Visualization of Acquired Data with Gnuplot

Quick on the Draw

Visualizing data quickly and effectively in diagrams is a common requirement. Gnuplot, the program we will be looking at in this article, has provided this functionality for years, although admittedly Gnuplot has never made the mainstream.

BY ALEXANDER REITERER

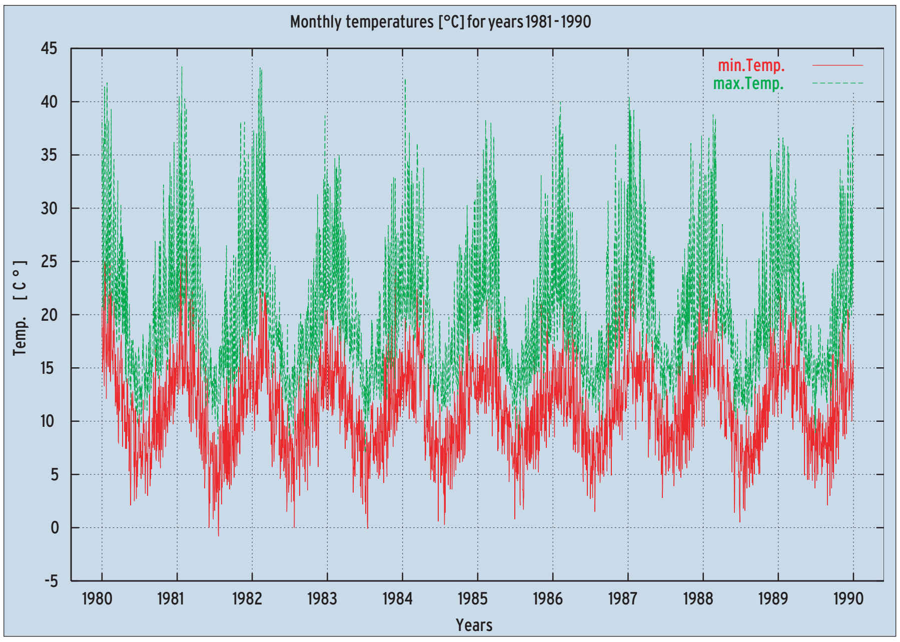

Figure 1: Minimum and maximum monthly temperatures in Melbourne

Visualization of acquired data might appear very specific at first glance, but you don't need to be a rocket scientist to find innumerable practical examples, such as remotely monitoring the ambient temperature, evaluating logfiles, or displaying share indices. For over ten years now Gnuplot has provided a tool that, despite its slightly Spartan character based interface, offers unimagined flexibility in combination with shell scripts and other tools.

Those of you who prefer a more comfortable interface will be pleased to hear that various front-ends provide more convenient control options. These include *Kile* [3], a Latex editor with integrated Gnuplot front-end, and *Unignuplot* [4]. We will be concentrating on the command line here, as it provides for system independent, automated data processing. Gnuplot offers 2 or 3 dimensional visualization facilities and can handle very large amounts of data.

Incidentally, the project has nothing to do with the Gnu Project run by the Free Software Foundation; any similarity is purely coincidental. The program is neither free software nor Open Source as defined by the Open Source Initiative. Modifying the program is allowed, although there is a stipulation that modifications must be published as patches.

Gnuplot is included in most major distributions and is launched by entering *gnuplot*, which will display a prompt to the user. You can now enter Gnuplot specific commands, such as *plot cos(x)*, to display a cosine function. Gnuplot then opens a separate window and plots graphs based on the current basic settings.

Listing 1: Plot file for monthly temperatures

```
set xrange [-150:3800]
set xtics ("1980" 0, "1981" 365, "1982" 365*2, "1983" 365*3,➘
"1984" 365*4, "1985" 365*5, "1986" 365*6, "1987" 365*7, "1988" 365*8,➘
"1989" 365*9, "1990" 365*10)
set title "Monthly temperatures [°C] for years 1981 - 1990"
set xlabel "Years"
set ylabel "Temp. [°C]"
set grid
plot 'tempmin.dat' title "min.Temp." with lines,➘
'tempmax.dat' title "max.Temp." with lines
```

Plotting Simple Datasets

The data to be visualized can be derived from a wide variety of sources. From measuring devices used to collate physical quantities (temperature, air-pressure, humidity), but also from the Internet (such as stock exchange stats). You can also use virtual data.

No matter what method you choose, or where your data originated, acquired data will be available as a dataset of some kind. As demonstrated in the previous section, the *plot* command is used to generate output. Our example simply passed a mathematical formula to the *plot* command. To process data from a file, you pass the file name to *plot* as an argument. The file format is extremely basic: Gnuplot parses the file line by line. A line represents a record, and can comprise multiple columns. Columns are separated by space characters.

Let us assume that our dataset is stored in a file called *example.dat* and comprises a single row of numbers (one value per line); in this case you could enter *plot 'example.dat'* to tell Gnuplot to plot the co-ordinates as points. If you need a different format to visualize your data, you can use one of the *plot* commands *style* attributes, which are as follows: lines, points, linespoints, dots, impulses, steps, fsteps, histeps, boxes, errorbars, xerrorbars, yerrorbars, xyerrorbars, boxerrorbars, boxxyerrorbars, vector, financebars, candlesticks (Figure 2 shows a selection of output formats). The syntax is always: *plot 'file' with style*

The *plot* command provides a number of useful parameters for visualizing large amounts of data. The requirement will be not to display the whole range of data, but a selection. For example *plot 'example.dat' using 2:3* would output only the second and third columns.

A Simple Example

The aim for this example is to plot an attractive graph of the monthly temperatures for Melbourne, Australia, for the years 1981 through 1990. The figures were supplied by a meteorological station and are stored in files called *tempmin.dat* and *tempmax.dat* (minimum and maximum monthly temperatures, respectively). The temperature figures in the files are stored in a single column.

When you are plotting complex graphs it may not be very effective to enter the commands involved sequentially. Thus, you have the option of creating a plot file. The following example creates a plot file called *temp.plt* that contains the commands required by our example. You will need an editor to enter the commands.

Gnuplot also allows you to write the commands you enter in the command line to a plot file. This facility can be enabled using the *save option filename* syntax, where *option* is either *functions*, *variables* or *set*. If you do not specify any options, the current sets, functions, variables and the last *plot* and/or *splot* command are saved. *load 'filename'* will load the file again.

The *~/.gnuplot* file plays a special role. If available, it is parsed on launching Gnuplot. In other words, you can use the file to store basic preferences, which are then applied whenever you launch the program.

Multiple Datasets

Our example will be plotting multiple datasets as a single graph. The following commands are required: *plot 'tempmax.dat' title 'max. Temp.' with lines, 'tempmin.dat' title 'min. Temp.' with lines*

set commands are used to define titles and labels. *set title "Title"* will set the diagram title, *set xlabel "X-label"* sets the label for X axis and *ylabel* for the Y axis. *set key x,y* will set the key to position x:y. If you do not require a key for the current diagram, you can use *set nokey* to hide it. The *set xtics* command allows you to apply individual labels to the X axis. *set grid* will display a grid. The following command defines the range for the plot: *set xrange [xmin:xmax]*. See the example in Listing 1.

Error Bars

One typical requirement is to plot erroneous measurements or deviations along with the acquired data. The *with errorbars* command provides this facility. To do so, you need to place the starting point and endpoint of the error bar in a column of their own within the data set. If you only use one column to define error bars, or if the use of columns is delimited by a command such as *using 1:2:3*, Gnuplot will refer to the third column to dimension the error bar. The

Special Functions

- You occasionally need to apply specific labels to a time axis. The following command is used for this purpose: *set xmtics* converts numbers to month names (1=Jan, 2=Feb, ...), *set xdtics* converts to weekdays (0=Sunday, 6=Saturday, ...). Of course, you could substitute y for x to apply this to the y axis. Undefined values are assigned a value, for example 13 will result in Jan.
- The following commands hide an axis: *set xzeroaxis, set yzeroaxis, set zzeroaxis*
- The *set logscale axis base* applies a logarithmic scale to an axis.

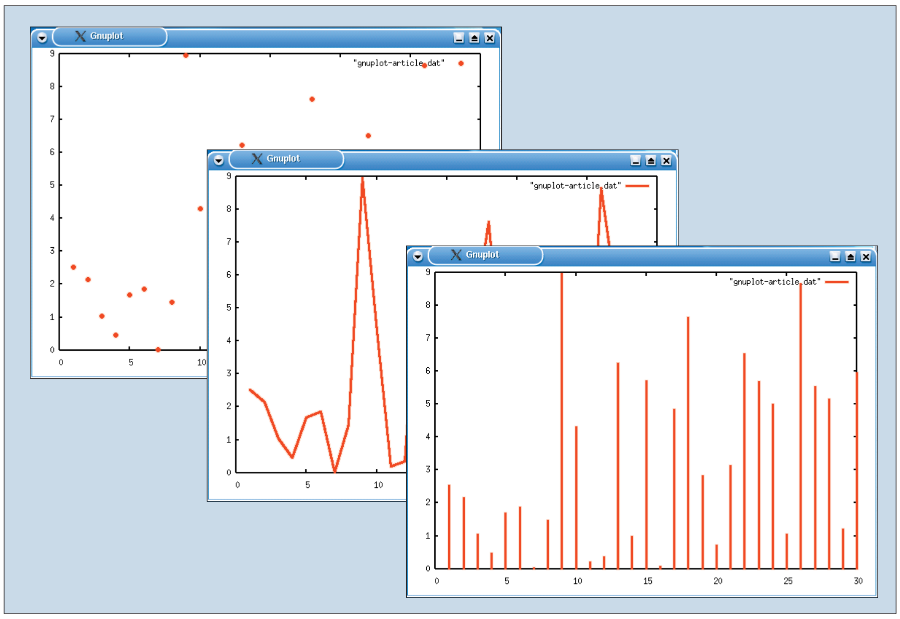

Figure 2: Three different output styles using the plot commands (*with points*, *with lines*, *with impulses*)

Listing 2: Plot file for error bars

```
set title "Average (fictional) ➘
income"
set xlabel "Years on staff"
set ylabel "Income [Dollars]"
set xrange [-1.5:48]
plot 'income.dat' title ➘
"Income" with boxes, ➘
'income.dat' title "Deviation" ➘
with errorbars
```

Listing 3: Extract from *3dpoints.dat*

```
-31 -194 -661
58 149 -605
-2 188 -599
...
```

Listing 4: Plot file for 3D-Plot example

```
set title "3D-Plot of ➘
co-ordinate points"
set data style lines
set contour base
set surface
set hidden3d
set view 50,10,1.0,1.0
set dgrid3d 40,51,3
show contour
splot '3dpoints.dat' title "Grid"
```

value is then interpreted as the difference to the value of the function (function value +/- deviation).

If the user specifies *with boxes* in addition to error bars, the acquired data will be displayed as a bar chart and the deviations as impulses. Listing 2 contains the plot file for our example.

3D Plots

Acquired data commonly needs to be visualized in 3D. Instead of the *plot* command, you can use the *splot* command to do so. The following example visualizes a set of 3 dimensional measuring points. The co-ordinates of these points are stored in *X Y Z* format in a file called *3dpoints.dat*:

There are a number of special commands for three-dimensional plotting with Gnuplot. *set view* manipulates the user's fictive view of 3D graph. The syntax is: *set view rot_x, rot_z, scale_x, scale_z*. You should be aware of the permissible rotation intervals; *rot_x* must lie between 0 and 180 degrees, *rot_z* between 0 and 360 degrees. The default settings are: 60 degrees for *rot_x*, and 30 degrees for *rot_z*. Our example: *set view 50, 10, 1.0, 1.0*. *set hidden3d* will hide any hidden line segments to enhance the 3D effect.

If you want Gnuplot to plot co-ordinates with an irregular distribution, you should look to closing the gaps in the distributed coordinates using interpolation. You can use the *set dgrid3d* to do so. You need to tell the command the number of intervals along the x and y axes, and additionally supply a weighting factor. In our case: *set dgrid3d 40,51,3*.

set ticslevel manipulates the starting point of the vertical axis. The default setting is 0.5.

Listing 4 shows the final plot file for our example.

Output

As most users tend to embellish other documents with diagrams, it is a good thing that Gnuplot provides a variety of export functions. You simply redirect the output to a file. The *set terminal* command is used for this purpose. *set output "filename"* then creates an output file. The following sections show commands used to create Postscript (see Listings 5 and 6) and Latex files (see Listing 7).

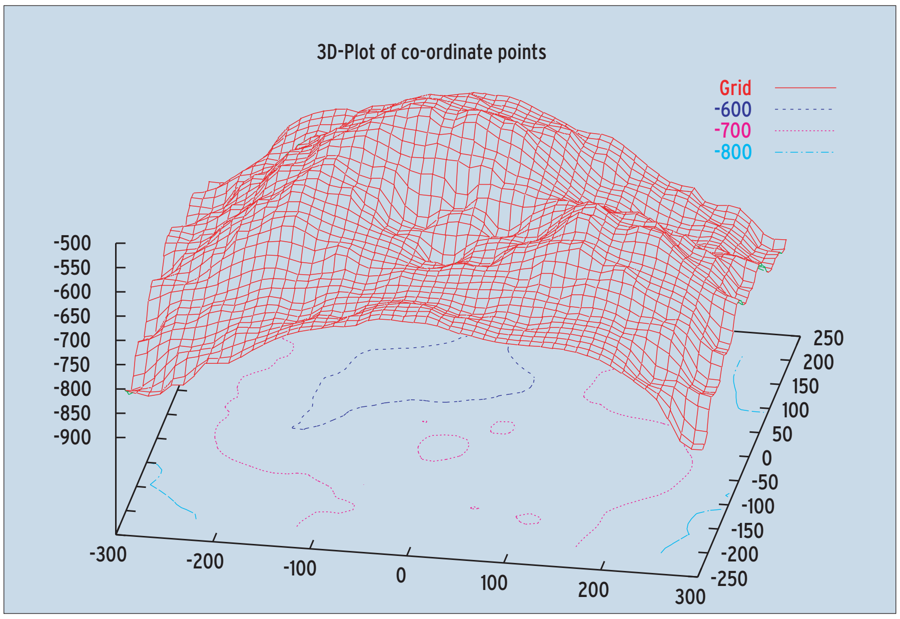

Figure 3: 3D-Plot

Listing 5: Postscript Output

```
set terminal postscript color
set output 'filename.ps'
replot
```

Listing 6: Encapsulated Postscript Output

```
set terminal postscript eps ➘
color
set output ' filename.eps'
replot
```

Listing 7: Latex Output

```
set terminal latex
set output 'filename.tex'
replot
```

The facility for issuing Latex-specific commands from within *set* commands (labels, titles, ...) using *$* and \\ is particularly useful. For example: *set title '\LaTeX\ - $ \gamma $ '*.

Help and Recommended Reading

As the full range of Gnuplot functions is far larger than the scope that this article could possibly cover, readers are recommended to investigate the comprehensive help system. Typing *help keyword* will provide help on the chosen keyword. Just typing *help* calls up the help system. Documentation on Gnuplot is available on the Internet from [2], [5] and [6], amongst other sites. The manual is available from the Gnuplot homepage [1]. ■

INFO

[1] Gnuplot Homepage: *http://www.gnuplot.info/*

[2] Gnuplot Tutorial: *http://www.duke.edu/~hpgavin/gnuplot.html*

[3] Kile Homepage: *http://kile.sourceforge.net/*

[4] Unignuplot Homepage: *http://unicalculus.sourceforge.net/*

[5] Gnuplot Manual: *http://www.tu-chemnitz.de/urz/anwendungen/grafik/gnuplotdoc.html*

[6] Gnuplot FAQ: *http://www.ucc.ie/gnuplot/gnuplot-faq.html*