eBook Gratuit

APPRENEZ Gnuplot

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#gnuplot

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: gnuplot

It is an unofficial and free Gnuplot 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 Gnuplot.

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

Chapitre 1: Démarrer avec Gnuplot

Remarques

Cette section fournit une vue d'ensemble de ce qu'est gnuplot et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans gnuplot, et établir un lien avec les sujets connexes. Comme la documentation de gnuplot est nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Dernier *patchlevel*	Dernière date de sortie
5.0.x	5.0.5	2016-10-09
4.6.x	4.6.7	2015-04-28
4.4.x	4.4.4	2011-11-13
4.2.x	4.2.6	2007-07-01
4.0.x	4.0.0	2004-04-01

Examples

Installation ou configuration

Gnuplot est un utilitaire de graphique portable piloté par ligne de commande. Cet exemple montre comment configurer gnuplot sur les différentes plates-formes.

les fenêtres

1. Téléchargez la dernière version du programme d'installation depuis le site gnuplot .
2. Exécutez le fichier téléchargé et laissez-le s'exécuter en tant qu'administrateur si nécessaire
3. Dans la fenêtre de configuration, sélectionnez la langue et suivez les instructions à l'écran.
4. (facultatif) Au cours de l'installation, vous pouvez sélectionner le gnuplot à ajouter au PATH qui vous permettra d'exécuter des commandes depuis n'importe où sur la ligne de commande. Si vous choisissez de ne pas le faire, vous pouvez l'ajouter manuellement plus tard ou `cd` dans le répertoire installé gnuplot avant d'exécuter les commandes.

L'emplacement d'installation par défaut de gnuplot sous Windows est `C:\Program Files (x86)\gnuplot`

NOTE: le nom du fichier sera au format: `gp<version>-win32-mingw.exe`

Linux

L'installation sous Linux peut être effectuée via les différents gestionnaires de packages comme suit.

Cambre

```
$ sudo pacman -S gnuplot
```

Debian et Ubuntu

```
$ sudo apt-get update
$ sudo apt-get install gnuplot
```

CentOS / RedHat

```
$ sudo yum check-update
$ sudo yum install gnuplot
```

Feutre

```
$ sudo dnf check-update
$ sudo dns install gnuplot
```

Mac OS X

Utiliser Homebrew

```
$ brew install gnuplot
```

Utiliser MacPorts

```
$ sudo port install gnuplot
```

Tester l'installation

Après avoir installé gnuplot, il est judicieux d'exécuter un exemple simple pour s'assurer que tout fonctionne correctement.

1. Ouvrez votre terminal
2. Tapez `gnuplot` .
3. Votre invite devrait maintenant passer à `gnuplot>`
4. Type: `plot sin(x)`

Si tout va bien, vous devriez maintenant voir un graphique sin (x) généré par gnuplot.

Remarque: si vous êtes sous Windows et que vous n'avez pas ajouté `gnuplot` à votre `PATH` vous devrez accéder au `<gnuplot_install_path>\bin` . L'emplacement par défaut est: `C:\Program Files (x86)\gnuplot\bin`

Introduction de base aux règles du langage de programmation

De la documentation en ligne officielle de gnuplot 5.0 :

> Le langage de commande de gnuplot est **sensible à la casse** , c'est-à-dire que les commandes et les noms de fonctions écrits en *minuscules* ne sont pas les mêmes que ceux écrits en *majuscules* . Tous les noms de commande peuvent être abrégés tant que l'abréviation n'est pas ambiguë. Un nombre quelconque de commandes peut apparaître sur une ligne, séparées par des points-virgules `;` . (T. Williams, C. Kelley - *gnuplot 5.0, un programme de traçage interactif*)

Quelques exemples de ces règles de base sont

1. Un langage sensible à la casse

Taper des commandes en *minuscules* définies en *majuscules* générera un avertissement de `invalid command` .

```
gnuplot> set xlabel "x"
gnuplot> Set xlabel "x"
         ^
         invalid command
```

De plus, la variable `N` sera différente de celle `n` .

2. Abréviations

Vous pouvez trouver une liste presque complète des abréviations ici . Quoi qu'il en soit, les trois premières lettres de toute commande dans *gnuplot* fonctionnent toujours comme des abréviations. Certaines commandes permettent également une contraction plus puissante. Un petit exemple est donné ci-dessous.

```
gnuplot> p sin(x)
gnuplot> rep
gnuplot> q
```

où `p` représente un `plot` , `rep` pour `replot` et `q` pour `quit` .

3. séparateurs

Le symbole utilisé pour séparer les commandes sur une ligne unique est `;`

```
set title "My First Plot"; plot 'data'; print "all done!"
```

5. Commentaires

> Les commentaires sont supportés comme suit: un `#` peut apparaître dans la plupart des endroits d'une ligne et gnuplot ignorera le reste de la ligne. Cela n'aura pas cet effet à l'intérieur des guillemets, des nombres internes (y compris des nombres complexes), des substitutions de commandes internes, etc. Bref, cela fonctionne partout où il est logique de travailler. (*Ibidem*)

Rappelez-vous simplement la règle simple *"où il est logique de travailler"* .

```
gnuplot> # this is a comment, nothing will happen
gnuplot> plot sin(x) # another valid comment
gnuplot> plot sin(#x)
                  ^
         invalid expression
```

4. Extension des commandes

> Les commandes peuvent s'étendre sur plusieurs lignes d'entrée en terminant chaque ligne, mais la dernière avec une barre oblique inverse (`\`). La barre oblique inverse doit être le dernier caractère de chaque ligne. L'effet est comme si le backslash et le newline n'étaient pas là. En d'autres termes, aucun espace blanc n'est implicite, et aucun commentaire n'est terminé. Par conséquent, commenter une ligne continue commente toute la commande. (*Ibidem*)

Par exemple, pour diviser la commande de `plot` sur plusieurs lignes,

```
plot\
    sin(x),\
    cos(x)
```

va tracer le même que

```
plot sin(x), cos(x)
```

Une petite note sur *"commenter une ligne continue commente toute la commande"* . Si vous tapez la commande

```
plot\
    sin(x),\ # I would like to comment here
    cos(x)
```

une erreur se produira:

```
gnuplot> plot\
>        sin(x),\ # I would like to comment here
```

```
                          ^
        invalid character \
```

Donc, il vaut mieux être prudent et respecter la règle *"partout où il est logique de travailler"* en utilisant `#` commentaires.

Lire Démarrer avec Gnuplot en ligne: https://riptutorial.com/fr/gnuplot/topic/3284/demarrer-avec-gnuplot

Chapitre 2: Ajuster les données avec gnuplot

Introduction

La commande ajustement peut ajuster une fonction définie par l' utilisateur à un ensemble de points de données `(x,y)` ou `(x,y,z)` , en utilisant une implémentation de la méthode des moindres carrés non linéaire **(NLLS)** algorithme de Levenberg-Marquardt.

Toute variable définie par l'utilisateur apparaissant dans le corps de la fonction peut servir de paramètre d'ajustement, mais le type de retour de la fonction doit être réel.

Syntaxe

- ***adapter*** *[xrange] [yrange] fonction "fichier* de *données"* en ***utilisant*** le *modificateur* ***via*** *parameter_file*

Paramètres

Paramètres	Détail
Paramètres d'ajustement `a` , `b` , `c` et toute lettre qui n'a pas été utilisée précédemment	Utilisez des lettres pour représenter les paramètres qui seront utilisés pour adapter une fonction. Ex: `f(x) = a * exp(b * x) + c` , `g(x,y) = a*x**2 + b*y**2 + c*x*y`
Paramètres de fichier `start.par`	Au lieu d'utiliser des paramètres non initialisés (le Marquardt-Levenberg initialise automatiquement pour vous `a=b=c=...=1`), vous pouvez les placer dans un fichier `start.par` et les appeler dans la section *parameter_file* . Ex: `fit f(x) 'data.dat' u 1:2 via 'start.par'` . Un exemple pour le fichier `start.par` est montré ci-dessous

Remarques

Brève introduction

`fit` est utilisé pour trouver un ensemble de paramètres qui correspondent le mieux à vos données à votre fonction définie par l'utilisateur. L'ajustement est jugé sur la base de la somme des différences au carré ou des «résidus» (SSR) entre les points de données d'entrée et les valeurs de fonction, évaluées aux mêmes endroits. Cette quantité est souvent appelée «chisquare» (c'est-à-dire la lettre grecque chi, à la puissance de 2). L'algorithme tente de minimiser le SSR, ou plus précisément le

> WSSR, car les résidus sont "pondérés" par les erreurs de données d'entrée (ou 1.0) avant d'être mis au carré. (*Ibidem*)

Le fichier `fit.log`

Après chaque étape d'itération, une information détaillée est donnée sur l'état de l'ajustement à la fois sur l'écran et sur ce que l'on appelle le fichier journal `fit.log` . Ce fichier ne sera jamais effacé mais toujours ajouté pour que l'historique de l'ajustement ne soit pas perdu.

Examples

Ajustement des données avec des erreurs

> Il peut y avoir jusqu'à 12 variables indépendantes, il y a toujours une variable dépendante et un nombre quelconque de paramètres peut être ajusté. En option, des estimations d'erreur peuvent être entrées pour pondérer les points de données. (T. Williams, C. Kelley - *gnuplot 5.0, un programme de traçage interactif*)

Si vous avez un ensemble de données et que vous souhaitez l'ajuster si la commande est très simple et naturelle:

```
fit f(x) "data_set.dat" using 1:2 via par1, par2, par3
```

où plutôt `f(x)` pourrait être aussi `f(x, y)` . Si vous avez également des estimations d'erreurs de données, ajoutez simplement le `{y | xy | z}errors` (`{ | }` représentent les choix possibles) dans l'option *modificateur* (voir **Syntaxe)** . Par exemple

```
fit f(x) "data_set.dat" using 1:2:3 yerrors via par1, par2, par3
```

où le `{y | xy | z}errors` options d' `{y | xy | z}errors` nécessitent respectivement 1 (`y`), 2 (`xy`), 1 (`z`) colonne spécifiant la valeur de l'estimation de l'erreur.

`xyerrors` exponentiel avec `xyerrors` d'un fichier

> Les estimations d'erreur de données sont utilisées pour calculer le poids relatif de chaque point de données lors de la détermination de la somme pondérée des résidus au carré, WSSR ou chisquare. Ils peuvent affecter les estimations des paramètres, car ils déterminent l'influence de la déviation de chaque point de données par rapport à la fonction ajustée sur les valeurs finales. Certaines des informations d'ajustement, y compris les estimations des erreurs de paramètres, sont plus significatives si des estimations précises des erreurs de données ont été fournies. (*Ibidem*)

Nous prendrons un ensemble de données échantillonné **`measured.dat`** , composé de 4 colonnes: les coordonnées de l'axe des abscisses (`Temperature (K)`), les coordonnées de l'axe des ordonnées (`Pressure (kPa)`), les estimations d'erreur x (`T_err (K)`) et les estimations d'erreur y (`P_err (kPa)`).

```
#### 'measured.dat' ####
### Dependence of boiling water from Temperature and Pressure
##Temperature (K) - Pressure (kPa) - T_err (K) - P_err (kPa)

368.5     73.332        0.66    1.5
364.2     62.668        0.66    1.0
359.2     52.004        0.66    0.8
354.5     44.006        0.66    0.7
348.7     34.675        0.66    1.2
343.7     28.010        0.66    1.6
338.7     22.678        0.66    1.2
334.2     17.346        0.66    1.5
329.0     14.680        0.66    1.6
324.0     10.681        0.66    1.2
319.1     8.015         0.66    0.8
314.6     6.682         0.66    1.0
308.7     5.349         0.66    1.5
```

Maintenant, il suffit de composer le prototype de la fonction qui, d'après la théorie, devrait se rapprocher de nos données. Dans ce cas:

```
Z = 0.001
f(x) = W * exp(x * Z)
```

où nous avons initialisé le paramètre `Z` car, sinon, évaluer la fonction exponentielle `exp(x * Z)` donne des valeurs énormes, ce qui conduit à Infinity et NaN dans l'algorithme d'adaptation de Marquardt-Levenberg, vous n'avez généralement pas besoin d'initialiser variables - regardez ici , si vous voulez en savoir plus sur Marquardt-Levenberg.

Il est temps d'adapter les données!

```
fit f(x) "measured.dat" u 1:2:3:4 xyerrors via W, Z
```

Le résultat ressemblera

```
After 360 iterations the fit converged.
final sum of squares of residuals : 10.4163
rel. change during last iteration : -5.83931e-07

degrees of freedom    (FIT_NDF)                        : 11
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.973105
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.946933
p-value of the Chisq distribution (FIT_P)              : 0.493377

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
W               = 1.13381e-05      +/- 4.249e-06    (37.47%)
Z               = 0.0426853        +/- 0.001047     (2.453%)

correlation matrix of the fit parameters:
                W      Z
W               1.000
Z              -0.999  1.000
```

Où maintenant `W` et `Z` sont remplis avec les paramètres désirés et les erreurs estimées sur ceux-là.

Le code ci-dessous produit le graphique suivant.

```
set term pos col
set out 'PvsT.ps'

set grid
set key center
set xlabel 'T (K)'
set ylabel 'P (kPa)'


Z = 0.001
f(x) = W * exp(x * Z)
fit f(x) "measured.dat" u 1:2:3:4 xyerrors via W, Z

p [305:] 'measured.dat' u 1:2:3:4 ps 1.3 pt 2 t 'Data' w xyerrorbars,\
f(x) t 'Fit'
```

Terrain avec ajustement de `measured.dat` En utilisant la commande `with xyerrorbars` affiche des erreurs sur les estimations x et des y. `set grid` placera une grille en pointillés sur les tics principaux.

80

70

, qui comprend les mètres carrés d'une maison dans une certaine ville et son prix en 1000 dollars.

```
### 'house_price.dat'
## X-Axis: House price (in $1000) - Y-Axis: Square meters (m^2)

245    426.72
312    601.68
279    518.16
308    571.50
199    335.28
219    472.44
405    716.28
324    546.76
319    534.34
255    518.16
```

Nous allons adapter ces paramètres avec *gnuplot* La commande elle-même est très simple, comme vous pouvez le constater avec la syntaxe, définissez simplement votre prototype, puis utilisez la commande `fit` pour obtenir le résultat:

```
## m, q will be our fitting parameters
f(x) = m * x + q
fit f(x) 'data_set.dat' using 1:2 via m, q
```

Mais il pourrait être intéressant d'utiliser également les paramètres obtenus dans l'intrigue elle-même. Le code ci-dessous correspond au fichier **`house_price.dat`** , puis trace les paramètres `m` et `q` pour obtenir la meilleure approximation de la courbe de l'ensemble de données. Une fois que vous avez les paramètres, vous pouvez calculer la `y-value` , dans ce cas le *prix Maison* , à partir de n'importe quelle `x-vaule` (*mètres carrés* de la maison) en remplaçant simplement la formule

```
y = m * x + q
```

la `x-value` appropriée. Commentons le code.

0. Définition du terme

```
set term pos col
set out 'house_price_fit.ps'
```

1. Administration ordinaire pour embellir le graphique

```
set title 'Linear Regression Example Scatterplot'
set ylabel 'House price (k$ = $1000)'
set xlabel 'Square meters (m^2)'
set style line 1 ps 1.5 pt 7 lc 'red'
set style line 2 lw 1.5 lc 'blue'

set grid
set key bottom center box height 1.4

set xrange [0:450]
set yrange [0:]
```

2. Le bon ajustement

Pour cela, il suffit de taper les commandes:

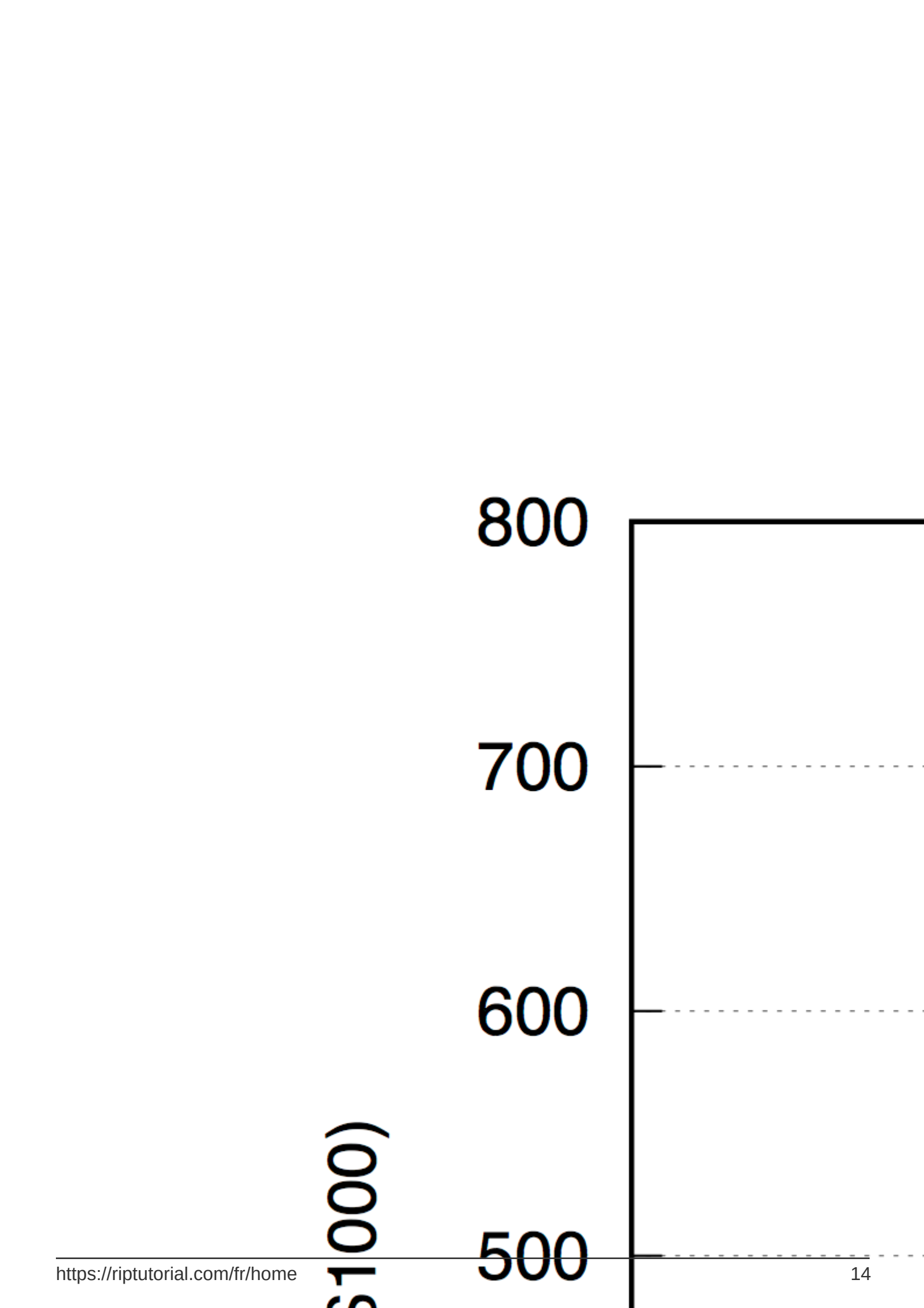
```
f(x) = m * x + q
fit f(x) 'house_price.dat' via m, q
```

3. Enregistrement des valeurs `m` et `q` dans une chaîne et tracé

Ici, nous utilisons la fonction `sprintf` pour préparer l'étiquette (encadrée dans le `object rectangle`) dans laquelle nous allons imprimer le résultat de l'ajustement. Enfin, nous traçons le graphique entier.

```
mq_value = sprintf("Parameters values\nm = %f k$/m^2\nq = %f k$", m, q)
set object 1 rect from 90,725 to 200, 650 fc rgb "white"
set label 1 at 100,700 mq_value


p 'house_price.dat' ls 1 t 'House price', f(x) ls 2 t 'Linear regression'
set out
```

La sortie ressemblera à ceci.


800
700
600
500

https://riptutorial.com/fr/gnuplot/topic/8825/ajuster-les-donnees-avec-gnuplot

Chapitre 3: Styles de traçage 2D

Examples

Sélection d'un style de tracé

Sélection explicite

Un style de traçage est généralement sélectionné avec le mot-clé `with` , comme

```
plot x with points
```

Cela permet d'utiliser différents styles de traçage pour chaque `plot` :

```
plot x with points, 2*x with lines
```

Taper `help with` dans la fenêtre de commande gnuplot donne une liste de tous les styles de tracé disponibles.

Sélection de style de tracé global

Les styles de traçage peuvent également être définis globalement pour toutes les commandes de tracé. Ici, gnuplot fait la distinction entre les tracés de fonctions et de données, pour lesquels différents styles par défaut peuvent être définis.

Pour les fonctions, utilisez `set style function` :

```
set style function linespoints
plot x, 2*x
```

Pour les fichiers de données, utilisez les `set style data` :

```
set style data lines
plot 'file.dat', 'other-file.dat'
```

Notez que pour les fonctions, le style par défaut est les `lines` et pour les fichiers de données, les `points` . Avec les `show style data` et la `show style function` vous pouvez inspecter les styles de tracé sélectionnés.

Lire Styles de traçage 2D en ligne: https://riptutorial.com/fr/gnuplot/topic/4302/styles-de-tracage-2d

Chapitre 4: Tracé de base des fichiers de données

Introduction

L'une des principales fonctionnalités de *gnuplot* est la possibilité de tracer **des fichiers de données** . Tracer un fichier de données est très simple avec *gnuplot* . En fait, une fois que vous avez ouvert le logiciel depuis le terminal, il vous suffit de numériser le `plot 'file'` pour obtenir un tracé automatique.

Tout d'abord, avant de tracer, vous devez être sûr d'être dans le même répertoire que le fichier de données, sinon vous obtiendrez éventuellement un `warning` .

Syntaxe

- tracer le *fichier de données en* utilisant *expression_colonne* avec *style*

Examples

Tracer un seul fichier de données

Le `plot` commande par défaut de *gnuplot* (également uniquement `p`) représente le jeu de données avec colonnes, sous la forme du fichier **`data_set.dat`** ci-dessous.

```
# Prototype of a gnuplot data set
# data_set.dat
# X -    X^2 -     2*X -     Random
0        0         0         5
1        1         2         15
1.4142   2         2.8284    1
2        4         4         30
3        9         6         26.46
3.1415   9.8696    6.2832    39.11
4        16        8         20
4.5627   20.8182   9.1254    17
5.0      25.0      10.0      25.50
6        36        12        0.908
```

Comme vous pouvez le voir, vous pouvez écrire dans votre ensemble de données en notation à virgule flottante. Maintenant, tout est prêt pour faire le tracé de données: en tapant uniquement

```
plot "data_set.dat"
```

gnuplot produira un graphique dans votre destination de `output` . Les paramètres par défaut utiliseront les deux premières colonnes de votre fichier de données, respectivement x et y. Pour spécifier les colonnes à tracer, utilisez le spécificateur **`using`**

```
plot "data_set.dat" using 2:4
```

ce qui signifie "tracer le fichier en utilisant la colonne 2 comme X et la colonne 4 comme Y". Dans le cas où votre fichier de données est un fichier tridimensionnel, utilisez simplement la `splot` ad pour ajouter la colonne z

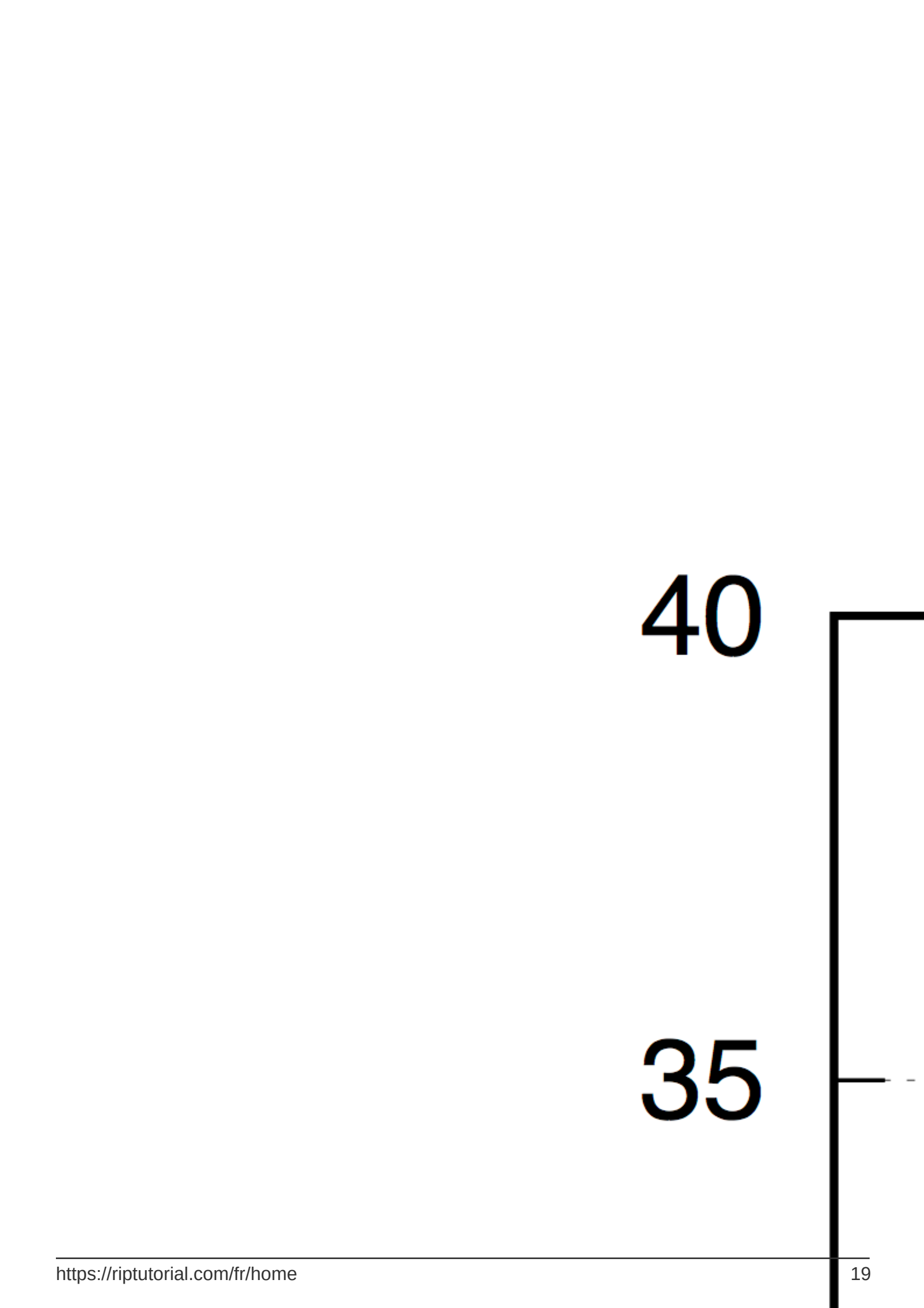
```
splot "data_set.dat" using 1:2:3
```

Il existe également des styles différents (voir la documentation gnuplot ou Sélection d'un style de traçage pour plus d'informations) pour tracer des points. Comme dit précédemment, le style par défaut est le `point`

```
plot "data_set.dat" using 1:4 with point
```

qui tracera la même chose que si vous ne tapez pas `with point` . Un point utile pour le traçage des données est le `linespoint` qui est évidemment "lignes + points". **PAR EXEMPLE:**

```
plot "data_set.dat" using 1:4 with linespoint
# the abbreviated form is completely equivalent:
# p "data_set.dat" u 1:4 w lp
```


40
35

, avec une étape décidée (si elle n'est pas spécifiée = 1). Par exemple, `for [i = 0:6:2]` , incrémente `i` de 0 à 6 en 2 étapes: `i = 0, 2, 4, 6` . Toutes les valeurs (start, stop et increment) sont converties en valeurs entières.

*** Grille**

La grille est souvent utile pour tracer un ensemble de données. Pour ajouter un type de grille

```
set grid
```

Tracer plusieurs fichiers de données

Première méthode - Concaténation de chaînes

La méthode la plus simple pour tracer plusieurs fichiers de données consiste à insérer une boucle `for` dans la commande `plot` de gnuplot. En supposant que vous avez `N` fichiers nommés en séquence, *c.-à-d.*

```
file_1.dat
file_2.dat
file_3.dat
...
file_N.dat
```

Exécuter la commande

```
plot for[i = 1:N] "file_".i.".dat"
```

`file_1.dat` tous les fichiers entre `file_1.dat` et `file_N.dat` dans le même graphique.

Exemple avec trois fichiers de données

Tableau de jeux de données

Axes X	Y-Axe file_1.dat	Y-Axe fichier_2.dat	Y-Axe file_3.dat
1	1	1	1
2	2	4	2
3	3	9	6
4	4	16	24
5	5	25	120

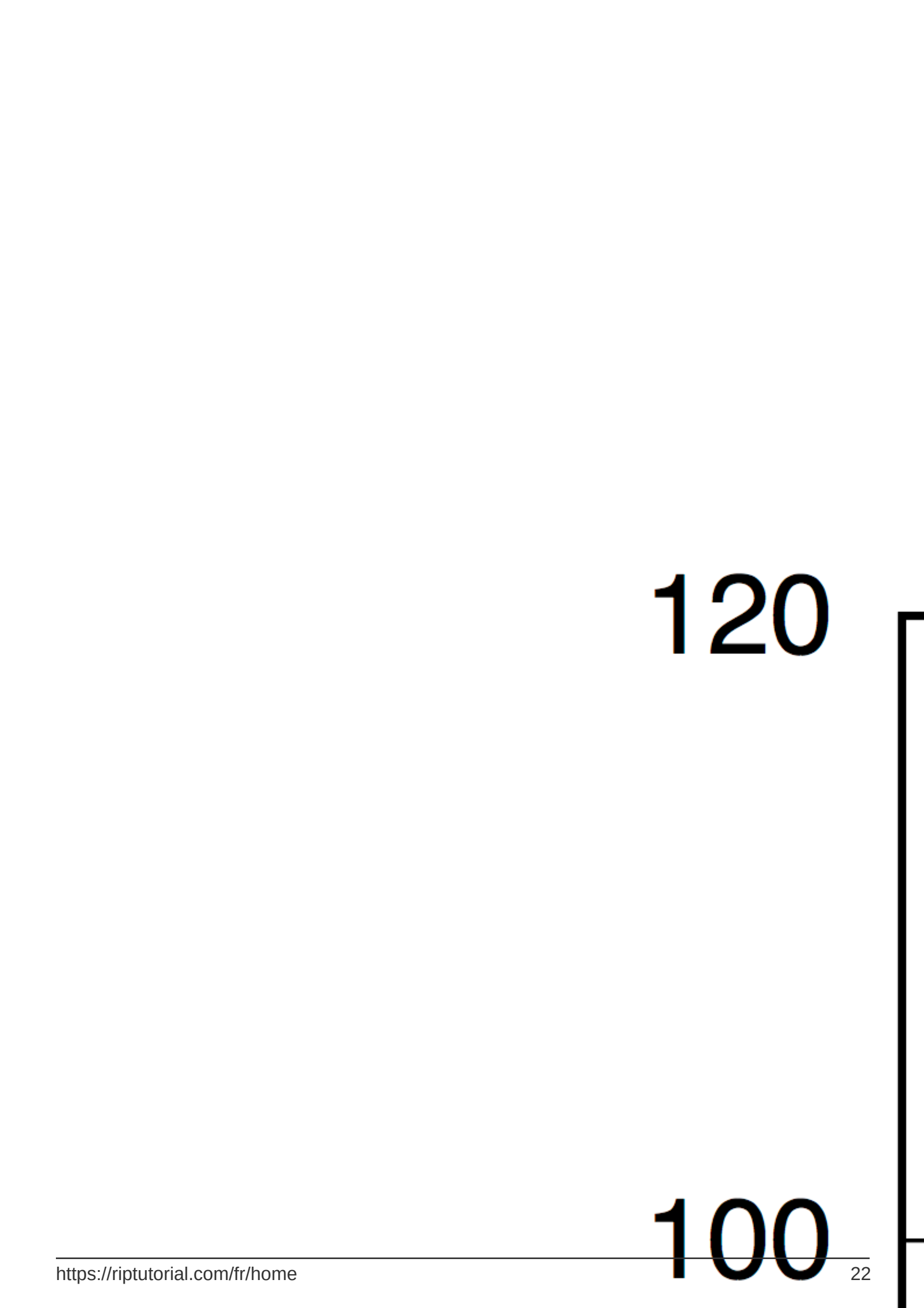
Commandes

```
set terminal postscript color noenhanced ##setting the term
set output "multiple_files.ps"

set key center ##legend placement

plot [1:5][1:120] \
    for [i = 1:3] "file_".i.".dat" \
    pointsize 1.3 linecolor i+4 \
    title "file\_".i.".dat" \
    with linespoint
```

La boucle commence par `for [i = 1:3] "file_".i.".dat"` et exécute la commande `plot` jusqu'à ce qu'elle atteigne `i = 3` . Le `.i.` est le nombre concaténé.

`title "file\_".i.".dat"` a été écrit avec le `\` pour que le symbole _ dans le nom des fichiers apparaisse comme un *soulignement* plutôt que comme un *indice* , et le spécificateur `noenhanced` est fondamental pour obtenir ce résultat .

Le résultat final est indiqué ci-dessous


120
100

comme le `sprintf` langage C. La bonne syntaxe, à partir de la documentation de gnuplot 5.1 est

```
sprintf("format", x, y, ...)
```

Un bref exemple clarifiera chaque doute.

```
file_name(n) = sprintf("file_%d.dat", n)
plot for[i = 1:N] file_name(i) title file_name(i)
```

Lire Tracé de base des fichiers de données en ligne:
https://riptutorial.com/fr/gnuplot/topic/3591/trace-de-base-des-fichiers-de-donnees

Chapitre 5: Utiliser des fichiers de script

Syntaxe

1. gnuplot -c scriptfile ARG1 ARG2 ...

Remarques

L'utilisation de base peut être affichée en tapant `gnuplot -h`

```
$ gnuplot -h
Usage: gnuplot [OPTION] ... [FILE]
  -V, --version
  -h, --help
  -p  --persist
  -d  --default-settings
  -c  scriptfile ARG1 ARG2 ...
  -e  "command1; command2; ..."
gnuplot 5.0 patchlevel 3
```

Examples

Fichier script simple

Gnuplot est capable de générer un graphique à partir d'un fichier script qui permet une séquence de commandes nécessaire pour dessiner un graphique à exécuter en séquence au lieu de saisir manuellement.

Dans cet exemple, nous allons créer un script simple pour dessiner un `sin(x)` .

Créer un fichier script

Créez un fichier `sinx.p` avec le contenu suivant:

```
# Set the output to a png file
set terminal png size 500,500
# The file we'll write to
set output 'sinx.png'
# The graphic title
set title 'Sin(x)'
#plot the graphic
plot sin(x)
```

Dans l'exemple ci-dessus, vous trouverez les commandes les plus courantes. Cependant, il existe plusieurs autres commandes à explorer, telles que `set xlabel` , `set ylabel` , etc.

Vous pouvez personnaliser la ligne de `set output` avec le chemin que vous souhaitez que le fichier

génère le fichier.

Exécuter le script

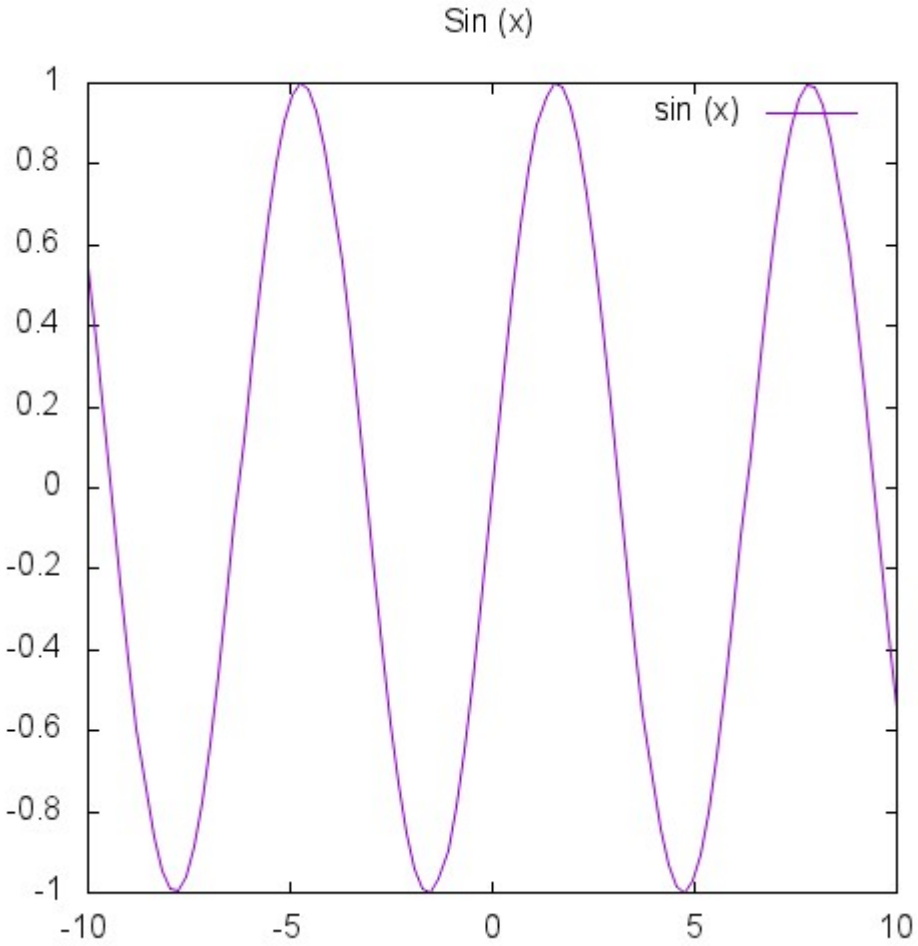
Ouvrez votre terminal et tapez:

```
gnuplot path/to/sinx.p
```

Si votre dossier actuel contient le script, vous pouvez entrer le texte suivant à la place:

```
gnuplot sinx.p
```

Le script s'exécutera et générera le fichier PNG à l'emplacement spécifié. Le graphique résultant devrait ressembler à ceci:



Lire Utiliser des fichiers de script en ligne: https://riptutorial.com/fr/gnuplot/topic/4013/utiliser-des-fichiers-de-script

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec Gnuplot	Community, Fawix, opisthofulax
2	Ajuster les données avec gnuplot	opisthofulax
3	Styles de traçage 2D	Christoph
4	Tracé de base des fichiers de données	Christoph, Matthew, opisthofulax, Tom Solid
5	Utiliser des fichiers de script	Christoph, Fawix