit der Basis-Plot-Funktion können Sie die erzeugten Raumpunkte spitzen.

```
plot(xySPoints, pch = ".")
```



## Importieren einer Formdatei (.shp)

### rgdal

ESRI-Shape-Dateien können einfach mit der Funktion `readOGR()` aus dem `rgdal` Paket in R importiert werden.

```
library(rgdal)
shp <- readOGR(dsn = "/path/to/your/file", layer = "filename")
```

Es ist wichtig zu wissen, dass die `dsn` nicht enden muss / und die `layer` erlaubt nicht die Datei mit der Endung (zB `.shp`)

# Raster

Eine andere Möglichkeit, Shapefiles zu importieren, ist über die `raster` Bibliothek und die `shapefile` Funktion:

```
library(raster)
shp <- shapefile("path/to/your/file.shp")
```

Beachten Sie, wie sich die Pfaddefinition von der Importanweisung `rgdal` unterscheidet.

---

## tmap

`tmap` Paket bietet einen schönen Wrapper für die Funktion `rgdal::readORG`.

```
library(tmap)
sph <- read_shape("path/to/your/file.shp")
```

räumliche Analyse online lesen: <https://riptutorial.com/de/r/topic/2093/raumliche-analyse>



# Kapitel 90: Rcpp

## Examples

### Inline-Code kompilieren

Rcpp bietet zwei Funktionen, die die Inline-Kompilierung von Code und den direkten Export in R ermöglichen: `cppFunction()` und `evalCpp()`. Eine dritte Funktion namens `sourceCpp()` existiert zum Lesen von C++ - Code in einer separaten Datei. Sie kann jedoch ähnlich wie `cppFunction()`.

Unten ist ein Beispiel für das Kompilieren einer C++ - Funktion in R. Beachten Sie die Verwendung von `"`, um die Quelle zu umgeben.

```
Note - This is R code.
cppFunction in Rcpp allows for rapid testing.
require(Rcpp)

Creates a function that multiplies each element in a vector
Returns the modified vector.
cppFunction("
NumericVector exfun(NumericVector x, int i){
 x = x*i;
 return x;
}")

Calling function in R
exfun(1:5, 3)
```

Um einen C++ - Ausdruck schnell zu verstehen, verwenden Sie:

```
Use evalCpp to evaluate C++ expressions
evalCpp("std::numeric_limits<double>::max()")
[1] 1.797693e+308
```

### Rcpp-Attribute

Rcpp Attributes macht das Arbeiten mit R und C++ unkompliziert. Die Form der Attribute hat folgende Eigenschaften:

```
// [[Rcpp::attribute]]
```

Die Verwendung von Attributen ist normalerweise verbunden mit:

```
// [[Rcpp::export]]
```

Das wird direkt über einem deklarierten Funktionsheader platziert, wenn eine C++ - Datei über `sourceCpp()`.

Nachfolgend finden Sie ein Beispiel für eine externe C ++ - Datei, die Attribute verwendet.

```
// Add code below into C++ file Rcpp_example.cpp

#include <Rcpp.h>
using namespace Rcpp;

// Place the export tag right above function declaration.
// [[Rcpp::export]]
double muRcpp(NumericVector x){

 int n = x.size(); // Size of vector
 double sum = 0; // Sum value

 // For loop, note cpp index shift to 0
 for(int i = 0; i < n; i++){
 // Shorthand for sum = sum + x[i]
 sum += x[i];
 }

 return sum/n; // Obtain and return the Mean
}

// Place dependent functions above call or
// declare the function definition with:
double muRcpp(NumericVector x);

// [[Rcpp::export]]
double varRcpp(NumericVector x, bool bias = true){

 // Calculate the mean using C++ function
 double mean = muRcpp(x);
 double sum = 0;

 int n = x.size();

 for(int i = 0; i < n; i++){
 sum += pow(x[i] - mean, 2.0); // Square
 }

 return sum/(n-bias); // Return variance
}
```

Um diese externe C ++ - Datei in R zu verwenden , führen wir folgende Schritte aus:

```
require(Rcpp)

Compile File
sourceCpp("path/to/file/Rcpp_example.cpp")

Make some sample data
x = 1:5

all.equal(muRcpp(x), mean(x))
TRUE

all.equal(varRcpp(x), var(x))
TRUE
```

## Rcpp mit Plugins erweitern

Innerhalb von C ++ können verschiedene Compilerflags gesetzt werden:

```
// [[Rcpp::plugins(name)]]
```

Liste der eingebauten Plugins:

```
// built-in C++11 plugin
// [[Rcpp::plugins(cpp11)]]

// built-in C++11 plugin for older g++ compiler
// [[Rcpp::plugins(cpp0x)]]

// built-in C++14 plugin for C++14 standard
// [[Rcpp::plugins(cpp14)]]

// built-in C++1y plugin for C++14 and C++17 standard under development
// [[Rcpp::plugins(cpp1y)]]

// built-in OpenMP++11 plugin
// [[Rcpp::plugins(openmp)]]
```

## Zusätzliche Build-Abhängigkeiten angeben

So verwenden Sie zusätzliche Pakete im RCPP Ökosystem, die korrekte Header - Datei möglicherweise nicht `Rcpp.h` aber `Rcpp<PACKAGE>.h` (wie zB für [RcppArmadillo](#) ). Normalerweise muss es importiert werden und dann wird die Abhängigkeit innerhalb angegeben

```
// [[Rcpp::depends(Rcpp<PACKAGE>)]]
```

Beispiele:

```
// Use the RcppArmadillo package
// Requires different header file from Rcpp.h
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]

// Use the RcppEigen package
// Requires different header file from Rcpp.h
#include <RcppEigen.h>
// [[Rcpp::depends(RcppEigen)]]
```

Rcpp online lesen: <https://riptutorial.com/de/r/topic/1404/rcpp>

# Kapitel 91: Rechenzeichen

## Bemerkungen

Fast alle Operatoren in R sind wirklich Funktionen. Zum Beispiel ist `+` eine als Funktion definierte `function (e1, e2) .Primitive("+")` wobei `e1` die linke Seite des Operators und `e2` die rechte Seite des Operators ist. Dies bedeutet, dass es möglich ist, eher nicht eingreifende Effekte zu erzielen, indem das `+` in der Basis mit einer benutzerdefinierten Funktion maskiert wird.

Zum Beispiel:

```
`+` <- function(e1, e2) {e1-e2}

> 3+10
[1] -7
```

## Examples

### Reichweite und Zusatz

Nehmen wir ein Beispiel für das Hinzufügen eines Werts zu einem Bereich (wie es zum Beispiel in einer Schleife möglich wäre):

```
3+1:5
```

Gibt:

```
[1] 4 5 6 7 8
```

Dies liegt daran, dass der Bereichsoperator `:` eine höhere Priorität hat als der Additionsoperator `+`.

Was passiert während der Auswertung ist wie folgt:

- `3+1:5`
- `3+c(1, 2, 3, 4, 5)` Erweiterung des Bereichsoperators, um einen Vektor aus ganzen Zahlen zu erzeugen.
- `c(4, 5, 6, 7, 8)` Zugabe von 3 zu jedem Element des Vektors.

Um dieses Verhalten zu vermeiden, müssen Sie dem R-Interpreter mitteilen, wie er die Operationen mit `( )` so anordnen soll:

```
(3+1):5
```

Jetzt berechnet R, was sich in den Klammern befindet, bevor der Bereich erweitert wird, und gibt Folgendes aus:

## Addition und Subtraktion

Die grundlegenden mathematischen Operationen werden hauptsächlich auf Zahlen oder auf Vektoren (Listen von Zahlen) ausgeführt.

### 1. Verwenden Sie einzelne Zahlen

Wir können einfach die mit + *verknüpften* Zahlen eingeben, um sie *hinzuzufügen* und – um sie *abzuziehen* :

```
> 3 + 4.5
[1] 7.5
> 3 + 4.5 + 2
[1] 9.5
> 3 + 4.5 + 2 - 3.8
[1] 5.7
> 3 + NA
[1] NA
> NA + NA
[1] NA
> NA - NA
[1] NA
> NaN - NA
[1] NaN
> NaN + NA
[1] NaN
```

Wir können die Zahlen *Variablen* (in diesem Fall Konstanten) zuweisen und dieselben Operationen ausführen:

```
> a <- 3; B <- 4.5; cc <- 2; Dd <- 3.8 ; na<-NA; nan<-NaN
> a + B
[1] 7.5
> a + B + cc
[1] 9.5
> a + B + cc - Dd
[1] 5.7
> B-nan
[1] NaN
> a+na-na
[1] NA
> a + na
[1] NA
> B-nan
[1] NaN
> a+na-na
[1] NA
```

### 2. Verwendung von Vektoren

In diesem Fall erstellen wir Vektoren von Zahlen und führen die Operationen unter Verwendung dieser Vektoren oder Kombinationen mit einzelnen Zahlen aus. In diesem Fall wird die Operation

unter Berücksichtigung jedes Elements des Vektors ausgeführt:

```
> A <- c(3, 4.5, 2, -3.8);
> A
[1] 3.0 4.5 2.0 -3.8
> A + 2 # Adding a number
[1] 5.0 6.5 4.0 -1.8
> 8 - A # number less vector
[1] 5.0 3.5 6.0 11.8
> n <- length(A) #number of elements of vector A
> n
[1] 4
> A[-n] + A[n] # Add the last element to the same vector without the last element
[1] -0.8 0.7 -1.8
> A[1:2] + 3 # vector with the first two elements plus a number
[1] 6.0 7.5
> A[1:2] - A[3:4] # vector with the first two elements less the vector with elements 3 and 4
[1] 1.0 8.3
```

Wir können auch die Funktion `sum`, um alle Elemente eines Vektors hinzuzufügen:

```
> sum(A)
[1] 5.7
> sum(-A)
[1] -5.7
> sum(A[-n]) + A[n]
[1] 5.7
```

Wir müssen beim *Recycling* aufpassen, was eine der Eigenschaften von `R` ist. Dieses Verhalten tritt bei mathematischen Operationen auf, bei denen die Länge der Vektoren unterschiedlich ist. *Kürzere Vektoren im Ausdruck werden so oft wie nötig rezykliert (möglicherweise unvollständig), bis sie der Länge des längsten Vektors entsprechen. Insbesondere wird eine Konstante einfach wiederholt*. In diesem Fall wird eine Warnung angezeigt.

```
> B <- c(3, 5, -3, 2.7, 1.8)
> B
[1] 3.0 5.0 -3.0 2.7 1.8
> A
[1] 3.0 4.5 2.0 -3.8
> A + B # the first element of A is repeated
[1] 6.0 9.5 -1.0 -1.1 4.8
Warning message:
In A + B : longer object length is not a multiple of shorter object length
> B - A # the first element of A is repeated
[1] 0.0 0.5 -5.0 6.5 -1.2
Warning message:
In B - A : longer object length is not a multiple of shorter object length
```

In diesem Fall müssen Sie nur die Elemente des kürzeren Vektors berücksichtigen:

```
> B[1:n] + A
[1] 6.0 9.5 -1.0 -1.1
> B[1:n] - A
[1] 0.0 0.5 -5.0 6.5
```

Bei Verwendung der `sum` werden wieder alle Elemente innerhalb der Funktion hinzugefügt.

```
> sum(A, B)
[1] 15.2
> sum(A, -B)
[1] -3.8
> sum(A)+sum(B)
[1] 15.2
> sum(A)-sum(B)
[1] -3.8
```

Rechenzeichen online lesen: <https://riptutorial.com/de/r/topic/4389/rechenzeichen>

# Kapitel 92: Recycling

## Bemerkungen

### Was ist Recycling in R?

Beim **Recycling** handelt es sich um ein Objekt, das bei bestimmten Vorgängen automatisch um die Länge eines anderen, längeren Objekts erweitert wird.

Zum Beispiel führt die vektorisierte Addition zu folgendem:

```
c(1,2,3) + c(1,2,3,4,5,6)
[1] 2 4 6 5 7 9
```

Aufgrund des Recyclings war der Vorgang, der tatsächlich erfolgte, folgende:

```
c(1,2,3,1,2,3) + c(1,2,3,4,5,6)
```

Wenn das längere Objekt kein Vielfaches des kürzeren Objekts ist, wird eine Warnmeldung angezeigt:

```
c(1,2,3) + c(1,2,3,4,5,6,7)
[1] 2 4 6 5 7 9 8
Warning message:
In c(1, 2, 3) + c(1, 2, 3, 4, 5, 6, 7) :
 longer object length is not a multiple of shorter object length
```

Ein anderes Beispiel für das Recycling:

```
matrix(nrow = 5, ncol = 2, 1:5)
 [,1] [,2]
[1,] 1 1
[2,] 2 2
[3,] 3 3
[4,] 4 4
[5,] 5 5
```

## Examples

### Recycling-Verwendung beim Subsetting

Recycling kann auf clevere Weise verwendet werden, um Code zu vereinfachen.

#### Subsetting

Wenn Sie jedes dritte Element eines Vektors beibehalten möchten, können Sie Folgendes tun:



```
my_vec <- c(1,2,3,4,5,6,7,8,9,10)
my_vec[c(TRUE, FALSE)]

[1] 1 3 5 7 9
```

Hier wurde der logische Ausdruck auf die Länge des Vektors erweitert.

Wir können auch Vergleiche mit Recycling durchführen:

```
my_vec <- c("foo", "bar", "soap", "mix")
my_vec == "bar"

[1] FALSE TRUE FALSE FALSE
```

Hier wird "Bar" recycelt.

Recycling online lesen: <https://riptutorial.com/de/r/topic/5649/recycling>

---

# Kapitel 93: Reguläre Ausdrücke (Regex)

## Einführung

Reguläre Ausdrücke (auch "Regex" oder "Regexp" genannt) definieren Muster, die [mit einer Zeichenfolge abgeglichen werden](#) können. Geben Sie `?regex` für die offizielle R-Dokumentation ein. Weitere Informationen finden Sie in den [Regex-](#) Dokumenten. Der wichtigste ‚Gotcha‘, die nicht in den SO regex / Themen gelernt wird, ist, dass die meisten R-regex Funktionen die Verwendung von gepaarten Schrägstrichen müssen in einem entkommen `pattern` - Parametern.

## Bemerkungen

---

## Charakterklassen

- "[AB]" könnte A oder B sein
- "[[:alpha:]]" könnte ein beliebiger Buchstabe sein
- "[[:lower:]]" steht für Kleinbuchstaben. Beachten Sie, dass "[az]" nahe ist, aber nicht übereinstimmt, z. B. `ú`.
- "[[:upper:]]" steht für Großbuchstaben. Beachten Sie, dass "[AZ]" nahe ist, aber nicht übereinstimmt, z. B. `ú`.
- "[[:digit:]]" steht für eine beliebige Ziffer: 0, 1, 2, ... oder 9 und entspricht "[0-9]" .

---

## Quantifizierer

`+`, `*` und `?` gelten wie üblich in Regex. `-` + stimmt mindestens einmal überein, `*` entspricht 0 oder mehrmals und `?` entspricht 0 oder 1 Mal.

---

## Anfangs- und Endzeilenindikatoren

Sie können die Position des Regex in der Zeichenfolge angeben:

- `"^..."` zwingt den regulären Ausdruck am Anfang der Zeichenfolge
- `"...$"` zwingt den regulären Ausdruck am Ende der Zeichenfolge

---

## Unterschiede zu anderen Sprachen

Bitte beachten Sie, dass reguläre Ausdrücke in R sucht oft *immer so leicht* von regulären Ausdrücken in anderen Sprachen verwendet.

- R erfordert doppelte Backslash-Escape-Zeichen (da `"\"` bereits generell in R-Strings ein

Escape-Zeichen enthält). Um beispielsweise Leerzeichen in den meisten regulären Ausdrucks-Engines zu erfassen, müssen Sie einfach `\s`, vs. `\\s` in R eingeben .

- UTF-8-Zeichen in R sollten mit einem Großbuchstaben U gekennzeichnet werden, z. B. `[\U{1F600}]` und `[\U1F600]` stimmen mit `whereas` überein, wohingegen in Ruby dies mit einem Kleinbuchstaben von `u` übereinstimmen würde.

---

## Zusätzliche Ressourcen

Die folgende Seite [reg101](#) ist ein guter Ort, um Online-Regex vor der Verwendung des R-Skripts zu überprüfen.

Das [R-Programming-Wikibook](#) enthält eine Seite für die Textverarbeitung mit vielen Beispielen mit regulären Ausdrücken.

## Examples

### Whitespace beseitigen

```
string <- ' some text on line one;
and then some text on line two '
```

### Leerzeichen trimmen

"Trimmen" Whitespace bezieht sich normalerweise auf das Entfernen von sowohl führenden als auch nachlaufenden Whitespaces aus einer Zeichenfolge. Dies kann unter Verwendung einer Kombination der vorherigen Beispiele erfolgen. `gsub` wird verwendet, um den Ersatz sowohl für das führende als auch für das `gsub` zu erzwingen.

Vor R 3.2.0

```
gsub(pattern = "^(^| +)$",
 replacement = "",
 x = string)

[1] "some text on line one; \nand then some text on line two"
```

R 3.2.0 und höher

```
trimws(x = string)

[1] "some text on line one; \nand then some text on line two"
```

### Führenden Whitespace entfernen

Vor R 3.2.0

```
sub(pattern = "^ +",
 replacement = "",
 x = string)
```

```
[1] "some text on line one; \nand then some text on line two "
```

## R 3.2.0 und höher

```
trimws(x = string,
 which = "left")
```

```
[1] "some text on line one; \nand then some text on line two "
```

## Nachlaufende Whitespace entfernen

### Vor R 3.2.0

```
sub(pattern = " +$",
 replacement = "",
 x = string)
```

```
[1] " some text on line one; \nand then some text on line two"
```

## R 3.2.0 und höher

```
trimws(x = string,
 which = "right")
```

```
[1] " some text on line one; \nand then some text on line two"
```

## Alle Leerzeichen entfernen

```
gsub(pattern = "\\s",
 replacement = "",
 x = string)
```

```
[1] "sometextonlineone;andthensometextonlinetwo"
```

Beachten Sie, dass dadurch auch Leerzeichen wie Tabulatoren ( `\t` ), Zeilenumbrüche ( `\r` und `\n` ) und Leerzeichen entfernt werden.

## Bestätigen Sie ein Datum im Format "JJJJMMTT"

Es ist eine übliche Praxis, Namen von Dateien mit dem Datum als Präfix in folgendem Format: `YYYYMMDD` , zum Beispiel: `20170101_results.csv` . Ein Datum in einem solchen Zeichenfolgenformat kann mit dem folgenden regulären Ausdruck überprüft werden:

```
\\d{4}(0[1-9]|1[012])(0[1-9]|12)[0-9]|3[01])
```

Der obige Ausdruck berücksichtigt Datumsangaben aus dem Jahr : 0000-9999 , Monate zwischen: 01-12 und den Tagen 01-31 .

Zum Beispiel:

```
> grepl("\\d{4}(0[1-9]|1[012])(0[1-9]|[12][0-9]|3[01])", "20170101")
[1] TRUE
> grepl("\\d{4}(0[1-9]|1[012])(0[1-9]|[12][0-9]|3[01])", "20171206")
[1] TRUE
> grepl("\\d{4}(0[1-9]|1[012])(0[1-9]|[12][0-9]|3[01])", "29991231")
[1] TRUE
```

**Hinweis :** Es validiert die Datumssyntax, aber wir können ein falsches Datum mit einer gültigen Syntax haben, zum Beispiel: 20170229 (2017 ist kein Schaltjahr).

```
> grepl("\\d{4}(0[1-9]|1[012])(0[1-9]|[12][0-9]|3[01])", "20170229")
[1] TRUE
```

Wenn Sie ein Datum validieren möchten, kann dies über diese benutzerdefinierte Funktion erfolgen:

```
is.Date <- function(x) {return(!is.na(as.Date(as.character(x), format = '%Y%m%d')))}
```

Dann

```
> is.Date(c("20170229", "20170101", 20170101))
[1] FALSE TRUE TRUE
```

## Validieren Sie die Postabkürzungen der US-Bundesstaaten

Der folgende `regex` umfasst 50 Staaten und auch Commonwealth / Territory (siehe [www.50states.com](http://www.50states.com)):

```
regex <-
"(A[LKSZJR])|(C[AOT])|(D[EC])|(F[ML])|(G[AU])|(HI)|(I[DLNA])|(K[SY])|(LA)|(M[EHDAINSOT])|(N[EVHJMYCD])|(O[HA])|(P[RIZ])|(T[X])|(V[VI])|(W[VA])|(Y[AK])|(Z[MI])"
```

Zum Beispiel:

```
> test <- c("AL", "AZ", "AR", "AJ", "AS", "DC", "FM", "GU", "PW", "FL", "AJ", "AP")
> grepl(us.states.pattern, test)
[1] TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
>
```

**Hinweis :**

Wenn Sie nur die 50 `state.abb` überprüfen möchten, empfehlen wir die Verwendung des R-Datensatzes: `state.abb` aus dem `state` .

```
> data(state)
> test %in% state.abb
[1] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE
```

Wir erhalten `TRUE` nur für Abkürzungen von 50 Staaten: `AL`, `AZ`, `AR`, `FL` .

## Bestätigen Sie die US-Telefonnummern

Der folgende reguläre Ausdruck:

```
us.phones.regex <- "^\\s*(\\+\\s*1(?:\\s+))*[0-9]{3}\\s*-?\\s*[0-9]{3}\\s*-?\\s*[0-9]{4}$"
```

Dieser Test prüft eine Telefonnummer in Form von: `+1-xxx-xxx-xxxx` , einschließlich optionalen führenden / Leerzeichen am Anfang / Ende jeder Gruppe von Zahlen, aber nicht in der Mitte, zum Beispiel Hinter: `+1-xxx-xxx-xx xx` ist nicht gültig. Die `-` Begrenzer kann durch Leerzeichen ersetzt werden: `xxx xxx xxx` oder ohne Trennzeichen: `xxxxxxxxxx` . Das `+1` Präfix ist optional.

Lass es uns überprüfen:

```
us.phones.regex <- "^\\s*(\\+\\s*1(?:\\s+))*[0-9]{3}\\s*-?\\s*[0-9]{3}\\s*-?\\s*[0-9]{4}$"

phones.OK <- c("305-123-4567", "305 123 4567", "+1-786-123-4567",
 "+1 786 123 4567", "7861234567", "786 - 123 4567", "+ 1 786 - 123 4567")

phones.NOK <- c("124-456-78901", "124-456-789", "124-456-78 90",
 "124-45 6-7890", "12 4-456-7890")
```

Gültige Fälle:

```
> grepl(us.phones.regex, phones.OK)
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
>
```

Ungültige Fälle:

```
> grepl(us.phones.regex, phones.NOK)
[1] FALSE FALSE FALSE FALSE FALSE
>
```

**Hinweis :**

- `\\s` Entspricht ein beliebiges Leerzeichen, Tabulatoren oder Zeilenende- Zeichen

## Escape-Zeichen in Regex-Mustern

Da sowohl R als auch Regex das `gsub "\"` , müssen beim `gsub` korrekter Muster für `grep` , `sub` , `gsub` oder einer anderen Funktion, die ein `gsub` akzeptiert, häufig Backslashes `gsub` . Wenn Sie einen Zeichenvektor mit drei Elementen erstellen, in dem ein Element einen Zeilenvorschub, ein anderes ein Tabulatorzeichen und kein Zeichen hat, und der Wunsch besteht, entweder den Zeilenvorschub oder den Tabulator in 4 Leerzeichen umzuwandeln, ist ein einzelner Backslash für die Konstruktion erforderlich aber paarweise Backslashes für das Matching:

```
x <- c("a\\nb", "c\\td", "e f")
x # how it's stored
```

```
[1] "a\nb" "c\td" "e f"
cat(x) # how it will be seen with cat
#a
#b c d e f

gsub(patt="\n|\t", repl=" ", x)
#[1] "a b" "c d" "e f"
```

Beachten Sie, dass das Musterargument (das optional ist, wenn es zuerst erscheint und nur eine teilweise Schreibweise benötigt) das einzige Argument ist, das diese Verdoppelung oder Paarung erfordert. Das Ersetzungsargument erfordert nicht die Verdoppelung von Zeichen, für die eine Escape-Aktion erforderlich ist. Wenn Sie möchten, dass alle Zeilenvorschübe und Vorkommen mit vier Leerzeichen durch Tabulatoren ersetzt werden, wäre dies:

```
gsub("\n| ", "\t", x)
#[1] "a\tb" "c\t d" "e\t f"
```

## Unterschiede zwischen Perl- und POSIX-Regex

In R gibt es zwei sehr unterschiedliche Engines für reguläre Ausdrücke. Die Standardeinstellung ist POSIX-konsistent. Alle Regex-Funktionen in R verfügen auch über eine Option zum Aktivieren des letzteren Typs: `perl = TRUE`.

## Vorausschau / Rückblick

`perl = TRUE` ermöglicht Vorausschau und Rückblick in regulären Ausdrücken.

- "`(?<=A)B`" entspricht einem Auftritt des Briefes `B` *nur dann*, wenn es durch vorangestellt ist `A`, das heißt "`ABACADABRA`" würde angepasst werden, aber "`abacadabra`" und "`aBacadabra`" würde nicht.

Reguläre Ausdrücke (Regex) online lesen: <https://riptutorial.com/de/r/topic/5748/regulare-ausdrucke--regex->

# Kapitel 94: Reguläre Ausdrücke in R

## Einführung

In diesem Dokument werden die Grundlagen für reguläre Ausdrücke, wie sie in R verwendet werden, vorgestellt. Weitere Informationen zur Syntax für reguläre Ausdrücke von R finden Sie unter "[?regex](#)". Eine umfassende Liste der Operatoren regulärer Ausdrücke finden Sie in [diesem ICU-Handbuch zu regulären Ausdrücken](#).

## Examples

Verwenden Sie ``grep``, um eine Zeichenfolge in einem Zeichenvektor zu finden

```
General syntax:
grep(<pattern>, <character vector>)

mystring <- c('The number 5',
 'The number 8',
 '1 is the loneliest number',
 'Company, 3 is',
 'Git SSH tag is git@github.com',
 'My personal site is www.personal.org',
 'path/to/my/file')

grep('5', mystring)
[1] 1
grep('@', mystring)
[1] 5
grep('number', mystring)
[1] 1 2 3
```

`x|y` bedeutet "x" oder "y" suchen

```
grep('5|8', mystring)
[1] 1 2
grep('com|org', mystring)
[1] 5 6
```

`.` ist ein Sonderzeichen in Regex. Es bedeutet "mit jedem Zeichen übereinstimmen"

```
grep('The number .', mystring)
[1] 1 2
```

Seien Sie vorsichtig, wenn Sie versuchen, Punkte zusammenzubringen!

```
tricky <- c('www.personal.org', 'My friend is a cyborg')
grep('.org', tricky)
[1] 1 2
```



Um ein Literalzeichen zu finden, müssen Sie die Zeichenfolge mit einem Backslash ( \ ) versehen. R versucht jedoch beim Erstellen von Zeichenfolgen nach Escape-Zeichen zu suchen. Daher müssen Sie den Backslash selbst mit Escapezeichen versehen (dh Sie müssen reguläre Ausdrücke *doppelt maskieren* .)

```
grep('\.org', tricky)
Error: '\.' is an unrecognized escape in character string starting "'\.'"
grep('\\.org', tricky)
[1] 1
```

Wenn Sie mit einem von mehreren Zeichen übereinstimmen möchten, können Sie diese Zeichen in Klammern einschließen ( [ ] ).

```
grep('[13]', mystring)
[1] 3 4
grep('[@/]', mystring)
[1] 5 7
```

Es kann nützlich sein, Zeichenfolgen anzugeben. Beispiel: [0-4] 0, 1, 2, 3 oder 4, [AZ] wird mit Großbuchstaben, [Az] mit Groß- oder Kleinbuchstaben und [A-z0-9] mit beliebigen Buchstaben übereinstimmen Buchstaben oder Zahlen (dh alle alphanumerischen Zeichen)

```
grep('[0-4]', mystring)
[1] 3 4
grep('[A-Z]', mystring)
[1] 1 2 4 5 6
```

R hat auch mehrere Shortcut-Klassen, die in Klammern verwendet werden können. Zum Beispiel ist [:lower:] eine Abkürzung für az , [:upper:] ist eine Abkürzung für AZ , [:alpha:] ist Az , [:digit:] ist 0-9 und [:alnum:] ist A-z0-9 . Beachten Sie, dass diese *ganzen Ausdrücke* in Klammern verwendet werden müssen. Um beispielsweise eine einzelne Ziffer abzugleichen, können Sie [[:digit:]] (beachten Sie die doppelten Klammern). Als weiteres Beispiel [@[[:digit:]]/] mit den Zeichen @ , / oder 0-9 überein.

```
grep('[[:digit:]]', mystring)
[1] 1 2 3 4
grep('[@[[:digit:]]/]', mystring)
[1] 1 2 3 4 5 7
```

Klammern können auch verwendet werden, um ein Match mit einem Karat ( ^ ) zu negieren. Zum Beispiel stimmt [^5] mit jedem anderen Zeichen als "5" überein.

```
grep('The number [^5]', mystring)
[1] 2
```

Reguläre Ausdrücke in R online lesen: <https://riptutorial.com/de/r/topic/9743/regulare-ausdrucke-in-r>

---

# Kapitel 95: Reproduzierbares R

## Einführung

Mit "Reproduzierbarkeit" meinen wir, dass jemand anderes (vielleicht Sie in der Zukunft) die von Ihnen durchgeführten Schritte wiederholen und dasselbe Ergebnis erzielen kann. Siehe die [reproduzierbare Forschungsaufgabensicht](#).

## Bemerkungen

Um reproduzierbare Ergebnisse zu erzielen, müssen alle Abweichungsquellen behoben werden. Wenn zum Beispiel ein (Pseudo) Zufallszahlengenerator verwendet wird, muss der Startwert korrigiert werden, wenn Sie dieselben Ergebnisse wiederherstellen möchten. Eine andere Möglichkeit, Abweichungen zu reduzieren, besteht darin, Text und Berechnung in demselben Dokument zu kombinieren.

---

## Verweise

- Peng, RD (2011). Reproduzierbare Forschung im Computational. Science, 334 (6060), 1226–1227. <http://doi.org/10.1126/science.1213847>
- Peng, Roger D. Report Writing für Data Science in R. Leanpub, 2015. <https://leanpub.com/reportwriting>.

## Examples

### Reproduzierbarkeit der Daten

`dput()` **und** `dget()`

Die einfachste Möglichkeit, einen (möglichst kleinen) `dput()` zu nutzen, ist die Verwendung der Basisfunktion `dput()`. Es wird ein R-Objekt in einem Nur-Text-Formular exportiert.

*Hinweis: Stellen Sie vor dem Erstellen der Beispieldaten sicher, dass Sie sich in einem leeren Ordner befinden, in den Sie schreiben können. Führen Sie `getwd()` und lesen Sie `?setwd` wenn Sie Ordner wechseln müssen.*

```
dput(mtcars, file = 'df.txt')
```

Dann kann jeder das präzise R-Objekt mit der Funktion `dget()` in seine GlobalEnvironment `dget()`.

```
df <- dget('df.txt')
```

Bei größeren R-Objekten gibt es mehrere Möglichkeiten, sie reproduzierbar zu speichern. Siehe [Eingabe und Ausgabe](#) .

## Paket-Reproduzierbarkeit

Die Reproduzierbarkeit von Paketen ist ein sehr häufiges Problem bei der Reproduktion von R-Code. Wenn verschiedene Pakete aktualisiert werden, können einige Verbindungen zwischen ihnen unterbrochen werden. Die ideale Lösung für dieses Problem besteht darin, das Abbild des Computers des R-Code-Brenners auf dem Computer zu dem Datum wiederzugeben, an dem der Code geschrieben wurde. Und hier kommt das `checkpoint` Paket.

Ab dem 17.09.2014 erstellen die Autoren des Pakets tägliche Kopien des gesamten CRAN-Paket-Repositorys in ihr eigenes Spiegel-Repository - Microsoft R Archived Network. Um beim Erstellen eines reproduzierbaren R-Projekts Probleme mit der Reproduzierbarkeit von Paketen zu vermeiden, müssen Sie nur Folgendes tun:

1. Stellen Sie sicher, dass alle Ihre Pakete (und die R-Version) auf dem neuesten Stand sind.
2. Fügen Sie die `checkpoint::checkpoint('YYYY-MM-DD')` Zeile `checkpoint::checkpoint('YYYY-MM-DD')` in Ihren Code ein.

`checkpoint` erstellt ein Verzeichnis `.checkpoint` in Ihrem `R_home`-Verzeichnis ( `"~/` "). In diesem technischen Verzeichnis werden alle Pakete installiert, die in Ihrem Projekt verwendet werden. Das heißt, `checkpoint .R` alle `.R` Dateien in Ihrem Projektverzeichnis, um alle `library()` oder `require()` Aufrufe `.R` und alle erforderlichen Pakete in der Form zu installieren, in der sie am angegebenen Datum bei CRAN vorhanden waren.

**PRO** Sie sind von der Frage der Reproduzierbarkeit des Pakets befreit.

**CONTRA** Für jedes angegebene Datum müssen Sie alle Pakete herunterladen und installieren, die in einem bestimmten Projekt verwendet werden, das Sie reproduzieren möchten. Das kann eine Weile dauern.

Reproduzierbares R online lesen: <https://riptutorial.com/de/r/topic/4087/reproduzierbares-r>

# Kapitel 96: RESTful R Services

## Einführung

OpenCPU verwendet Standard-R- [Pakete](#) zur Entwicklung, Auslieferung und Bereitstellung von Webanwendungen.

## Examples

### opencpu Apps

Die offizielle Website enthält ein gutes Beispiel für Apps: <https://www.opencpu.org/apps.html>

Der folgende Code wird zum Bereitstellen einer R-Sitzung verwendet:

```
library(opencpu)
opencpu$start(port = 5936)
```

Nachdem dieser Code ausgeführt wurde, können Sie URLs verwenden, um auf die Funktionen der R-Sitzung zuzugreifen. Das Ergebnis kann XML, HTML, JSON oder andere definierte Formate sein.

Beispielsweise kann auf die vorherige R-Sitzung mit einem cURL-Aufruf zugegriffen werden:

```
#curl uses http post method for -X POST or -d "arg=value"
curl http://localhost:5936/ocpu/library/MASS/scripts/ch01.R -X POST
curl http://localhost:5936/ocpu/library/stats/R/rnorm -d "n=10&mean=5"
```

Der Anruf ist asynchron, dh die R-Sitzung wird nicht blockiert, während auf den Abschluss des Anrufs gewartet wird (im Gegensatz zu glänzend).

Das Anrufergebnis wird in einer temporären Sitzung gespeichert, die in `/ocpu/tmp/` gespeichert ist.

Ein Beispiel zum Abrufen der temporären Sitzung:

```
curl https://public.opencpu.org/ocpu/library/stats/R/rnorm -d n=5
/ocpu/tmp/x009f9e7630/R/.val
/ocpu/tmp/x009f9e7630/stdout
/ocpu/tmp/x009f9e7630/source
/ocpu/tmp/x009f9e7630/console
/ocpu/tmp/x009f9e7630/info
```

`x009f9e7630` ist der Name der Sitzung.

Unter Hinweis auf `/ocpu/tmp/x009f9e7630/R/.val` wird den Wert von Rück resultierenden `rnorm(5)` , `/ocpu/tmp/x009f9e7630/R/console` wird , den Inhalt der Konsole Rückgabe `rnorm(5)` , etc ..

RESTful R Services online lesen: <https://riptutorial.com/de/r/topic/8323/restful-r-services>

# Kapitel 97: RMarkdown und Knitr-Präsentation

## Syntax

- Header:
  - YAML-Format, wird verwendet, wenn das Skript zur Definition allgemeiner Parameter und Metadaten kompiliert wird

## Parameter

| Parameter | Definition                                                                                                                                                          |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Titel     | der Titel des Dokuments                                                                                                                                             |
| Autor     | Der Autor des Dokuments                                                                                                                                             |
| Datum     | Das Datum des Dokuments: Kann " <code>r format(Sys.time(), '%d %B, %Y')</code> "                                                                                    |
| Autor     | Der Autor des Dokuments                                                                                                                                             |
| Ausgabe   | Das Ausgabeformat des Dokuments: mindestens 10 Formate verfügbar. Für das HTML-Dokument <code>html_output</code> . Für PDF-Dokument, <code>pdf_document</code> , .. |

## Bemerkungen

## Unteroptionsparameter:

| Unteroption      | Beschreibung                                                                 | html | pdf | Wort | odt | rtf | md | github | ioslides |
|------------------|------------------------------------------------------------------------------|------|-----|------|-----|-----|----|--------|----------|
| citation_package | Das LaTeX-Paket zur Bearbeitung von Zitaten, Natbib, Biblatex oder keiner    |      | X   |      |     |     | X  |        |          |
| code_folding     | Leser können die Anzeige des R-Codes umschalten, "none", "hide" oder "show". | X    |     |      |     |     |    |        |          |
| Colortheme       | Zu verwendendes Beamer-Farbthema                                             |      |     |      |     |     |    |        |          |

| Unteroption           | Beschreibung                                                                                  | html | pdf | Wort | odt | rtf | md | github | ioslides |
|-----------------------|-----------------------------------------------------------------------------------------------|------|-----|------|-----|-----|----|--------|----------|
| css                   | CSS-Datei, die zum Formatieren des Dokuments verwendet wird                                   | X    |     |      |     |     |    |        | X        |
| dev                   | Grafikgerät zur Ausgabe von Zahlen (z. B. "png")                                              | X    | X   |      |     |     | X  | X      | X        |
| Dauer                 | Fügen Sie einen Countdown-Timer (in Minuten) zur Fußzeile der Folien hinzu                    |      |     |      |     |     |    |        |          |
| fig_caption           | Sollen Zahlen mit Bildunterschriften gerendert werden?                                        | X    | X   | X    | X   |     |    |        | X        |
| fig_height, fig_width | Standardmäßige Höhe und Breite (in Zoll) des Dokuments                                        | X    | X   | X    | X   | X   | X  | X      | X        |
| Markieren             | Syntax-Hervorhebung: "Tango", "Pygments", "Kate", "Zenburn", "Textmate"                       | X    | X   | X    |     |     |    |        |          |
| beinhaltet            | Datei des Inhalts, der im Dokument platziert werden soll (in_header, before_body, after_body) | X    | X   |      | X   |     | X  | X      | X        |
| inkrementell          | Sollten Kugeln einzeln angezeigt werden (auf Mausklicks des Moderators)?                      |      |     |      |     |     |    |        | X        |
| keep_md               | Speichern Sie eine Kopie der .md-Datei, die die Knitr-Ausgabe enthält                         | X    |     | X    | X   | X   |    |        | X        |
| keep_tex              | Speichern Sie eine Kopie der .tex-Datei, die die Knitr-Ausgabe enthält                        |      | X   |      |     |     |    |        |          |
| latex_engine          | Engine zum Rendern von Latex oder "" pdflatex ", " xelatex ", lualatex"                       |      | X   |      |     |     |    |        |          |

| Unteroption     | Beschreibung                                                                                          | html | pdf | Wort | odt | rtf | md | github | ioslides |
|-----------------|-------------------------------------------------------------------------------------------------------|------|-----|------|-----|-----|----|--------|----------|
| lib_dir         | Verzeichnis der zu verwendenden Abhängigkeitsdateien (Bootstrap, MathJax usw.)                        | X    |     |      |     |     |    |        | X        |
| mathjax         | Legen Sie local oder eine URL fest, um eine lokale / URL-Version von MathJax zum Rendern zu verwenden | X    |     |      |     |     |    |        | X        |
| md_extensions   | Markdown-Erweiterungen zum Hinzufügen zur Standarddefinition oder R-Markdown                          | X    | X   | X    | X   | X   | X  | X      | X        |
| number_sections | Fügen Sie den Kopfzeilen Abschnittsnummerierungen hinzu                                               | X    | X   |      |     |     |    |        |          |
| pandoc_args     | Zusätzliche Argumente für Pandoc                                                                      | X    | X   | X    | X   | X   | X  | X      | X        |
| preserve_yaml   | YAML-Frontsache im endgültigen Dokument beibehalten?                                                  |      |     |      |     |     | X  |        |          |
| reference_docx  | docx-Datei, deren Stile beim Erstellen der docx-Ausgabe kopiert werden sollen                         |      |     | X    |     |     |    |        |          |
| selbstbehalten  | Einbetten von Abhängigkeiten in das Dokument                                                          | X    |     |      |     |     |    |        | X        |
| slide_level     | Die unterste Überschriftsebene, die die einzelnen Folien definiert                                    |      |     |      |     |     |    |        |          |
| kleiner         | Verwenden Sie die kleinere Schriftgröße in der Präsentation?                                          |      |     |      |     |     |    |        | X        |
| Clever          | Konvertieren Sie gerade Anführungszeichen in                                                          | X    |     |      |     |     |    |        | X        |

| Unteroption | Beschreibung                                                                               | html | pdf | Wort | odt | rtf | md | github | ioslides |
|-------------|--------------------------------------------------------------------------------------------|------|-----|------|-----|-----|----|--------|----------|
|             | geschweifte, Bindestriche in Bindestriche, ... in Ellipsen usw.                            |      |     |      |     |     |    |        |          |
| Vorlage     | Pandoc-Vorlage zum Rendern der Datei                                                       | X    | X   |      | X   |     |    |        |          |
|             |                                                                                            |      |     |      |     |     |    |        |          |
| Thema       | Bootswatch- oder Beamer-Design für die Seite                                               | X    |     |      |     |     |    |        |          |
| toc         | Fügen Sie am Anfang des Dokuments ein Inhaltsverzeichnis hinzu                             | X    | X   | X    |     | X   | X  | X      |          |
| toc_depth   | Die unterste Ebene der Überschriften, die dem Inhaltsverzeichnis hinzugefügt werden sollen | X    | X   | X    |     | X   | X  | X      |          |
| toc_float   | Verschieben Sie das Inhaltsverzeichnis links vom Hauptinhalt                               | X    |     |      |     |     |    |        |          |

## Examples

### Rstudio Beispiel

Dies ist ein Skript, das als .Rmd gespeichert wird, im Gegensatz zu r-Skripten, die als .R gespeichert werden.

Um das Skript zu stricken, entweder die `render` - Funktion oder die Shortcut - Taste in Rstudio verwenden.

```

title: "Rstudio example of a rmd file"
author: 'stack user'
date: "22 July 2016"
output: html_document

```

The header is used to define the general parameters and the metadata.

```
R Markdown
```

This is an R Markdown document.  
It is a script written in markdown with the possibility to insert chunk of R code in it.  
To insert R code, it needs to be encapsulated into inverted quote.



Like that for a long piece of code:

```
```{r cars}
summary(cars)
```
```

And like ``r cat("that")`` for small piece of code.

## Including Plots

You can also embed plots, for example:

```
```{r echo=FALSE}
plot(pressure)
```
```

## Hinzufügen einer Fußzeile zu einer Ioslides-Präsentation

Das Hinzufügen einer Fußzeile ist nicht nativ möglich. Glücklicherweise können wir mit jQuery und CSS eine Fußzeile zu den Folien einer mit knitr gerenderten Ioslides-Präsentation hinzufügen. Zunächst müssen wir das jQuery-Plugin hinzufügen. Dies geschieht durch die Leitung

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
```

Jetzt können wir jQuery verwenden, um das DOM ( *Document Object Model* ) unserer Präsentation zu ändern. Mit anderen Worten: Wir ändern die HTML-Struktur des Dokuments. Sobald die Präsentation geladen ist ( `$(document).ready(function() { ... })` ), wählen wir alle Folien aus, die nicht die Klassenattribute `.title-slide` , `.backdrop` oder `.segue` und haben Fügen Sie das Tag `<footer></footer>` direkt vor dem Schließen jeder Folie ein (also vor `</slide>` ). Das Attribut `label` trägt den Inhalt , die später angezeigt.

Jetzt müssen wir nur noch unsere Fußzeile mit CSS gestalten:

Nach jedem `<footer>` ( `footer::after` ):

- Anzeigen des Inhalts der `label`
- Schriftgröße 12 verwenden
- Positionieren Sie die Fußzeile (20 Pixel von der Unterseite der Folie und 60 px von links).

(Die anderen Eigenschaften können ignoriert werden, müssen jedoch möglicherweise geändert werden, wenn die Präsentation eine andere Stilvorlage verwendet.)

```

title: "Adding a footer to presentaion slides"
author: "Martin Schmelzer"
date: "26 Juli 2016"
output: ioslides_presentation

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
```
```

```

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>

<script>
 $(document).ready(function() {
 $('slide:not(.title-slide, .backdrop, .segue)').append('<footer label=\\"My amazing
 footer!\\ "></footer>');
 })
</script>

<style>
 footer:after {
 content: attr(label);
 font-size: 12pt;
 position: absolute;
 bottom: 20px;
 left: 60px;
 line-height: 1.9;
 }
</style>

Slide 1

This is slide 1.

Slide 2

This is slide 2

Test

Slide 3

And slide 3.

```

Das Ergebnis sieht so aus:

# Slide 1

This is slide 1.

My amazing footer

RMarkdown und Knitr-Präsentation online lesen: <https://riptutorial.com/de/r/topic/2999/rmarkdown-und-knitr-prasentation>

# Kapitel 98: RODBC

## Examples

### Verbindung zu Excel-Dateien über RODBC

Während `RODBC` auf Windows-Computer mit einer kompatiblen Architektur zwischen R und jedem Ziel-RDMS beschränkt ist, besteht eine der Hauptflexibilitäten darin, mit Excel-Dateien so zu arbeiten, als wären sie SQL-Datenbanken.

```
require(RODBC)
con = odbcConnectExcel("myfile.xlsx") # open a connection to the Excel file
sqlTables(con)$TABLE_NAME # show all sheets
df = sqlFetch(con, "Sheet1") # read a sheet
df = sqlQuery(con, "select * from [Sheet1 $]") # read a sheet (alternative SQL syntax)
close(con) # close the connection to the file
```

### SQL Server-Verwaltungsdatenbankverbindung zum Abrufen einzelner Tabellen

Eine weitere Verwendung von RODBC ist die Verbindung mit der SQL Server-Verwaltungsdatenbank. Wir müssen hier den 'Treiber', dh SQL Server, den Datenbanknamen "Atilla" `sqlQuery` und dann die `sqlQuery`, um entweder die vollständige Tabelle oder einen Bruchteil davon zu extrahieren.

```
library(RODBC)
cn <- odbcDriverConnect(connection="Driver={SQL
Server};server=localhost;database=Atilla;trusted_connection=yes;")
tbl <- sqlQuery(cn, 'select top 10 * from table_1')
```

### Verbindung zu relationalen Datenbanken

```
library(RODBC)
con <- odbcDriverConnect("driver={Sql Server};server=servername;trusted connection=true")
dat <- sqlQuery(con, "select * from table");
close(con)
```

Dadurch wird eine Verbindung zu einer SQL Server-Instanz hergestellt. Weitere Informationen dazu, wie Ihre Verbindungszeichenfolge aussehen soll, finden Sie unter [connectionstrings.com](https://connectionstrings.com)

Da keine Datenbank angegeben ist, sollten Sie sicherstellen, dass Sie das Objekt, das Sie abfragen möchten, vollständig wie folgt `database.schema.objectname` qualifizieren

RODBC online lesen: <https://riptutorial.com/de/r/topic/2471/rodbc>

# Kapitel 99: Rohoperatoren (%>% und andere)

## Einführung

Pipe-Operatoren, die in `magrittr`, `dplyr` und anderen R-Paketen verfügbar sind, verarbeiten ein Datenobjekt unter Verwendung einer Folge von Operationen, indem das Ergebnis eines Schritts als Eingabe für den nächsten Schritt unter Verwendung von Infix-Operatoren anstelle der üblicheren R-Methode von verschachtelten übergeben wird Funktionsaufrufe.

Beachten Sie, dass das Ziel der Pipe-Operatoren darin besteht, die Lesbarkeit von geschriebenem Code für den Menschen zu verbessern. Weitere Informationen zur Leistung finden Sie im Abschnitt Anmerkungen.

## Syntax

- `lhs%>% rhs` # Pipe-Syntax für `rhs(lhs)`
- `lhs%>% rhs (a = 1)` # Pipe-Syntax für `rhs(lhs, a = 1)`
- `lhs%>% rhs (a = 1, b =.)` # `rhs(a = 1, b = lhs)` für `rhs(a = 1, b = lhs)`
- `lhs% <>% rhs` # Pipe-Syntax für `lhs <- rhs(lhs)`
- `lhs% $% rhs (a)` # Pipe-Syntax für `with(lhs, rhs(lhs$a))`
- `lhs% T>% rhs` # Pipe-Syntax für `{ rhs(lhs); lhs }`

## Parameter

| lhs                                     | rhs                                                        |
|-----------------------------------------|------------------------------------------------------------|
| Ein Wert oder der Magrittr-Platzhalter. | Ein Funktionsaufruf unter Verwendung der Magrittr-Semantik |

## Bemerkungen

### Pakete, die %>%

Der Pipe-Operator ist im `magrittr` Paket definiert, er gewann jedoch mit dem `dplyr` Paket (das die Definition aus `magrittr`) große Sichtbarkeit und Beliebtheit. Jetzt ist es Teil von `tidyverse`, einer Sammlung von Paketen, die *"harmonisch zusammenarbeiten, weil sie gemeinsame Datenrepräsentationen und API-Design verwenden"*.

Das `magrittr` Paket bietet auch mehrere Variationen des Pipe-Operators für diejenigen, die mehr Flexibilität beim Paspeln wünschen, wie z. B. die zusammengesetzte Zuweisungspipe `%<>%`, die Expositionspipe `$$` und den T-Operator `%T>%`. Es bietet auch eine Reihe von Alias-Funktionen, um gängige Funktionen mit einer speziellen Syntax (`+`, `[`, `[[` usw.) zu ersetzen, sodass sie problemlos innerhalb einer Pipeline-Kette verwendet werden können.

## Dokumentation finden

Wie bei jedem *Infix-Operator* (wie `+`, `*`, `^`, `&`, `%in%`) finden Sie die offizielle Dokumentation, wenn Sie sie in Anführungszeichen setzen: `? '%>%'` Oder `help('%>%')` (Angenommen, Sie haben ein Paket geladen, das `pkg:magrittr`) `pkg:magrittr`.

## Hotkeys

In [RStudio](#) gibt es einen speziellen Hotkey für den Pipe-Operator: `Ctrl+Shift+M` (*Windows und Linux*), `Cmd+Shift+M` (*Mac*).

## Überlegungen zur Leistung

Beachten Sie, dass der Pipe-Operator nützlich ist. Beachten Sie jedoch, dass die Leistung hauptsächlich durch den Overhead der Verwendung beeinträchtigt wird. Berücksichtigen Sie bei der Verwendung des Pipe-Operators die folgenden beiden Punkte sorgfältig:

- Maschinenleistung (Schleifen)
- Auswertung (`object %>% rm()` entfernt `object`)

## Examples

### Grundlegende Verwendung und Verkettung

Der Pipe-Operator `%>%` wird verwendet, um ein Argument in eine Funktion einzufügen. Es ist keine `magrittr` der Sprache und kann nur verwendet werden, nachdem ein Paket, das es bereitstellt, wie z. B. `magrittr`, `magrittr`. Der Pipe-Operator nimmt die linke Seite (LHS) der Pipe und verwendet sie als erstes Argument der Funktion auf der rechten Seite (RHS) der Pipe. Zum Beispiel:

```
library(magrittr)

1:10 %>% mean
[1] 5.5

is equivalent to
mean(1:10)
[1] 5.5
```

Die Pipe kann verwendet werden, um eine Folge von Funktionsaufrufen zu ersetzen. Mehrere Pipes ermöglichen das Lesen und Schreiben der Sequenz von links nach rechts und nicht von innen nach außen. Angenommen, wir haben `years` als Faktor definiert, möchten diesen jedoch in eine Zahl umwandeln. Um möglichen Informationsverlust zu vermeiden, konvertieren wir zuerst in

## Zeichen und dann in Zahlen.

```
years <- factor(2008:2012)

nesting
as.numeric(as.character(years))

piping
years %>% as.character %>% as.numeric
```

Wenn wir nicht möchten, dass die LHS (linke Seite) als *erstes* Argument auf der rechten Seite (rechte Seite) verwendet wird, gibt es Abhilfemaßnahmen, z . um anzugeben, wohin die Pipe-Eingabe geht.

```
example with grepl
its syntax:
grepl(pattern, x, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)

note that the `substring` result is the *2nd* argument of grepl
grepl("Wo", substring("Hello World", 7, 11))

piping while naming other arguments
"Hello World" %>% substring(7, 11) %>% grepl(pattern = "Wo")

piping with .
"Hello World" %>% substring(7, 11) %>% grepl("Wo", .)

piping with . and curly braces
"Hello World" %>% substring(7, 11) %>% { c(paste('Hi', .)) }
#[1] "Hi World"

#using LHS multiple times in argument with curly braces and .
"Hello World" %>% substring(7, 11) %>% { c(paste(. , 'Hi', .)) }
#[1] "World Hi World"
```

## Funktionsabläufe

Bei einer Reihe von Schritten, die wir wiederholt verwenden, ist es oft praktisch, sie in einer Funktion zu speichern. Pipes ermöglichen das Speichern solcher Funktionen in einem lesbaren Format, indem eine Sequenz mit einem Punkt wie folgt gestartet wird:

```
. %>% RHS
```

Angenommen, wir haben Faktordaten und möchten das Jahr extrahieren:

```
library(magrittr) # needed to include the pipe operators
library(lubridate)
read_year <- . %>% as.character %>% as.Date %>% year

Creating a dataset
df <- data.frame(now = "2015-11-11", before = "2012-01-01")
now before
1 2015-11-11 2012-01-01
```

```
Example 1: applying `read_year` to a single character-vector
df$now %>% read_year
[1] 2015

Example 2: applying `read_year` to all columns of `df`
df %>% lapply(read_year) %>% as.data.frame # implicit `lapply(df, read_year)`
now before
1 2015 2012

Example 3: same as above using `mutate_all`
library(dplyr)
df %>% mutate_all(funs(read_year))
if an older version of dplyr use `mutate_each`
now before
1 2015 2012
```

Wir können die Zusammensetzung der Funktion überprüfen, indem Sie ihren Namen eingeben oder `functions` :

```
read_year
Functional sequence with the following components:
#
1. as.character(.)
2. as.Date(.)
3. year(.)
#
Use 'functions' to extract the individual functions.
```

Wir können auf jede Funktion auch über ihre Position in der Reihenfolge zugreifen:

```
read_year[[2]]
function (.)
as.Date(.)
```

Im Allgemeinen kann dieser Ansatz nützlich sein, wenn Klarheit wichtiger ist als Geschwindigkeit.

## Zuordnung mit `%<>%`

Das `magrittr` Paket enthält einen zusammengesetzten Zuweisungs-Infix-Operator `%<>%`, der einen Wert aktualisiert, indem er ihn zuerst in einen oder mehrere `rhs` Ausdrücke `rhs` und dann das Ergebnis zuweist. Dadurch müssen Sie den Objektnamen nicht zweimal eingeben (einmal auf jeder Seite des Zuweisungsoperators `<-`). `%<>%` muss der erste Infix-Operator in einer Kette sein:

```
library(magrittr)
library(dplyr)

df <- mtcars
```

Anstatt zu schreiben

```
df <- df %>% select(1:3) %>% filter(mpg > 20, cyl == 6)
```

oder



```
df %>% select(1:3) %>% filter(mpg > 20, cyl == 6) -> df
```

Der zusammengesetzte Zuweisungsoperator leitet `df` sowohl weiter als auch neu:

```
df %>% select(1:3) %>% filter(mpg > 20, cyl == 6)
```

## Inhalte mit `%$%` verfügbar machen

Der Exposition-Pipe-Operator `%$%` macht die Spaltennamen als R-Symbole im linken Objekt für den Ausdruck auf der rechten Seite verfügbar. Dieser Operator ist praktisch, wenn in Funktionen Rohrleitungen, die keine haben `data` (im Gegensatz zu, sagen wir, Argument `lm`) und nehmen Sie nicht eine `data.frame` und Spaltennamen als Argumente (die meisten der wichtigsten dplyr Funktionen).

Mit dem Exposition-Pipeline-Operator `%$%` kann ein Benutzer verhindern, dass eine Pipeline beschädigt wird, wenn auf Spaltennamen verwiesen werden muss. Angenommen, Sie möchten ein `data.frame` filtern und dann mit `cor.test` einen Korrelationstest für zwei Spalten `cor.test`:

```
library(magrittr)
library(dplyr)
mtcars %>%
 filter(wt > 2) %$%
 cor.test(hp, mpg)

#>
#> Pearson's product-moment correlation
#>
#> data: hp and mpg
#> t = -5.9546, df = 26, p-value = 2.768e-06
#> alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#> -0.8825498 -0.5393217
#> sample estimates:
#> cor
#> -0.7595673
```

Hier durchläuft die standardmäßige Pipe `%>%` das `data.frame` an `filter()`, während die Pipe `%$%` die Spaltennamen für `cor.test()`.

Die Expositionspipe funktioniert wie eine Pipe-Version der Basis R `with()` Funktionen, und die gleichen Objekte auf der linken Seite werden als Eingaben akzeptiert.

## Verwendung der Pipe mit Dplyr und ggplot2

Der Operator `%>%` kann auch verwendet werden, um die Ausgabe von dplyr an ggplot zu übergeben. Dadurch wird eine EDA-Pipeline (Unified Exploratory Data Analysis) geschaffen, die leicht anpassbar ist. Diese Methode ist schneller als die Aggregation intern in ggplot und hat den zusätzlichen Vorteil, dass unnötige Zwischenvariablen vermieden werden.

```
library(dplyr)
library(ggplot2)
```

```
diamonds %>%
 filter(depth > 60) %>%
 group_by(cut) %>%
 summarize(mean_price = mean(price)) %>%
 ggplot(aes(x = cut, y = mean_price)) +
 geom_bar(stat = "identity")
```

## Nebeneffekte mit %T>% erstellen

Einige Funktionen in R erzeugen einen Nebeneffekt (dh Speichern, Drucken, Plotten usw.) und geben nicht immer einen sinnvollen oder gewünschten Wert zurück.

%T>% (Abschlagoperator) können Sie einen Wert in eine Nebeneffektfunktion `lhs` wobei der ursprüngliche `lhs` Wert erhalten bleibt. Mit anderen Worten: Der Tee-Operator funktioniert wie %>% , nur dass die Rückgabewerte `lhs` selbst sind und nicht das Ergebnis der `rhs` Funktion / des `rhs` Ausdrucks.

Beispiel: Ein Objekt erstellen, übermitteln, schreiben und zurückgeben. Wenn in diesem Beispiel %>% anstelle von %T>% , enthält die Variable `all_letters` `NULL` und nicht den Wert des sortierten Objekts.

```
all_letters <- c(letters, LETTERS) %>%
 sort %T>%
 write.csv(file = "all_letters.csv")

read.csv("all_letters.csv") %>% head()
x
1 a
2 A
3 b
4 B
5 c
6 C
```

Warnung: Wenn Sie ein nicht benanntes Objekt an `save()` weiterleiten, wird ein Objekt mit dem Namen erzeugt . wenn mit `load()` in den Arbeitsbereich geladen. Eine Problemumgehung mithilfe einer Hilfsfunktion ist jedoch möglich (die auch als anonyme Funktion inline geschrieben werden kann).

```
all_letters <- c(letters, LETTERS) %>%
 sort %T>%
 save(file = "all_letters.RData")

load("all_letters.RData", e <- new.env())

get("all_letters", envir = e)
Error in get("all_letters", envir = e) : object 'all_letters' not found

get(".", envir = e)
[1] "a" "A" "b" "B" "c" "C" "d" "D" "e" "E" "f" "F" "g" "G" "h" "H" "i" "I" "j" "J"
[21] "k" "K" "l" "L" "m" "M" "n" "N" "o" "O" "p" "P" "q" "Q" "r" "R" "s" "S" "t" "T"
[41] "u" "U" "v" "V" "w" "W" "x" "X" "y" "Y" "z" "Z"
```

```
Work-around
save2 <- function(. = ., name, file = stop("'file' must be specified")) {
 assign(name, .)
 call_save <- call("save", ... = name, file = file)
 eval(call_save)
}

all_letters <- c(letters, LETTERS) %>%
 sort %T>%
 save2("all_letters", "all_letters.RData")
```

Rohroperatoren (%>% und andere) online lesen:

<https://riptutorial.com/de/r/topic/652/rohroperatoren----gt---und-andere->

# Kapitel 100: roxygen2

## Parameter

| Parameter       | Einzelheiten                                                                        |
|-----------------|-------------------------------------------------------------------------------------|
| Autor           | Autor des Pakets                                                                    |
| Beispiele       | Die folgenden Zeilen enthalten Beispiele zur Verwendung der dokumentierten Funktion |
| Export          | Funktion exportieren - dh von Benutzern des Pakets aufrufbar machen                 |
| eingeführen     | Paket (s) Namespace (s) zum Importieren                                             |
| Importieren von | Funktionen, die aus dem Paket importiert werden sollen (Vorname der Liste)          |
| param           | Parameter der zu dokumentierenden Funktion                                          |

## Examples

### Dokumentation eines Pakets mit roxygen2

## Schreiben mit roxygen2

[roxygen2](#) ist ein Paket, das von Hadley Wickham zur Vereinfachung der Dokumentation erstellt wurde.

Es erlaubt, die Dokumentation in das R-Skript aufzunehmen, beginnend mit `#'`. Die verschiedenen Parameter, die an die Dokumentation übergeben werden, beginnen mit einem `@`. Der Ersteller eines Pakets wird beispielsweise wie folgt geschrieben:

```
#' @author The Author
```

Zum Beispiel, wenn wir die folgende Funktion dokumentieren wollten:

```
mean<-function(x) sum(x)/length(x)
```

Wir möchten eine kleine Beschreibung dieser Funktion schreiben und die Parameter mit folgendem erläutern (jede Zeile wird erklärt und detailliert beschrieben):

```
#' Mean
#'
```

```
#' A function to compute the mean of a vector
#' @param x A numeric vector
#' @keyword mean
#' @importFrom base sum
#' @export
#' @examples
#' mean(1:3)
#' \dontrun{ mean(1:1e99) }
mean<-function(x) sum(x)/length(x)
```

- Die erste Zeile `#' Mean` ist der Titel der Dokumentation, die folgenden Zeilen bilden den Korpus.
- Jeder Parameter einer Funktion muss durch ein relevantes `@param` detailliert sein. `@export` an, dass dieser Funktionsname exportiert werden sollte und daher aufgerufen werden kann, wenn das Paket geladen wird.
- `@keyword` liefert relevante Schlüsselwörter, wenn Sie nach Hilfe suchen
- `@importFrom` listet alle Funktionen auf, die aus einem Paket importiert werden sollen, das in dieser Funktion oder in `@importFrom` Paket verwendet wird. Beachten Sie, dass der Import des vollständigen Namespaces eines Pakets mit `@import`
- Die Beispiele werden dann unter dem Tag `@example` .
  - Die erste wird ausgewertet, wenn das Paket erstellt wird.
  - Der zweite Befehl wird aufgrund des `\dontrun` normalerweise nicht verwendet, um lange Berechnungen zu verhindern.

## Erstellung der Dokumentation

Die Dokumentation kann mit `devtools::document()` . Beachten Sie auch, dass `devtools::check()` automatisch eine Dokumentation erstellt und fehlende Argumente in der Dokumentation der Funktionen als Warnungen meldet.

roxygen2 online lesen: <https://riptutorial.com/de/r/topic/5171/roxygen2>

---

# Kapitel 101: R-Version wird aktualisiert

## Einführung

Durch das Installieren oder Aktualisieren Ihrer Software erhalten Sie Zugriff auf neue Funktionen und Fehlerbehebungen. Die Aktualisierung Ihrer R-Installation kann auf verschiedene Arten erfolgen. Eine einfache Möglichkeit ist, die [R-Website aufzurufen](https://cran.r-project.org/) und die neueste Version für Ihr System herunterzuladen.

## Examples

### Installation von der R-Website

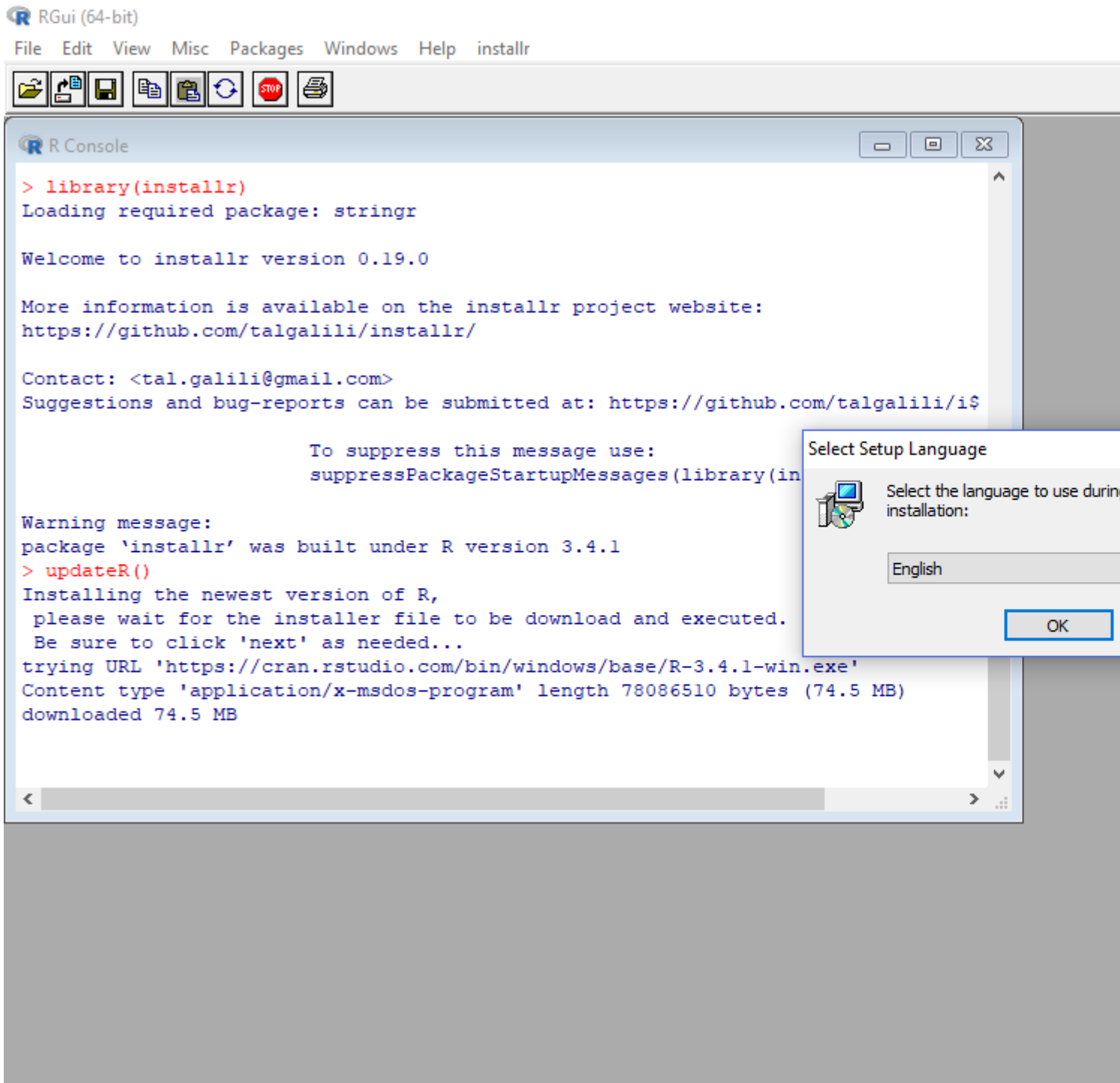
Um die neueste Version zu erhalten, gehen Sie zu <https://cran.r-project.org/> und laden Sie die Datei für Ihr Betriebssystem herunter. Öffnen Sie die heruntergeladene Datei und befolgen Sie die Installationsschritte auf dem Bildschirm. Alle Einstellungen können standardmäßig beibehalten werden, es sei denn, Sie möchten ein bestimmtes Verhalten ändern.

### Aktualisierung von R aus mit dem installr-Paket

Sie können R auch von R aus aktualisieren, indem Sie ein praktisches Paket namens **installr** verwenden .

Öffnen Sie R Console (NICHT RStudio, dies funktioniert nicht von RStudio aus) und führen Sie den folgenden Code aus, um das Paket zu installieren und das Update zu starten.

```
install.packages("installr")
library("installr")
updateR()
```



## Entscheidung über die alten Pakete

Klicken Sie nach Abschluss der Installation auf die Schaltfläche Fertig stellen.

Nun werden Sie gefragt, ob Sie Ihre Pakete für die ältere Version von R in die neuere Version von R kopieren möchten. Wenn Sie Ja wählen, wird das gesamte Paket in die neuere Version von R kopiert.



```
> library(installr)
Loading required package: stringr

Welcome to installr version 0.19.0

More information is available on the installr project website:
https://github.com/talgalili/installr/

Contact: <tal.galili@gmail.com>
Suggestions and bug-reports can be submitted at: https://github.com/talgalili/i$

To suppress this message use:
suppressPackageStartupMessages()

Warning message:
package 'installr' was built under R version 3.4.1
> updateR()
Installing the newest version of R,
please wait for the installer file to be download and
Be sure to click 'next' as needed...
trying URL 'https://cran.rstudio.com/bin/windows/base/R341.exe'
Content type 'application/x-msdos-program' length 7808640 bytes
downloaded 74.5 MB
```

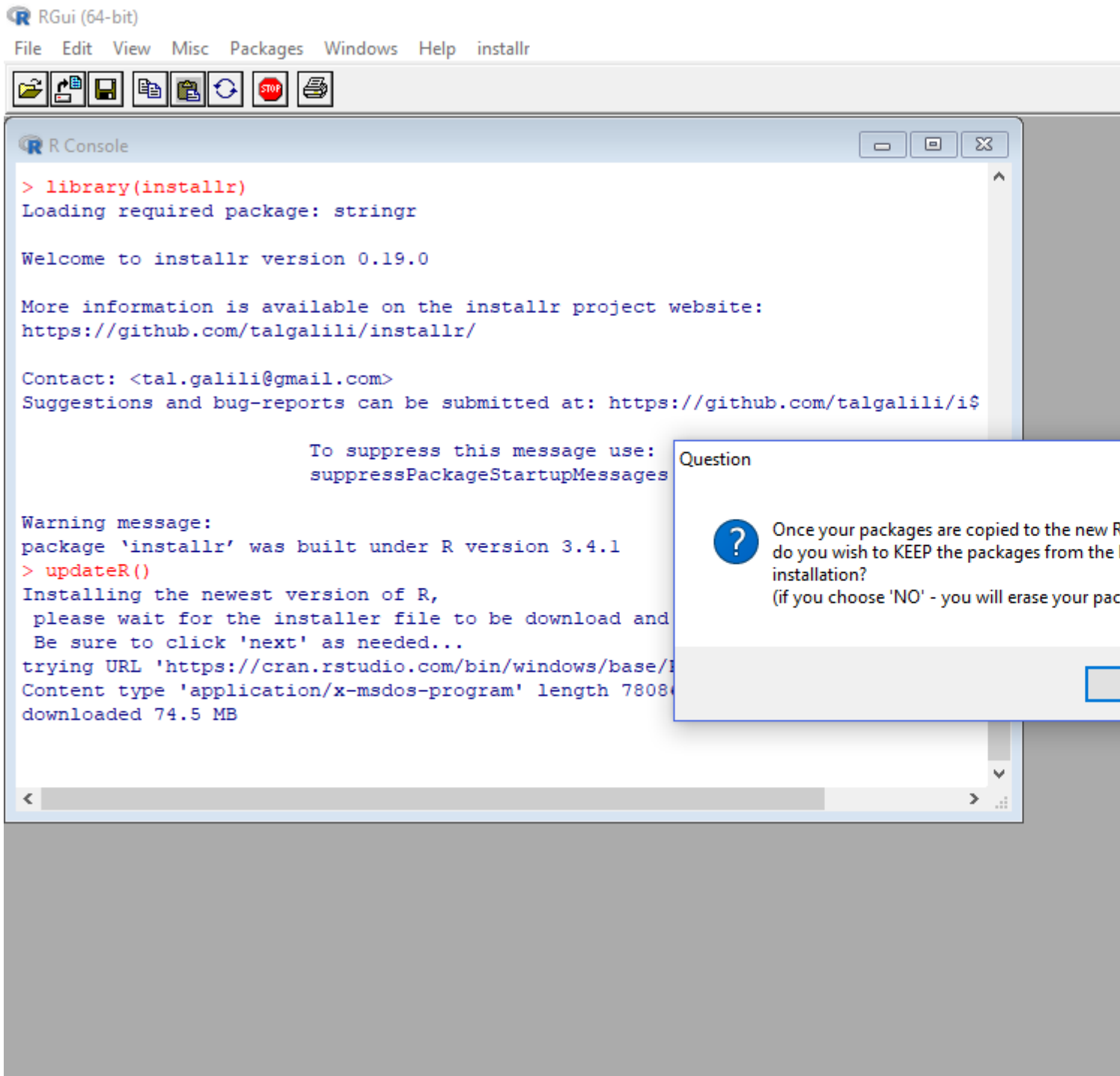
## Question



Do you wish to copy your packages from the newer version of R?

Danach können Sie wählen, ob Sie die alten Pakete noch behalten oder löschen möchten.





Sie können Ihre Rprofile.site sogar von einer älteren Version verschieben, um alle benutzerdefinierten Einstellungen beizubehalten.



```
> library(installr)
Loading required package: stringr

Welcome to installr version 0.19.0

More information is available on the installr project website:
https://github.com/talgalili/installr/

Contact: <tal.galili@gmail.com>
Suggestions and bug-reports can be submitted at: https://github.com/talgalili/i$

To suppress this message use:
suppressPackageStartupMessages()

Warning message:
package 'installr' was built under R version 3.4.1
> updateR()
Installing the newest version of R,
please wait for the installer file to be download and
Be sure to click 'next' as needed...
trying URL 'https://cran.rstudio.com/bin/windows/base/R
Content type 'application/x-msdos-program' length 78086
downloaded 74.5 MB
```

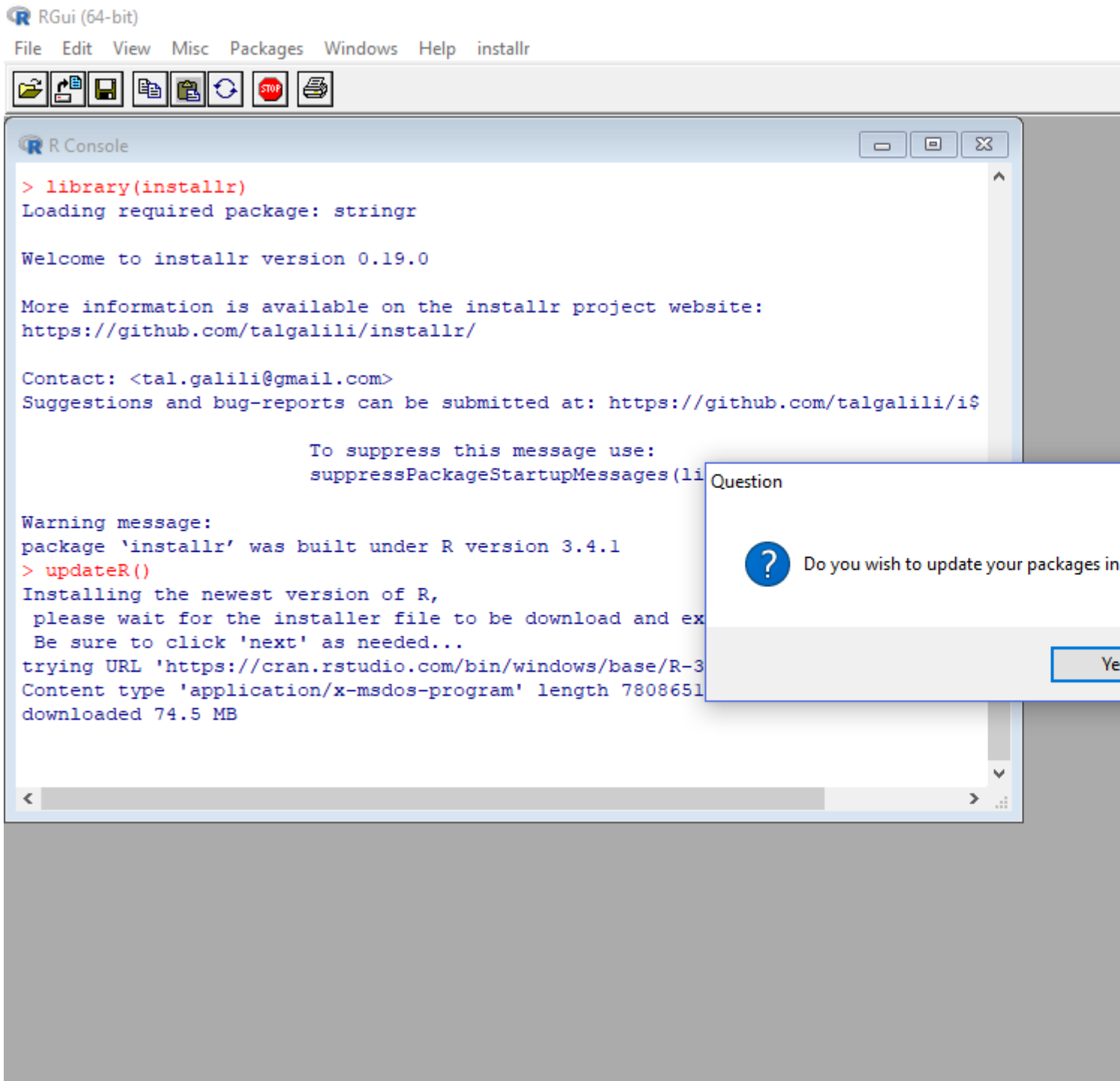
## Question



Do you wish to copy your 'Rprofile.site' from the newer version of R?

## Pakete aktualisieren

Sie können Ihre installierten Pakete aktualisieren, sobald die Aktualisierung von R abgeschlossen ist.



Sobald es fertig ist, starten Sie R neu und genießen Sie die Erkundung.

## Überprüfen Sie die R-Version

Sie können die R-Version über die Konsole überprüfen

```
version
```

R-Version wird aktualisiert online lesen: <https://riptutorial.com/de/r/topic/10729/r-version-wird-aktualisiert>

# Kapitel 102: Schreibfunktionen in R

## Examples

### Benannte Funktionen

R ist voll mit Funktionen, es ist immerhin eine [funktionale Programmiersprache](#), aber manchmal ist die genaue Funktion, die Sie benötigen, nicht in den Basisressourcen enthalten. Es ist denkbar, dass Sie [ein Paket installieren](#), das die Funktion enthält. Vielleicht sind Ihre Anforderungen jedoch so spezifisch, dass keine vorgefertigte Funktion passt. Dann haben Sie die Möglichkeit, Ihre eigenen zu machen.

Eine Funktion kann sehr einfach sein, so dass sie ziemlich sinnlos ist. Es braucht nicht einmal ein Argument:

```
one <- function() { 1 }
one()
[1] 1

two <- function() { 1 + 1 }
two()
[1] 2
```

Zwischen den geschweiften Klammern { } ist die eigentliche Funktion. Solange Sie alles in eine einzige Zeile einfügen können, sind sie nicht unbedingt erforderlich, können aber hilfreich sein, um die Dinge in Ordnung zu halten.

Eine Funktion kann sehr einfach und dennoch sehr spezifisch sein. Diese Funktion nimmt als Eingabe einen Vektor (in diesem Beispiel `vec`) und gibt den gleichen Vektor aus, wobei die Länge des Vektors (in diesem Fall 6) von jedem der Elemente des Vektors abgezogen wird.

```
vec <- 4:9
subtract.length <- function(x) { x - length(x) }
subtract.length(vec)
[1] -2 -1 0 1 2 3
```

Beachten Sie, dass `length()` an sich eine vorab gelieferte (dh *Basis-*) Funktion ist. Sie können natürlich eine zuvor selbst erstellte Funktion innerhalb einer anderen selbst erstellten Funktion verwenden sowie Variablen zuweisen und andere Operationen ausführen, während Sie sich über mehrere Zeilen erstrecken:

```
vec2 <- (4:7)/2

msdf <- function(x, multiplier=4) {
 mult <- x * multiplier
 subl <- subtract.length(x)
 data.frame(mult, subl)
}
```

```
msdf(vec2, 5)
 mult subl
1 10.0 -2.0
2 12.5 -1.5
3 15.0 -1.0
4 17.5 -0.5
```

`multiplier=4` stellt sicher, dass 4 der Standardwert des Argumentmultiplikators `multiplier`, wenn beim Aufruf der Funktion 4 kein Wert angegeben wird.

Die oben genannten sind Beispiele für *benannte* Funktionen, die einfach genannt werden, weil ihnen Namen gegeben wurden (`one`, `two`, `subtract.length` usw.).

## Anonyme Funktionen

Eine anonyme Funktion erhält, wie der Name schon sagt, keinen Namen. Dies kann nützlich sein, wenn die Funktion Teil einer größeren Operation ist, an sich aber nicht viel Platz einnimmt. Ein häufiger Anwendungsfall für anonyme Funktionen ist innerhalb der `*apply` Familie von Basisfunktionen.

Berechnen Sie den quadratischen Mittelwert für jede Spalte in einem `data.frame`:

```
df <- data.frame(first=5:9, second=(0:4)^2, third=-1:3)

apply(df, 2, function(x) { sqrt(sum(x^2)) })
 first second third
15.968719 18.814888 3.872983
```

Erstellen Sie eine Sequenz von Schrittlänge eins vom kleinsten zum größten Wert für jede Zeile in einer Matrix.

```
x <- sample(1:6, 12, replace=TRUE)
mat <- matrix(x, nrow=3)

apply(mat, 1, function(x) { seq(min(x), max(x)) })
```

Eine anonyme Funktion kann auch für sich alleine stehen:

```
(function() { 1 })()
[1] 1
```

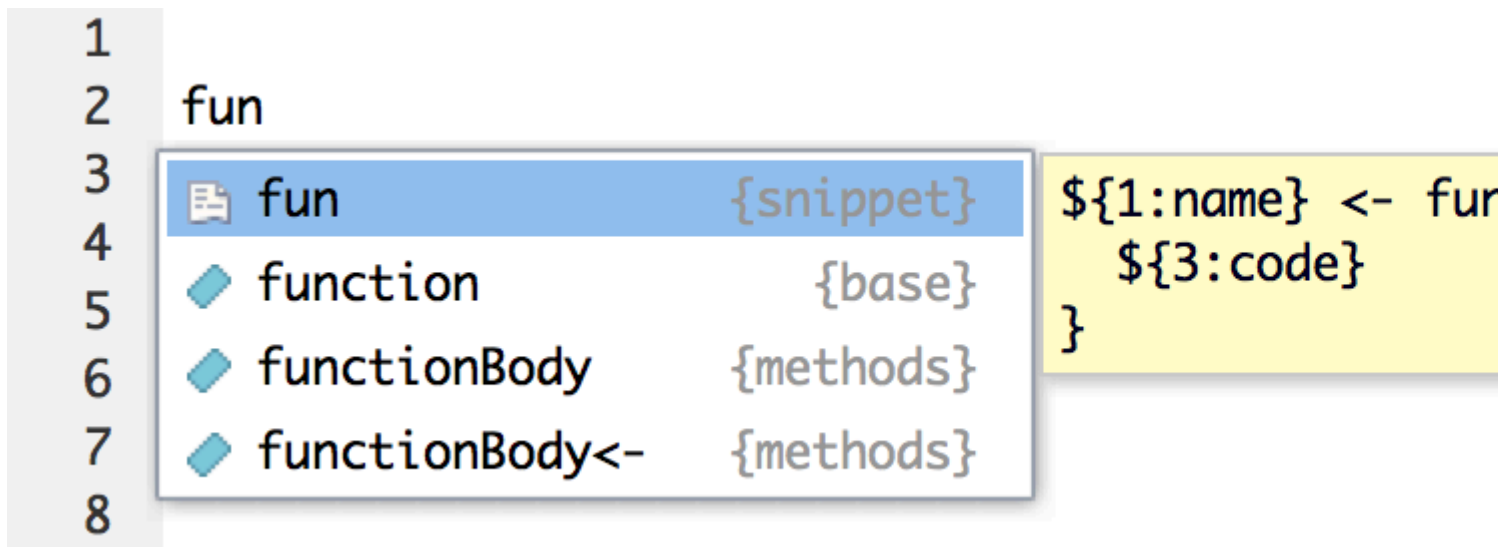
ist äquivalent zu

```
f <- function() { 1 }
f()
[1] 1
```

## RStudio-Codeausschnitte

Dies ist nur ein kleiner Hack für diejenigen, die oft selbst definierte Funktionen verwenden.

Geben Sie "fun" RStudio IDE ein und drücken Sie TAB.



Das Ergebnis wird ein Skelett einer neuen Funktion sein.

```
name <- function(variables) {

}
```

Man kann leicht eine eigene Snippet-Vorlage definieren, dh wie die folgende

```
name <- function(df, x, y) {
 require(tidyverse)
 out <-
 return(out)
}
```

Die Option ist `Edit Snippets` im Menü `Global Options -> Code`.

## Spaltennamen als Argument einer Funktion übergeben

Manchmal möchte man Namen von Spalten aus einem Datenrahmen an eine Funktion übergeben. Sie können als Strings bereitgestellt und in einer Funktion mit `[[` . Schauen wir uns das folgende Beispiel an, das die Grundstatistiken der ausgewählten Variablen in der R-Konsole ausgibt:

```
basic.stats <- function(dset, vars){
 for(i in 1:length(vars)){
 print(vars[i])
 print(summary(dset[[vars[i]]]))
 }
}

basic.stats(iris, c("Sepal.Length", "Petal.Width"))
```

Als Ergebnis der Ausführung des angegebenen Codes werden die Namen der ausgewählten Variablen und ihre grundlegenden Zusammenfassungsstatistiken (Minima, erste Quantile,

Mittelwerte, Mittelwerte, dritte Quantile und Maxima) in der R-Konsole gedruckt. Der Code `dset[[vars[i]]]` wählt das *i*-te Element aus den Argumenten `vars` und wählt eine entsprechende Spalte im deklarierten Eingabedatensatz `dset`. Wenn Sie beispielsweise `iris[["Sepal.Length"]]` `Sepal.Length` Spalte `Sepal.Length` aus dem `iris` Datensatz als Vektor `Sepal.Length`.

Schreibfunktionen in R online lesen: <https://riptutorial.com/de/r/topic/7937/schreibfunktionen-in-r>

# Kapitel 103: Selbstbeobachtung

## Examples

### Funktionen zum Lernen von Variablen

In R möchten Sie oft etwas über ein Objekt oder eine Variable wissen, mit der Sie arbeiten. Dies kann hilfreich sein, wenn Sie den Code einer anderen Person oder sogar Ihren eigenen Code lesen, insbesondere wenn Sie für Sie neue Pakete verwenden.

Angenommen, wir erstellen eine Variable `a` :

```
a <- matrix(1:9, 3, 3)
```

Welcher Datentyp ist das? Sie können es mit herausfinden

```
> class(a)
[1] "matrix"
```

Es ist eine Matrix, also arbeiten Matrixoperationen daran:

```
> a %*% t(a)
 [,1] [,2] [,3]
[1,] 66 78 90
[2,] 78 93 108
[3,] 90 108 126
```

Was sind die Dimensionen `a` ?

```
> dim(a)
[1] 3 3
> nrow(a)
[1] 3
> ncol(a)
[2] 3
```

Andere nützliche Funktionen, die für verschiedene Datentypen funktionieren, sind `head` , `tail` und `str` :

```
> head(a, 1)
 [,1] [,2] [,3]
[1,] 1 4 7
> tail(a, 1)
 [,1] [,2] [,3]
[3,] 3 6 9
> str(a)
int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
```

Dies ist bei großen Objekten (z. B. großen Datensätzen) viel nützlicher. `str` eignet sich auch gut



zum Lernen der Verschachtelung von Listen. Jetzt neu formt `a` in etwa so:

```
a <- c(a)
```

Bleibt die Klasse gleich?

```
> class(a)
[1] "integer"
```

Nein, `a` ist keine Matrix mehr. Ich bekomme keine gute Antwort, wenn ich jetzt nach Abmessungen frage:

```
> dim(a)
NULL
```

Stattdessen kann ich nach der Länge fragen:

```
> length(a)
[1] 9
```

Was ist mit jetzt:

```
> class(a * 1.0)
[1] "numeric"
```

Oft können Sie mit `data.frames` :

```
a <- as.data.frame(a)
names(a) <- c("var1", "var2", "var3")
```

Siehe die Variablennamen:

```
> names(a)
[1] "var1" "var2" "var3"
```

Diese Funktionen können bei der Verwendung von `R` viele Arten helfen.

Selbstbeobachtung online lesen: <https://riptutorial.com/de/r/topic/3565/selbstbeobachtung>

---

# Kapitel 104: Spaltenweiser Betrieb

## Examples

### Summe jeder Spalte

Angenommen, wir müssen die `sum` jeder Spalte in einem Dataset durchführen

```
set.seed(20)
df1 <- data.frame(ID = rep(c("A", "B", "C"), each = 3), V1 = rnorm(9), V2 = rnorm(9))
m1 <- as.matrix(df1[-1])
```

Dafür gibt es viele Möglichkeiten. Bei Verwendung von `base R` wäre `colSums` die beste Option

```
colSums(df1[-1], na.rm = TRUE)
```

Hier haben wir die erste Spalte entfernt, da sie nicht numerisch ist, und die `sum` jeder Spalte angegeben, wobei `na.rm = TRUE` (falls NA im Datensatz vorhanden sind).

Dies funktioniert auch mit `matrix`

```
colSums(m1, na.rm = TRUE)
```

Dies kann in einer Schleife mit `lapply/sapply/vapply`

```
lapply(df1[-1], sum, na.rm = TRUE)
```

Es ist zu beachten, dass die Ausgabe eine `list` . Wenn wir eine `vector` benötigen

```
sapply(df1[-1], sum, na.rm = TRUE)
```

Oder

```
vapply(df1[-1], sum, na.rm = TRUE, numeric(1))
```

Wenn Sie für Matrizen Spalten durchlaufen möchten, verwenden Sie `apply` mit `MARGIN = 1`

```
apply(m1, 2, FUN = sum, na.rm = TRUE)
```

---

Es gibt Möglichkeiten, dies mit Paketen wie `dplyr` oder `data.table` zu tun

```
library(dplyr)
df1 %>%
 summarise_at(vars(matches("^V\\d+")), sum, na.rm = TRUE)
```

Hier übergeben wir einen regulären Ausdruck, um den Spaltennamen zu entsprechen, die wir benötigen, um die `sum` in `summarise_at`. Der reguläre Ausdruck stimmt mit allen Spalten überein, die mit `v` gefolgt von einer oder mehreren Zahlen (`\\d+`).

Eine `data.table` Option ist

```
library(data.table)
setDT(df1)[, lapply(.SD, sum, na.rm = TRUE), .SDcols = 2:ncol(df1)]
```

Wir konvertieren 'data.frame' in 'data.table' (`setDT(df1)`), spezifizieren die Spalten, auf die die Funktion in `.SDcols` angewendet werden `.SDcols`, durchlaufen die Teilmenge von Data.table (`.SD`) und erhalten die `sum`.

---

Wenn Sie eine Gruppe nach Operation verwenden müssen, können Sie dies leicht tun, indem Sie die Gruppe nach Spalte / Spalte angeben

```
df1 %>%
 group_by(ID) %>%
 summarise_at(vars(matches("^V\\d+")), sum, na.rm = TRUE)
```

In Fällen, in denen wir die brauchen `sum` aller Spalten, `summarise_each` kann anstelle verwendet werden `summarise_at`

```
df1 %>%
 group_by(ID) %>%
 summarise_each(funs(sum(., na.rm = TRUE)))
```

Die Option `data.table` lautet

```
setDT(df1)[, lapply(.SD, sum, na.rm = TRUE), by = ID]
```

Spaltenweiser Betrieb online lesen: <https://riptutorial.com/de/r/topic/2212/spaltenweiser-betrieb>

---

# Kapitel 105: Spark-API (SparkR)

## Bemerkungen

Mit dem `SparkR` Paket können Sie mit verteilten Datenrahmen über einem [Spark-Cluster arbeiten](#) . Damit können Sie Vorgänge wie Auswahl, Filterung und Aggregation für sehr große Datensätze durchführen. [SparkR-Übersicht](#) [SparkR-Paketdokumentation](#)

## Examples

### Spark-Kontext einrichten

### Spark-Kontext in R einrichten

Um mit verteilten Sparks-Datenrahmen arbeiten zu können, müssen Sie Ihr R-Programm mit einem vorhandenen Spark-Cluster verbinden.

```
library(SparkR)
sc <- sparkR.init() # connection to Spark context
sqlContext <- sparkRSQL.init(sc) # connection to SQL context
```

[Hier erfahren Sie](#), wie Sie Ihre IDE mit einem Spark-Cluster verbinden.

### Spark-Cluster abrufen

Es gibt ein [Einführungsthema von Apache Spark](#) mit Installationsanweisungen. Grundsätzlich können Sie einen Spark - Cluster lokal über Java (verwenden [siehe Anleitung](#) ) oder verwenden (non-free) Cloud - Anwendungen (zB [Microsoft Azure \[Themenseite\]](#) , [IBM](#) ).

### Daten zwischenspeichern

Was:

Durch das Caching kann die Berechnung in Spark optimiert werden. Die Zwischenspeicherung speichert Daten im Speicher und ist ein Sonderfall von Persistenz. [Hier wird erklärt](#), was passiert, wenn Sie eine RDD in Spark zwischenspeichern.

Warum:

Grundsätzlich speichert das Zwischenspeichern ein Zwischenergebnis - normalerweise nach Transformationen - Ihrer Originaldaten. Wenn Sie also die zwischengespeicherte RDD verwenden, wird auf die bereits umgewandelten Daten aus dem Speicher zugegriffen, ohne die vorherigen Umwandlungen neu zu berechnen.

Wie:

Hier ein Beispiel, wie Sie schnell auf große Daten (*hier 3 GB große csv*) zugreifen können, wenn Sie mehrmals darauf zugreifen:

```
library(SparkR)
next line is needed for direct csv import:
Sys.setenv('SPARKR_SUBMIT_ARGS'='"--packages" "com.databricks:spark-csv_2.10:1.4.0" "sparkr-shell"')
sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)

loading 3 GB big csv file:
train <- read.df(sqlContext, "/train.csv", source = "com.databricks.spark.csv", inferSchema = "true")
cache(train)
system.time(head(train))
output: time elapsed: 125 s. This action invokes the caching at this point.
system.time(head(train))
output: time elapsed: 0.2 s (!!)
```

## Erstellen Sie RDDs (Resilient Distributed Datasets)

### Vom Datenrahmen:

```
mtrdd <- createDataFrame(sqlContext, mtcars)
```

### Von csv:

Für csvs müssen Sie das [csv-Paket](#) zur Umgebung hinzufügen, bevor Sie den Spark-Kontext initiieren:

```
Sys.setenv('SPARKR_SUBMIT_ARGS'='"--packages" "com.databricks:spark-csv_2.10:1.4.0" "sparkr-shell"') # context for csv import read csv ->
sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)
```

Dann können Sie die csv laden, indem Sie auf das Datenschema der Daten in den Spalten schließen:

```
train <- read.df(sqlContext, "/train.csv", header= "true", source = "com.databricks.spark.csv", inferSchema = "true")
```

Oder indem Sie vorher das Datenschema angeben:

```
customSchema <- structType(
 structField("margin", "integer"),
 structField("gross", "integer"),
 structField("name", "string"))

train <- read.df(sqlContext, "/train.csv", header= "true", source = "com.databricks.spark.csv", schema = customSchema)
```

Spark-API (SparkR) online lesen: <https://riptutorial.com/de/r/topic/5349/spark-api--sparkr->

# Kapitel 106: Split-Funktion

## Examples

### Grundlegende Verwendung von Split

`split` können Sie einen Vektor oder einen `data.frame` in Buckets hinsichtlich einer Faktor- / Gruppenvariablen unterteilen. Diese Belüftung in Buckets erfolgt in Form einer Liste, `lapply` `sapply` gruppenweise Berechnungen ( `for` Schleifen oder `lapply` / `sapply` ) durchgeführt werden können.

Das erste Beispiel zeigt die Verwendung von `split` für einen Vektor:

Betrachten Sie den folgenden Buchstabenvektor:

```
testdata <- c("e", "o", "r", "g", "a", "y", "w", "q", "i", "s", "b", "v", "x", "h", "u")
```

Ziel ist es, diese Briefe in `vowels` und `consonants` `vowels` , dh entsprechend dem Buchstabentyp aufzuteilen.

Zuerst erstellen wir einen Gruppierungsvektor:

```
vowels <- c('a','e','i','o','u','y')
letter_type <- ifelse(testdata %in% vowels, "vowels", "consonants")
```

Beachten Sie, dass `letter_type` dieselbe Länge hat wie unsere `testdata` . Jetzt können wir `split` diese Testdaten in den beiden Gruppen, `vowels` und `consonants` :

```
split(testdata, letter_type)
#$consonants
#[1] "r" "g" "w" "q" "s" "b" "v" "x" "h"

#$vowels
#[1] "e" "o" "a" "y" "i" "u"
```

Das Ergebnis ist also eine Liste, deren Namen von unserem Gruppierungsvektor / Faktor `letter_type` .

`split` hat auch eine Methode, um mit `data.frames` umzugehen.

Betrachten Sie zum Beispiel `iris` :

```
data(iris)
```

Mit `split` kann man eine Liste mit einem `data.frame` pro Irisart erstellen (variabel: `Species`):

```
> liris <- split(iris, iris$Species)
> names(liris)
[1] "setosa" "versicolor" "virginica"
```

```
> head(liris$setosa)
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
```

(enthält nur Daten für die Setosa-Gruppe).

Eine Beispieloperation wäre die Berechnung der Korrelationsmatrix pro Irisart; man würde dann `lapply` :

```
> (lcor <- lapply(liris, FUN=function(df) cor(df[,1:4])))

$setosa
 Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length 1.0000000 0.7425467 0.2671758 0.2780984
Sepal.Width 0.7425467 1.0000000 0.1777000 0.2327520
Petal.Length 0.2671758 0.1777000 1.0000000 0.3316300
Petal.Width 0.2780984 0.2327520 0.3316300 1.0000000

$versicolor
 Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length 1.0000000 0.5259107 0.7540490 0.5464611
Sepal.Width 0.5259107 1.0000000 0.5605221 0.6639987
Petal.Length 0.7540490 0.5605221 1.0000000 0.7866681
Petal.Width 0.5464611 0.6639987 0.7866681 1.0000000

$virginica
 Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length 1.0000000 0.4572278 0.8642247 0.2811077
Sepal.Width 0.4572278 1.0000000 0.4010446 0.5377280
Petal.Length 0.8642247 0.4010446 1.0000000 0.3221082
Petal.Width 0.2811077 0.5377280 0.3221082 1.0000000
```

Dann können wir pro Gruppe das beste Paar von korrelierten Variablen abrufen:  
(Korrelationsmatrix wird umgeformt / geschmolzen, Diagonale wird herausgefiltert und die Auswahl des besten Datensatzes wird durchgeführt)

```
> library(reshape)
> (topcor <- lapply(lcor, FUN=function(cormat){
 correlations <- melt(cormat,variable_name="correlation");
 filtered <- correlations[correlations$X1 != correlations$X2,];
 filtered[which.max(filtered$correlation),]
}))

$setosa
 X1 X2 correlation
2 Sepal.Width Sepal.Length 0.7425467

$versicolor
 X1 X2 correlation
12 Petal.Width Petal.Length 0.7866681

$virginica
```



|   | X1           | X2           | correlation |
|---|--------------|--------------|-------------|
| 3 | Petal.Length | Sepal.Length | 0.8642247   |

Beachten Sie, dass eine Berechnung auf einer solchen gruppenweisen Ebene durchgeführt wird. Möglicherweise interessieren Sie sich für das Stapeln der Ergebnisse. Dies kann folgendermaßen durchgeführt werden:

```
> (result <- do.call("rbind", topcor))
```

|            | X1           | X2           | correlation |
|------------|--------------|--------------|-------------|
| setosa     | Sepal.Width  | Sepal.Length | 0.7425467   |
| versicolor | Petal.Width  | Petal.Length | 0.7866681   |
| virginica  | Petal.Length | Sepal.Length | 0.8642247   |

## Verwenden von Split im Split-Apply-Combine-Paradigma

Eine beliebte Form der Datenanalyse ist das [Aufteilen-Anwenden-Kombinieren](#), bei dem Sie Ihre Daten in Gruppen aufteilen, für jede Gruppe eine Art Verarbeitung durchführen und dann die Ergebnisse kombinieren.

Betrachten wir eine Datenanalyse, bei der wir die zwei Autos mit den besten Meilen pro Gallone (mpg) für jede Zylinderanzahl (cyl) im eingebauten mtcars-Datensatz erhalten möchten. Zuerst teilen wir den `mtcars` nach der Zylinderanzahl auf:

```
(spl <- split(mtcars, mtcars$cyl))
```

```
$`4`
```

|              | mpg  | cyl | disp  | hp | drat | wt    | qsec  | vs | am | gear | carb |
|--------------|------|-----|-------|----|------|-------|-------|----|----|------|------|
| # Datsun 710 | 22.8 | 4   | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1  | 1  | 4    | 1    |
| # Merc 240D  | 24.4 | 4   | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1  | 0  | 4    | 2    |
| # Merc 230   | 22.8 | 4   | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1  | 0  | 4    | 2    |
| # Fiat 128   | 32.4 | 4   | 78.7  | 66 | 4.08 | 2.200 | 19.47 | 1  | 1  | 4    | 1    |
| # ...        |      |     |       |    |      |       |       |    |    |      |      |

```
$`6`
```

|                  | mpg  | cyl | disp  | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|------------------|------|-----|-------|-----|------|-------|-------|----|----|------|------|
| # Mazda RX4      | 21.0 | 6   | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0  | 1  | 4    | 4    |
| # Mazda RX4 Wag  | 21.0 | 6   | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0  | 1  | 4    | 4    |
| # Hornet 4 Drive | 21.4 | 6   | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1  | 0  | 3    | 1    |
| # Valiant        | 18.1 | 6   | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1  | 0  | 3    | 1    |
| # ...            |      |     |       |     |      |       |       |    |    |      |      |

```
$`8`
```

|                     | mpg  | cyl | disp  | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|---------------------|------|-----|-------|-----|------|-------|-------|----|----|------|------|
| # Hornet Sportabout | 18.7 | 8   | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0  | 0  | 3    | 2    |
| # Duster 360        | 14.3 | 8   | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0  | 0  | 3    | 4    |
| # Merc 450SE        | 16.4 | 8   | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0  | 0  | 3    | 3    |
| # Merc 450SL        | 17.3 | 8   | 275.8 | 180 | 3.07 | 3.730 | 17.60 | 0  | 0  | 3    | 3    |
| # ...               |      |     |       |     |      |       |       |    |    |      |      |

Dies hat eine Liste von Datenrahmen zurückgegeben, einen für jede Zylinderanzahl. Wie durch die Ausgabe angegeben ist, könnten wir den entsprechenden Datenrahmen mit `spl$`4``, `spl$`6`` und `spl$`8`` (einige könnten es optisch ansprechender zu verwenden `spl$"4"` oder `spl[["4"]]` statt.

Jetzt können wir `lapply` um durch diese Liste zu `lapply`, indem wir unsere Funktion verwenden, die die Autos mit den besten 2 mpg-Werten aus jedem Listenelement extrahiert:

```
(best2 <- lapply(spl, function(x) tail(x[order(x$mpg)], 2)))
$`4`
mpg cyl disp hp drat wt qsec vs am gear carb
Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
#
$`6`
mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
#
$`8`
mpg cyl disp hp drat wt qsec vs am gear carb
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Pontiac Firebird 19.2 8 400 175 3.08 3.845 17.05 0 0 3 2
```

Schließlich können wir alles mit `rbind`. Wir wollen `rbind(best2[["4"]], best2[["6"]], best2[["8"]])` nennen `rbind(best2[["4"]], best2[["6"]], best2[["8"]])`, aber das wäre langweilig, wenn wir eine riesige Liste hätten. Als Ergebnis verwenden wir:

```
do.call(rbind, best2)
mpg cyl disp hp drat wt qsec vs am gear carb
4.Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
4.Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
6.Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4
6.Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1
8.Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2
8.Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2
```

Dies gibt das Ergebnis von `rbind` (Argument 1, eine Funktion) zurück, wobei alle Elemente von `best2` (Argument 2, eine Liste) als Argumente übergeben werden.

Mit einfachen Analysen wie dieser kann es kompakter sein (und möglicherweise viel weniger lesbar!), Wenn Sie die gesamte Split-Apply-Kombination in einer einzigen Codezeile ausführen:

```
do.call(rbind, lapply(split(mtcars, mtcars$cyl), function(x) tail(x[order(x$mpg)], 2)))
```

Es ist auch erwähnenswert, dass die `lapply(split(x, f), FUN)` alternativ mit der Funktion `?by` `lapply(split(x, f), FUN)` gerahmt werden kann:

```
by(mtcars, mtcars$cyl, function(x) tail(x[order(x$mpg)], 2))
do.call(rbind, by(mtcars, mtcars$cyl, function(x) tail(x[order(x$mpg)], 2)))
```

Split-Funktion online lesen: <https://riptutorial.com/de/r/topic/1073/split-funktion>

# Kapitel 107: sqldf

## Examples

### Grundlegende Anwendungsbeispiele

`sqldf()` aus dem Paket `sqldf` ermöglicht die Verwendung von SQLite-Abfragen zum Auswählen und Bearbeiten von Daten in R. SQL-Abfragen werden als Zeichenfolgen eingegeben.

So wählen Sie beispielsweise die ersten 10 Zeilen des Datensatzes "Diamonds" aus dem Paket `ggplot2` aus:

```
data("diamonds")
head(diamonds)
```

```
A tibble: 6 x 10
 carat cut color clarity depth table price x y z
<dbl> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43
2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31
3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31
4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63
5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75
6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48
```

```
require(sqldf)
sqldf("select * from diamonds limit 10")
```

```
 carat cut color clarity depth table price x y z
1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43
2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31
3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31
4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63
5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75
6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48
7 0.24 Very Good I VVS1 62.3 57 336 3.95 3.98 2.47
8 0.26 Very Good H SI1 61.9 55 337 4.07 4.11 2.53
9 0.22 Fair E VS2 65.1 61 337 3.87 3.78 2.49
10 0.23 Very Good H VS1 59.4 61 338 4.00 4.05 2.39
```

So wählen Sie die ersten 10 Zeilen für die Farbe "E" aus:

```
sqldf("select * from diamonds where color = 'E' limit 10")
```

```
 carat cut color clarity depth table price x y z
1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43
2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31
3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31
4 0.22 Fair E VS2 65.1 61 337 3.87 3.78 2.49
5 0.20 Premium E SI2 60.2 62 345 3.79 3.75 2.27
```

|    |      |           |   |     |      |    |     |      |      |      |
|----|------|-----------|---|-----|------|----|-----|------|------|------|
| 6  | 0.32 | Premium   | E | I1  | 60.9 | 58 | 345 | 4.38 | 4.42 | 2.68 |
| 7  | 0.23 | Very Good | E | VS2 | 63.8 | 55 | 352 | 3.85 | 3.92 | 2.48 |
| 8  | 0.23 | Very Good | E | VS1 | 60.7 | 59 | 402 | 3.97 | 4.01 | 2.42 |
| 9  | 0.23 | Very Good | E | VS1 | 59.5 | 58 | 402 | 4.01 | 4.06 | 2.40 |
| 10 | 0.23 | Good      | E | VS1 | 64.1 | 59 | 402 | 3.83 | 3.85 | 2.46 |

Beachten Sie im obigen Beispiel, dass in Anführungszeichen stehende Zeichenfolgen in der SQL-Abfrage mit " in Anführungszeichen gesetzt werden, wenn die gesamte Abfrage mit "" in Anführungszeichen gesetzt wird (dies funktioniert auch umgekehrt).

Angenommen, wir möchten eine neue Spalte hinzufügen, um die Anzahl der Diamanten im Premium-Schliff über 1 Karat zu zählen:

```
sqldf("select count(*) from diamonds where carat > 1 and color = 'E'")
```

|   | count(*) |
|---|----------|
| 1 | 1892     |

Ergebnisse der erstellten Werte können auch als neue Spalten zurückgegeben werden:

```
sqldf("select *, count(*) as cnt_big_E_colored_stones from diamonds where carat > 1 and color = 'E' group by clarity")
```

|                          | carat | cut       | color | clarity | depth | table | price | x    | y    | z    |
|--------------------------|-------|-----------|-------|---------|-------|-------|-------|------|------|------|
| cnt_big_E_colored_stones |       |           |       |         |       |       |       |      |      |      |
| 1                        | 1.30  | Fair      | E     | I1      | 66.5  | 58    | 2571  | 6.79 | 6.75 | 4.50 |
| 65                       |       |           |       |         |       |       |       |      |      |      |
| 2                        | 1.28  | Ideal     | E     | IF      | 60.7  | 57    | 18700 | 7.09 | 6.99 | 4.27 |
| 28                       |       |           |       |         |       |       |       |      |      |      |
| 3                        | 2.02  | Very Good | E     | SI1     | 59.8  | 59    | 18731 | 8.11 | 8.20 | 4.88 |
| 499                      |       |           |       |         |       |       |       |      |      |      |
| 4                        | 2.03  | Premium   | E     | SI2     | 61.5  | 59    | 18477 | 8.24 | 8.16 | 5.04 |
| 666                      |       |           |       |         |       |       |       |      |      |      |
| 5                        | 1.51  | Ideal     | E     | VS1     | 61.5  | 57    | 18729 | 7.34 | 7.40 | 4.53 |
| 158                      |       |           |       |         |       |       |       |      |      |      |
| 6                        | 1.72  | Very Good | E     | VS2     | 63.4  | 56    | 18557 | 7.65 | 7.55 | 4.82 |
| 318                      |       |           |       |         |       |       |       |      |      |      |
| 7                        | 1.20  | Ideal     | E     | VVS1    | 61.8  | 56    | 16256 | 6.78 | 6.87 | 4.22 |
| 52                       |       |           |       |         |       |       |       |      |      |      |
| 8                        | 1.55  | Ideal     | E     | VVS2    | 62.5  | 55    | 18188 | 7.38 | 7.40 | 4.62 |
| 106                      |       |           |       |         |       |       |       |      |      |      |

Wenn man interessiert wäre, wie hoch ist der **maximale** price des Diamanten **entsprechend** dem cut :

```
sqldf("select cut, max(price) from diamonds group by cut")
```

|   | cut       | max(price) |
|---|-----------|------------|
| 1 | Fair      | 18574      |
| 2 | Good      | 18788      |
| 3 | Ideal     | 18806      |
| 4 | Premium   | 18823      |
| 5 | Very Good | 18818      |

sqldf online lesen: <https://riptutorial.com/de/r/topic/2100/sqldf>

---

# Kapitel 108: Standardisieren Sie Analysen, indem Sie eigenständige R-Skripts schreiben

## Einführung

Wenn Sie eine R-Analyse routinemäßig auf viele separate Datendateien anwenden oder anderen Personen eine wiederholbare Analysemethode zur Verfügung stellen möchten, ist ein ausführbares R-Skript eine benutzerfreundliche Methode. Anstelle von Ihnen oder Ihrem Benutzer mit anrufen `R` und das Skript innerhalb R über ausführen `source(.)` Oder einen Funktionsaufruf, kann Ihr Benutzer einfach rufen Sie das Skript selbst , als ob es ein Programm.

## Bemerkungen

Verwenden Sie zur Darstellung der Standard-Eingabe- / Ausgabekanäle die Funktionsdatei `file("stdin")` (Eingabe vom Terminal oder einem anderen Programm über Pipe), `stdout()` (Standardausgabe) und `stderr()` (Standardfehler). Beachten Sie, dass die Funktion `stdin()` zwar nicht verwendet werden kann, wenn ein fertiges Skript an R übergeben wird, da die nächsten Zeilen dieses Skripts anstelle der Benutzereingabe gelesen werden.

## Examples

Die grundlegende Struktur des Standalone-R-Programms und wie es aufgerufen wird

---

## Das erste eigenständige R-Skript

Eigenständige R-Skripts werden nicht vom Programm `R` (`R.exe` unter Windows) ausgeführt, sondern von einem Programm namens `Rscript` (`Rscript.exe`), das standardmäßig in Ihrer `R` Installation enthalten ist.

Um auf diese Tatsache hinzuweisen, beginnen eigenständige R-Skripts mit einer speziellen Zeile namens **Shebang-Zeile** , die den folgenden Inhalt enthält: `#!/usr/bin/env Rscript` . Unter Windows ist eine zusätzliche Maßnahme erforderlich, die später näher erläutert wird.

Das folgende einfache eigenständige R-Skript speichert ein Histogramm unter dem Dateinamen "hist.png" aus den Zahlen, die es als Eingabe erhält:

```
#!/usr/bin/env Rscript

User message (\n = end the line)
cat("Input numbers, separated by space:\n")
Read user input as one string (n=1 -> Read only one line)
input <- readLines(file('stdin'), n=1)
```

```
Split the string at each space (\\s == any space)
input <- strsplit(input, "\\s")[[1]]
convert the obtained vector of strings to numbers
input <- as.numeric(input)

Open the output picture file
png("hist.png",width=400, height=300)
Draw the histogram
hist(input)
Close the output file
dev.off()
```

Sie können mehrere Schlüsselemente eines eigenständigen R-Skripts sehen. In der ersten Zeile sehen Sie die Shebang-Linie. Anschließend wird `cat("....\n")` verwendet, um dem Benutzer eine Nachricht zu drucken. Verwenden Sie die `file("stdin")` immer dann, wenn Sie "User input on console" als `file("stdin")` angeben möchten. Dies kann anstelle eines Dateinamens in mehreren `read.table ( scan , read.table , read.csv , ... )` verwendet werden. Nachdem die Benutzereingabe von Zeichenfolgen in Zahlen konvertiert wurde, beginnt die Zeichnung. Dort ist zu sehen, dass Plotbefehle, die in eine Datei geschrieben werden sollen, in zwei Befehlen eingeschlossen sein müssen. Dies sind in diesem Fall `png(.)` Und `dev.off()` . Die erste Funktion hängt vom gewünschten Ausgabeformat ab (andere gängige Optionen sind `jpeg(.)` Und `pdf(.)` ). Die zweite Funktion `dev.off()` ist immer erforderlich. Es schreibt die Zeichnung in die Datei und beendet den Zeichenvorgang.

## Eigenständiges R-Skript vorbereiten

### Linux / Mac

Die Datei des Standalone-Skripts muss zuerst ausführbar gemacht werden. Dies kann geschehen, indem Sie mit der rechten Maustaste auf die Datei klicken, "Eigenschaften" im sich öffnenden Menü öffnen und das Kontrollkästchen "Ausführbare Datei" auf der Registerkarte "Berechtigungen" aktivieren. Alternativ den Befehl

```
chmod +x PATH/TO/SCRIPT/SCRIPTNAME.R
```

kann in einem Terminal angerufen werden.

### Windows

Für jedes eigenständige Skript muss eine Batchdatei mit folgendem Inhalt geschrieben werden:

```
"C:\Program Files\R-XXXXXXX\bin\Rscript.exe" "%~dp0\XXXXXXX.R" %*
```

Eine Batchdatei ist eine normale Textdatei, die jedoch die Erweiterung `*.bat` Ausnahme der Erweiterung `*.txt` . Erstellen Sie es mit einem Texteditor wie `notepad` (nicht `Word` ) oder ähnlichem und setzen Sie den Dateinamen in die Anführungszeichen ( `"FILENAME.bat"` ) im Sicherungsdialogfeld. Um eine vorhandene Batchdatei zu bearbeiten, klicken Sie mit der rechten

Maustaste darauf und wählen Sie "Bearbeiten".

Sie müssen den oben gezeigten Code überall dort anpassen, wo `xxx...` geschrieben ist:

- Fügen Sie den richtigen Ordner ein, in dem sich Ihre R-Installation befindet
- Geben Sie den korrekten Namen Ihres Skripts ein und platzieren Sie es in demselben Verzeichnis wie diese Batchdatei.

Erklärung der Elemente im Code: Der erste Teil `"C:\...\Rscript.exe"` teilt Windows mit, wo sich das Programm `Rscript.exe` . Der zweite Teil `"%~dp0\XXX.R"` weist `Rscript` an, das von Ihnen geschriebene R-Skript auszuführen, das sich in demselben Ordner befindet wie die Batchdatei (`%~dp0` steht für den Batchdateiordner). Schließlich leitet `%*` alle Befehlszeilenargumente, die Sie der Batchdatei angeben, an das R-Skript weiter.

Wenn Sie auf die Batchdatei doppelklicken, wird das R-Skript ausgeführt. Wenn Sie Dateien in die Batchdatei ziehen, werden dem R-Skript die entsprechenden Dateinamen als Befehlszeilenargumente zugewiesen.

## Verwenden von `littler` zum Ausführen von R-Skripten

`littler` (ausgesprochen *kleines r*) ( [cran](#) ) bietet neben anderen Funktionen zwei Möglichkeiten, R-Skripts mit dem Befehl `r littler` (wenn man mit Linux oder MacOS arbeitet) von der Kommandozeile aus auszuführen.

## Kleiner installieren

Von R:

```
install.packages("littler")
```

Der Pfad von `r` wird im Terminal ausgegeben, wie

```
You could link to the 'r' binary installed in
'/home/*USER*/R/x86_64-pc-linux-gnu-library/3.4/littler/bin/r'
from '/usr/local/bin' in order to use 'r' for scripting.
```

Um `r` über die Befehlszeile des Systems aufrufen zu können, ist ein Symlink erforderlich:

```
ln -s /home/*USER*/R/x86_64-pc-linux-gnu-library/3.4/littler/bin/r /usr/local/bin/r
```

## Verwenden von `apt-get` (Debian, Ubuntu):

```
sudo apt-get install littler
```

## Verwenden von `littler` mit Standard-`.r`-Skripten



Mit `r` von `littler` können Standalone-R-Skripts ohne Änderungen am Skript ausgeführt werden.  
Beispielskript:

```
User message (\n = end the line)
cat("Input numbers, separated by space:\n")
Read user input as one string (n=1 -> Read only one line)
input <- readLines(file('stdin'), n=1)
Split the string at each space (\\s == any space)
input <- strsplit(input, "\\s")[[1]]
convert the obtained vector of strings to numbers
input <- as.numeric(input)

Open the output picture file
png("hist.png",width=400, height=300)
Draw the histogram
hist(input)
Close the output file
dev.off()
```

Beachten Sie, dass sich kein Shebang an der Spitze der Skripte befindet. Wenn es als `hist.r`, kann es direkt vom Systembefehl `hist.r` werden:

```
r hist.r
```

## Littler für *Shebanged*- Skripte verwenden

Es ist auch möglich, ausführbare R-Skripte mit Hilfe von `shebang` mit `littler` zu erstellen

```
#!/usr/bin/env r
```

am Anfang des Skripts. Das entsprechende R-Skript muss mit `chmod +X /path/to/script.r` ausführbar `chmod +X /path/to/script.r` und ist direkt vom Systemendgerät aus aufrufbar.

Standardisieren Sie Analysen, indem Sie eigenständige R-Skripts schreiben online lesen:  
<https://riptutorial.com/de/r/topic/9937/standardisieren-sie-analysen--indem-sie-eigenstandige-r-skripts-schreiben>

---

# Kapitel 109: String-Manipulation mit dem Stringi-Paket

## Bemerkungen

Um das Paket zu installieren, führen Sie einfach Folgendes aus:

```
install.packages("stringi")
```

um es zu laden:

```
require("stringi")
```

## Examples

### Zählmuster im String

Mit festem Muster

```
stri_count_fixed("babab", "b")
[1] 3
stri_count_fixed("babab", "ba")
[1] 2
stri_count_fixed("babab", "bab")
[1] 1
```

*Nativ:*

```
length(gregexpr("b", "babab")[[1]])
[1] 3
length(gregexpr("ba", "babab")[[1]])
[1] 2
length(gregexpr("bab", "babab")[[1]])
[1] 1
```

Funktion wird über String und Muster vektorisiert:

```
stri_count_fixed("babab", c("b", "ba"))
[1] 3 2
stri_count_fixed(c("babab", "bbb", "bca", "abc"), c("b", "ba"))
[1] 3 0 1 0
```

*Eine Basis-R-Lösung :*

```
sapply(c("b", "ba"), function(x) length(gregexpr(x, "babab")[[1]]))
b ba
```

## Mit Regex

Erstes Beispiel - finden Sie `a` und ein beliebiges Zeichen nach

Zweites Beispiel - finde `a` und eine beliebige Ziffer danach

```
stri_count_regex("a1 b2 a3 b4 aa", "a.")
[1] 3
stri_count_regex("a1 b2 a3 b4 aa", "a\\d")
[1] 2
```

## Zeichenketten duplizieren

```
stri_dup("abc", 3)
[1] "abcabcabc"
```

*Eine Basis-R-Lösung, die dasselbe tut, würde folgendermaßen aussehen:*

```
paste0(rep("abc", 3), collapse = "")
[1] "abcabcabc"
```

## Vektoren einfügen

```
stri_paste(LETTERS, "-", 1:13)
[1] "A-1" "B-2" "C-3" "D-4" "E-5" "F-6" "G-7" "H-8" "I-9" "J-10" "K-11" "L-12" "M-13"
[14] "N-1" "O-2" "P-3" "Q-4" "R-5" "S-6" "T-7" "U-8" "V-9" "W-10" "X-11" "Y-12" "Z-13"
```

*Nativ könnten wir das in R machen über:*

```
> paste(LETTERS, 1:13, sep="-")
[1] "A-1" "B-2" "C-3" "D-4" "E-5" "F-6" "G-7" "H-8" "I-9" "J-10" "K-11" "L-12" "M-13"
[14] "N-1" "O-2" "P-3" "Q-4" "R-5" "S-6" "T-7" "U-8" "V-9" "W-10" "X-11" "Y-12" "Z-13"
```

## Text nach einem festen Muster aufteilen

Textvektor nach einem Muster aufteilen:

```
stri_split_fixed(c("To be or not to be.", "This is very short sentence."), " ")
[[1]]
[1] "To" "be" "or" "not" "to" "be."
#
[[2]]
[1] "This" "is" "very" "short" "sentence."
```

Teilen Sie einen Text mit vielen Mustern:

```
stri_split_fixed("Apples, oranges and pineaplles.", c(" ", ",", ".", "s"))
[[1]]
[1] "Apples," "oranges" "and" "pineaplles."
#
[[2]]
[1] "Apples" " oranges and pineaplles."
#
[[3]]
[1] "Apple" ", orange" " and pineaplle" "."
```

String-Manipulation mit dem Stringi-Paket online lesen:

<https://riptutorial.com/de/r/topic/1670/string-manipulation-mit-dem-stringi-paket>

# Kapitel 110: Strsplit-Funktion

## Syntax

- `strsplit (`
- `x`
- `Teilt`
- `fixed = FALSE`
- `perl = FALSE`
- `useBytes = FALSE)`

## Examples

### Einführung

`strsplit` ist eine nützliche Funktion, um einen Vektor in eine Liste eines Zeichenmusters aufzuteilen. Bei typischen R-Werkzeugen kann die gesamte Liste wieder in einen Datenrahmen eingefügt werden. Ein Teil der Liste kann in einer Grafikübung verwendet werden.

In der `strsplit` : Unterbrechen Sie einen Zeichenvektor entlang eines Trennzeichens:

```
temp <- c("this,that,other", "hat,scarf,food", "woman,man,child")
get a list split by commas
myList <- strsplit(temp, split=",")
print myList
myList
[[1]]
[1] "this" "that" "other"

[[2]]
[1] "hat" "scarf" "food"

[[3]]
[1] "woman" "man" "child"
```

Wie oben angedeutet, ist das Split-Argument nicht auf Zeichen beschränkt, sondern kann einem Muster folgen, das von einem regulären Ausdruck vorgegeben wird. Zum Beispiel ist `temp2` mit `temp` identisch, mit der Ausnahme, dass die Trennzeichen für jedes Element geändert wurden. Wir können die Tatsache ausnutzen, dass das Split-Argument reguläre Ausdrücke akzeptiert, um die Unregelmäßigkeit im Vektor zu verringern.

```
temp2 <- c("this, that, other", "hat,scarf ,food", "woman; man ; child")
myList2 <- strsplit(temp2, split=" ?[,;] ?")
myList2
[[1]]
[1] "this" "that" "other"

[[2]]
[1] "hat" "scarf" "food"

[[3]]
[1] "woman" "man" "child"
```

```
[[3]]
[1] "woman" "man" "child"
```

*Anmerkungen :*

1. Die Aufgliederung der Syntax für reguläre Ausdrücke ist für dieses Beispiel nicht möglich.
2. Manchmal kann das Abgleichen von regulären Ausdrücken einen Prozess verlangsamen.  
Wie bei vielen R-Funktionen, die die Verwendung regulärer Ausdrücke zulassen, steht das feste Argument zur Verfügung, um R mit den geteilten Zeichen wörtlich übereinzustimmen.

Strsplit-Funktion online lesen: <https://riptutorial.com/de/r/topic/2762/strsplit-funktion>

# Kapitel 111: Subsetting

## Einführung

In Anbetracht eines R-Objekts benötigen wir möglicherweise eine separate Analyse für einen oder mehrere Teile der darin enthaltenen Daten. Das `subsetting` dieser Teile der Daten von einem bestimmten Objekt wird als `subsetting` .

## Bemerkungen

### Fehlende Werte:

Fehlende Werte ( `NA` ), die bei der Subsetting mit `[` `return NA` seit einem `NA` Index verwendet werden wählt ein unbekanntes Element aus und gibt `NA` im entsprechenden Element zurück.

Die „Standard“ Typ von `NA` ist „logische“ ( `typeof(NA)` ) , was bedeutet , dass, da jede „logische“ Vektor in subsetting verwendet, um die Länge des subsetted Objekt übereinstimmen **recycle** wird. Also ist `x[NA]` äquivalent zu `x[as.logical(NA)]` was äquivalent zu `x[rep_len(as.logical(NA), length(x))]` und folglich einen fehlenden Wert ( `NA` ) zurückgibt. für jedes Element von `x` . Als Beispiel:

```
x <- 1:3
x[NA]
[1] NA NA NA
```

Während der Indizierung mit "numerisch" / "Ganzzahl" wählt `NA` ein einzelnes `NA` Element aus (für jede `NA` im Index):

```
x[as.integer(NA)]
[1] NA

x[c(NA, 1, NA, NA)]
[1] NA 1 NA NA
```

### Subsetting außerhalb der Grenzen:

Der Operator `[` mit einem übergebenen Argument ermöglicht Indizes, die `> length(x)` und gibt `NA` für Atomvektoren oder `NULL` für generische Vektoren zurück. Im Gegensatz dazu wird bei `[[` und wenn `[` mehr Argumente übergeben werden (dh Objekte außerhalb der Grenzen mit `length(dim(x)) > 2` ) ein Fehler zurückgegeben:

```
(1:3)[10]
[1] NA
(1:3)[[10]]
Error in (1:3)[[10]] : subscript out of bounds
as.matrix(1:3)[10]
[1] NA
```

```
as.matrix(1:3)[, 10]
Error in as.matrix(1:3)[, 10] : subscript out of bounds
list(1, 2, 3)[10]
[[1]]
NULL
list(1, 2, 3)[[10]]
Error in list(1, 2, 3)[[10]] : subscript out of bounds
```

Das Verhalten ist beim Subsetting mit "Zeichen" -Vektoren gleich, die auch im Attribut "names" des Objekts nicht übereinstimmen:

```
c(a = 1, b = 2)["c"]
<NA>
NA
list(a = 1, b = 2)["c"]
<NA>
NULL
```

## Hilfethemen:

Siehe `?Extract` für weitere Informationen.

# Examples

## Atomare Vektoren

Atomare Vektoren (die Listen und Ausdrücke, die ebenfalls Vektoren sind, ausschließen) werden mit dem Operator `[]` subset:

```
create an example vector
v1 <- c("a", "b", "c", "d")

select the third element
v1[3]
[1] "c"
```

Der Operator `[]` kann auch einen Vektor als Argument verwenden. So wählen Sie beispielsweise das erste und das dritte Element aus:

```
v1 <- c("a", "b", "c", "d")

v1[c(1, 3)]
[1] "a" "c"
```

Manchmal müssen wir einen bestimmten Wert aus dem Vektor auslassen. Dies kann durch ein negatives Vorzeichen ( `-` ) vor dem Index dieses Wertes erreicht werden. Wenn Sie beispielsweise den ersten Wert in `v1` nicht `v1[-1]` möchten, verwenden Sie `v1[-1]` . Dieser kann auf einfache Weise auf mehrere Werte erweitert werden. Zum Beispiel `v1[-c(1, 3)]` .

```
> v1[-1]
[1] "b" "c" "d"
```



```
> v1[-c(1,3)]
[1] "b" "d"
```

In manchen Fällen möchten wir insbesondere, wenn die Länge des Vektors groß ist, den Index eines bestimmten Werts wissen, falls vorhanden:

```
> v1=="c"
[1] FALSE FALSE TRUE FALSE
> which(v1=="c")
[1] 3
```

Wenn das Atomvektornamen (a hat `names` Attribut), kann es ein Zeichen Vektor Namen Subset werden:

```
v <- 1:3
names(v) <- c("one", "two", "three")

v
one two three
1 2 3

v["two"]
two
2
```

Der Operator `[]` kann auch zum Indizieren atomarer Vektoren verwendet werden, mit dem Unterschied, dass er einen Indizierungsvektor mit einer Länge von 1 akzeptiert und vorhandene Namen entfernt:

```
v[[c(1, 2)]]
Error in v[[c(1, 2)]] :
attempt to select more than one element in vectorIndex

v[["two"]]
[1] 2
```

Vektoren können auch mit einem logischen Vektor untergeordnet werden. Im Gegensatz zum Subsetting mit numerischen und Zeichenvektoren muss der für die Teilmenge verwendete logische Vektor der Länge des Vektors entsprechen, dessen Elemente extrahiert werden. Wenn also ein logischer Vektor `y` für die Teilmenge `x`, dh `x[y]`, falls `length(y) < length(x)` dann wird `y` wiederverwendet, um der `length(x)` :

```
v[c(TRUE, FALSE, TRUE)]
one three
1 3

v[c(FALSE, TRUE)] # recycled to 'c(FALSE, TRUE, FALSE)'
two
2

v[TRUE] # recycled to 'c(TRUE, TRUE, TRUE)'
one two three
1 2 3
```

```
v[FALSE] # handy to discard elements but save the vector's type and basic structure
named integer(0)
```

## Listen

Eine Liste kann mit `[]` :

```
l1 <- list(c(1, 2, 3), 'two' = c("a", "b", "c"), list(10, 20))
l1
[[1]]
[1] 1 2 3
##
$two
[1] "a" "b" "c"
##
[[3]]
[[3]][[1]]
[1] 10
##
[[3]][[2]]
[1] 20

l1[1]
[[1]]
[1] 1 2 3

l1['two']
$two
[1] "a" "b" "c"

l1[[2]]
[1] "a" "b" "c"

l1[['two']]
[1] "a" "b" "c"
```

Beachten Sie, dass das Ergebnis von `l1[2]` immer noch eine Liste ist, da der Operator `[]` Elemente einer Liste auswählt und eine kleinere Liste zurückgibt. Der Operator `[[` extrahiert Listenelemente und gibt ein Objekt vom Typ des Listenelements zurück.

Elemente können nach Nummer oder einer Zeichenfolge des Namens (falls vorhanden) indiziert werden. Mit `[]` können mehrere Elemente ausgewählt werden, indem ein Vektor mit Zahlen oder Namensfolgen übergeben wird. Indizierung mit einem Vektor der `length > 1` in `[]` und `[[` gibt eine "Liste" mit den angegebenen Elementen bzw. einer rekursiven Teilmenge (falls verfügbar) zurück :

```
l1[c(3, 1)]
[[1]]
[[1]][[1]]
[1] 10
##
[[1]][[2]]
[1] 20
##
##
[[2]]
```

```
[1] 1 2 3
```

Verglichen mit:

```
l1[[c(3, 1)]]
[1] 10
```

was äquivalent ist zu:

```
l1[[3]][[1]]
[1] 10
```

Mit dem Operator `$` können Sie Listenelemente nur nach Namen auswählen. Anders als bei `[]` und `[[` sind keine Anführungszeichen erforderlich. Als Infix-Operator kann `$` nur einen einzigen Namen annehmen:

```
l1$two
[1] "a" "b" "c"
```

Außerdem erlaubt der Operator `$` standardmäßig die teilweise Übereinstimmung:

```
l1$t
[1] "a" "b" "c"
```

Im Gegensatz zu `[[` muss angegeben werden, ob ein partieller Abgleich zulässig ist:

```
l1[["t"]]
NULL
l1[["t", exact = FALSE]]
[1] "a" "b" "c"
```

Einstellungsoptionen `options(warnPartialMatchDollar = TRUE)` , eine Warnung wird ausgegeben, wenn ein teilweiser Abgleich mit `$` :

```
l1$t
[1] "a" "b" "c"
Warning message:
In l1$t : partial match of 't' to 'two'
```

## Matrizen

Für jede Dimension eines Objekts `[]` Operator `[]` ein Argument. Vektoren haben eine Dimension und ein Argument. Matrizen und Datenrahmen haben zwei Dimensionen und nehmen zwei Argumente an, gegeben als `[i, j]` wobei `i` die Zeile und `j` die Spalte ist. Die Indizierung beginnt bei 1.

```
a sample matrix
mat <- matrix(1:6, nrow = 2, dimnames = list(c("row1", "row2"), c("col1", "col2", "col3")))
```

```
mat
col1 col2 col3
row1 1 3 5
row2 2 4 6
```

`mat[i, j]` ist das Element in der `i` - ten Reihe, `j` - ten Spalte der Matrix - `mat` . Ein `i` Wert von `2` und ein `j` Wert von `1` geben beispielsweise die Anzahl in der zweiten Zeile und der ersten Spalte der Matrix an. Das Auslassen von `i` oder `j` gibt alle Werte in dieser Dimension zurück.

```
mat[, 3]
row1 row2
5 6

mat[1,]
col1 col2 col3
1 3 5
```

Wenn die Matrix Zeilen- oder Spaltennamen hat (nicht erforderlich), können diese zum Subsetting verwendet werden:

```
mat[, 'col1']
row1 row2
1 2
```

Standardmäßig wird das Ergebnis einer Teilmenge wenn möglich vereinfacht. Wenn die Teilmenge wie in den obigen Beispielen nur eine Dimension hat, ist das Ergebnis ein eindimensionaler Vektor und keine zweidimensionale Matrix. Dieser Standardwert kann mit dem Argument `drop = FALSE` auf `[ : ]` überschrieben werden.

```
This selects the first row as a vector
class(mat[1,])
[1] "integer"

Whereas this selects the first row as a 1x3 matrix:
class(mat[1, , drop = F])
[1] "matrix"
```

Dimensionen können natürlich nicht gelöscht werden, wenn die Auswahl selbst zwei Dimensionen hat:

```
mat[1:2, 2:3] ## A 2x2 matrix
col2 col3
row1 3 5
row2 4 6
```

## Auswahl einzelner Matrixeinträge anhand ihrer Positionen

Es ist auch möglich, eine `Nx2`-Matrix zu verwenden, um `N` einzelne Elemente aus einer Matrix auszuwählen (wie ein Koordinatensystem funktioniert). Wenn Sie in einem Vektor die Einträge einer Matrix in `(1st row, 1st column)`, `(1st row, 3rd column)`, `(2nd row, 3rd column)`, `(2nd row, 1st column)`

extrahieren möchten, können Sie dies tun Sie können dies leicht tun, indem Sie eine Indexmatrix mit diesen Koordinaten erstellen und diese zum Untersetzen der Matrix verwenden:

```
mat
col1 col2 col3
row1 1 3 5
row2 2 4 6

ind = rbind(c(1, 1), c(1, 3), c(2, 3), c(2, 1))
ind
[,1] [,2]
[1,] 1 1
[2,] 1 3
[3,] 2 3
[4,] 2 1

mat[ind]
[1] 1 5 6 2
```

In dem obigen Beispiel bezieht sich die 1. Spalte der `ind` Matrix auf Zeilen in `mat` , die 2. Spalte von `ind` auf Spalten in `mat` .

## Datenrahmen

**Das Unterteilen eines Datenrahmens in einen kleineren Datenrahmen** kann auf dieselbe Weise erfolgen wie das Unterteilen einer Liste.

```
> df3 <- data.frame(x = 1:3, y = c("a", "b", "c"), stringsAsFactors = FALSE)

> df3
x y
1 1 a
2 2 b
3 3 c

> df3[1] # Subset a variable by number
x
1 1
2 2
3 3

> df3["x"] # Subset a variable by name
x
1 1
2 2
3 3

> is.data.frame(df3[1])
TRUE

> is.list(df3[1])
TRUE
```

**Das Unterteilen eines Datenrahmens in einen Spaltenvektor** kann mit doppelten Klammern `[[ ]]` oder dem Dollarzeichenoperator `$` durchgeführt werden.

```

> df3[[2]] # Subset a variable by number using [[]]
[1] "a" "b" "c"

> df3[["y"]] # Subset a variable by name using [[]]
[1] "a" "b" "c"

> df3$x # Subset a variable by name using $
[1] 1 2 3

> typeof(df3$x)
"integer"

> is.vector(df3$x)
TRUE

```

**Das Unterteilen von Daten als zweidimensionale Matrix** kann unter Verwendung von `i` und `j` Ausdrücken erreicht werden.

```

> df3[1, 2] # Subset row and column by number
[1] "a"

> df3[1, "y"] # Subset row by number and column by name
[1] "a"

> df3[2,] # Subset entire row by number
x y
2 2 b

> df3[, 1] # Subset all first variables
[1] 1 2 3

> df3[, 1, drop = FALSE]
x
1 1
2 2
3 3

```

Anmerkung: Das Subsetting by `j` (column) vereinfacht sich allein auf den eigenen Typ der Variablen, aber das Subsetting by `i` allein gibt einen `data.frame`, da die verschiedenen Variablen unterschiedliche Typen und Klassen haben können. Wenn Sie den `drop` Parameter auf `FALSE` bleibt der Datenrahmen erhalten.

```

> is.vector(df3[, 2])
TRUE

> is.data.frame(df3[2,])
TRUE

> is.data.frame(df3[, 2, drop = FALSE])
TRUE

```

## Andere Objekte

Die Operatoren `[]` und `[[` sind generische Funktionen. Dies bedeutet, dass jedes *Objekt* in R (insbesondere `isTRUE(is.object(x))` - es hat ein explizites "class" -Attribut) sein eigenes Verhalten

haben kann, wenn es subsetting wird. dh hat eigene *Methoden* für `[]` und `/` oder `[]`.

Dies ist zum Beispiel der Fall bei "data.frame" (`is.object(iris)`) Objekten, bei `[]` `.data.frame` Methoden `[]` `.data.frame` und `[]` `.data.frame` definiert sind und beide "Matrix" -ähnlich sind und "list" -ähnliches Subsetting. Mit einem Fehler zu zwingen, wenn ein „data.frame“ subsetting, sehen wir, dass, tatsächlich, eine Funktion `[]` `.data.frame` genannt wurde, als wir früher -just- `[]`.

```
iris[invalidArgument,]
Error in `[.data.frame`(iris, invalidArgument,) :
object 'invalidArgument' not found
```

Ohne weitere Details zum aktuellen Thema, ein Beispiel `[]` Methode:

```
x = structure(1:5, class = "myClass")
x[c(3, 2, 4)]
[1] 3 2 4
' [.myClass' = function(x, i) cat(sprintf("We'd expect '%s[%s]'" to be returned but this a
custom `[` method and should have a `?[.myClass` help page for its behaviour\n",
deparse(substitute(x)), deparse(substitute(i))))

x[c(3, 2, 4)]
We'd expect 'x[c(3, 2, 4)]' to be returned but this a custom `[` method and should have a
`?[.myClass` help page for its behaviour
NULL
```

Das Methoden-Dispatching von `[]` kann überwunden werden, indem der äquivalente, nicht-generische `.subset` (und `.subset2` für `[]`) verwendet wird. Dies ist besonders nützlich und effizient, wenn Sie unsere eigenen "Klassen" programmieren und `unclass(x)` (wie `unclass(x)`) vermeiden möchten, wenn Sie auf unseren "Klassen" effizient berechnen (Methodenversand und Kopieren von Objekten vermeiden):

```
.subset(x, c(3, 2, 4))
[1] 3 2 4
```

## Vektor-Indizierung

In diesem Beispiel verwenden wir den Vektor:

```
> x <- 11:20
> x
[1] 11 12 13 14 15 16 17 18 19 20
```

R-Vektoren sind 1-indiziert, so dass zum Beispiel `x[1]` 11. Wir können auch einen Untervektor von `x` extrahieren, indem Sie einen Indexvektor an den Klammeroperator übergeben:

```
> x[c(2, 4, 6)]
[1] 12 14 16
```

Wenn wir einen Vektor negativer Indizes übergeben, gibt R einen Untervektor mit den angegebenen Indizes zurück:

```
> x[c(-1,-3)]
[1] 12 14 15 16 17 18 19 20
```

Wir können auch einen booleschen Vektor an den Klammeroperator übergeben. In diesem Fall gibt er einen Untervektor zurück, der den Koordinaten entspricht, bei denen der Indexierungsvektor `TRUE` :

```
> x[c(rep(TRUE,5),rep(FALSE,5))]
[1] 11 12 13 14 15 16
```

Ist der Indexierungsvektor kürzer als die Länge des Arrays, wird er wie folgt wiederholt:

```
> x[c(TRUE,FALSE)]
[1] 11 13 15 17 19
> x[c(TRUE,FALSE,FALSE)]
[1] 11 14 17 20
```

## Elementweise Matrixoperationen

Sei A und B zwei Matrizen gleicher Dimension. Die Operatoren `+`, `-`, `/`, `*`, `^` wenn sie mit Matrizen derselben Dimension verwendet werden, die erforderlichen Operationen für die entsprechenden Elemente der Matrizen aus und geben eine neue Matrix derselben Dimension zurück. Diese Operationen werden normalerweise als elementweise Operationen bezeichnet.

| Operator | A op B   | Bedeutung                                                                   |
|----------|----------|-----------------------------------------------------------------------------|
| +        | A + B    | Zugabe entsprechender Elemente von A und B                                  |
| -        | A - B    | Subtrahiert die Elemente von B von den entsprechenden Elementen von A       |
| /        | A / B    | Dividiert die Elemente von A durch die entsprechenden Elemente von B        |
| *        | A * B    | Multipliziert die Elemente von A mit den entsprechenden Elementen von B     |
| ^        | A ^ (-1) | Beispielsweise ergibt sich eine Matrix, deren Elemente Kehrwerte von A sind |

Verwenden Sie für "echte" Matrixmultiplikationen, wie in der *linearen Algebra* zu sehen, `%*%`. Die Multiplikation von A mit B ist beispielsweise: `A %*% B`. Die `ncol()` bestehen darin, dass `ncol()` von A mit `nrow()` von B `nrow()`

## Einige Funktionen, die mit Matrizen verwendet werden



| Funktion       | Beispiel             | Zweck                                                                                  |
|----------------|----------------------|----------------------------------------------------------------------------------------|
| nrow ()        | nrow (A)             | bestimmt die Anzahl der Reihen von A                                                   |
| ncol ()        | ncol (A)             | bestimmt die Anzahl der Spalten von A                                                  |
| rownames<br>() | Rufnamen (A)         | druckt die Zeilennamen der Matrix A aus                                                |
| Colnames<br>() | Colnames (A)         | gibt die Spaltennamen der Matrix A aus                                                 |
| rowMeans<br>() | rowMeans (A)         | berechnet die Mittelwerte für jede Zeile der Matrix A                                  |
| colMeans<br>() | colMeans (A)         | berechnet die Mittelwerte für jede Spalte der Matrix A                                 |
| upper.tri ()   | obere.Tri (A)        | gibt einen Vektor zurück, dessen Elemente die oberen sind                              |
|                |                      | Dreiecksmatrix der quadratischen Matrix A                                              |
| lower.tri ()   | niedriger.Tri<br>(A) | gibt einen Vektor zurück, dessen Elemente niedriger sind                               |
|                |                      | Dreiecksmatrix der quadratischen Matrix A                                              |
| det ()         | det (A)              | ergibt die Determinante der Matrix A                                                   |
| lösen()        | lösen (A)            | ergibt die Inverse der nicht singulären Matrix A                                       |
| diag ()        | diag (A)             | gibt eine Diagonalmatrix zurück, deren Elemente außerhalb der Diagnose Nullen und sind |
|                |                      | Diagonalen sind die gleichen wie die der quadratischen Matrix A                        |
| t ()           | t (A)                | gibt die Transponierte der Matrix A zurück                                             |
| Eigen ()       | Eigen (A)            | stellt die Eigenwerte und Eigenvektoren der Matrix A wieder her                        |
| is.matrix ()   | is.matrix (A)        | gibt TRUE oder FALSE zurück, je nachdem, ob A eine Matrix ist oder nicht.              |
| as.matrix ()   | as.matrix (x)        | erstellt eine Matrix aus dem Vektor x                                                  |

Subsetting online lesen: <https://riptutorial.com/de/r/topic/1686/subsetting>

---

# Kapitel 112: Text-Mining

## Examples

### Daten auskratzen, um N-Gramm-Word-Clouds zu erstellen

Im folgenden Beispiel wird das `tm` Text-Mining-Paket verwendet, um Textdaten aus dem Web zu kratzen und zu extrahieren, um Wortwolken mit symbolischer Schattierung und Reihenfolge zu erstellen.

```
require(RWeka)
require(tau)
require(tm)
require(tm.plugin.webmining)
require(wordcloud)

Scrape Google Finance -----
googlefinance <- WebCorpus(GoogleFinanceSource("NASDAQ:LFVN"))

Scrape Google News -----
lv.googlenews <- WebCorpus(GoogleNewsSource("LifeVantage"))
p.googlenews <- WebCorpus(GoogleNewsSource("Protandim"))
ts.googlenews <- WebCorpus(GoogleNewsSource("TrueScience"))

Scrape NYTimes -----
lv.nytimes <- WebCorpus(NYTimesSource(query = "LifeVantage", appid = nytimes_appid))
p.nytimes <- WebCorpus(NYTimesSource("Protandim", appid = nytimes_appid))
ts.nytimes <- WebCorpus(NYTimesSource("TrueScience", appid = nytimes_appid))

Scrape Reuters -----
lv.reutersnews <- WebCorpus(ReutersNewsSource("LifeVantage"))
p.reutersnews <- WebCorpus(ReutersNewsSource("Protandim"))
ts.reutersnews <- WebCorpus(ReutersNewsSource("TrueScience"))

Scrape Yahoo! Finance -----
lv.yahoofinance <- WebCorpus(YahooFinanceSource("LFVN"))

Scrape Yahoo! News -----
lv.yahoonews <- WebCorpus(YahooNewsSource("LifeVantage"))
p.yahoonews <- WebCorpus(YahooNewsSource("Protandim"))
ts.yahoonews <- WebCorpus(YahooNewsSource("TrueScience"))

Scrape Yahoo! Inplay -----
lv.yahooinplay <- WebCorpus(YahooInplaySource("LifeVantage"))

Text Mining the Results -----
corpus <- c(googlefinance, lv.googlenews, p.googlenews, ts.googlenews, lv.yahoofinance,
lv.yahoonews, p.yahoonews,
ts.yahoonews, lv.yahooinplay) #lv.nytimes, p.nytimes, ts.nytimes,lv.reutersnews,
p.reutersnews, ts.reutersnews,

inspect(corpus)
wordlist <- c("lfvn", "lifevantage", "protandim", "truescience", "company", "fiscal",
"nasdaq")

ds0.1g <- tm_map(corpus, content_transformer(tolower))
```

```

ds1.1g <- tm_map(ds0.1g, content_transformer(removeWords), wordlist)
ds1.1g <- tm_map(ds1.1g, content_transformer(removeWords), stopwords("english"))
ds2.1g <- tm_map(ds1.1g, stripWhitespace)
ds3.1g <- tm_map(ds2.1g, removePunctuation)
ds4.1g <- tm_map(ds3.1g, stemDocument)

tdm.1g <- TermDocumentMatrix(ds4.1g)
dtm.1g <- DocumentTermMatrix(ds4.1g)

findFreqTerms(tdm.1g, 40)
findFreqTerms(tdm.1g, 60)
findFreqTerms(tdm.1g, 80)
findFreqTerms(tdm.1g, 100)

findAssocs(dtm.1g, "skin", .75)
findAssocs(dtm.1g, "scienc", .5)
findAssocs(dtm.1g, "product", .75)

tdm89.1g <- removeSparseTerms(tdm.1g, 0.89)
tdm9.1g <- removeSparseTerms(tdm.1g, 0.9)
tdm91.1g <- removeSparseTerms(tdm.1g, 0.91)
tdm92.1g <- removeSparseTerms(tdm.1g, 0.92)

tdm2.1g <- tdm92.1g

Creates a Boolean matrix (counts # docs w/terms, not raw # terms)
tdm3.1g <- inspect(tdm2.1g)
tdm3.1g[tdm3.1g>=1] <- 1

Transform into a term-term adjacency matrix
termMatrix.1gram <- tdm3.1g %*% t(tdm3.1g)

inspect terms numbered 5 to 10
termMatrix.1gram[5:10,5:10]
termMatrix.1gram[1:10,1:10]

Create a WordCloud to Visualize the Text Data -----
notsparse <- tdm2.1g
m = as.matrix(notsparse)
v = sort(rowSums(m),decreasing=TRUE)
d = data.frame(word = names(v),freq=v)

Create the word cloud
pal = brewer.pal(9,"BuPu")
wordcloud(words = d$word,
 freq = d$freq,
 scale = c(3,.8),
 random.order = F,
 colors = pal)

```



Beachten Sie die Verwendung von `random.order` und einer sequenziellen Palette von `RColorBrewer`, die es dem Programmierer ermöglicht, weitere Informationen in der Cloud zu erfassen, indem er der Reihenfolge und Farbe der Begriffe Bedeutung zuweist.

Oben ist der 1-Gramm-Fall.

Wir können einen großen Sprung zu N-Gramm-Wortwolken machen, und wir werden sehen, wie wir fast jede Text-Mining-Analyse flexibel genug machen können, um mit N-Gramm durch Transformation unseres TDM umzugehen.

Die anfängliche Schwierigkeit, die Sie mit N-Gramm in R haben, ist, dass `tm`, das beliebteste Paket für Text-Mining, die Tokenisierung von Bi-Gramm oder N-Gramm nicht grundsätzlich unterstützt. Tokenisierung ist der Prozess der Darstellung eines Wortes, eines Wortteils oder einer Gruppe von Wörtern (oder Symbolen) als ein einzelnes Datenelement, das als Token bezeichnet wird.

Glücklicherweise gibt es einige Hacks, mit denen wir `tm` mit einem aktualisierten Tokenizer weiter verwenden können. Es gibt mehr als einen Weg, dies zu erreichen. Wir können unseren eigenen einfachen Tokenizer mit der Funktion `textcnt()` von `tau` schreiben:

```
tokenize_ngrams <- function(x, n=3)
 return(rownames(as.data.frame(unclass(textcnt(x, method="string", n=n)))))
```

oder wir können den Tokenizer von `RWeka` innerhalb von `tm` aufrufen:

```
BigramTokenize
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
```

Von diesem Punkt aus können Sie wie im 1-Gramm-Fall vorgehen:



# Kapitel 113: Überlebensanalyse

## Examples

### Random Forest Survival-Analyse mit randomForestSRC

So wie der [Random-Forest](#)-Algorithmus auf Regressions- und Klassifizierungsaufgaben angewendet werden kann, kann er auch auf die Überlebensanalyse erweitert werden.

Im folgenden Beispiel wird ein Überlebensmodell `randomForestSRC` und für die Vorhersage, Bewertung und Leistungsanalyse mit dem Paket `randomForestSRC` [von CRAN verwendet](#).

```
require(randomForestSRC)

set.seed(130948) #Other seeds give similar comparative results
x1 <- runif(1000)
y <- rnorm(1000, mean = x1, sd = .3)
data <- data.frame(x1 = x1, y = y)
head(data)
```

|   | x1        | y          |
|---|-----------|------------|
| 1 | 0.9604353 | 1.3549648  |
| 2 | 0.3771234 | 0.2961592  |
| 3 | 0.7844242 | 0.6942191  |
| 4 | 0.9860443 | 1.5348900  |
| 5 | 0.1942237 | 0.4629535  |
| 6 | 0.7442532 | -0.0672639 |

```
(modRFSRC <- rfsrc(y ~ x1, data = data, ntree=500, nodesize = 5))
```

```
 Sample size: 1000
 Number of trees: 500
 Minimum terminal node size: 5
 Average no. of terminal nodes: 208.258
No. of variables tried at each split: 1
 Total no. of variables: 1
 Analysis: RF-R
 Family: regr
 Splitting rule: mse
 % variance explained: 32.08
 Error rate: 0.11
```

```
x1new <- runif(10000)
ynew <- rnorm(10000, mean = x1new, sd = .3)
newdata <- data.frame(x1 = x1new, y = ynew)

survival.results <- predict(modRFSRC, newdata = newdata)
survival.results
```

```
Sample size of test (predict) data: 10000
```

```
Number of grow trees: 500
Average no. of grow terminal nodes: 208.258
Total no. of grow variables: 1
 Analysis: RF-R
 Family: regr
% variance explained: 34.97
Test set error rate: 0.11
```

## Einführung - Grundlegende Anpassung und Darstellung von parametrischen Überlebensmodellen mit dem Überlebenspaket

`survival` ist das am häufigsten verwendete Paket für die Überlebensanalyse in R. Mit dem integrierten `lung` Dataset können Sie mit der Überlebensanalyse beginnen, indem Sie ein Regressionsmodell mit der `survreg()` Funktion `survreg()`, eine Kurve mit `survfit()` und das geplottete `survfit()` Überlebenskurven durch Aufrufen der `predict` für dieses Paket mit neuen Daten.

In dem folgenden Beispiel stellen wir zwei vorhergesagte Kurven dar und variieren das `sex` zwischen den zwei neuen Datensätzen, um deren Wirkung zu veranschaulichen:

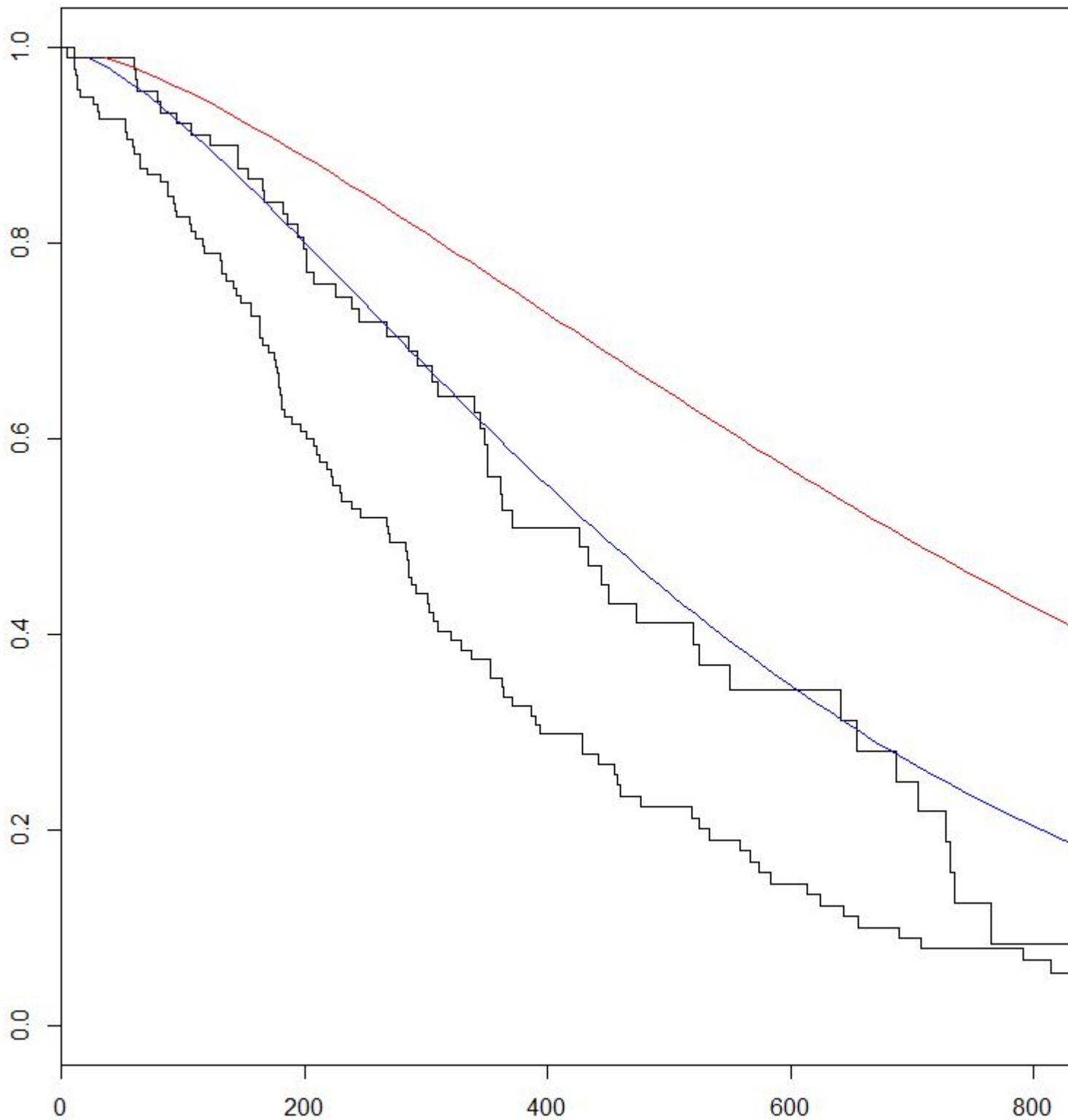
```
require(survival)
s <- with(lung, Surv(time, status))

sWei <- survreg(s ~ as.factor(sex)+age+ph.ecog+wt.loss+ph.karno, dist='weibull', data=lung)

fitKM <- survfit(s ~ sex, data=lung)
plot(fitKM)

lines(predict(sWei, newdata = list(sex = 1,
 age = 1,
 ph.ecog = 1,
 ph.karno = 90,
 wt.loss = 2),
 type = "quantile",
 p = seq(.01, .99, by = .01)),
 seq(.99, .01, by = -.01),
 col = "blue")

lines(predict(sWei, newdata = list(sex = 2,
 age = 1,
 ph.ecog = 1,
 ph.karno = 90,
 wt.loss = 2),
 type = "quantile",
 p = seq(.01, .99, by = .01)),
 seq(.99, .01, by = -.01),
 col = "red")
```



**Kaplan Meier schätzt die Überlebenskurven und Risikotabellen mit Überlebenden**

Grundplot

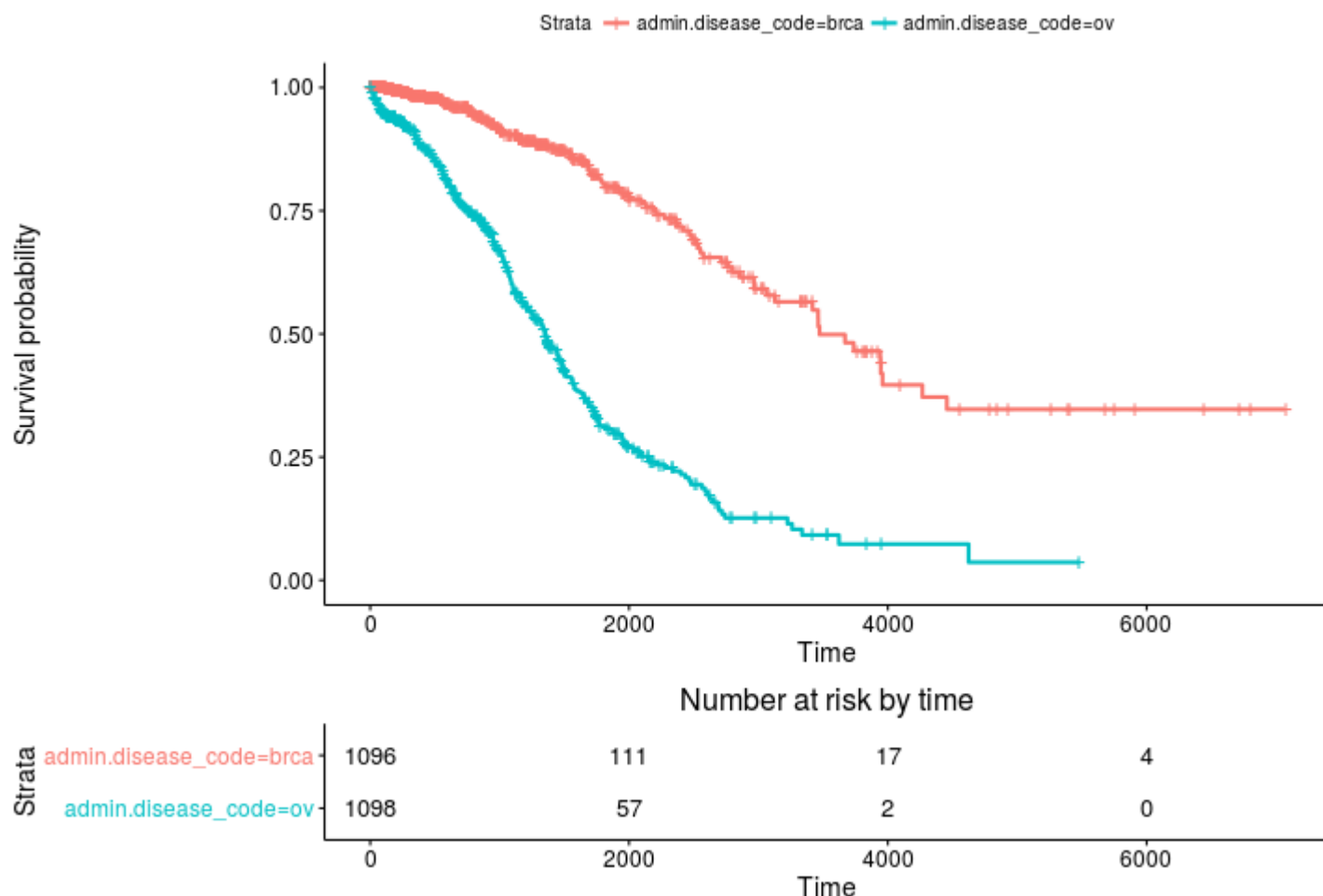
```
install.packages('survminer')
source("https://bioconductor.org/biocLite.R")
```



```

biocLite("RTCGA.clinical") # data for examples
library(RTCGA.clinical)
survivalTCGA(BRCA.clinical, OV.clinical,
 extract.cols = "admin.disease_code") -> BRCAOV.survInfo
library(survival)
fit <- survfit(Surv(times, patient.vital_status) ~ admin.disease_code,
 data = BRCAOV.survInfo)
library(survminer)
ggsurvplot(fit, risk.table = TRUE)

```

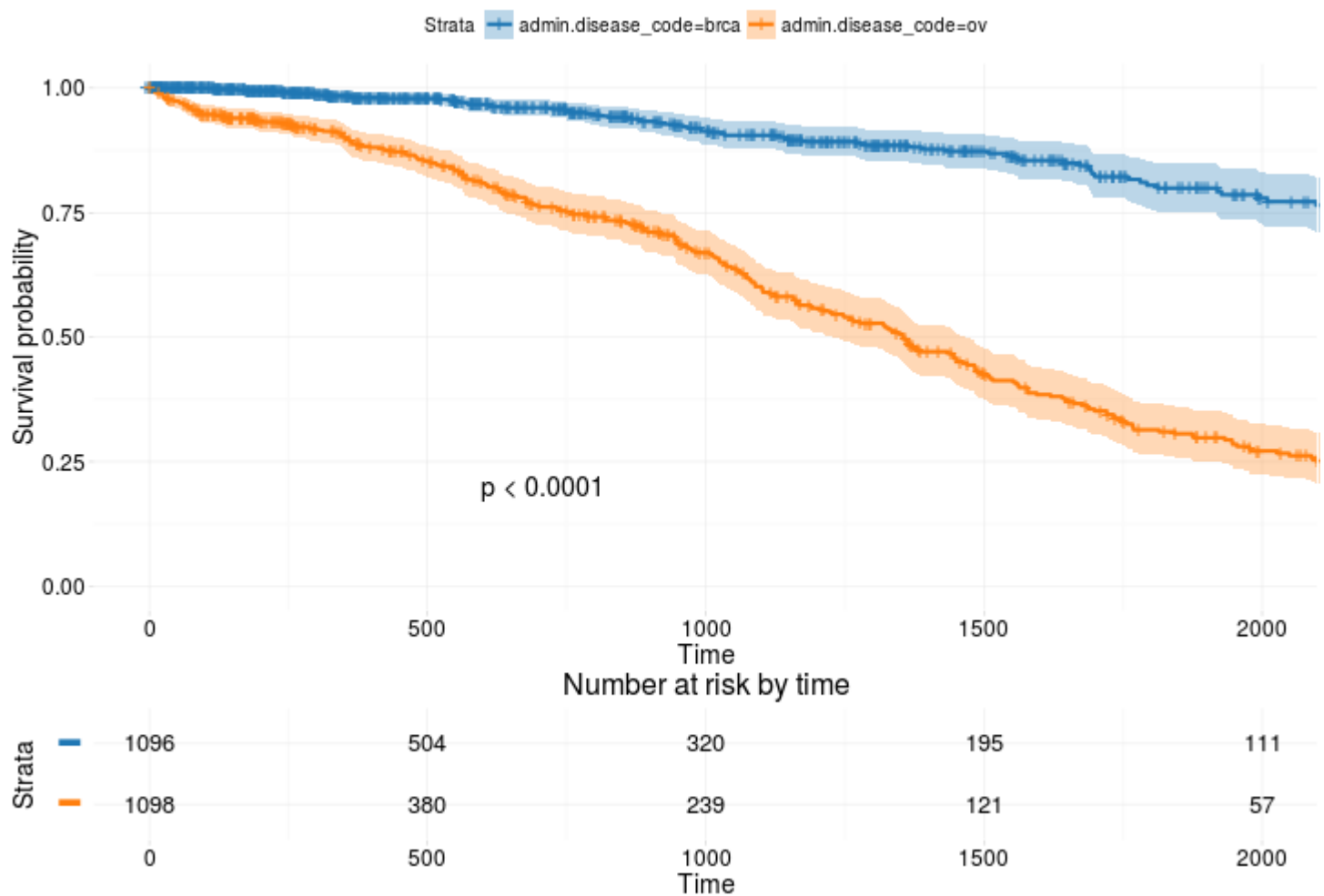


## Fortgeschrittener

```

ggsurvplot(
 fit, # survfit object with calculated statistics.
 risk.table = TRUE, # show risk table.
 pval = TRUE, # show p-value of log-rank test.
 conf.int = TRUE, # show confidence intervals for
 # point estimates of survival curves.
 xlim = c(0,2000), # present narrower X axis, but not affect
 # survival estimates.
 break.time.by = 500, # break X axis in time intervals by 500.
 ggtheme = theme_RTCGA(), # customize plot and risk table with a theme.
 risk.table.y.text.col = T, # colour risk table text annotations.
 risk.table.y.text = FALSE # show bars instead of names in text annotations
 # in legend of risk table
)

```



Beyogen auf

<http://r-addict.com/2016/05/23/Informative-Survival-Plots.html>

Überlebensanalyse online lesen: <https://riptutorial.com/de/r/topic/3788/uberlebensanalyse>

# Kapitel 114: Umfang der Variablen

## Bemerkungen

Die häufigste Fallgrube mit Spielraum entsteht in der Parallelisierung. Alle Variablen und Funktionen müssen in eine neue Umgebung übergeben werden, die in jedem Thread ausgeführt wird.

## Examples

### Umgebungen und Funktionen

Innerhalb einer Funktion deklarierte Variablen existieren nur innerhalb dieser Funktion.

```
x <- 1

foo <- function(x) {
 y <- 3
 z <- x + y
 return(z)
}

y
```

Fehler: Objekt 'y' nicht gefunden

In eine Funktion übergebene und dann neu zugewiesene Variablen werden überschrieben, *jedoch nur innerhalb der Funktion*.

```
foo <- function(x) {
 x <- 2
 y <- 3
 z <- x + y
 return(z)
}

foo(1)
x
```

5

1

Variablen, die in einer höheren Umgebung als eine Funktion zugewiesen wurden, sind in dieser Funktion vorhanden, ohne dass sie übergeben werden.

```
foo <- function() {
 y <- 3
 z <- x + y
```

```
 return(z)
 }

foo()
```

4

## Unterfunktionen

Funktionen, die innerhalb einer Funktion aufgerufen werden (dh Unterfunktionen), müssen innerhalb dieser Funktion definiert werden, um auf Variablen zuzugreifen, die in der lokalen Umgebung definiert sind, ohne weitergegeben zu werden.

Das schlägt fehl:

```
bar <- function() {
 z <- x + y
 return(z)
}

foo <- function() {
 y <- 3
 z <- bar()
 return(z)
}

foo()
```

Fehler in bar (): Objekt 'y' nicht gefunden

Das funktioniert:

```
foo <- function() {

 bar <- function() {
 z <- x + y
 return(z)
 }

 y <- 3
 z <- bar()
 return(z)
}

foo()
```

4

## Globale Zuordnung

Variablen können mit `<<-` aus jeder Umgebung global zugewiesen werden. `bar()` kann jetzt auf `y` zugreifen.

```
bar <- function() {
 z <- x + y
 return(z)
}

foo <- function() {
 y <- 3
 z <- bar()
 return(z)
}

foo()
```

4

Von der globalen Zuweisung wird dringend abgeraten. Die Verwendung einer Wrapper-Funktion oder das explizite Aufrufen von Variablen aus einer anderen lokalen Umgebung wird stark bevorzugt.

## Explizite Zuweisung von Umgebungen und Variablen

Umgebungen in R können explizit aufgerufen und benannt werden. Variablen können explizit zugewiesen werden und von oder zu diesen Umgebungen aufrufen.

Eine häufig erstellte Umgebung ist eine, die `package:base` oder eine Teilumgebung innerhalb von `package:base` einschließt.

```
e1 <- new.env(parent = baseenv())
e2 <- new.env(parent = e1)
```

Variablen können explizit zugewiesen werden und von oder zu diesen Umgebungen aufrufen.

```
assign("a", 3, envir = e1)
get("a", envir = e1)
get("a", envir = e2)
```

3

3

Da `e2` von `e1` erbt, ist `a` 3 sowohl in `e1` und `e2`. Durch die Zuweisung von `a` innerhalb von `e2` wird der Wert von `a` in `e1` nicht geändert.

```
assign("a", 2, envir = e2)
get("a", envir = e2)
get("a", envir = e1)
```

3

2

## Funktion beenden

Die Funktion `on.exit()` ist praktisch für das Bereinigen von Variablen, wenn globale Variablen zugewiesen werden müssen.

Einige Parameter, insbesondere die für die Grafik, können nur global eingestellt werden. Diese kleine Funktion ist bei der Erstellung spezialisierter Diagramme üblich.

```
new_plot <- function(...) {

 old_pars <- par(mar = c(5,4,4,2) + .1, mfrow = c(1,1))
 on.exit(par(old_pars))
 plot(...)
}
```

## Pakete und Maskierung

Funktionen und Objekte in verschiedenen Paketen können denselben Namen haben. Das später geladene Paket 'maskiert' das frühere Paket und eine Warnmeldung wird gedruckt. Beim Aufruf der Funktion über den Namen wird die Funktion des zuletzt geladenen Pakets ausgeführt. Auf die frühere Funktion kann explizit zugegriffen werden.

```
library(plyr)
library(dplyr)
```

Paket anhängen: 'dplyr'

Die folgenden Objekte werden von 'package: plyr' maskiert:

ordnen, zählen, desc, failwith, id, mutieren, umbenennen, zusammenfassen,  
zusammenfassen

Die folgenden Objekte werden von 'package: stats' maskiert:

filter, lag

Die folgenden Objekte werden von 'package: base' maskiert:

schneiden, setdiff, setequal, union

Beim Schreiben von Code empfiehlt es sich immer, Funktionen explizit mit `package::function()` aufzurufen, um dieses Problem zu vermeiden.

**Umfang der Variablen online lesen:** <https://riptutorial.com/de/r/topic/3138/umfang-der-variablen>

# Kapitel 115: Umformen mit Tidyr

## Einführung

tidyr hat zwei Tools zur Daten Umformung: `gather` (breit lang) und `spread` (lang breit).

Weitere [Informationen finden](#) Sie unter [Umformen von Daten](#) .

## Examples

### Umformung vom langen in das breite Format mit Spread ()

```
library(tidyr)

example data
set.seed(123)
df <- data.frame(
 name = rep(c("firstName", "secondName"), each=4),
 numbers = rep(1:4, 2),
 value = rnorm(8)
)
df
name numbers value
1 firstName 1 -0.56047565
2 firstName 2 -0.23017749
3 firstName 3 1.55870831
4 firstName 4 0.07050839
5 secondName 1 0.12928774
6 secondName 2 1.71506499
7 secondName 3 0.46091621
8 secondName 4 -1.26506123
```

Wir können die "Zahlen" -Spalte in separate Spalten "spreizen":

```
spread(data = df,
 key = numbers,
 value = value)
name 1 2 3 4
1 firstName -0.5604756 -0.2301775 1.5587083 0.07050839
2 secondName 0.1292877 1.7150650 0.4609162 -1.26506123
```

Oder spreizen Sie die Spalte "Name" in separate Spalten:

```
spread(data = df,
 key = name,
 value = value)
numbers firstName secondName
1 1 -0.56047565 0.1292877
2 2 -0.23017749 1.7150650
3 3 1.55870831 0.4609162
4 4 0.07050839 -1.2650612
```

## Umformen von Breit zu Langformat mit Gather ()

```
library(tidyr)

example data
df <- read.table(text =" numbers firstName secondName
1 1 1.5862639 0.4087477
2 2 0.1499581 0.9963923
3 3 0.4117353 0.3740009
4 4 -0.4926862 0.4437916", header = T)
df
numbers firstName secondName
1 1 1.5862639 0.4087477
2 2 0.1499581 0.9963923
3 3 0.4117353 0.3740009
4 4 -0.4926862 0.4437916
```

Wir können die Spalten zusammenfassen, indem wir "Zahlen" als Schlüsselspalte verwenden:

```
gather(data = df,
 key = numbers,
 value = myValue)
numbers numbers myValue
1 1 firstName 1.5862639
2 2 firstName 0.1499581
3 3 firstName 0.4117353
4 4 firstName -0.4926862
5 1 secondName 0.4087477
6 2 secondName 0.9963923
7 3 secondName 0.3740009
8 4 secondName 0.4437916
```

Umformen mit Tidyr online lesen: <https://riptutorial.com/de/r/topic/9195/umformen-mit-tidyr>



# Kapitel 116: Variablen

## Examples

### Variablen, Datenstrukturen und grundlegende Operationen

In R werden Datenobjekte mit Hilfe benannter Datenstrukturen bearbeitet. Die Namen der Objekte können als "Variablen" bezeichnet werden, obwohl dieser Begriff in der offiziellen R-Dokumentation keine spezielle Bedeutung hat. Bei R-Namen wird zwischen *Groß-* und *Kleinschreibung unterschieden*. Sie können alphanumerische Zeichen ( *az* , *Az* , *0-9* ), den Punkt / Punkt ( *.* ) Und den Unterstrich ( *\_* ) enthalten. Um Namen für die Datenstrukturen zu erstellen, müssen wir die folgenden Regeln beachten:

- Namen, die mit einer Ziffer oder einem Unterstrich beginnen (z. B. *1a* ), oder Namen, die gültige numerische Ausdrücke sind (z. B. *.11* ), oder Namen mit Bindestrichen ('-') oder Leerzeichen können nur verwendet werden, wenn sie in Anführungszeichen gesetzt sind: *`1a`* und *`.11`* . Die Namen werden mit Backticks gedruckt:

```
list('.11' ="a")
#$`.11`
#[1] "a"
```

- Alle anderen Kombinationen von alphanumerischen Zeichen, Punkten und Unterstrichen können frei verwendet werden, wobei die Referenz mit oder ohne Backticks auf dasselbe Objekt verweist.
- Namen , die mit beginnen *.* werden als Systemnamen betrachtet und sind mit der *ls()* - Funktion nicht immer sichtbar.

Die Anzahl der Zeichen in einem Variablennamen ist nicht beschränkt.

Beispiele für gültige Objektnamen sind: *foobar* , *foo.bar* , *foo\_bar* , *.foobar*

In R werden Variablen Werte mit dem Infix-Zuweisungsoperator *<-* zugewiesen. Der Operator *=* kann auch zum Zuweisen von Werten zu Variablen verwendet werden. Die richtige Verwendung ist jedoch die Zuordnung von Werten zu Parameternamen in Funktionsaufrufen. Beachten Sie, dass das Auslassen von Leerzeichen um Operatoren zu Verwirrung für Benutzer führen kann. Der Ausdruck *a<-1* wird als Zuordnung (analysiert *a <- 1* ) und nicht als logischen Vergleich ( *a < -1* ).

```
> foo <- 42
> fooEquals = 43
```

*foo* erhält also den Wert *42* . *foo* in der Konsole *foo* *fooEquals* wird *42* ausgegeben, während Sie *fooEquals* wird *43* ausgegeben.

```
> foo
[1] 42
```

```
> fooEquals
[1] 43
```

Der folgende Befehl weist der Variablen `x` einen Wert zu und gibt den Wert gleichzeitig aus:

```
> (x <- 5)
[1] 5
actually two function calls: first one to `<-`; second one to the `()`-function
> is.function(`()`)
[1] TRUE # Often used in R help page examples for its side-effect of printing.
```

Zuweisungen an Variablen können auch mit `->` .

```
> 5 -> x
> x
[1] 5
>
```

## Arten von Datenstrukturen

Es gibt keine skalaren Datentypen in R. Vektoren der Länge Eins verhalten sich wie Skalare.

- **Vektoren:** Atomare Vektoren müssen eine Folge von Objekten derselben Klasse sein: eine Folge von Zahlen oder eine Folge von logischen Zeichen oder eine Folge von Zeichen. `v <- c(2, 3, 7, 10)`, `v2 <- c("a", "b", "c")` sind beide Vektoren.
- **Matrizen:** Eine Matrix aus Zahlen, logischen Zeichen oder Zeichen. `a <- matrix(data = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), nrow = 4, ncol = 3, byrow = F)`. Wie Vektoren muss die Matrix aus Elementen derselben Klasse bestehen. Um Elemente aus Matrixzeilen und -spalten zu extrahieren, müssen Sie Folgendes angeben: `a[1,2]` gibt `[1] 5`, dh das Element in der ersten Zeile, zweiten Spalte.
- **Listen:** Verkettung verschiedener Elemente `mylist <- list(course = 'stat', date = '04/07/2009', num_isc = 7, num_cons = 6, num_mat = as.character(c(45020, 45679, 46789, 43126, 42345, 47568, 45674)), results = c(30, 19, 29, NA, 25, 26, 27))`. Das Extrahieren von Elementen aus einer Liste kann nach Name (wenn die Liste benannt ist) oder nach Index erfolgen. In diesem Beispiel erhalten `mylist$results` und `mylist[[6]]` dasselbe Element. Warnung: Wenn Sie `mylist[6]` versuchen, gibt R `mylist[6]` Fehler aus, sondern extrahiert das Ergebnis als Liste. Während `mylist[[6]][2]` erlaubt ist (es gibt 19), gibt `mylist[6][2]` einen Fehler.
- **Datenrahmen:** Objekt mit Spalten, die Vektoren gleicher Länge, jedoch (möglicherweise) verschiedener Typen sind. Sie sind keine Matrizen. `exam <- data.frame(matr = as.character(c(45020, 45679, 46789, 43126, 42345, 47568, 45674)), res_S = c(30, 19, 29, NA, 25, 26, 27), res_O = c(3, 3, 1, NA, 3, 2, NA), res_TOT = c(30, 22, 30, NA, 28, 28, 27))`. Die Spalten können nach Name `exam$matr`, `exam$matr`, `exam[, 'matr']` oder nach `exam[1]`, `exam[,1]` gelesen werden. Zeilen können auch nach Namenstest `exam['rowname', ]` oder `exam['rowname', ] exam[1,]` gelesen werden. Dataframes sind eigentlich nur Listen mit einer bestimmten Struktur (Rownames-Attribut und Komponenten gleicher Länge).

## Gemeinsame Operationen und einige Warnhinweise

Standardvorgänge werden Element für Element ausgeführt. Siehe `?Syntax` für die Regeln der Operatorvorrangigkeit. Die meisten Operatoren (und andere Funktionen in der Basis-R) verfügen über Recyclingregeln, die Argumente unterschiedlicher Länge zulassen. Angesichts dieser Objekte:

## Beispielobjekte

```
> a <- 1
> b <- 2
> c <- c(2,3,4)
> d <- c(10,10,10)
> e <- c(1,2,3,4)
> f <- 1:6
> W <- cbind(1:4,5:8,9:12)
> Z <- rbind(rep(0,3),1:3,rep(10,3),c(4,7,1))
```

## Einige Vektoroperationen

```
> a+b # scalar + scalar
[1] 3
> c+d # vector + vector
[1] 12 13 14
> a*b # scalar * scalar
[1] 2
> c*d # vector * vector (componentwise!)
[1] 20 30 40
> c+a # vector + scalar
[1] 3 4 5
> c^2 #
[1] 4 9 16
> exp(c)
[1] 7.389056 20.085537 54.598150
```

## Einige Vektoroperationen Warnungen!

```
> c+e # warning but.. no errors, since recycling is assumed to be desired.
[1] 3 5 7 6
Warning message:
In c + e : longer object length is not a multiple of shorter object length
```

R summiert, was es kann, und verwendet dann den kürzeren Vektor erneut, um die Lücken auszufüllen ... Die Warnung wurde nur ausgegeben, weil die beiden Vektoren Längen haben, die nicht genau Vielfache sind. `c + f` # keine Warnung.

## Einige Matrixoperationen Warnung!

```
> Z+W # matrix + matrix #(componentwise)
> Z*W # matrix* matrix#(Standard product is always componentwise)
```

Um eine Matrix zu verwenden, multiplizieren Sie: `V%*%W`

```
> W + a # matrix+ scalar is still componentwise
 [,1] [,2] [,3]
[1,] 2 6 10
[2,] 3 7 11
[3,] 4 8 12
[4,] 5 9 13

> W + c # matrix + vector... : no warnings and R does the operation in a column-wise manner
 [,1] [,2] [,3]
[1,] 3 8 13
[2,] 5 10 12
[3,] 7 9 14
[4,] 6 11 16
```

## "Private" Variablen

Ein führender Punkt in einem Namen einer Variablen oder Funktion in R wird im Allgemeinen verwendet, um anzuzeigen, dass die Variable oder Funktion verborgen sein soll.

So deklarieren Sie die folgenden Variablen

```
> foo <- 'foo'
> .foo <- 'bar'
```

Wenn Sie dann mit der `ls` Funktion Objekte `ls` , wird nur das erste Objekt angezeigt.

```
> ls()
[1] "foo"
```

`all.names = TRUE` Sie jedoch `all.names = TRUE` an die Funktion übergeben, wird die private Variable `all.names = TRUE`

```
> ls(all.names = TRUE)
[1] ".foo" "foo"
```

Variablen online lesen: <https://riptutorial.com/de/r/topic/9013/variablen>

# Kapitel 117: Vektoren erstellen

## Examples

### Zahlenfolge

Verwenden Sie die `:` Operatorsequenzen von Zahlen zu erzeugen, wie zum Beispiel für den Einsatz in Vektorisierung größere Brocken des Codes:

```
x <- 1:5
x
[1] 1 2 3 4 5
```

Dies funktioniert in beide Richtungen

```
10:4
[1] 10 9 8 7 6 5 4
```

und sogar mit Fließkommazahlen

```
1.25:5
[1] 1.25 2.25 3.25 4.25
```

oder Negative

```
-4:4
[1] -4 -3 -2 -1 0 1 2 3 4
```

### seq ()

`seq` ist eine flexiblere Funktion als der `:` Bediener ermöglicht zu spezifizieren Schritte außer 1.

Die Funktion erstellt eine Sequenz vom `start` (Standardeinstellung ist 1) bis zum Ende, einschließlich dieser Nummer.

Sie können nur den Endparameter ( `to` ) angeben

```
seq(5)
[1] 1 2 3 4 5
```

Ebenso wie der Start

```
seq(2, 5) # or seq(from=2, to=5)
[1] 2 3 4 5
```

Und zum Schluss der Schritt ( `by` )

```
seq(2, 5, 0.5) # or seq(from=2, to=5, by=0.5)
[1] 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

`seq` kann optional auf die (gleichmäßig beabstandeten) Schritte schließen, wenn alternativ die gewünschte Länge des Ausgangs ( `length.out` ) geliefert wird

```
seq(2,5, length.out = 10)
[1] 2.0 2.3 2.6 2.9 3.2 3.5 3.8 4.1 4.4 4.7 5.0
```

Wenn die Sequenz die gleiche Länge wie ein anderer Vektor haben muss, können Sie die `along.with` als Abkürzung für `length.out = length(x)`

```
x = 1:8
seq(2,5,along.with = x)
[1] 2.000000 2.428571 2.857143 3.285714 3.714286 4.142857 4.571429 5.000000
```

Es gibt zwei nützliche vereinfachte Funktionen in der `seq` Familie: `seq_along` , `seq_len` und `seq.int` .  
`seq_along` Funktionen `seq_along` und `seq_len` konstruieren die natürlichen `seq_len` von 1 bis N, wobei N durch das Funktionsargument, die Länge eines Vektors oder einer Liste mit `seq_along` und das Ganzzahlargument mit `seq_len` .

```
seq_along(x)
[1] 1 2 3 4 5 6 7 8
```

Beachten Sie, dass `seq_along` die Indizes eines vorhandenen Objekts zurückgibt.

```
counting numbers 1 through 10
seq_len(10)
[1] 1 2 3 4 5 6 7 8 9 10
indices of existing vector (or list) with seq_along
letters[1:10]
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
seq_along(letters[1:10])
[1] 1 2 3 4 5 6 7 8 9 10
```

`seq.int` ist das gleiche wie `seq` , das für alte Kompatibilität beibehalten wurde.

Es gibt auch eine alte `sequence` , die einen Vektor von Sequenzen aus einem nicht negativen Argumente erzeugt.

```
sequence(4)
[1] 1 2 3 4
sequence(c(3, 2))
[1] 1 2 3 1 2
sequence(c(3, 2, 5))
[1] 1 2 3 1 2 1 2 3 4 5
```

## Vektoren

Vektoren in R können verschiedene Typen haben (z. B. Ganzzahl, logisch, Zeichen). Die

allgemeinste Art, einen Vektor zu definieren, ist die Verwendung des Funktionsvektors `vector()` .

```
vector('integer',2) # creates a vector of integers of size 2.
vector('character',2) # creates a vector of characters of size 2.
vector('logical',2) # creates a vector of logicals of size 2.
```

In R sind die Abkürzungsfunktionen jedoch allgemein beliebter.

```
integer(2) # is the same as vector('integer',2) and creates an integer vector with two
elements
character(2) # is the same as vector('integer',2) and creates an character vector with two
elements
logical(2) # is the same as vector('logical',2) and creates an logical vector with two
elements
```

Das Erstellen von Vektoren mit anderen Werten als den Standardwerten ist ebenfalls möglich. Oft wird dazu die Funktion `c()` verwendet. Das `c` steht für kombinieren oder verketteten.

```
c(1, 2) # creates a integer vector of two elements: 1 and 2.
c('a', 'b') # creates a character vector of two elements: a and b.
c(T,F) # creates a logical vector of two elements: TRUE and FALSE.
```

Hierbei ist zu beachten, dass R eine ganze Zahl (zB 1) als Ganzzahlvektor der Größe Eins interpretiert. Gleiches gilt für Numerik (zB 1.1), Logische (zB T oder F) oder Zeichen (zB 'a'). Daher kombinieren Sie im Wesentlichen Vektoren, die wiederum Vektoren sind.

Achten Sie darauf, dass Sie immer ähnliche Vektoren kombinieren müssen. Andernfalls versucht R, die Vektoren in Vektoren desselben Typs zu konvertieren.

```
c(1,1.1,'a',T) # all types (integer, numeric, character and logical) are converted to the
'lowest' type which is character.
```

Das Finden von Elementen in Vektoren kann mit dem Operator `[]` werden.

```
vec_int <- c(1,2,3)
vec_char <- c('a','b','c')
vec_int[2] # accessing the second element will return 2
vec_char[2] # accessing the second element will return 'b'
```

Dies kann auch zum Ändern von Werten verwendet werden

```
vec_int[2] <- 5 # change the second value from 2 to 5
vec_int # returns [1] 1 5 3
```

Schließlich kann mit dem Operator `:` (kurz für die Funktion `seq()` ) schnell ein Zahlenvektor erstellt werden.

```
vec_int <- 1:10
vec_int # returns [1] 1 2 3 4 5 6 7 8 9 10
```

Dies kann auch verwendet werden, um Vektoren zu subsetieren (von einfachen zu komplexeren Subsets).

```
vec_char <- c('a','b','c','d','e')
vec_char[2:4] # returns [1] "b" "c" "d"
vec_char[c(1,3,5)] # returns [1] "a" "c" "e"
```

## Benannte Vektoren erstellen

Der benannte Vektor kann auf verschiedene Arten erstellt werden. Mit `c` :

```
xc <- c('a' = 5, 'b' = 6, 'c' = 7, 'd' = 8)
```

was in ... resultiert:

```
> xc
a b c d
5 6 7 8
```

mit `list` :

```
x1 <- list('a' = 5, 'b' = 6, 'c' = 7, 'd' = 8)
```

was in ... resultiert:

```
> x1
$a
[1] 5

$b
[1] 6

$c
[1] 7

$d
[1] 8
```

Mit der Funktion `setNames` können zwei Vektoren gleicher Länge verwendet werden, um einen benannten Vektor zu erstellen:

```
x <- 5:8
y <- letters[1:4]

xy <- setNames(x, y)
```

was zu einem benannten Integer-Vektor führt:

```
> xy
a b c d
5 6 7 8
```



Wie zu sehen ist, ergibt dies das gleiche Ergebnis wie die `c` Methode.

Sie können auch die `names` Funktion verwenden, um dasselbe Ergebnis zu erhalten:

```
xy <- 5:8
names(xy) <- letters[1:4]
```

Mit einem solchen Vektor ist es auch möglich, Elemente nach Namen auszuwählen:

```
> xy["c"]
c
7
```

Dieses Merkmal ermöglicht die Verwendung eines solchen benannten Vektors als Nachschlagvektor / -tabelle, um die Werte mit den Werten eines anderen Vektors oder einer anderen Spalte im Datenrahmen abzugleichen. Betrachten Sie den folgenden Datenrahmen:

```
mydf <- data.frame(let = c('c','a','b','d'))

> mydf
 let
1 c
2 a
3 b
4 d
```

Angenommen, Sie möchten eine neue Variable mit dem Namen `num` mit den korrekten Werten aus `xy` in den Zeilen im `mydf` Datenrahmen erstellen. Mit der `match` Funktion können die entsprechenden Werte von `xy` ausgewählt werden:

```
mydf$num <- xy[match(mydf$let, names(xy))]
```

was in ... resultiert:

```
> mydf
 let num
1 c 7
2 a 5
3 b 6
4 d 8
```

## Erweitern eines Vektors mit der Funktion `rep()`

Die Funktion `rep` kann verwendet werden, um einen Vektor ziemlich flexibel zu wiederholen.

```
repeat counting numbers, 1 through 5 twice
rep(1:5, 2)
[1] 1 2 3 4 5 1 2 3 4 5

repeat vector with incomplete recycling
rep(1:5, 2, length.out=7)
```

```
[1] 1 2 3 4 5 1 2
```

Jedes Argument ist besonders nützlich für die Erweiterung eines Statistikvektors von Beobachtungs- / Experimentaleinheiten in einen Vektorvektor mit wiederholten Beobachtungen dieser Einheiten.

```
same except repeat each integer next to each other
rep(1:5, each=2)
[1] 1 1 2 2 3 3 4 4 5 5
```

`rep` Bezug auf die Erweiterung einer solchen Datenstruktur besonders interessant, da die Erweiterung eines Vektors zu einem unsymmetrischen Feld erreicht werden kann, indem das Längenargument durch einen Vektor ersetzt wird, der die Anzahl der Wiederholungen jedes Elements im Vektor festlegt:

```
automated length repetition
rep(1:5, 1:5)
[1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
hand-fed repetition length vector
rep(1:5, c(1,1,1,2,2))
[1] 1 2 3 4 4 5 5
```

Dies sollte die Möglichkeit aufzeigen, dass eine externe Funktion das zweite Argument von `rep`, um dynamisch einen Vektor zu konstruieren, der sich entsprechend den Daten ausdehnt.

Wie bei `seq` sind schnellere, vereinfachte Versionen von `rep`, `rep_len` und `rep.int`. Dadurch werden einige Attribute `rep`, die von `rep` beibehalten werden. `rep` kann es in Situationen, in denen Geschwindigkeit von Belang ist und zusätzliche Aspekte des wiederholten Vektors nicht erforderlich sind, am nützlichsten sein.

```
repeat counting numbers, 1 through 5 twice
rep.int(1:5, 2)
[1] 1 2 3 4 5 1 2 3 4 5

repeat vector with incomplete recycling
rep_len(1:5, length.out=7)
[1] 1 2 3 4 5 1 2
```

## Vektoren von eingebauten Konstanten: Folgen von Buchstaben und Monatsnamen

`R` hat eine Reihe von eingebauten Konstanten. Folgende Konstanten stehen zur Verfügung:

- `LETTERS` : die 26 Großbuchstaben des römischen Alphabets
- `letters` : die 26 Kleinbuchstaben des römischen Alphabets
- `month.abb` : Die aus drei Buchstaben bestehenden Abkürzungen für die englischen Monatsnamen
- `month.name` : die englischen Namen für die Monate des Jahres

- $\pi$  : das Verhältnis des Umfangs eines Kreises zu seinem Durchmesser

Aus den Buchstaben und Monatskonstanten können Vektoren erstellt werden.

## 1) Buchstabenfolgen:

```
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v"
"w" "x" "y" "z"

> LETTERS[7:9]
[1] "G" "H" "I"

> letters[c(1,5,3,2,4)]
[1] "a" "e" "c" "b" "d"
```

## 2) Sequenzen von Monatsabkürzungen oder Monatsnamen:

```
> month.abb
[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"

> month.name[1:4]
[1] "January" "February" "March" "April"

> month.abb[c(3,6,9,12)]
[1] "Mar" "Jun" "Sep" "Dec"
```

Vektoren erstellen online lesen: <https://riptutorial.com/de/r/topic/1088/vektoren-erstellen>

# Kapitel 118: Verarbeitung natürlicher Sprache

## Einführung

Natural Language Processing (NLP) ist ein Bereich der Informatik, der sich auf das Abrufen von Informationen aus der von Menschen generierten Texteingabe konzentriert.

## Examples

### Erstellen Sie eine Termfrequenzmatrix

Die einfachste Herangehensweise an das Problem (und die bis jetzt am häufigsten verwendete) besteht darin, Sätze in *Token* aufzuteilen. Vereinfachend haben *Wörter* abstrakte und subjektive Bedeutungen für die Menschen, die sie verwenden und erhalten. *Token* haben eine objektive Interpretation: eine geordnete Folge von Zeichen (oder Bytes). Sobald die Sätze aufgeteilt sind, wird die Reihenfolge des Tokens ignoriert. Diese Herangehensweise an das Problem ist als **Bag of Words**-Modell bekannt.

Eine **Begriffshäufigkeit** ist ein Wörterbuch, in dem jedem Token eine *Gewichtung* zugewiesen wird. Im ersten Beispiel konstruieren wir eine Termfrequenzmatrix aus einem Corpus **Corpus** (einer Sammlung von **Dokumenten**) mit dem R-Paket `tm`.

```
require(tm)
doc1 <- "drugs hospitals doctors"
doc2 <- "smog pollution environment"
doc3 <- "doctors hospitals healthcare"
doc4 <- "pollution environment water"
corpus <- c(doc1, doc2, doc3, doc4)
tm_corpus <- Corpus(VectorSource(corpus))
```

In diesem Beispiel haben wir einen Korpus der Klasse `Corpus`, der durch das Paket `tm` mit den zwei Funktionen `Corpus` und `VectorSource`, die ein `VectorSource` Objekt aus einem Zeichenvektor zurückgibt. Das Objekt `tm_corpus` ist eine Liste unserer Dokumente mit zusätzlichen (und optionalen) Metadaten, um jedes Dokument zu beschreiben.

```
str(tm_corpus)
List of 4
 $ 1:List of 2
 ..$ content: chr "drugs hospitals doctors"
 ..$ meta :List of 7
 $ author : chr(0)
 $ timestamp: POSIXlt[1:1], format: "2017-06-03 00:31:34"
 $ description : chr(0)
 $ heading : chr(0)
 $ id : chr "1"
 $ language : chr "en"
 $ origin : chr(0)
 .. -- attr(*, "class")= chr "TextDocumentMeta"
 .. -- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
```

```
[truncated]
```

Sobald wir einen `Corpus`, können wir mit der Vorverarbeitung der im `Corpus` enthaltenen Token fortfahren, um die Qualität der endgültigen Ausgabe (der Begriff Frequenzmatrix) zu verbessern. Dazu verwenden wir die `tm` Funktion `tm_map`, die ähnlich wie die `apply` Funktionsfamilie die Dokumente im Korpus transformiert, indem auf jedes Dokument eine Funktion `tm_map` wird.

```
tm_corpus <- tm_map(tm_corpus, tolower)
tm_corpus <- tm_map(tm_corpus, removeWords, stopwords("english"))
tm_corpus <- tm_map(tm_corpus, removeNumbers)
tm_corpus <- tm_map(tm_corpus, PlainTextDocument)
tm_corpus <- tm_map(tm_corpus, stemDocument, language="english")
tm_corpus <- tm_map(tm_corpus, stripWhitespace)
tm_corpus <- tm_map(tm_corpus, PlainTextDocument)
```

Nach diesen Transformationen erstellen wir schließlich die Termfrequenzmatrix mit

```
tdm <- TermDocumentMatrix(tm_corpus)
```

was gibt ein

```
<<TermDocumentMatrix (terms: 8, documents: 4)>>
Non-/sparse entries: 12/20
Sparsity : 62%
Maximal term length: 9
Weighting : term frequency (tf)
```

das können wir sehen, indem wir es in eine Matrix transformieren

```
as.matrix(tdm)
```

|           | Docs         |              |              |              |
|-----------|--------------|--------------|--------------|--------------|
| Terms     | character(0) | character(0) | character(0) | character(0) |
| doctor    | 1            | 0            | 1            | 0            |
| drug      | 1            | 0            | 0            | 0            |
| environ   | 0            | 1            | 0            | 1            |
| healthcar | 0            | 0            | 1            | 0            |
| hospit    | 1            | 0            | 1            | 0            |
| pollut    | 0            | 1            | 0            | 1            |
| smog      | 0            | 1            | 0            | 0            |
| water     | 0            | 0            | 0            | 1            |

Jede Zeile stellt die Frequenz jedes Token - dass, wie Sie (zB bemerkt dämmt wurden `environment` zu `environ` in jedem Dokument (4 Dokumente, 4 Spalten) -).

In den vorherigen Zeilen haben wir jedes Token / Dokument jedes Paares mit der absoluten Häufigkeit (dh der Anzahl von Instanzen des Tokens, die im Dokument vorkommen) gewichtet.

Verarbeitung natürlicher Sprache online lesen:

<https://riptutorial.com/de/r/topic/10119/verarbeitung-natuerlicher-sprache>

---

# Kapitel 119: Veröffentlichung

## Einführung

Es gibt viele Möglichkeiten, R-Code, Tabellen und Diagramme für die Veröffentlichung zu formatieren.

## Bemerkungen

R-Benutzer möchten häufig Analysen und Ergebnisse auf reproduzierbare Weise veröffentlichen. Weitere Informationen finden Sie unter [Reproduzierbares R](#).

## Examples

### Tabellen formatieren

Unter "Tabelle" ist hier allgemein zu `data.frame` (`data.frame`, `data.frame`, `table`

---

## Drucken in einfachem Text

Das Drucken (wie in der Konsole zu sehen) reicht möglicherweise aus, um ein Nur-Text-Dokument in Monospaced-Schrift anzuzeigen:

*Hinweis: Stellen Sie vor dem Erstellen der Beispieldaten sicher, dass Sie sich in einem leeren Ordner befinden, in den Sie schreiben können. Führen Sie `getwd()` und lesen Sie `?setwd` wenn Sie Ordner wechseln müssen.*

```
..w = options()$width
options(width = 500) # reduce text wrapping
sink(file = "mytab.txt")
 summary(mtcars)
sink()
options(width = ..w)
rm(..w)
```

---

## Drucken von Tabellen mit Trennzeichen

Eine weitere Option ist das Schreiben in CSV (oder ein anderes allgemeines Format) und das anschließende Öffnen in einem Tabellenkalkulations-Editor.

*Hinweis: Stellen Sie vor dem Erstellen der Beispieldaten sicher, dass Sie sich in einem leeren Ordner befinden, in den Sie schreiben können. Führen Sie `getwd()` und lesen Sie `?setwd` wenn Sie Ordner wechseln müssen.*

```
write.csv(mtcars, file="mytab.csv")
```

---

## Weitere Ressourcen

- `knitr::kable`
- Sterngucker
- `tables::tabular`
- [Texreg](#)
- `xtabelle`

### Ganze Dokumente formatieren

`Sweave` aus dem Paket `utils` ermöglicht die gemeinsame Formatierung von Code, Prosa, Diagrammen und Tabellen in einem LaTeX-Dokument.

---

## Weitere Ressourcen

- Knitr und RMarkdown

Veröffentlichung online lesen: <https://riptutorial.com/de/r/topic/9039/veroffentlichung>

---

# Kapitel 120: Verteilungsfunktionen

## Einführung

R hat viele integrierte Funktionen, um mit Wahrscheinlichkeitsverteilungen zu arbeiten, wobei offizielle Dokumente bei `?Distributions`.

## Bemerkungen

Es gibt im Allgemeinen vier Präfixe:

- **d** -Der **Dichtefunktion** für die gegebene Verteilung
- **p** Die kumulative Verteilungsfunktion
- **q** -Holen das **Quantil** mit der gegebenen Wahrscheinlichkeit zugeordnet
- **r** -Holen eine **Stichprobe**

Die in der Basisinstallation von R integrierten `?Distributions`.

## Examples

### Normalverteilung

Als Beispiel verwenden wir `*norm`. Aus der Dokumentation:

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Wenn ich also den Wert einer Standard-Normalverteilung bei 0 wissen wollte, würde ich das tun

```
dnorm(0)
```

Das gibt uns `0.3989423`, eine vernünftige Antwort.

Auf dieselbe Weise `pnorm(0) .5`. Dies ist wiederum sinnvoll, da sich die Hälfte der Verteilung links von 0 befindet.

`qnorm` wird im Wesentlichen das Gegenteil von `pnorm`. `qnorm(.5)` ergibt 0.

Schließlich gibt es noch die Funktion `rnorm`:

```
rnorm(10)
```

Erzeugt 10 Proben aus dem Normalzustand.



Wenn Sie die Parameter einer bestimmten Verteilung ändern möchten, ändern Sie sie einfach so

```
rnorm(10, mean=4, sd= 3)
```

## Binomialverteilung

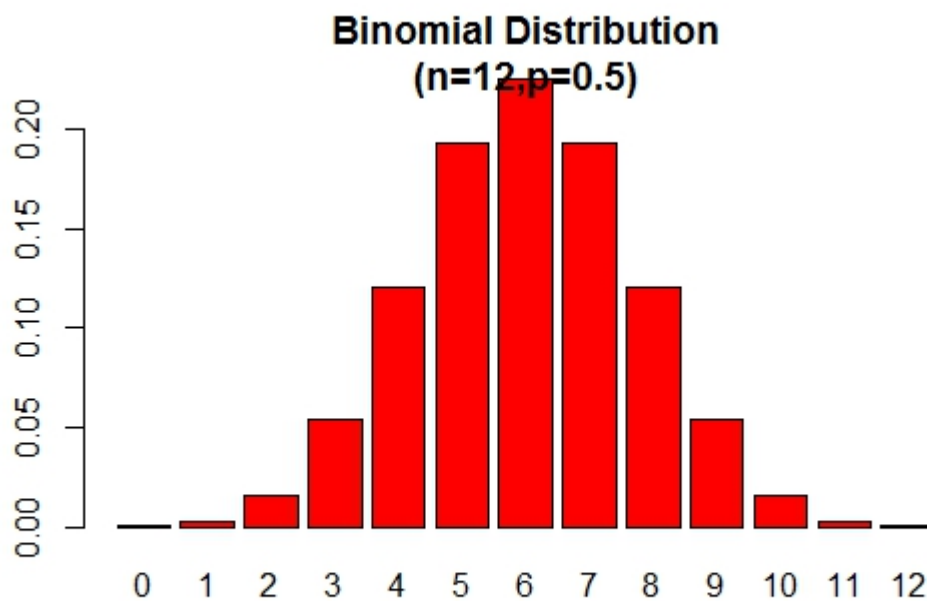
Wir erläutern nun die für die *Binomialverteilung* definierten Funktionen `dbinom`, `pbinom`, `qbinom` und `rbinom`.

Die Funktion `dbinom()` gibt die Wahrscheinlichkeiten für verschiedene Werte der binomialen Variablen an. Minimal erfordert es drei Argumente. Das erste Argument für diese Funktion muss ein Vektor von Quantilen sein (die möglichen Werte der Zufallsvariablen  $x$ ). Das zweite und dritte Argument sind die *defining parameters* der Verteilung, nämlich  $n$  (die Anzahl unabhängiger Versuche) und  $p$  (die Erfolgswahrscheinlichkeit in jedem Versuch). Für eine Binomialverteilung mit  $n = 5$ ,  $p = 0.5$  sind die möglichen Werte für  $X$  beispielsweise  $0, 1, 2, 3, 4, 5$ . Das heißt, die Funktion `dbinom(x,n,p)` ergibt die Wahrscheinlichkeitswerte  $P(X = x)$  für  $x = 0, 1, 2, 3, 4, 5$ .

```
#Binom(n = 5, p = 0.5) probabilities
> n <- 5; p<- 0.5; x <- 0:n
> dbinom(x,n,p)
[1] 0.03125 0.15625 0.31250 0.31250 0.15625 0.03125
#To verify the total probability is 1
> sum(dbinom(x,n,p))
[1] 1
>
```

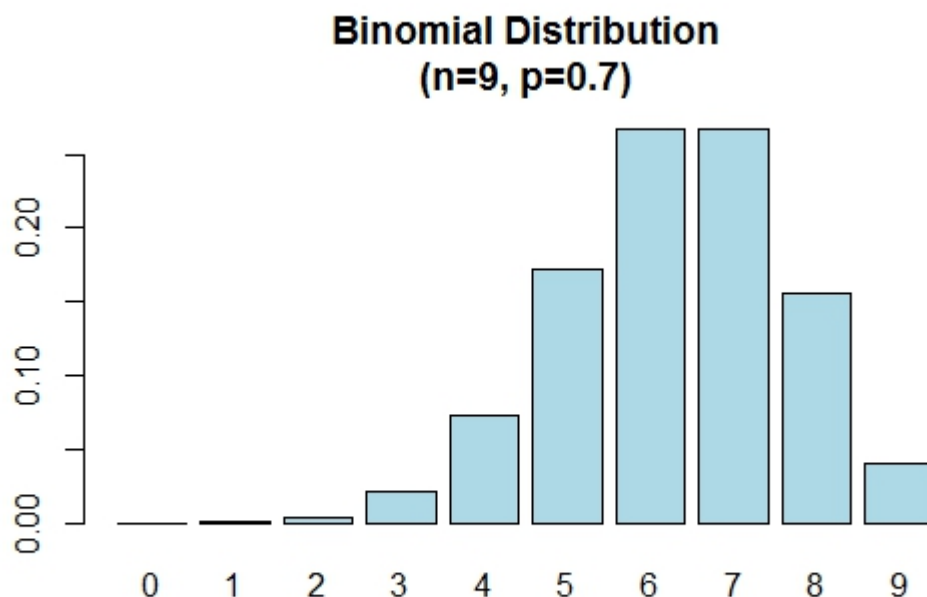
Das binomische Wahrscheinlichkeitsverteilungsdiagramm kann wie in der folgenden Abbildung dargestellt werden:

```
> x <- 0:12
> prob <- dbinom(x,12,.5)
> barplot(prob,col = "red",ylim = c(0,.2),names.arg=x,
 main="Binomial Distribution\n(n=12,p=0.5) ")
```



Beachten Sie, dass die Binomialverteilung symmetrisch ist, wenn  $p = 0.5$ . Betrachten Sie das folgende Beispiel, um zu zeigen, dass die Binomialverteilung negativ verzerrt ist, wenn  $p$  größer als 0.5 ist.

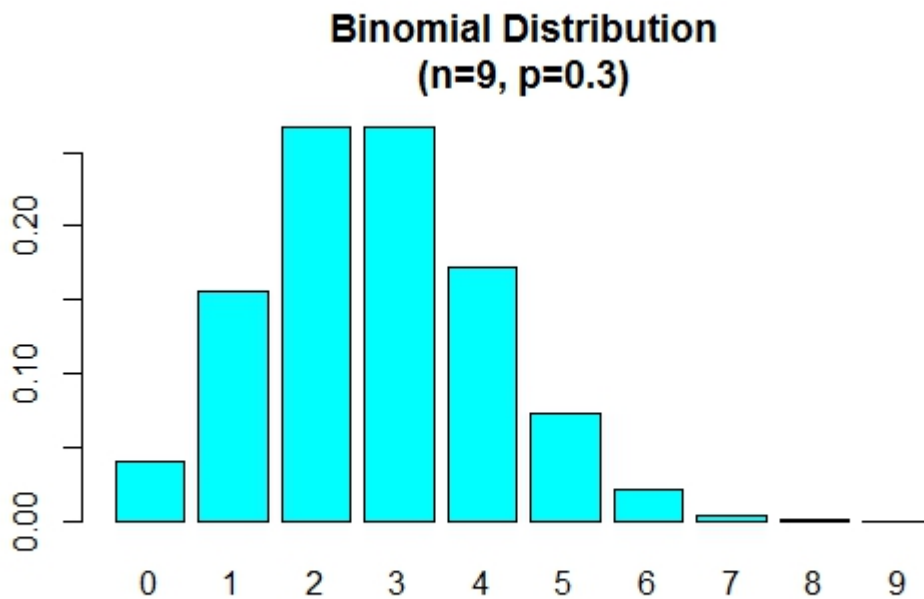
```
> n=9; p=.7; x=0:n; prob=dbinom(x,n,p);
> barplot(prob, names.arg = x, main="Binomial Distribution\n(n=9, p=0.7)", col="lightblue")
```



Wenn  $p$  kleiner als 0.5 die Binomialverteilung wie unten gezeigt positiv geneigt.

```
> n=9; p=.3; x=0:n; prob=dbinom(x,n,p);
```

```
> barplot(prob, names.arg = x, main="Binomial Distribution\n(n=9, p=0.3)", col="cyan")
```



Wir werden nun die Verwendung der kumulativen Verteilungsfunktion `pbinom()`. Mit dieser Funktion können Wahrscheinlichkeiten wie  $P(X \leq x)$  berechnet werden. Das erste Argument für diese Funktion ist ein Vektor von Quantilen (Werte von  $x$ ).

```
Calculating Probabilities
P(X <= 2) in a Bin(n=5,p=0.5) distribution
> pbinom(2,5,0.5)
[1] 0.5
```

Die obige Wahrscheinlichkeit kann auch wie folgt erhalten werden:

```
P(X <= 2) = P(X=0) + P(X=1) + P(X=2)
> sum(dbinom(0:2,5,0.5))
[1] 0.5
```

Zur Berechnung von Wahrscheinlichkeiten des Typs:  $P(a \leq X \leq b)$

```
P(3 <= X <= 5) = P(X=3) + P(X=4) + P(X=5) in a Bin(n=9,p=0.6) dist
> sum(dbinom(c(3,4,5),9,0.6))
[1] 0.4923556
>
```

Darstellung der Binomialverteilung in Form einer Tabelle:

```
> n = 10; p = 0.4; x = 0:n;
> prob = dbinom(x,n,p)
> cdf = pbinom(x,n,p)
> distTable = cbind(x,prob,cdf)
> distTable
```

```

 x prob cdf
[1,] 0 0.0060466176 0.006046618
[2,] 1 0.0403107840 0.046357402
[3,] 2 0.1209323520 0.167289754
[4,] 3 0.2149908480 0.382280602
[5,] 4 0.2508226560 0.633103258
[6,] 5 0.2006581248 0.833761382
[7,] 6 0.1114767360 0.945238118
[8,] 7 0.0424673280 0.987705446
[9,] 8 0.0106168320 0.998322278
[10,] 9 0.0015728640 0.999895142
[11,] 10 0.0001048576 1.000000000
>

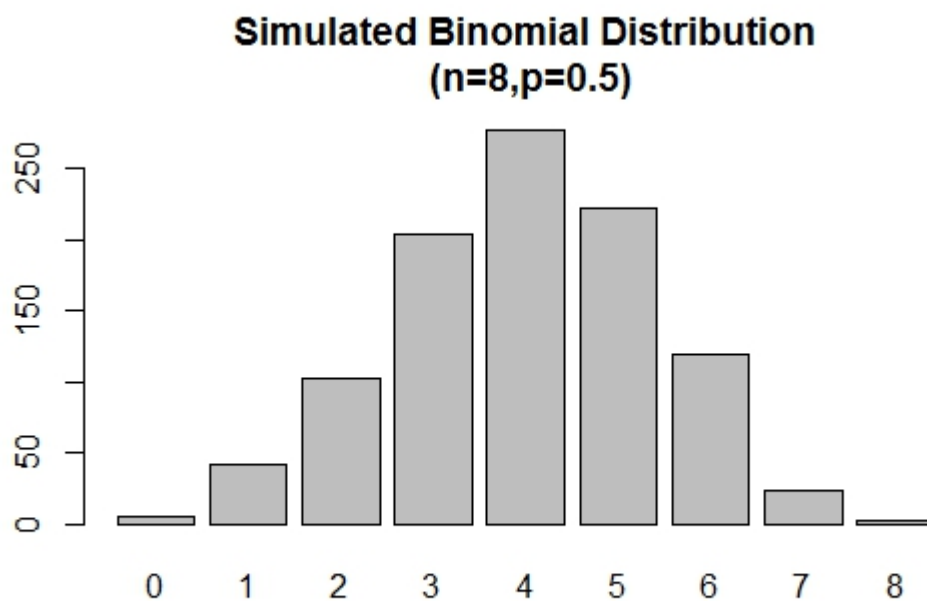
```

Das `rbinom()` wird verwendet, um Zufallsstichproben von angegebenen Größen mit einem gegebenen Parameterwert zu erzeugen.

```

Simulation
> xVal<-names(table(rbinom(1000,8,.5)))
> barplot(as.vector(table(rbinom(1000,8,.5))),names.arg =xVal,
 main="Simulated Binomial Distribution\n (n=8,p=0.5)")

```



Verteilungsfunktionen online lesen: <https://riptutorial.com/de/r/topic/1885/verteilungsfunktionen>

---

# Kapitel 121: Verwenden von Texreg zum Exportieren von Modellen in Papierform

## Einführung

Das texreg-Paket hilft beim Exportieren eines Modells (oder mehrerer Modelle) auf saubere Weise. Das Ergebnis kann als HTML oder .doc (MS Office Word) exportiert werden.

## Bemerkungen

---

## Links

- [CRAN-Seite](#)

## Examples

### Lineare Regressionsergebnisse drucken

```
models
fit1 <- lm(mpg ~ wt, data = mtcars)
fit2 <- lm(mpg ~ wt+hp, data = mtcars)
fit3 <- lm(mpg ~ wt+hp+cyl, data = mtcars)

export to html
texreg::htmlreg(list(fit1, fit2, fit3), file='models.html')

export to doc
texreg::htmlreg(list(fit1, fit2, fit3), file='models.doc')
```

Das Ergebnis sieht aus wie eine Tabelle in einer Zeitung.

|             | Model 1            | Model 2            | Model 3            |
|-------------|--------------------|--------------------|--------------------|
| (Intercept) | 37.29***<br>(1.88) | 37.23***<br>(1.60) | 38.75***<br>(1.79) |
| wt          | -5.34***<br>(0.56) | -3.88***<br>(0.63) | -3.17***<br>(0.74) |
| hp          |                    | -0.03**<br>(0.01)  | -0.02<br>(0.01)    |
| cyl         |                    |                    | -0.94<br>(0.55)    |
| R2          | 0.75               | 0.83               | 0.84               |
| Adj. R2     | 0.74               | 0.81               | 0.83               |
| Num. obs.   | 32                 | 32                 | 32                 |
| RMSE        | 3.05               | 2.59               | 2.51               |

\*\*\*p < 0.001, \*\*p < 0.01, \*p < 0.05

Statistical models

Die `texreg::htmlreg()` Funktion enthält mehrere weitere nützliche Parameter. Hier ist ein Anwendungsfall für die hilfreichsten Parameter.

```
export to html
texreg::htmlreg(list(fit1, fit2, fit3), file='models.html',
 single.row = T,
 custom.model.names = LETTERS[1:3],
 leading.zero = F,
 digits = 3)
```

Welches Ergebnis in so einer Tabelle

|             | A                 | B                 | C                 |
|-------------|-------------------|-------------------|-------------------|
| (Intercept) | 37.285 (1.878)*** | 37.227 (1.599)*** | 38.752 (1.787)*** |
| wt          | -5.344 (0.559)*** | -3.878 (0.633)*** | -3.167 (0.741)*** |
| hp          |                   | -0.032 (0.009)**  | -0.018 (0.012)    |
| cyl         |                   |                   | -0.942 (0.551)    |
| R2          | 0.753             | 0.827             | 0.843             |
| Adj. R2     | 0.745             | 0.815             | 0.826             |
| Num. obs.   | 32                | 32                | 32                |
| RMSE        | 3.046             | 2.593             | 2.512             |

\*\*\*p < 0.001, \*\*p < 0.01, \*p < 0.05

Statistical models

Verwenden von Texreg zum Exportieren von Modellen in Papierform online lesen:

<https://riptutorial.com/de/r/topic/9037/verwenden-von-texreg-zum-exportieren-von-modellen-in-papierform>

---

# Kapitel 122: Verwendung der Rohrzuordnung in Ihrem eigenen Paket

## Einführung

Um die Pipe in einem vom Benutzer erstellten Paket verwenden zu können, muss sie wie jede andere zu importierende Funktion im NAMESPACE aufgeführt sein.

## Examples

### Pipe in eine Utility-Funktionsdatei einfügen

Eine Möglichkeit, dies zu tun, besteht darin, die Pipe aus dem Paket selbst zu exportieren. Dies kann in den "traditionellen" `zzz.R` oder `utils.R` Dateien `utils.R`, die viele Pakete für nützliche kleine Funktionen verwenden, die nicht als Teil des Pakets exportiert werden. Zum Beispiel:

```
#' Pipe operator
#'
#' @name %>%
#' @rdname pipe
#' @keywords internal
#' @export
#' @importFrom magrittr %>%
#' @usage lhs \%>\% rhs
NULL
```

Verwendung der Rohrzuordnung in Ihrem eigenen Paket online lesen:

<https://riptutorial.com/de/r/topic/10547/verwendung-der-rohrzuordnung-in-ihrem-eigenen-paket>

---

# Kapitel 123: Wahrscheinlichkeitsverteilungen mit R

## Examples

### PDF und PMF für verschiedene Distributionen in R

#### PMF FÜR DIE BINOMIALVERTEILUNG

Angenommen, ein fairer Würfel wird zehnmal gewürfelt. Wie hoch ist die Wahrscheinlichkeit, genau zwei Sechser zu werfen?

Sie können die Frage mit der Funktion `dbinom` beantworten:

```
> dbinom(2, 10, 1/6)
[1] 0.29071
```

#### PMF FÜR DIE POISSON-VERTRIEB

Es ist bekannt, dass die Anzahl der in einem Restaurant an einem bestimmten Tag bestellten Sandstücke einem Mittelwert von 20 entspricht. Wie hoch ist die Wahrscheinlichkeit, dass genau achtzehn Sandsteine bestellt werden, die morgen bestellt werden?

Sie können die Frage mit der Funktion `dpois` beantworten:

```
> dpois(18, 20)
[1] 0.08439355
```

#### PDF FÜR DIE NORMALE VERTEILUNG

Um den Wert des pdf bei  $x = 2,5$  für eine Normalverteilung mit einem Mittelwert von 5 und einer Standardabweichung von 2 zu ermitteln, verwenden Sie den Befehl:

```
> dnorm(2.5, mean=5, sd=2)
[1] 0.09132454
```

Wahrscheinlichkeitsverteilungen mit R online lesen:

<https://riptutorial.com/de/r/topic/4333/wahrscheinlichkeitsverteilungen-mit-r>



---

# Kapitel 124: Web Scraping und Parsing

## Bemerkungen

*Scraping* bezieht sich auf die Verwendung eines Computers zum Abrufen des Codes einer Webseite. Nachdem der Code erhalten wurde, muss er in eine nützliche Form zur weiteren Verwendung in R *geparst* werden.

In Base R sind nicht viele der für diese Prozesse erforderlichen Tools vorhanden, sodass das Scraping und die Analyse normalerweise mit Paketen durchgeführt werden. Einige Pakete sind am nützlichsten für das Scraping ( `RSelenium` , `httr` , `curl` , `RCurl` ), einige für die Analyse ( `XML` , `xml2` ) und einige für beide ( `rvest` ).

Ein verwandter Prozess ist das Scraping einer Web-API, die im Gegensatz zu einer Webseite Daten zurückgibt, die maschinenlesbar sein sollen. Viele der gleichen Pakete werden für beide verwendet.

---

## Rechtmäßigkeit

Einige Websites lehnen Scrap ab, sei es aufgrund einer erhöhten Serverauslastung oder aufgrund von Bedenken hinsichtlich des Dateneigentums. Wenn eine Website das Scraping in den Nutzungsbedingungen verbietet, ist das Scrapen illegal.

## Examples

### Grundschaben mit Rvest

`rvest` ist ein Paket für Web Scraping und Parsing von Hadley Wickham, inspiriert von Pythons [Beautiful Soup](#) . Es nutzt die `libxml2` Bindungen des Hadley- `xml2` Pakets für die HTML-Analyse.

Als Teil des Tidyverse wird die `rvest` [geleitet](#) . Es verwendet

- `xml2::read_html` um den HTML- `xml2::read_html` einer Webseite zu kratzen,
- das kann dann mit seinen `html_node` und `html_nodes` Funktionen unter Verwendung von CSS- oder XPath-Selektoren subset werden, und
- In R-Objekten mit Funktionen wie `html_text` und `html_table` .

Um die Tabelle der Meilensteine von [der Wikipedia-Seite zu R](#) zu kratzen, würde der Code aussehen

```
library(rvest)

url <- 'https://en.wikipedia.org/wiki/R_(programming_language) '

scrape HTML from website
```

```
url %>% read_html() %>%
 # select HTML tag with class="wikitable"
 html_node(css = '.wikitable') %>%
 # parse table into data.frame
 html_table() %>%
 # trim for printing
 dplyr::mutate(Description = substr(Description, 1, 70))
```

|       | Release | Date       | Description                                                  |
|-------|---------|------------|--------------------------------------------------------------|
| ## 1  | 0.16    |            | This is the last alpha version developed primarily by Ihaka  |
| ## 2  | 0.49    | 1997-04-23 | This is the oldest source release which is currently availab |
| ## 3  | 0.60    | 1997-12-05 | R becomes an official part of the GNU Project. The code is h |
| ## 4  | 0.65.1  | 1999-10-07 | First versions of update.packages and install.packages funct |
| ## 5  | 1.0     | 2000-02-29 | Considered by its developers stable enough for production us |
| ## 6  | 1.4     | 2001-12-19 | S4 methods are introduced and the first version for Mac OS X |
| ## 7  | 2.0     | 2004-10-04 | Introduced lazy loading, which enables fast loading of data  |
| ## 8  | 2.1     | 2005-04-18 | Support for UTF-8 encoding, and the beginnings of internatio |
| ## 9  | 2.11    | 2010-04-22 | Support for Windows 64 bit systems.                          |
| ## 10 | 2.13    | 2011-04-14 | Adding a new compiler function that allows speeding up funct |
| ## 11 | 2.14    | 2011-10-31 | Added mandatory namespaces for packages. Added a new paralle |
| ## 12 | 2.15    | 2012-03-30 | New load balancing functions. Improved serialization speed f |
| ## 13 | 3.0     | 2013-04-03 | Support for numeric index values 231 and larger on 64 bit sy |

Während dies eine `data.frame` zurückgibt, beachten Sie, dass für geschabte Daten typisch ist, gibt es noch die Reinigung weiterer Daten ist zu tun: hier, Termine Formatierung, Einfügen `NA` s, und so weiter.

Beachten Sie, dass Daten in einem weniger konsistenten rechteckigen Format möglicherweise durchlaufen werden müssen, um erfolgreich analysiert zu werden. Wenn auf der Website jQuery oder andere Mittel zum Einfügen von Inhalten verwendet werden, reicht `read_html` möglicherweise nicht aus, um einen `RSelenium` , und ein robusterer `RSelenium` wie `RSelenium` kann erforderlich sein.

## Verwendung von `rvest`, wenn ein Login erforderlich ist

Bei der Verschrottung eines Web tritt häufig das Problem auf, wie Sie eine Benutzer-ID und ein Kennwort eingeben, um sich bei einer Website anzumelden.

In diesem Beispiel, das ich erstellt habe, um meine Antworten zu verfolgen, die hier zum Stack-Überlauf gepostet wurden. Der gesamte Ablauf besteht darin, sich anzumelden, zu einer Webseite zu gelangen, Informationen zu sammeln, einen Datenrahmen hinzuzufügen und dann zur nächsten Seite zu gelangen.

```
library(rvest)

#Address of the login webpage
login<-
"https://stackoverflow.com/users/login?ssrc=head&returnurl=http%3a%2f%2fstackoverflow.com%2f"

#create a web session with the desired login address
pgsession<-html_session(login)
pgform<-html_form(pgsession)[[2]] #in this case the submit is the 2nd form
filled_form<-set_values(pgform, email="*****", password="*****")
submit_form(pgsession, filled_form)

#pre allocate the final results dataframe.
```

```

results<-data.frame()

#loop through all of the pages with the desired info
for (i in 1:5)
{
 #base address of the pages to extract information from
 url<-"http://stackoverflow.com/users/*****?tab=answers&sort=activity&page="
 url<-paste0(url, i)
 page<-jump_to(pgsession, url)

 #collect info on the question votes and question title
 summary<-html_nodes(page, "div .answer-summary")
 question<-matrix(html_text(html_nodes(summary, "div"), trim=TRUE), ncol=2, byrow = TRUE)

 #find date answered, hyperlink and whether it was accepted
 dateans<-html_node(summary, "span") %>% html_attr("title")
 hyperlink<-html_node(summary, "div a") %>% html_attr("href")
 accepted<-html_node(summary, "div") %>% html_attr("class")

 #create temp results then bind to final results
 rtemp<-cbind(question, dateans, accepted, hyperlink)
 results<-rbind(results, rtemp)
}

#Dataframe Clean-up
names(results)<-c("Votes", "Answer", "Date", "Accepted", "HyperLink")
results$Votes<-as.integer(as.character(results$Votes))
results$Accepted<-ifelse(results$Accepted=="answer-votes default", 0, 1)

```

Die Schleife ist in diesem Fall auf nur 5 Seiten begrenzt, dies muss sich ändern, um zu Ihrer Anwendung zu passen. Ich habe die benutzerspezifischen Werte durch \*\*\*\*\* ersetzt, hoffentlich gibt dies einige Hinweise für Ihr Problem.

Web Scraping und Parsing online lesen: <https://riptutorial.com/de/r/topic/2890/web-scraping-und-parsing>

---

# Kapitel 125: Web-Crawling in R

## Examples

### Standard-Kratzverfahren mit dem RCurl-Paket

Wir versuchen imdb Top-Chartfilme und Bewertungen zu extrahieren

```
R> library(RCurl)
R> library(XML)
R> url <- "http://www.imdb.com/chart/top"
R> top <- getURL(url)
R> parsed_top <- htmlParse(top, encoding = "UTF-8")
R> top_table <- readHTMLTable(parsed_top)[[1]]
R> head(top_table[1:10, 1:3])
```

```
Rank & Title IMDb Rating
1 1. The Shawshank Redemption (1994) 9.2
2 2. The Godfather (1972) 9.2
3 3. The Godfather: Part II (1974) 9.0
4 4. The Dark Knight (2008) 8.9
5 5. Pulp Fiction (1994) 8.9
6 6. The Good, the Bad and the Ugly (1966) 8.9
7 7. Schindler's List (1993) 8.9
8 8. 12 Angry Men (1957) 8.9
9 9. The Lord of the Rings: The Return of the King (2003) 8.9
10 10. Fight Club (1999) 8.8
```

Web-Crawling in R online lesen: <https://riptutorial.com/de/r/topic/4336/web-crawling-in-r>

# Kapitel 126: xgboost

## Examples

### Cross Validation und Tuning mit xgboost

```
library(caret) # for dummyVars
library(RCurl) # download https data
library(Metrics) # calculate errors
library(xgboost) # model

#####
Load data from UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets.html)
urlfile <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
x <- getURL(urlfile, ssl.verifypeer = FALSE)
adults <- read.csv(textConnection(x), header=F)

adults <- read.csv('https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data', header=F)
names(adults)=c('age', 'workclass', 'fnlwgt', 'education', 'educationNum',
 'maritalStatus', 'occupation', 'relationship', 'race',
 'sex', 'capitalGain', 'capitalLoss', 'hoursWeek',
 'nativeCountry', 'income')

clean up data
adults$income <- ifelse(adults$income==' <=50K',0,1)
binarize all factors
library(caret)
dmy <- dummyVars(" ~ .", data = adults)
adultsTrsf <- data.frame(predict(dmy, newdata = adults))
#####

what we're trying to predict adults that make more than 50k
outcomeName <- c('income')
list of features
predictors <- names(adultsTrsf)[!names(adultsTrsf) %in% outcomeName]

play around with settings of xgboost - eXtreme Gradient Boosting (Tree) library
https://github.com/tqchen/xgboost/wiki/Parameters
max.depth - maximum depth of the tree
nrounds - the max number of iterations

take first 10% of the data only!
trainPortion <- floor(nrow(adultsTrsf)*0.1)

trainSet <- adultsTrsf[1:floor(trainPortion/2),]
testSet <- adultsTrsf[(floor(trainPortion/2)+1):trainPortion,]

smallestError <- 100
for (depth in seq(1,10,1)) {
 for (rounds in seq(1,20,1)) {

 # train
 bst <- xgboost(data = as.matrix(trainSet[,predictors]),
 label = trainSet[,outcomeName],
 max.depth=depth, nround=rounds,
 objective = "reg:linear", verbose=0)

 gc()
```

```

 # predict
 predictions <- predict(bst, as.matrix(testSet[,predictors]),
outputmargin=TRUE)
 err <- rmse(as.numeric(testSet[,outcomeName]), as.numeric(predictions))

 if (err < smallestError) {
 smallestError = err
 print(paste(depth,rounds,err))
 }
 }
}

cv <- 30
trainSet <- adultsTrsf[1:trainPortion,]
cvDivider <- floor(nrow(trainSet) / (cv+1))

smallestError <- 100
for (depth in seq(1,10,1)) {
 for (rounds in seq(1,20,1)) {
 totalError <- c()
 indexCount <- 1
 for (cv in seq(1:cv)) {
 # assign chunk to data test
 dataTestIndex <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
 dataTest <- trainSet[dataTestIndex,]
 # everything else to train
 dataTrain <- trainSet[-dataTestIndex,]

 bst <- xgboost(data = as.matrix(dataTrain[,predictors]),
 label = dataTrain[,outcomeName],
 max.depth=depth, nround=rounds,
 objective = "reg:linear", verbose=0)

 gc()
 predictions <- predict(bst, as.matrix(dataTest[,predictors]),
outputmargin=TRUE)

 err <- rmse(as.numeric(dataTest[,outcomeName]),
as.numeric(predictions))
 totalError <- c(totalError, err)
 }
 if (mean(totalError) < smallestError) {
 smallestError = mean(totalError)
 print(paste(depth,rounds,smallestError))
 }
 }
}

#####
Test both models out on full data set

trainSet <- adultsTrsf[1:trainPortion,]

assign everything else to test
testSet <- adultsTrsf[(trainPortion+1):nrow(adultsTrsf),]

bst <- xgboost(data = as.matrix(trainSet[,predictors]),
 label = trainSet[,outcomeName],
 max.depth=4, nround=19, objective = "reg:linear", verbose=0)
pred <- predict(bst, as.matrix(testSet[,predictors]), outputmargin=TRUE)
rmse(as.numeric(testSet[,outcomeName]), as.numeric(pred))

```

```
bst <- xgboost(data = as.matrix(trainSet[,predictors]),
 label = trainSet[,outcomeName],
 max.depth=3, nround=20, objective = "reg:linear", verbose=0)
pred <- predict(bst, as.matrix(testSet[,predictors]), outputmargin=TRUE)
rmse(as.numeric(testSet[,outcomeName]), as.numeric(pred))
```

xgboost online lesen: <https://riptutorial.com/de/r/topic/3239/xgboost>

---

# Kapitel 127: Zeichenfolgen lesen und schreiben

## Bemerkungen

Verwandte Dokumente:

- [Benutzereingabe abrufen](#)

## Examples

### Zeichenketten drucken und anzeigen

R verfügt über mehrere integrierte Funktionen, die zum Drucken oder Anzeigen von Informationen verwendet werden können, aber `print` und `cat` sind die grundlegendsten Funktionen. Da R eine [interpretierte Sprache ist](#), können Sie diese direkt in der R-Konsole ausprobieren:

```
print("Hello World")
#[1] "Hello World"
cat("Hello World\n")
#Hello World
```

Beachten Sie den Unterschied in Eingang und Ausgang für die beiden Funktionen. (Hinweis: Der Wert von `x` keine Anführungszeichen, die mit `x <- "Hello World"`. Sie werden bei der Ausgabe durch `print` hinzugefügt.)

`cat` einen oder mehrere Zeichenvektoren als Argumente und gibt sie an die Konsole aus. Wenn der Zeichenvektor eine Länge von mehr als 1 hat, werden die Argumente standardmäßig durch ein Leerzeichen getrennt:

```
cat(c("hello", "world", "\n"))
#hello world
```

Ohne das neue Zeilenzeichen ( `\n` ) wäre die Ausgabe:

```
cat("Hello World")
#Hello World>
```

Die Eingabeaufforderung für den nächsten Befehl erscheint unmittelbar nach der Ausgabe. (Einige Konsolen, wie die von RStudio, fügen automatisch eine Newline an Zeichenfolgen an, die nicht mit einer Newline enden.)

`print` ist ein Beispiel für eine "generische" Funktion. Dies bedeutet, dass die Klasse des ersten übergebenen Arguments erkannt wird und eine klassenspezifische *Methode* zur Ausgabe verwendet wird. Bei einem Zeichenvektor wie `"Hello World"` ist das Ergebnis ähnlich wie bei der



Ausgabe von `cat` . Die Zeichenfolge wird jedoch in Anführungszeichen gesetzt und eine Zahl `[1]` ausgegeben, um das erste Element eines Zeichenvektors anzugeben (in diesem Fall das erste und einzige Element):

```
print("Hello World")
#[1] "Hello World"
```

Diese Standarddruckmethode sehen wir auch, wenn wir einfach R bitten, eine Variable zu drucken. Beachten Sie, dass die Ausgabe von `s` ist wie das Aufrufen von `print(s)` oder `print("Hello World")` :

```
s <- "Hello World"
s
#[1] "Hello World"
```

Oder auch ohne etwas zuzuordnen:

```
"Hello World"
#[1] "Hello World"
```

Wenn wir eine zweite Zeichenkette als zweites Element des Vektors hinzufügen (mit der Funktion `c()` , **um** die Elemente zusammen zu ketten), unterscheidet sich das Verhalten von `print()` deutlich von dem von `cat` :

```
print(c("Hello World", "Here I am. "))
#[1] "Hello World" "Here I am."
```

Beachten Sie, dass die `c()` Funktion *nicht* String-Verkettung macht. (Man muss dazu `paste` verwenden.) R zeigt an, dass der Zeichenvektor zwei Elemente hat, indem er sie separat zitiert. Wenn wir einen Vektor haben, der lang genug ist, um mehrere Zeilen zu überspannen, druckt R den Index des Elements, beginnend mit jeder Zeile, so wie es `[1]` am Anfang der ersten Zeile steht.

```
c("Hello World", "Here I am!", "This next string is really long.")
#[1] "Hello World" "Here I am!"
#[3] "This next string is really long."
```

Das spezielle Verhalten von `print` hängt von der Klasse des Objekts ab, das an die Funktion übergeben wird.

Wenn wir `print` ein Objekt mit einer anderen Klasse aufrufen, z. B. "numerisch" oder "logisch", werden die Anführungszeichen in der Ausgabe weggelassen, um anzuzeigen, dass es sich um ein Objekt handelt, das keine Zeichenklasse ist:

```
print(1)
#[1] 1
print(TRUE)
#[1] TRUE
```

Factor-Objekte werden auf dieselbe Art und Weise wie Zeichenvariablen gedruckt, was häufig zu Mehrdeutigkeiten führt, wenn die Konsolenausgabe zum Anzeigen von Objekten in SO-Fragenkörpern verwendet wird. Es ist selten, `cat` oder `print` außer in einem interaktiven Kontext. Das explizite Aufrufen von `print()` ist besonders selten (es sei denn, Sie wollten die Darstellung der Anführungszeichen unterdrücken oder ein Objekt anzeigen, das von einer Funktion als `invisible` wird), da die Eingabe von `foo` an der Konsole eine Abkürzung für `print(foo)` . Die interaktive Konsole von R ist als REPL (Read-Eval-Print-Loop) bekannt. Die `cat` Funktion wird am besten für spezielle Zwecke (z. B. Schreiben der Ausgabe in eine offene Dateiverbindung) gespeichert. Manchmal wird es in Funktionen verwendet (bei denen Aufrufe von `print()` unterdrückt werden). Die **Verwendung von `cat()` in einer Funktion zum Erzeugen von Ausgaben für die Konsole ist jedoch eine schlechte Praxis** . Die bevorzugte Methode ist `message()` oder `warning()` für Zwischennachrichten. Sie verhalten sich ähnlich wie `cat` , können aber vom Endbenutzer optional unterdrückt werden. Das Endergebnis sollte einfach zurückgegeben werden, damit der Benutzer es ggf. speichern kann.

```
message("hello world")
#hello world
suppressMessages(message("hello world"))
```

## Lesen oder Schreiben einer Dateiverbindung

Nicht immer haben wir die Freiheit, von einem lokalen Systempfad zu lesen oder in ihn zu schreiben. Wenn zum Beispiel R-Code-Streaming-Map-Reduce die Dateiverbindung lesen und beschreiben muss. Es kann auch andere Szenarien geben, in denen man über das lokale System hinausgeht. Mit dem Aufkommen von Cloud und Big Data wird dies immer häufiger. Eine Möglichkeit, dies zu tun, ist in logischer Reihenfolge.

Stellen Sie eine Dateiverbindung zum Lesen mit dem Befehl `file()` ("r" steht für den Lesemodus):

```
conn <- file("/path/example.data", "r") #when file is in local system
conn1 <- file("stdin", "r") #when just standard input/output for files are available
```

Da dies nur eine Dateiverbindung herstellt, können Sie die Daten von diesen Dateiverbindungen wie folgt lesen:

```
line <- readLines(conn, n=1, warn=FALSE)
```

Hier lesen wir die Daten von der Dateiverbindung `conn` Zeile für Zeile als `n=1` . man kann den Wert von `n` (z. B. 10, 20 usw.) ändern, um Datenblöcke für ein schnelleres Lesen zu lesen (10 oder 20 Zeilen Block in einem Durchgang). Um die komplette Datei in einem Schritt zu lesen, setzen Sie `n=-1` .

Nach der Datenverarbeitung oder Modellausführung; Sie können die Ergebnisse mit vielen verschiedenen Befehlen wie `writeLines()`, `cat()` usw. in die Dateiverbindung `writeLines()`, `cat()` die in eine Dateiverbindung schreiben können. Alle diese Befehle nutzen jedoch die zum Schreiben eingerichtete Dateiverbindung. Dies kann mit dem Befehl `file()` folgt durchgeführt werden:

```
conn2 <- file("/path/result.data", "w") #when file is in local system
conn3 <- file("stdout", "w") #when just standard input/output for files are available
```

Dann schreibe die Daten wie folgt:

```
writeLines("text",conn2, sep = "\n")
```

Erfassen Sie die Ausgabe des Betriebssystembefehls

## Funktionen, die einen Zeichenvektor zurückgeben

Base  $\mathbb{R}$  hat zwei Funktionen zum Aufrufen eines Systembefehls. Beide erfordern einen zusätzlichen Parameter, um die Ausgabe des Systembefehls zu erfassen.

```
system("top -a -b -n 1", intern = TRUE)
system2("top", "-a -b -n 1", stdout = TRUE)
```

Beide geben einen Zeichenvektor zurück.

```
[1] "top - 08:52:03 up 70 days, 15:09, 0 users, load average: 0.00, 0.00, 0.00"
[2] "Tasks: 125 total, 1 running, 124 sleeping, 0 stopped, 0 zombie"
[3] "Cpu(s): 0.9%us, 0.3%sy, 0.0%ni, 98.7%id, 0.1%wa, 0.0%hi, 0.0%si, 0.0%st"
[4] "Mem: 12194312k total, 3613292k used, 8581020k free, 216940k buffers"
[5] "Swap: 12582908k total, 2334156k used, 10248752k free, 1682340k cached"
[6] ""
[7] " PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND "
[8] "11300 root 20 0 1278m 375m 3696 S 0.0 3.2 124:40.92 trala "
[9] " 6093 user1 20 0 1817m 269m 1888 S 0.0 2.3 12:17.96 R "
[10] " 4949 user2 20 0 1917m 214m 1888 S 0.0 1.8 11:16.73 R "
```

Zur Veranschaulichung wird der UNIX-Befehl `top -a -b -n 1` verwendet. Dies ist betriebssystemspezifisch und muss möglicherweise geändert werden, um die Beispiele auf Ihrem Computer auszuführen.

Package `devtools` hat die Funktion, einen Systembefehl auszuführen und die Ausgabe ohne zusätzlichen Parameter zu erfassen. Es gibt auch einen Zeichenvektor zurück.

```
devtools::system_output("top", "-a -b -n 1")
```

## Funktionen, die einen Datenrahmen zurückgeben

Die Funktion `fread` im Paket `data.table` ermöglicht das Ausführen eines Shell-Befehls und das

Lesen der Ausgabe wie bei `read.table` . Es gibt eine `data.table` oder einen `data.frame` .

```
fread("top -a -b -n 1", check.names = TRUE)
```

|    | PID   | USER  | PR | NI | VIRT  | RES  | SHR  | S | X.CPU | X.MEM | TIME.     | COMMAND |
|----|-------|-------|----|----|-------|------|------|---|-------|-------|-----------|---------|
| 1: | 11300 | root  | 20 | 0  | 1278m | 375m | 3696 | S | 0     | 3.2   | 124:40.92 | trala   |
| 2: | 6093  | user1 | 20 | 0  | 1817m | 269m | 1888 | S | 0     | 2.3   | 12:18.56  | R       |
| 3: | 4949  | user2 | 20 | 0  | 1917m | 214m | 1888 | S | 0     | 1.8   | 11:17.33  | R       |
| 4: | 7922  | user3 | 20 | 0  | 3094m | 131m | 1892 | S | 0     | 1.1   | 21:04.95  | R       |

Beachten Sie, dass `fread` die oberen 6 Kopfzeilen automatisch übersprungen hat.

Hier wurde der Parameter `check.names = TRUE` hinzugefügt, um `%CPU` , `%MEM` und `TIME+` in syntaktisch gültige Spaltennamen zu konvertieren.

Zeichenfolgen lesen und schreiben online lesen:

<https://riptutorial.com/de/r/topic/5541/zeichenfolgen-lesen-und-schreiben>

# Kapitel 128: Zeitreihen und Prognosen

## Bemerkungen

Prognosen und Zeitreihenanalysen können mit üblichen Funktionen des `stats` wie `glm()` oder einer großen Anzahl von Spezialpaketen behandelt werden. Die [CRAN-Taskansicht](#) für die Zeitreihenanalyse bietet eine detaillierte Auflistung der wichtigsten Pakete nach Themen mit kurzen Beschreibungen.

## Examples

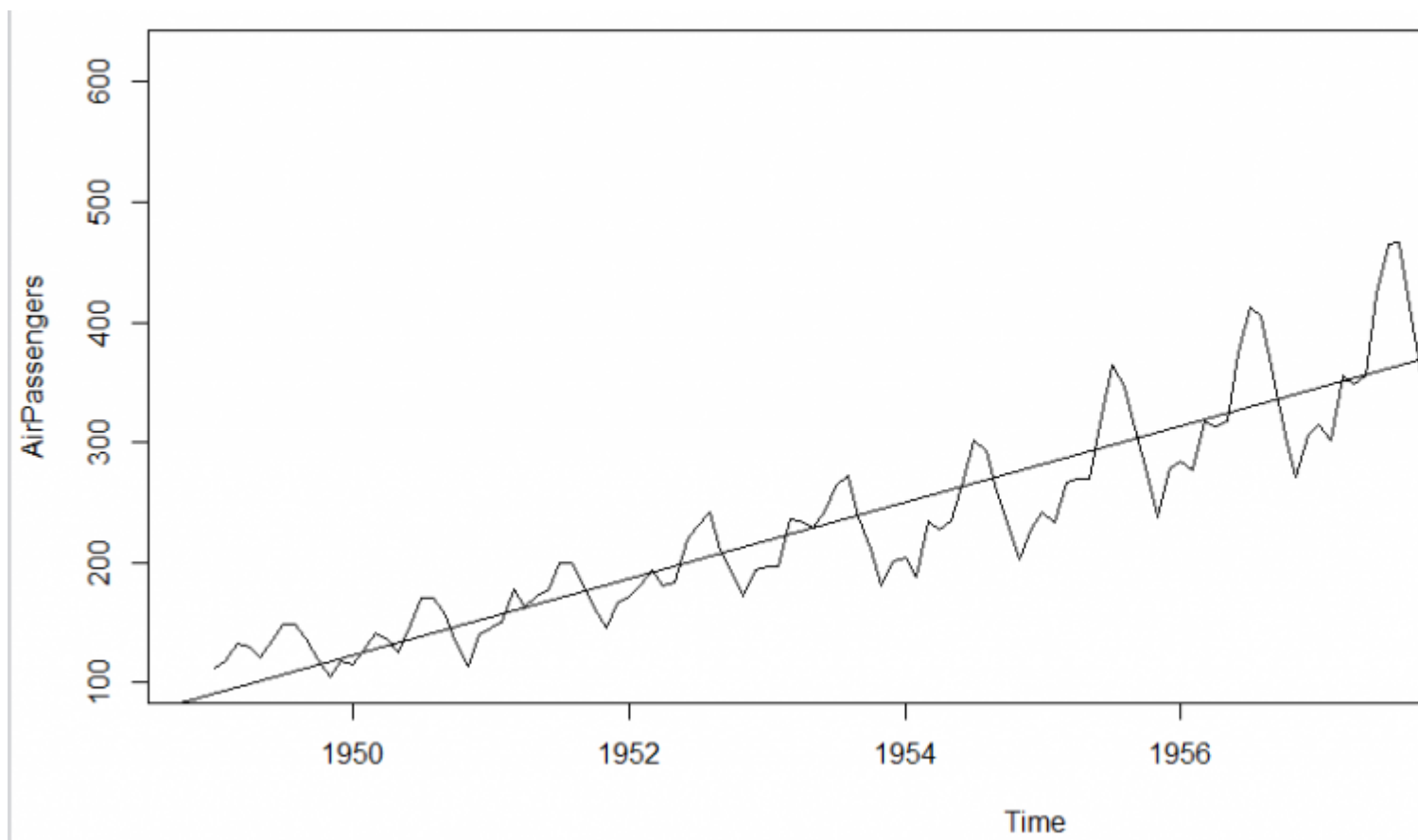
### Explorative Datenanalyse mit Zeitreihendaten

```
data(AirPassengers)
class(AirPassengers)
```

1 "ts"

Im Sinne der Exploratory Data Analysis (EDA) ist es ein guter erster Schritt, eine grafische Darstellung Ihrer Zeitreihendaten zu betrachten:

```
plot(AirPassengers) # plot the raw data
abline(reg=lm(AirPassengers~time(AirPassengers))) # fit a trend line
```

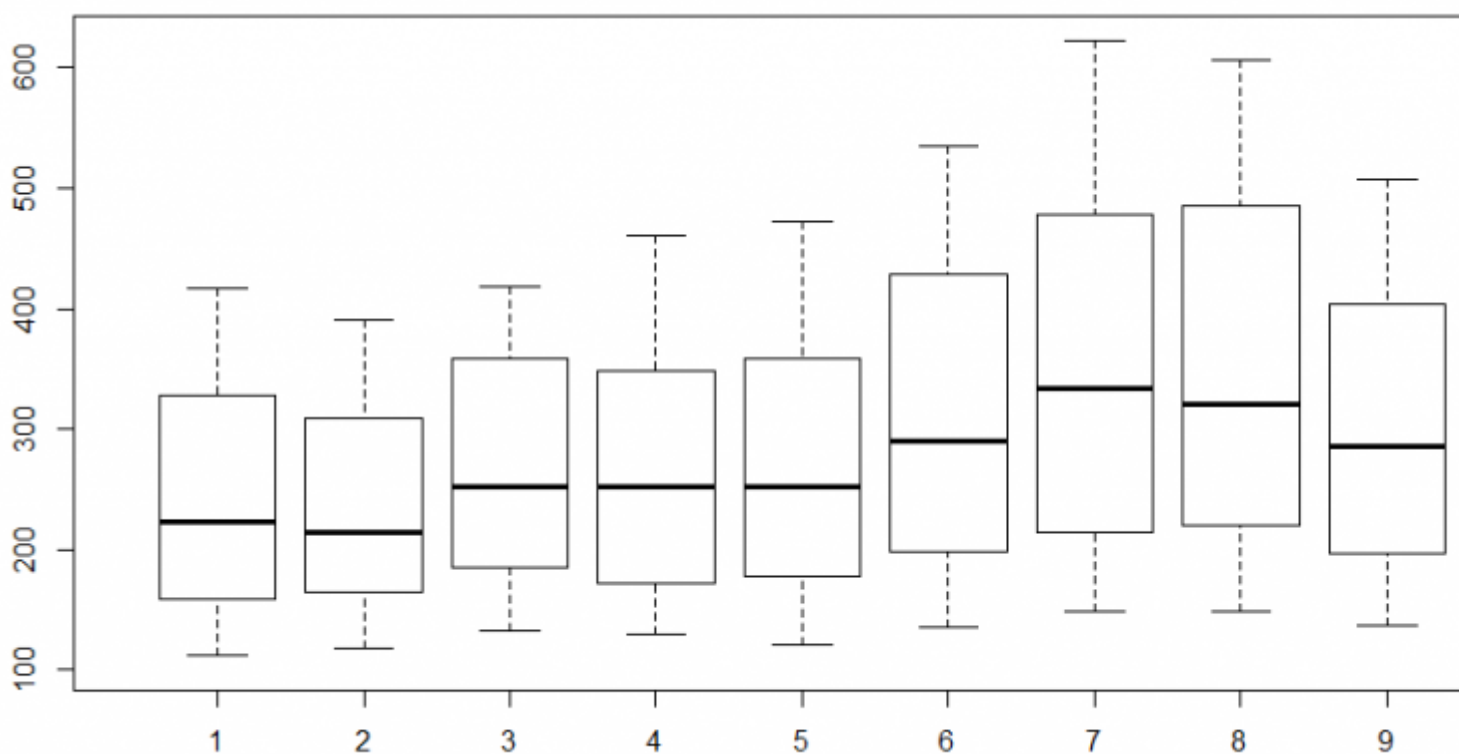


Für weitere EDA untersuchen wir Zyklen über Jahre hinweg:

```
cycle(AirPassengers)
```

|      | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1949 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1950 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1951 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1952 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1953 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1954 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1955 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1956 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1957 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1958 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1959 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |
| 1960 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  |

```
boxplot(AirPassengers~cycle(AirPassengers)) #Box plot across months to explore seasonal effects
```



## Ts-Objekt erstellen

Zeitreihendaten können als `ts` Objekt gespeichert werden. `ts` Objekte enthalten Informationen zur saisonalen Häufigkeit, die von ARIMA-Funktionen verwendet werden. Es ermöglicht auch das Aufrufen von Elementen in der Serie nach Datum mit dem Befehl `window`.

```
#Create a dummy dataset of 100 observations
x <- rnorm(100)
```

```

#Convert this vector to a ts object with 100 annual observations
x <- ts(x, start = c(1900), freq = 1)

#Convert this vector to a ts object with 100 monthly observations starting in July
x <- ts(x, start = c(1900, 7), freq = 12)

#Alternatively, the starting observation can be a number:
x <- ts(x, start = 1900.5, freq = 12)

#Convert this vector to a ts object with 100 daily observations and weekly frequency starting
in the first week of 1900
x <- ts(x, start = c(1900, 1), freq = 7)

#The default plot for a ts object is a line plot
plot(x)

#The window function can call elements or sets of elements by date

#Call the first 4 weeks of 1900
window(x, start = c(1900, 1), end = (1900, 4))

#Call only the 10th week in 1900
window(x, start = c(1900, 10), end = (1900, 10))

#Call all weeks including and after the 10th week of 1900
window(x, start = c(1900, 10))

```

Es ist möglich, `ts` Objekte mit mehreren Serien zu erstellen:

```

#Create a dummy matrix of 3 series with 100 observations each
x <- cbind(rnorm(100), rnorm(100), rnorm(100))

#Create a multi-series ts with annual observation starting in 1900
x <- ts(x, start = 1900, freq = 1)

#R will draw a plot for each series in the object
plot(x)

```

Zeitreihen und Prognosen online lesen: <https://riptutorial.com/de/r/topic/2701/zeitreihen-und-prognosen>

# Kapitel 129: Zufallsforstalgorithmus

## Einführung

RandomForest ist eine Ensemble-Methode zur Klassifizierung oder Regression, die die Wahrscheinlichkeit einer Überanpassung der Daten verringert. Details zur Methode finden Sie im [Wikipedia-Artikel zu Zufallswäldern](#). Die Hauptimplementierung für R befindet sich im randomForest-Paket, es gibt jedoch auch andere Implementierungen. Siehe die [CRAN-Ansicht zum maschinellen Lernen](#).

## Examples

### Grundlegende Beispiele - Klassifizierung und Regression

```
Used for both Classification and Regression examples
library(randomForest)
library(car) ## For the Soils data
data(Soils)

#####
RF Classification Example
set.seed(656) ## for reproducibility
S_RF_Class = randomForest(Gp ~ ., data=Soils[,c(4,6:14)])
Gp_RF = predict(S_RF_Class, Soils[,6:14])
length(which(Gp_RF != Soils$Gp)) ## No Errors

Naive Bayes for comparison
library(e1071)
S_NB = naiveBayes(Soils[,6:14], Soils[,4])
Gp_NB = predict(S_NB, Soils[,6:14], type="class")
length(which(Gp_NB != Soils$Gp)) ## 6 Errors
```

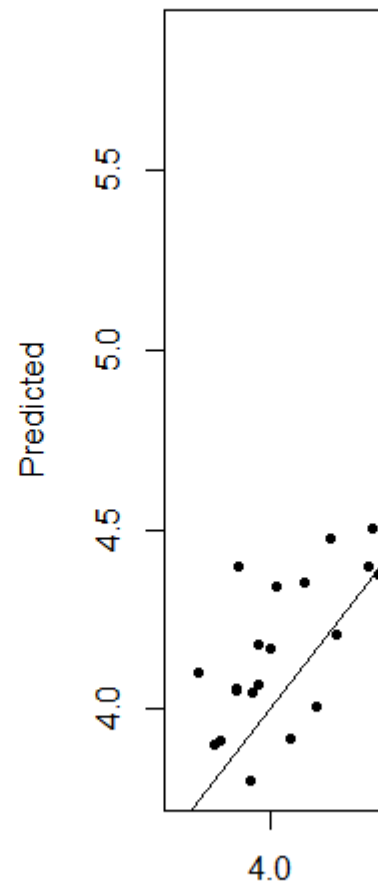
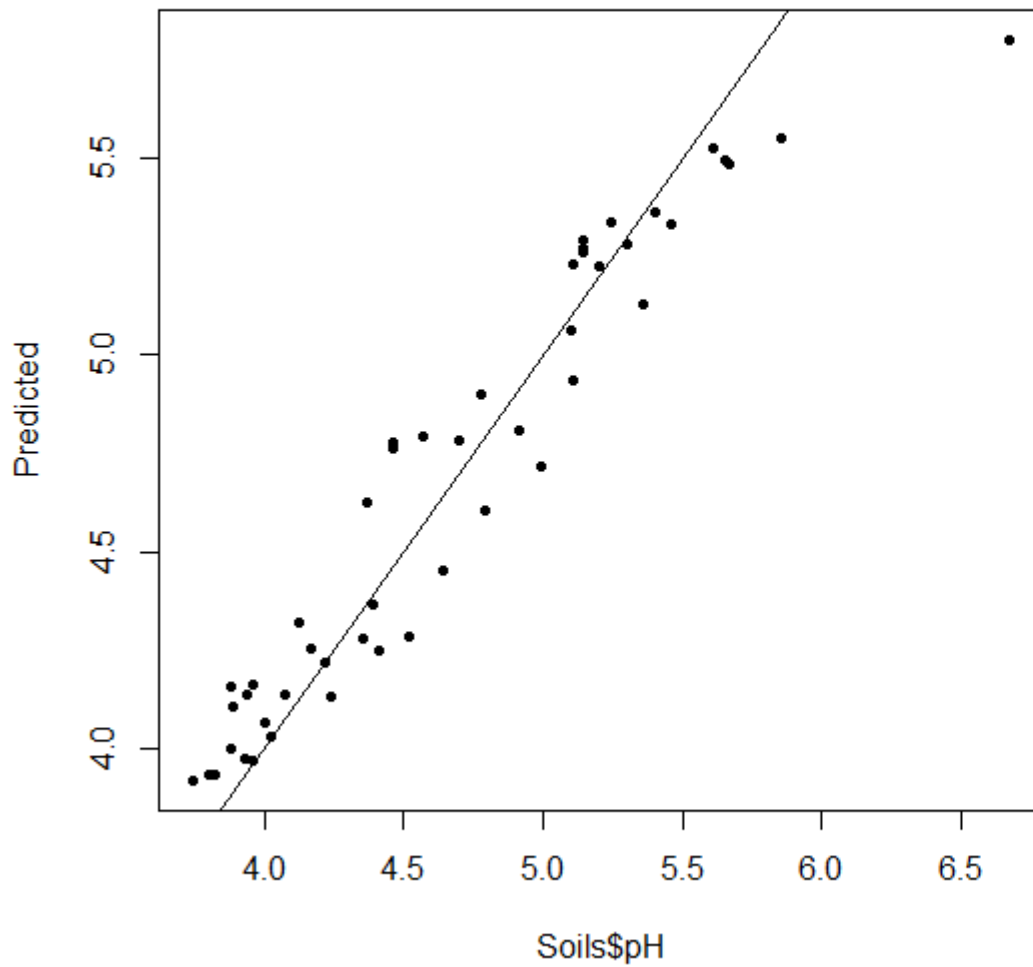
Dieses Beispiel hat die Trainingsdaten getestet, zeigt jedoch, dass RF sehr gute Modelle machen kann.

```
#####
RF Regression Example
set.seed(656) ## for reproducibility
S_RF_Reg = randomForest(pH ~ ., data=Soils[,6:14])
pH_RF = predict(S_RF_Reg, Soils[,6:14])

Compare Predictions with Actual values for RF and Linear Model
S_LM = lm(pH ~ ., data=Soils[,6:14])
pH_LM = predict(S_LM, Soils[,6:14])
par(mfrow=c(1,2))
plot(Soils$pH, pH_RF, pch=20, ylab="Predicted", main="Random Forest")
abline(0,1)
plot(Soils$pH, pH_LM, pch=20, ylab="Predicted", main="Linear Model")
abline(0,1)
```



## Random Forest



Zufallsforstalgorithmus online lesen: <https://riptutorial.com/de/r/topic/8088/zufallsforstalgorithmus>

# Kapitel 130: Zufallszahlengenerator

## Examples

### Zufällige Permutationen

Um zufällige Permutation von 5 Zahlen zu erzeugen:

```
sample(5)
[1] 4 5 3 1 2
```

So erzeugen Sie eine zufällige Permutation eines Vektors:

```
sample(10:15)
[1] 11 15 12 10 14 13
```

Man könnte auch das Paket `pracma`

```
randperm(a, k)
Generates one random permutation of k of the elements a, if a is a vector,
or of 1:a if a is a single integer.
a: integer or numeric vector of some length n.
k: integer, smaller as a or length(a).

Examples
library(pracma)
randperm(1:10, 3)
[1] 3 7 9

randperm(10, 10)
[1] 4 5 10 8 2 7 6 9 3 1

randperm(seq(2, 10, by=2))
[1] 6 4 10 2 8
```

### Reproduzierbarkeit des Zufallszahlengenerators

Wenn erwartet wird, dass jemand einen R-Code mit zufälligen Elementen reproduziert, ist die Funktion `set.seed()` sehr praktisch. Beispielsweise erzeugen diese beiden Zeilen immer eine andere Ausgabe (da dies der ganze Punkt von Zufallszahlengeneratoren ist):

```
> sample(1:10,5)
[1] 6 9 2 7 10
> sample(1:10,5)
[1] 7 6 1 2 10
```

Diese beiden erzeugen auch unterschiedliche Ausgaben:

```
> rnorm(5)
```

```
[1] 0.4874291 0.7383247 0.5757814 -0.3053884 1.5117812
> rnorm(5)
[1] 0.38984324 -0.62124058 -2.21469989 1.12493092 -0.04493361
```

Wenn wir jedoch den Samen in beiden Fällen auf einen identischen Wert setzen (die meisten Leute verwenden zur Vereinfachung 1), erhalten wir zwei identische Beispiele:

```
> set.seed(1)
> sample(letters,2)
[1] "g" "j"
> set.seed(1)
> sample(letters,2)
[1] "g" "j"
```

und das gleiche mit, sagen wir, `rexp()` zeichnet:

```
> set.seed(1)
> rexp(5)
[1] 0.7551818 1.1816428 0.1457067 0.1397953 0.4360686
> set.seed(1)
> rexp(5)
[1] 0.7551818 1.1816428 0.1457067 0.1397953 0.4360686
```

## Zufallszahlen mit verschiedenen Dichtefunktionen erzeugen

Nachfolgend finden Sie Beispiele für die Generierung von 5 Zufallszahlen mit verschiedenen Wahrscheinlichkeitsverteilungen.

## Gleichmäßige Verteilung zwischen 0 und 10

```
runif(5, min=0, max=10)
[1] 2.1724399 8.9209930 6.1969249 9.3303321 2.4054102
```

## Normalverteilung mit 0 Mittelwert und Standardabweichung von 1

```
rnorm(5, mean=0, sd=1)
[1] -0.97414402 -0.85722281 -0.08555494 -0.37444299 1.20032409
```

## Binomialverteilung mit 10 Versuchen und einer Erfolgswahrscheinlichkeit von 0,5

```
rbinom(5, size=10, prob=0.5)
[1] 4 3 5 2 3
```

## Geometrische Verteilung mit 0,2 Erfolgswahrscheinlichkeit

```
rgeom(5, prob=0.2)
[1] 14 8 11 1 3
```

## Hypergeometrische Verteilung mit 3 weißen Kugeln, 10 schwarzen Kugeln und 5 Unentschieden

```
rhyper(5, m=3, n=10, k=5)
[1] 2 0 1 1 1
```

## Negative Binomialverteilung mit 10 Versuchen und einer Erfolgswahrscheinlichkeit von 0,8

```
rnbinom(5, size=10, prob=0.8)
[1] 3 1 3 4 2
```

## Poissonverteilung mit Mittelwert und Varianz (Lambda) von 2

```
rpois(5, lambda=2)
[1] 2 1 2 3 4
```

## Exponentialverteilung mit der Rate von 1,5

```
rexp(5, rate=1.5)
[1] 1.8993303 0.4799358 0.5578280 1.5630711 0.6228000
```

## Logistische Verteilung mit 0 Standort und Skala von 1

```
rlogis(5, location=0, scale=1)
[1] 0.9498992 -1.0287433 -0.4192311 0.7028510 -1.2095458
```

## Chi-Quadrat- Verteilung mit 15 Freiheitsgraden

```
rchisq(5, df=15)
[1] 14.89209 19.36947 10.27745 19.48376 23.32898
```

## Beta- Verteilung mit den Formparametern $a = 1$ und $b = 0,5$

```
rbeta(5, shape1=1, shape2=0.5)
[1] 0.1670306 0.5321586 0.9869520 0.9548993 0.9999737
```

## Gamma- Verteilung mit Formparameter 3 und Skala = 0,5

```
rgamma(5, shape=3, scale=0.5)
[1] 2.2445984 0.7934152 3.2366673 2.2897537 0.8573059
```

## Cauchy- Verteilung mit 0 Standort und Skala von 1

```
rcauchy(5, location=0, scale=1)
[1] -0.01285116 -0.38918446 8.71016696 10.60293284 -0.68017185
```

## Log-Normalverteilung mit 0-Mittelwert und Standardabweichung von 1 (auf Log-Skala)

```
rlnorm(5, meanlog=0, sdlog=1)
[1] 0.8725009 2.9433779 0.3329107 2.5976206 2.8171894
```

## Weibullverteilung mit Formparameter 0,5 und Skala von 1

```
rweibull(5, shape=0.5, scale=1)
[1] 0.337599112 1.307774557 7.233985075 5.840429942 0.005751181
```

## Wilcoxon- Verteilung mit 10 Beobachtungen in der ersten Probe und 20 in der zweiten.

```
rwilcox(5, 10, 20)
[1] 111 88 93 100 124
```

## Multinomialverteilung mit 5 Objekt und 3 Boxen mit den angegebenen Wahrscheinlichkeiten

```
rmultinom(5, size=5, prob=c(0.1,0.1,0.8))
 [,1] [,2] [,3] [,4] [,5]
[1,] 0 0 1 1 0
[2,] 2 0 1 1 0
[3,] 3 5 3 3 5
```

Zufallszahlengenerator online lesen: <https://riptutorial.com/de/r/topic/1578/zufallszahlengenerator>

---

# Kapitel 131: Zwang

## Einführung

Zwang geschieht in R, wenn der Typ der Objekte während der Berechnung entweder implizit oder unter Verwendung von Funktionen für expliziten Zwang geändert wird (wie z. B. numerisch, als.data.frame usw.).

## Examples

### Impliziter Zwang

Zwang geschieht bei Datentypen in R oft implizit, sodass die Daten alle Werte aufnehmen können. Zum Beispiel,

```
x = 1:3
x
[1] 1 2 3
typeof(x)
#[1] "integer"

x[2] = "hi"
x
#[1] "1" "hi" "3"
typeof(x)
#[1] "character"
```

Beachten Sie, dass `x` vom Typ `integer`. Bei der Zuweisung von `x[2] = "hi"` wurden alle Elemente von `x` in `character` da Vektoren in R nur Daten eines einzigen Typs enthalten können.

Zwang online lesen: <https://riptutorial.com/de/r/topic/9793/zwang>

# Credits

| S. No | Kapitel                                             | Contributors                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | Erste Schritte mit R Language                       | <a href="#">42-</a> , <a href="#">akraf</a> , <a href="#">Ale</a> , <a href="#">Andrea Cirillo</a> , <a href="#">Andrew Brēza</a> , <a href="#">Axeman</a> , <a href="#">Community</a> , <a href="#">Craig Vermeer</a> , <a href="#">d.b</a> , <a href="#">dotancohen</a> , <a href="#">Francesco Dondi</a> , <a href="#">Frank</a> , <a href="#">G5W</a> , <a href="#">George Bonebright</a> , <a href="#">GForce</a> , <a href="#">Giorgos K</a> , <a href="#">Gregor</a> , <a href="#">H. Pauwelyn</a> , <a href="#">kartoffelsalat</a> , <a href="#">kdopen</a> , <a href="#">Konrad Rudolph</a> , <a href="#">L.V.Rao</a> , <a href="#">Imckeogh</a> , <a href="#">Lovy</a> , <a href="#">Matt</a> , <a href="#">mnoronha</a> , <a href="#">pitosalas</a> , <a href="#">polka</a> , <a href="#">Rahul Saini</a> , <a href="#">RetractedAndRetired</a> , <a href="#">russellpierce</a> , <a href="#">Steve_Corrin</a> , <a href="#">theArun</a> , <a href="#">Thomas</a> , <a href="#">torina</a> , <a href="#">user2100721</a> , <a href="#">while</a> |
| 2     | * wende die Familie der Funktionen (Funktionale) an | <a href="#">Benjamin</a> , <a href="#">FisherDisinformation</a> , <a href="#">Gavin Simpson</a> , <a href="#">jcb</a> , <a href="#">Karolis Koncevičius</a> , <a href="#">kneijenhuijs</a> , <a href="#">Maximilian Kohl</a> , <a href="#">nrussell</a> , <a href="#">omar</a> , <a href="#">Robert</a> , <a href="#">seasmith</a> , <a href="#">zacdav</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 3     | .Rprofile                                           | <a href="#">42-</a> , <a href="#">Dirk Eddelbuettel</a> , <a href="#">ikashnitsky</a> , <a href="#">Karolis Koncevičius</a> , <a href="#">Nikos Alexandris</a> , <a href="#">Stedy</a> , <a href="#">Thomas</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 4     | Aggregation von Datenrahmen                         | <a href="#">Florian</a> , <a href="#">Frank</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 5     | Aktualisieren von R und der Paketbibliothek         | <a href="#">Eric Lecoutre</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 6     | Analysieren Sie Tweets mit R                        | <a href="#">Umberto</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 7     | Ändern von Strings durch Ersetzen                   | <a href="#">Alex</a> , <a href="#">David Leal</a> , <a href="#">Frank</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 8     | ANOVA                                               | <a href="#">Ben Bolker</a> , <a href="#">DataTx</a> , <a href="#">kneijenhuijs</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9     | Arima-Modelle                                       | <a href="#">Andrew Bryk</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 10    | Ausdruck: parse + eval                              | <a href="#">YCR</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 11    | Balkendiagramm                                      | <a href="#">L.V.Rao</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 12    | Basisplot                                           | <a href="#">42-</a> , <a href="#">Alexey Shiklomanov</a> , <a href="#">catastrophic-failure</a> , <a href="#">FisherDisinformation</a> , <a href="#">Frank</a> , <a href="#">Giorgos K</a> , <a href="#">K.Daisey</a> , <a href="#">maRtin</a> , <a href="#">MichaelChirico</a> , <a href="#">RamenChef</a> , <a href="#">Robert</a> , <a href="#">symbolrush</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 13    | Benutzereingabe abrufen                             | <a href="#">Ashish</a> , <a href="#">DeveauP</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

|    |                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----|--------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14 | Beschleunigen Sie Code, der stark zu vektorisieren ist | <a href="#">egnha</a> , <a href="#">josliber</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 15 | Bewährte Methoden für die Vektorisierung von R-Codes   | <a href="#">Axeman</a> , <a href="#">David Arenburg</a> , <a href="#">snaut</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 16 | Bibliographie in RMD                                   | <a href="#">J_F</a> , <a href="#">RamenChef</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 17 | Box-Plot                                               | <a href="#">Carlos Cinelli</a> , <a href="#">Christophe D.</a> , <a href="#">Karolis Koncevičius</a> , <a href="#">L.V.Rao</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 18 | Caret                                                  | <a href="#">highBandWidth</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 19 | Code-Profiling                                         | <a href="#">Ben Bolker</a> , <a href="#">Glen Moutrie</a> , <a href="#">Jav</a> , <a href="#">SymbolixAU</a> , <a href="#">USER_1</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 20 | Daten reinigen                                         | <a href="#">Derek Corcoran</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21 | Daten zwischen langen und breiten Formularen umformen  | <a href="#">Charmgoggles</a> , <a href="#">David Arenburg</a> , <a href="#">demonplus</a> , <a href="#">Frank</a> , <a href="#">Jeromy Anglim</a> , <a href="#">kneijenhuijs</a> , <a href="#">Imo</a> , <a href="#">Steve_Corrin</a> , <a href="#">SymbolixAU</a> , <a href="#">takje</a> , <a href="#">user2100721</a> , <a href="#">zx8754</a>                                                                                                                                                                                                                                                                                                                                                                   |
| 22 | Datenerfassung                                         | <a href="#">ikashnitsky</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 23 | Datenrahmen                                            | <a href="#">Alex</a> , <a href="#">Andrea Ianni</a> , <a href="#">Batanichek</a> , <a href="#">Carlos Cinelli</a> , <a href="#">Christophe D.</a> , <a href="#">DataTx</a> , <a href="#">David Arenburg</a> , <a href="#">David Robinson</a> , <a href="#">dayne</a> , <a href="#">Frank</a> , <a href="#">Gregor</a> , <a href="#">Hack-R</a> , <a href="#">kaksat</a> , <a href="#">R. Schifini</a> , <a href="#">scoa</a> , <a href="#">Sumedh</a> , <a href="#">Thomas</a> , <a href="#">Tomás Barcellos</a> , <a href="#">user2100721</a>                                                                                                                                                                      |
| 24 | Datentabelle                                           | <a href="#">akrun</a> , <a href="#">Allen Wang</a> , <a href="#">bartektartanus</a> , <a href="#">cderv</a> , <a href="#">David</a> , <a href="#">David Arenburg</a> , <a href="#">Dean MacGregor</a> , <a href="#">Eric Lecoutre</a> , <a href="#">Frank</a> , <a href="#">Jaap</a> , <a href="#">jogo</a> , <a href="#">L Co</a> , <a href="#">leogama</a> , <a href="#">Mallick Hossain</a> , <a href="#">micstr</a> , <a href="#">Nathan Werth</a> , <a href="#">oshun</a> , <a href="#">Peter Humburg</a> , <a href="#">Sowmya S. Manian</a> , <a href="#">stanekam</a> , <a href="#">Steve_Corrin</a> , <a href="#">Sumedh</a> , <a href="#">Tensibai</a> , <a href="#">user2100721</a> , <a href="#">Uwe</a> |
| 25 | Datum und Uhrzeit                                      | <a href="#">AkselA</a> , <a href="#">alistaire</a> , <a href="#">Angelo</a> , <a href="#">coatless</a> , <a href="#">David Leal</a> , <a href="#">Dean MacGregor</a> , <a href="#">Frank</a> , <a href="#">kneijenhuijs</a> , <a href="#">MichaelChirico</a> , <a href="#">scoa</a> , <a href="#">SymbolixAU</a> , <a href="#">takje</a> , <a href="#">theArun</a> , <a href="#">thelatemail</a>                                                                                                                                                                                                                                                                                                                    |
| 26 | Datum-Zeit-Klassen (POSIXct und POSIXlt)               | <a href="#">AkselA</a> , <a href="#">alistaire</a> , <a href="#">coatless</a> , <a href="#">Frank</a> , <a href="#">MichaelChirico</a> , <a href="#">SymbolixAU</a> , <a href="#">thelatemail</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 27 | Debuggen                                               | <a href="#">James Elderfield</a> , <a href="#">russellpierce</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 28 | Die Charakterklasse                                    | <a href="#">Frank</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 29 | Die Date-Klasse                                        | <a href="#">alistaire</a> , <a href="#">coatless</a> , <a href="#">Frank</a> , <a href="#">L.V.Rao</a> , <a href="#">MichaelChirico</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



|    |                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----|-----------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30 | Die logische Klasse                                             | <a href="#">42-</a> , <a href="#">Frank</a> , <a href="#">Gregor</a> , <a href="#">L.V.Rao</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                         |
| 31 | dplyr                                                           | <a href="#">4444</a> , <a href="#">Alihan Zihna</a> , <a href="#">ikashnitsky</a> , <a href="#">Robert</a> , <a href="#">skoh</a> , <a href="#">Sumedh</a> , <a href="#">theArun</a>                                                                                                                                                                                                                                                                                                  |
| 32 | Durchführen eines Permutationstests                             | <a href="#">Stephen Leppik</a> , <a href="#">tenCupMaximum</a>                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 33 | E / A für das Binärformat von R                                 | <a href="#">Frank</a> , <a href="#">ikashnitsky</a> , <a href="#">Mario</a> , <a href="#">russellpierce</a> , <a href="#">zacdav</a> , <a href="#">zx8754</a>                                                                                                                                                                                                                                                                                                                         |
| 34 | E / A für Datenbanktabellen                                     | <a href="#">Frank</a> , <a href="#">JHowIX</a> , <a href="#">SommerEngineering</a>                                                                                                                                                                                                                                                                                                                                                                                                    |
| 35 | E / A für Fremdtabellen (Excel, SAS, SPSS, Stata)               | <a href="#">42-</a> , <a href="#">Alex</a> , <a href="#">alistaire</a> , <a href="#">Andrea Cirillo</a> , <a href="#">Carlos Cinelli</a> , <a href="#">Charmgoggles</a> , <a href="#">Crops</a> , <a href="#">Frank</a> , <a href="#">Jaap</a> , <a href="#">Jeromy Anglim</a> , <a href="#">kaksat</a> , <a href="#">Ken S.</a> , <a href="#">kitman0804</a> , <a href="#">Imo</a> , <a href="#">Miha</a> , <a href="#">Parfait</a> , <a href="#">polka</a> , <a href="#">Thomas</a> |
| 36 | E / A für geographische Daten (Shapefiles usw.)                 | <a href="#">Alex</a> , <a href="#">Frank</a> , <a href="#">ikashnitsky</a>                                                                                                                                                                                                                                                                                                                                                                                                            |
| 37 | E / A für Rasterbilder                                          | <a href="#">Frank</a> , <a href="#">loki</a>                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 38 | Einführung in geographische Karten                              | <a href="#">4444</a> , <a href="#">AkselA</a> , <a href="#">alistaire</a> , <a href="#">beetroot</a> , <a href="#">Carson</a> , <a href="#">Frank</a> , <a href="#">Hack-R</a> , <a href="#">HypnoGenX</a> , <a href="#">Robert</a> , <a href="#">russellpierce</a> , <a href="#">SymbolixAU</a> , <a href="#">symbolrush</a>                                                                                                                                                         |
| 39 | Eingabe und Ausgabe                                             | <a href="#">Frank</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 40 | Erinnerung an Beispiele                                         | <a href="#">Lovy</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 41 | Erstellen von Berichten mit RMarkdown                           | <a href="#">ikashnitsky</a> , <a href="#">Karolis Koncevičius</a> , <a href="#">Martin Schmelzer</a>                                                                                                                                                                                                                                                                                                                                                                                  |
| 42 | Extrahieren und Auflisten von Dateien in komprimierten Archiven | <a href="#">catastrophic-failure</a> , <a href="#">Jeff</a>                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 43 | Faktoren                                                        | <a href="#">42-</a> , <a href="#">Benjamin</a> , <a href="#">dash2</a> , <a href="#">Frank</a> , <a href="#">Gavin Simpson</a> , <a href="#">JulioSergio</a> , <a href="#">kneijenhuijs</a> , <a href="#">Nathan Werth</a> , <a href="#">omar</a> , <a href="#">Rich Scriven</a> , <a href="#">Robert</a> , <a href="#">Steve_Corrin</a>                                                                                                                                              |
| 44 | Farbschemata für Grafiken                                       | <a href="#">ikashnitsky</a> , <a href="#">munirbe</a>                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 45 | Fehlende Werte                                                  | <a href="#">Amit Kohli</a> , <a href="#">Artem Klevtsov</a> , <a href="#">Axeman</a> , <a href="#">Eric Lecoutre</a> , <a href="#">Frank</a> , <a href="#">Gregor</a> , <a href="#">Jaap</a> , <a href="#">kitman0804</a> , <a href="#">Imo</a> , <a href="#">seasmith</a> , <a href="#">Steve_Corrin</a> , <a href="#">theArun</a> , <a href="#">user2100721</a>                                                                                                                     |
| 46 | Fehlertoleranter / belastbarer Code                             | <a href="#">Rappster</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|    |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----|---------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 47 | Formel                                                              | <a href="#">42-</a> , <a href="#">Axeman</a> , <a href="#">Qaswed</a> , <a href="#">Sathish</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 48 | Fourier-Serie und Transformationen                                  | <a href="#">Hack-R</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 49 | Funktionale Programmierung                                          | <a href="#">Karolis Koncevičius</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 50 | Funktionsauswahl in R - Entfernen überflüssiger Funktionen          | <a href="#">Joy</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 51 | Generalisierte lineare Modelle                                      | <a href="#">Ben Bolker</a> , <a href="#">YCR</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 52 | ggplot2                                                             | <a href="#">akraf</a> , <a href="#">Alex</a> , <a href="#">alistaire</a> , <a href="#">Andrea Cirillo</a> , <a href="#">Artem Klevtsov</a> , <a href="#">Axeman</a> , <a href="#">baptiste</a> , <a href="#">blmoore</a> , <a href="#">Boern</a> , <a href="#">gitblame</a> , <a href="#">ikashnitsky</a> , <a href="#">Jaap</a> , <a href="#">jmax</a> , <a href="#">loki</a> , <a href="#">Matt</a> , <a href="#">Mine Cetinkaya-Rundel</a> , <a href="#">Paolo</a> , <a href="#">smci</a> , <a href="#">Steve_Corrin</a> , <a href="#">Sumedh</a> , <a href="#">Taylor Ostberg</a> , <a href="#">theArun</a> , <a href="#">void</a> , <a href="#">YCR</a> , <a href="#">Yun Ching</a> |
| 53 | Glänzend                                                            | <a href="#">alistaire</a> , <a href="#">CClaire</a> , <a href="#">Christophe D.</a> , <a href="#">JvH</a> , <a href="#">russellpierce</a> , <a href="#">SymbolixAU</a> , <a href="#">tuomastik</a> , <a href="#">zx8754</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 54 | GPU-beschleunigtes Computing                                        | <a href="#">cdeterman</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 55 | Hashmaps                                                            | <a href="#">nrussell</a> , <a href="#">russellpierce</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 56 | Heatmap und Heatmap.2                                               | <a href="#">AndreyAkinshin</a> , <a href="#">Nanami</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 57 | Hierarchische lineare Modellierung                                  | <a href="#">Ben Bolker</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 58 | Hierarchisches Clustering mit hclust                                | <a href="#">Frank</a> , <a href="#">G5W</a> , <a href="#">Tal Galili</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 59 | Implementieren Sie das State Machine Pattern mithilfe der S4-Klasse | <a href="#">David Leal</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 60 | JSON                                                                | <a href="#">SymbolixAU</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 61 | Klassen                                                             | <a href="#">42-</a> , <a href="#">AkselA</a> , <a href="#">David Heckmann</a> , <a href="#">dayne</a> , <a href="#">Frank</a> , <a href="#">Gregor</a> , <a href="#">Jaap</a> , <a href="#">kneijenhuijs</a> , <a href="#">L.V.Rao</a> , <a href="#">Nathan Werth</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                     |
| 62 | Kombinatorik                                                        | <a href="#">Frank</a> , <a href="#">Karolis Koncevičius</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 63 | Kontrollieren Sie Flussstrukturen                                   | <a href="#">Benjamin</a> , <a href="#">David Arenburg</a> , <a href="#">nrussell</a> , <a href="#">Robert</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|    |                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----|--------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 64 | Lauf längencodierung                                                     | <a href="#">Frank</a> , <a href="#">josliber</a> , <a href="#">Psidom</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 65 | Lesen und Schreiben von Tabellendaten in Klartextdateien (CSV, TSV usw.) | <a href="#">a.powell</a> , <a href="#">Aaghaz Hussain</a> , <a href="#">abhiieor</a> , <a href="#">Alex</a> , <a href="#">alistaire</a> , <a href="#">Andrea Cirillo</a> , <a href="#">bartektartanus</a> , <a href="#">Carl Witthoft</a> , <a href="#">Carlos Cinelli</a> , <a href="#">catastrophic-failure</a> , <a href="#">cdrini</a> , <a href="#">Charmgoggles</a> , <a href="#">Crops</a> , <a href="#">DaveRGP</a> , <a href="#">David Arenburg</a> , <a href="#">Dawny33</a> , <a href="#">Derwin McGeary</a> , <a href="#">EDi</a> , <a href="#">Eric Lecoutre</a> , <a href="#">FoldedChromatin</a> , <a href="#">Frank</a> , <a href="#">Gavin Simpson</a> , <a href="#">gitblame</a> , <a href="#">Hairizuan Noorazman</a> , <a href="#">herbaman</a> , <a href="#">ikashnitsky</a> , <a href="#">Jaap</a> , <a href="#">Jeromy Anglim</a> , <a href="#">JHowlX</a> , <a href="#">joeyreid</a> , <a href="#">Jordan Kassof</a> , <a href="#">K.Daisey</a> , <a href="#">kitman0804</a> , <a href="#">kneijenhuijs</a> , <a href="#">Imo</a> , <a href="#">loki</a> , <a href="#">Miha</a> , <a href="#">PAC</a> , <a href="#">polka</a> , <a href="#">russellpierce</a> , <a href="#">Sam Firke</a> , <a href="#">stats-hb</a> , <a href="#">Thomas</a> , <a href="#">Uwe</a> , <a href="#">zacdav</a> , <a href="#">zelite</a> , <a href="#">zx8754</a> |
| 66 | Lineare Modelle (Regression)                                             | <a href="#">Amstell</a> , <a href="#">Ben Bolker</a> , <a href="#">Carl</a> , <a href="#">Carlos Cinelli</a> , <a href="#">David Robinson</a> , <a href="#">fortune_p</a> , <a href="#">Frank</a> , <a href="#">highBandWidth</a> , <a href="#">ikashnitsky</a> , <a href="#">jaySf</a> , <a href="#">Robert</a> , <a href="#">russellpierce</a> , <a href="#">thelatemail</a> , <a href="#">USER_1</a> , <a href="#">WAF</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 67 | Listen                                                                   | <a href="#">Andrea Ianni</a> , <a href="#">BarkleyBG</a> , <a href="#">dayne</a> , <a href="#">Frank</a> , <a href="#">Hack-R</a> , <a href="#">Hairizuan Noorazman</a> , <a href="#">Peter Humburg</a> , <a href="#">RamenChef</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 68 | Lösen von ODEs in R                                                      | <a href="#">J_F</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 69 | Lubridat                                                                 | <a href="#">alistaire</a> , <a href="#">Angelo</a> , <a href="#">Frank</a> , <a href="#">gitblame</a> , <a href="#">Hendrik</a> , <a href="#">scoa</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 70 | Maschinelles lernen                                                      | <a href="#">loki</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 71 | Matrizen                                                                 | <a href="#">dayne</a> , <a href="#">Frank</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 72 | Meta: Dokumentationsrichtlinien                                          | <a href="#">Frank</a> , <a href="#">Gregor</a> , <a href="#">Stephen Leppik</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 73 | Musterabgleich und Ersatz                                                | <a href="#">Abdou</a> , <a href="#">Alex</a> , <a href="#">Artem Klevtsov</a> , <a href="#">David Arenburg</a> , <a href="#">David Leal</a> , <a href="#">Frank</a> , <a href="#">Gavin Simpson</a> , <a href="#">Jaap</a> , <a href="#">NWaters</a> , <a href="#">R. Schifini</a> , <a href="#">SommerEngineering</a> , <a href="#">Steve_Corrin</a> , <a href="#">Tensibai</a> , <a href="#">thelatemail</a> , <a href="#">user2100721</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 74 | Netzwerkanalyse mit dem igraph-Paket                                     | <a href="#">Boysenb3rry</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 75 | Nichtstandardmäßige Bewertung und Standardbewertung                      | <a href="#">PAC</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 76 | Numerische Klassen und Speichermodi                                      | <a href="#">Frank</a> , <a href="#">Steve_Corrin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 77 | Objektorientierte Programmierung in R                                    | <a href="#">Jon Ericson</a> , <a href="#">rcorty</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|    |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 78 | Operationen einstellen               | <a href="#">DeveauP</a> , <a href="#">FisherDisinformation</a> , <a href="#">Frank</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 79 | ordentlich                           | <a href="#">David Robinson</a> , <a href="#">egnha</a> , <a href="#">Frank</a> , <a href="#">ikashnitsky</a> , <a href="#">RamenChef</a> , <a href="#">Sumedh</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 80 | Pakete installieren                  | <a href="#">Aaghaz Hussain</a> , <a href="#">akraf</a> , <a href="#">alko989</a> , <a href="#">Andrew Brēza</a> , <a href="#">Artem Klevtsov</a> , <a href="#">Arun Balakrishnan</a> , <a href="#">Christophe D.</a> , <a href="#">CL.</a> , <a href="#">Frank</a> , <a href="#">gitblame</a> , <a href="#">Hack-R</a> , <a href="#">hongsy</a> , <a href="#">Jaap</a> , <a href="#">kaksat</a> , <a href="#">kneijenhuijs</a> , <a href="#">Imckeogh</a> , <a href="#">loki</a> , <a href="#">Marc Brinkmann</a> , <a href="#">Miha</a> , <a href="#">Peter Humburg</a> , <a href="#">Pragyaditya Das</a> , <a href="#">Raj Padmanabhan</a> , <a href="#">seasmith</a> , <a href="#">SymbolixAU</a> , <a href="#">theArun</a> , <a href="#">user890739</a> , <a href="#">xamgore</a> , <a href="#">zx8754</a> |
| 81 | Pakete mit devtools erstellen        | <a href="#">Frank</a> , <a href="#">Lovy</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 82 | Pakete prüfen                        | <a href="#">Frank</a> , <a href="#">Sowmya S. Manian</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 83 | Parallelverarbeitung                 | <a href="#">Artem Klevtsov</a> , <a href="#">jameselmore</a> , <a href="#">K.Daisey</a> , <a href="#">Imo</a> , <a href="#">loki</a> , <a href="#">russellpierce</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 84 | Pivot und Unpivot mit data.table     | <a href="#">Sun Bee</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 85 | R in LaTeX mit Knitr                 | <a href="#">JHowIX</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 86 | R Markdown-Notizbücher (von RStudio) | <a href="#">dmail</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 87 | Randomisierung                       | <a href="#">TARehman</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 88 | Raster- und Bildanalyse              | <a href="#">Frank</a> , <a href="#">loki</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 89 | räumliche Analyse                    | <a href="#">beetroot</a> , <a href="#">ikashnitsky</a> , <a href="#">loki</a> , <a href="#">maRtin</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 90 | Rcpp                                 | <a href="#">Artem Klevtsov</a> , <a href="#">coatless</a> , <a href="#">Dirk Eddebuettel</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 91 | Rechenzeichen                        | <a href="#">Batanichek</a> , <a href="#">FisherDisinformation</a> , <a href="#">Matt Sandgren</a> , <a href="#">Robert</a> , <a href="#">russellpierce</a> , <a href="#">Tensibai</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 92 | Recycling                            | <a href="#">Frank</a> , <a href="#">USER_1</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 93 | Reguläre Ausdrücke (Regex)           | <a href="#">42-</a> , <a href="#">Benjamin</a> , <a href="#">David Leal</a> , <a href="#">etienne</a> , <a href="#">Frank</a> , <a href="#">MichaelChirico</a> , <a href="#">PAC</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 94 | Reguläre Ausdrücke in R              | <a href="#">Alexey Shiklomanov</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 95 | Reproduzierbares R                   | <a href="#">Charmgoggles</a> , <a href="#">Frank</a> , <a href="#">ikashnitsky</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 96 | RESTful R Services                   | <a href="#">YCR</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 97 | RMarkdown und Knitr-Präsentation     | <a href="#">Martin Schmelzer</a> , <a href="#">YCR</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

|     |                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 98  | RODBC                                                                     | <a href="#">akrun</a> , <a href="#">Hack-R</a> , <a href="#">Parfait</a> , <a href="#">Tim Coker</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 99  | Rohroperatoren (%>% und andere)                                           | <a href="#">42-</a> , <a href="#">Alexandru Papiu</a> , <a href="#">Alihan Zihna</a> , <a href="#">alistaire</a> , <a href="#">AndreyAkinshin</a> , <a href="#">Artem Klevtsov</a> , <a href="#">Atish</a> , <a href="#">Axeman</a> , <a href="#">Benjamin</a> , <a href="#">Carlos Cinelli</a> , <a href="#">CMichael</a> , <a href="#">DrPositron</a> , <a href="#">Franck Dernoncourt</a> , <a href="#">Frank</a> , <a href="#">Gal Dreiman</a> , <a href="#">Gavin Simpson</a> , <a href="#">Gregor</a> , <a href="#">ikashnitsky</a> , <a href="#">James McCalden</a> , <a href="#">Kay Brodersen</a> , <a href="#">Matt</a> , <a href="#">polka</a> , <a href="#">RamenChef</a> , <a href="#">Ryan Hilbert</a> , <a href="#">Sam Firke</a> , <a href="#">seasmith</a> , <a href="#">Shawn Mehan</a> , <a href="#">Simplans</a> , <a href="#">Spacedman</a> , <a href="#">SymbolixAU</a> , <a href="#">thelatemail</a> , <a href="#">tomw</a> , <a href="#">TriskalJM</a> , <a href="#">user2100721</a> |
| 100 | roxygen2                                                                  | <a href="#">DeveauP</a> , <a href="#">PAC</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 101 | R-Version wird aktualisiert                                               | <a href="#">dmail</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 102 | Schreibfunktionen in R                                                    | <a href="#">AkselA</a> , <a href="#">ikashnitsky</a> , <a href="#">kaksat</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 103 | Selbstbeobachtung                                                         | <a href="#">Jason</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 104 | Spaltenweiser Betrieb                                                     | <a href="#">akrun</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 105 | Spark-API (SparkR)                                                        | <a href="#">Maximilian Kohl</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 106 | Split-Funktion                                                            | <a href="#">Eric Lecoutre</a> , <a href="#">etienne</a> , <a href="#">josliber</a> , <a href="#">Sathish</a> , <a href="#">Tensibai</a> , <a href="#">thelatemail</a> , <a href="#">user2100721</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 107 | sqldf                                                                     | <a href="#">Hack-R</a> , <a href="#">Miha</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 108 | Standardisieren Sie Analysen, indem Sie eigenständige R-Skripts schreiben | <a href="#">akraf</a> , <a href="#">herbaman</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 109 | String-Manipulation mit dem Stringi-Paket                                 | <a href="#">bartektartanus</a> , <a href="#">FisherDisinformation</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 110 | Strsplit-Funktion                                                         | <a href="#">Imo</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 111 | Subsetting                                                                | <a href="#">42-</a> , <a href="#">Agriculturist</a> , <a href="#">alexis_laz</a> , <a href="#">alistaire</a> , <a href="#">dayne</a> , <a href="#">Frank</a> , <a href="#">Gavin Simpson</a> , <a href="#">Gregor</a> , <a href="#">L.V.Rao</a> , <a href="#">Mario</a> , <a href="#">mrip</a> , <a href="#">RamenChef</a> , <a href="#">smci</a> , <a href="#">user2100721</a> , <a href="#">zx8754</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 112 | Text-Mining                                                               | <a href="#">Hack-R</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 113 | Überlebensanalyse                                                         | <a href="#">42-</a> , <a href="#">Axeman</a> , <a href="#">Hack-R</a> , <a href="#">Marcin Kosiński</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 114 | Umfang der Variablen                                                      | <a href="#">Artem Klevtsov</a> , <a href="#">K.Daisey</a> , <a href="#">RamenChef</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 115 | Umformen mit TidyR                                                        | <a href="#">Charmgoggles</a> , <a href="#">Frank</a> , <a href="#">Jeromy Anglim</a> , <a href="#">SymbolixAU</a> , <a href="#">user2100721</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|     |                                                                 |                                                                                                                                                                                                                                                                                     |
|-----|-----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 116 | Variablen                                                       | <a href="#">42-</a> , <a href="#">Ale</a> , <a href="#">Axeman</a> , <a href="#">Craig Vermeer</a> , <a href="#">Frank</a> , <a href="#">L.V.Rao</a> , <a href="#">Imckeogh</a>                                                                                                     |
| 117 | Vektoren erstellen                                              | <a href="#">alistic</a> , <a href="#">bartekartanus</a> , <a href="#">Jaap</a> , <a href="#">Karsten W.</a> , <a href="#">Imo</a> , <a href="#">Rich Scriven</a> , <a href="#">Robert</a> , <a href="#">Robin Gertenbach</a> , <a href="#">smci</a> , <a href="#">takje</a>         |
| 118 | Verarbeitung natürlicher Sprache                                | <a href="#">CptNemo</a>                                                                                                                                                                                                                                                             |
| 119 | Veröffentlichung                                                | <a href="#">Frank</a>                                                                                                                                                                                                                                                               |
| 120 | Verteilungsfunktionen                                           | <a href="#">FisherDisinformation</a> , <a href="#">Frank</a> , <a href="#">L.V.Rao</a> , <a href="#">tenCupMaximum</a>                                                                                                                                                              |
| 121 | Verwenden von Texreg zum Exportieren von Modellen in Papierform | <a href="#">Frank</a> , <a href="#">ikashnitsky</a>                                                                                                                                                                                                                                 |
| 122 | Verwendung der Rohrzuordnung in Ihrem eigenen Paket             | <a href="#">RobertMc</a>                                                                                                                                                                                                                                                            |
| 123 | Wahrscheinlichkeitsverteilungen mit R                           | <a href="#">Pankaj Sharma</a>                                                                                                                                                                                                                                                       |
| 124 | Web Scraping und Parsing                                        | <a href="#">alistic</a> , <a href="#">Dave2e</a>                                                                                                                                                                                                                                    |
| 125 | Web-Crawling in R                                               | <a href="#">Pankaj Sharma</a>                                                                                                                                                                                                                                                       |
| 126 | xgboost                                                         | <a href="#">Hack-R</a>                                                                                                                                                                                                                                                              |
| 127 | Zeichenfolgen lesen und schreiben                               | <a href="#">42-</a> , <a href="#">4444</a> , <a href="#">abhiieor</a> , <a href="#">cdrini</a> , <a href="#">dotancohen</a> , <a href="#">Frank</a> , <a href="#">Gregor</a> , <a href="#">kdopen</a> , <a href="#">Rich Scriven</a> , <a href="#">Thomas</a> , <a href="#">Uwe</a> |
| 128 | Zeitreihen und Prognosen                                        | <a href="#">Andras Deak</a> , <a href="#">Andrew Bryk</a> , <a href="#">coatless</a> , <a href="#">Hack-R</a> , <a href="#">JGreenwell</a> , <a href="#">Pankaj Sharma</a> , <a href="#">Steve_Corrin</a> , <a href="#">μ Muthupandian</a>                                          |
| 129 | Zufallsforstalgorithmus                                         | <a href="#">G5W</a>                                                                                                                                                                                                                                                                 |
| 130 | Zufallszahlengenerator                                          | <a href="#">bartekartanus</a> , <a href="#">FisherDisinformation</a> , <a href="#">Karolis Koncevičius</a> , <a href="#">Miha</a> , <a href="#">mnoronha</a>                                                                                                                        |
| 131 | Zwang                                                           | <a href="#">d.b</a>                                                                                                                                                                                                                                                                 |