Visualisation Lab: gnuplot

Anton Gerdelan

February 2, 2012

What is gnuplot?

gnuplot is a tool for creating graphs and charts. gnuplot has a terminal. You can enter commands to tell gnuplot how to format your graph, or provide a file containing a list of commands.

The major advantages of gnuplot are:

- clean, publication-quality graphs are created
- error bars are easy to display
- input data is easy to format
- data or commands can be piped in from `stdin`
- you have control over every aspect of the display
- you can output to a variety of pixel and vector formats

This means that gnuplot will play well with LaTeX, and with any home-brewed data processing pipeline. It can be a little clunky to learn the terminal environment, so this tutorial gives some examples of common tasks.

NB: gnuplot is not related to the GNU project, and is usually not spelled with any capital letters.

1 Install

Homepage is: `http://gnuplot.info/`.

- lab machine: I installed `gp46rc1-win32-setup.exe` on a lab machine. You will need to put in the admin user name when installing. Say yes to the query about putting the application directory on PATH (allows you to pipe to gnuplot from anywhere)
- Debian/Ubuntu/mint Linux: `sudo apt-get install gnuplot`
- mac: I haven't tried it, but mac ports is the best bet. Read this: `http://youinfinitesnake.blogspot.com/2011/02/attractive-scientific-plots-with.html`

The homepage has a number of links to tutorials and demos. You may also find the "Not so Frequently Asked Questions" website useful:

`http://t16web.lanl.gov/Kawano/gnuplot/index-e.html`

2 Exercises

2.1 start a gnuplot terminal

Type in

```
gnuplot
```

in a console window to get to the plain gnuplot terminal. You can also run *via* the icon to get to a graphical terminal. To close a gnuplot terminal, type

```
quit
```

Many commands in gnuplot can be abbreviated to their first letter, so you can just type

```
q
```

to quit.

2.2 browse the help system

Type

```
help
```

to get a list of help topics. On Windows you get a browseable help thingy. On other systems you may find it helpful to type

```
help COMMAND
```

for specific help, where COMMAND is the name of any command.

2.3 plot a function

Try

```
plot sin(x)
```

In Windows and X11 the plot will be displayed in a window. Some buttons

are available to tweak formatting - these settings are saved internally for future plots.

Function plots like this are useful for fitting a function to a data set (to show that a trend follows an equation).

2.4 plot a more complex function

You can use the same mathematical operators as in the C language to build a more complex function. You can use any integer or real values, and you can use any of the built-in functions in Table 1. Type in

```
help expressions functions
```

for a complete list. Try using a combination of functions to plot a curve.

Table 1: Mathematical functions built into gnuplot, source `http://w2.syronex.com/jmr/tex/texsym.old.html`

Function	**Returns**
abs(x)	absolute value of x, $\|x\|$
acos(x)	arc-cosine of x
asin(x)	arc-sine of x
atan(x)	arc-tangent of x
cos(x)	cosine of x, x is in radians.
cosh(x)	hyperbolic cosine of x, x is in radians
erf(x)	error function of x
exp(x)	exponential function of x, base e
inverf(x)	inverse error function of x
invnorm(x)	inverse normal distribution of x
log(x)	log of x, base e
log10(x)	log of x, base 10
norm(x)	normal Gaussian distribution function
rand(x)	pseudo-random number generator
sgn(x)	1 if $x > 0$, -1 if $x < 0$, 0 if $x = 0$
sin(x)	sine of x, x is in radians
sinh(x)	hyperbolic sine of x, x is in radians
sqrt(x)	the square root of x
tan(x)	tangent of x, x is in radians
tanh(x)	hyperbolic tangent of x, x is in radians

Notice that the function used appears as a label in the key in the graph output. This is good for explaining what the fit is to a reader.

2.5 terminal tip

You can use the up and down arrows to cycle through recent commands.

2.6 plot some data from a file

Create a plain text file with 2 columns. Something like:

```
# x y
0 10.11
2 20.01
6 28.00
14 35.66
30 42.10
```

If needed (usually not) you can set the working directory to the same one as your file with gnuplot's `cd` command, e.g.

```
cd C:\temp
```

To plot data from a file use double quotes:

```
plot "myfile.xy"
```

Note: on Windows watch out for hidden file extensions - it might add a ***.txt*** *to the end that you can't see.*
To add a title to your data set:

```
plot "myfile.xy" title 'group A'
```

Note: it's better to rewrite rather than copy-paste these commands as quote characters can get mangled
In the plot window:

- scroll mouse wheel to change y-axis range
- scroll mouse wheel with shift held down to change x-axis
- scroll mouse wheel with ctrl held down to change zoom

2.7 using a script instead of the terminal

Most of the time you will want to run gnuplot automatically. You use the same commands as in the terminal.
Make a plain text file `myscript.gp`:

```
set terminal wxt persist
plot sin(x)
```

`set terminal wxt` tells gnuplot to display our results to the screen using wxWidgets, and `persist` tells it to keep the display window open. Normally we would

just write to a file. You can run gnuplot from the command-line:

```
gnuplot myscript.gp
```

You could might write a script or a programme to do that. You can also open the file from within a gnuplot terminal:

```
load "myscript.gp"
```

Default gnuplot graphs are ugly, so we can use scripts to store presets and formatting for our graphs.

2.8 output plot to an image file

To output a plot to a file there are three steps:

1. Tell gnuplot **what type** of graphics to generate `set terminal TYPE`
2. Tell gnuplot **where** to put the output file `set output FILE`
   (creates an empty file of this name)
3. Use the `plot` or `replot` command to **write** the file

If we have already done a `plot` command, `replot` is shorthand. You can check the available graphics types by typing:

```
set terminal
```

For something amusing try making a gnuplot script containing:

```
set terminal dumb
set output "output.txt"
plot sin(x)
```

When saving files we prefer vector images, because they scale to any size without loss in image quality:

- LaTeX uses `eps` (enhanced post script) natively,
- we can also save `svg` (scalable vector graphics) for use on the web.

If we prefer raster/bitmap graphics (perhaps for making an animation using GIMP or ImageMagick), then our best option is probably `png` (portable network graphics), because it is compressed, but also **lossless**, and is native to web browsers. `jpeg` (joint photographics expert group) is there as an option, but I would avoid it because it uses **lossy** compression which will damage your image quality, especially if you need to resize it later. The different terminal types have slightly different options. Some good defaults are:

```
set terminal svg size 512,256 fname 'Verdana' fsize 10
set output 'output.svg'

set terminal png size 512,256 enhanced font 'Verdana,10'
set output 'output1.png'
```

If your system supports it, the `pngcairo` type is usually nicer-looking than the plain `png` type:

```
set terminal pngcairo size 512,256 enhanced font 'Verdana,10'
set output 'output2.png'
```

Compare the difference. To output an eps file, the best way is to use the postscript option with eps extensions:

```
set terminal postscript eps enhanced color
set output 'output.eps'
```

Most pdf viewer programmes should be able to load an eps. You will see that by default the fonts are tiny compared to the png output, and the lines very thin. To double the font size, and line thickness we need to tell postscript the size of the image in **inches** (pixel size divided by 100), and then give it double-sized font and line thickness.

```
set terminal postscript eps size 3.5,2.62 enhanced color font 'Helvetica,20'
linewidth 2
```

Any bigger than 3.5 can be a bit big for the image boundaries.
If you are in a terminal, you can set the output back to the display window:

`set terminal wxt` (wxWidgets toolkit is used to display a window)
`replot`

NB: You will see that there are also options to save directly in LaTeX formats. If you try this you will see that the output is not an image at all, but actually a long series of commands, which tell LaTeX how to build the graph out of vectors, step-by-step. (I would avoid this option and stick with eps).

2.9 multiple plots in one chart

To plot several data sets or functions in one chart, use the same `plot` command, but separate the plots with commas:

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

2.10 manually set data ranges

You can use the terminal to specify the ranges on each axis that should be used. This is an option to the `plot` command.

```
plot [-pi:pi] [-1.2:1.2] sin(x)
```

The first two variables tell gnuplot to use $\pm\pi$ on the x (dependent) axis, and we have some sensible-looking ranges to fit a sine curve on the y-axis.

```
plot [0:50] [0:50] "myfile.xy", log(x)
```

Here we plot our data file with x and y ranges that fit our values. Maybe we zoomed out in the previous exercise and found roughly 0 to 50 to be a good fit? But we are also plotting a log function, because it looks like our values have some sort of growth function, and we think we can approximate this to a logarithmic function of x.

If the key labels are cluttering the display, we should increase the axis ranges. Our log function is probably too small. You will need to tweak the values...perhaps:

```
plot [0:60] [0:60] "myfile.xy", log(x) * 12
```

We probably want to add a title to the data set from the file, but not to the fitted plot.

2.11 adding lines or error bars to a plot

Try a plot command as before, but add `with ...` to the end.

```
plot "myfile.xy" with lines
```

Other 'with' options are `linespoints, dots, impulses, steps`. If you add a third column to your data file you can use `with yerrorbars` or `with xerrorbars`. A fourth column allows `with xyerrorbars`.

In the sciences, you would normally plot points with yerrorbars, and perhaps provide a second plot that approximates a fit. If you want to plot lines with error bars you need to plot the data twice:

```
plot "myfile.xy" with lines, "myfile.xy" with yerrorbars
```

2.12 titles, axes, keys, and tics

All of these labels are controlled with the `set` command. You need to do these before plotting, so if you make a change afterwards, do a `replot`.

`help set` (full list)
`help set border` (help about borders)
`plot sin(x)` (draw a basic plot)
`set title "graph title"`
`set ylabel "manure (lbs)"`
`set xlabel "feed per day (teaspoons)"`
`set xtics nomirror` (take tics off opposite side)
`set ytics nomirror` (take tics off opposite side)
`set border 3` (draw only left, bottom borders)
`set key at 6,1` (move the key to a data point position)
`replot`

2.13 style

You can add style to your plots to make lines easier to differentiate, and graphs either prettier for display, or with thicker lines for printing clearly. I wrote a guide but have since found this website, which has a much nicer-looking example to try:

`http://www.gnuplotting.org/attractive-plots/`

2.14 making a bar graph

And you can draw box plots, but it's quite hard to find in the help system:

```
plot sin(x) with boxes
```

Don't forget to add a error bars plot to these too (where appropriate).

3 And Furthermore...

This website is awesome. It has lots of articles with code on how to do some more interesting types of visualisation with gnuplot, **including heatmaps, 3D plots, colour maps, animations**, and making your gnuplot visualisations look pretty.
`http://www.gnuplotting.org/`

3.1 draw a 3D surface plot (splot)

`plot` draws in 2D, `splot` draws in 3D. Type in:

```
help splot
```

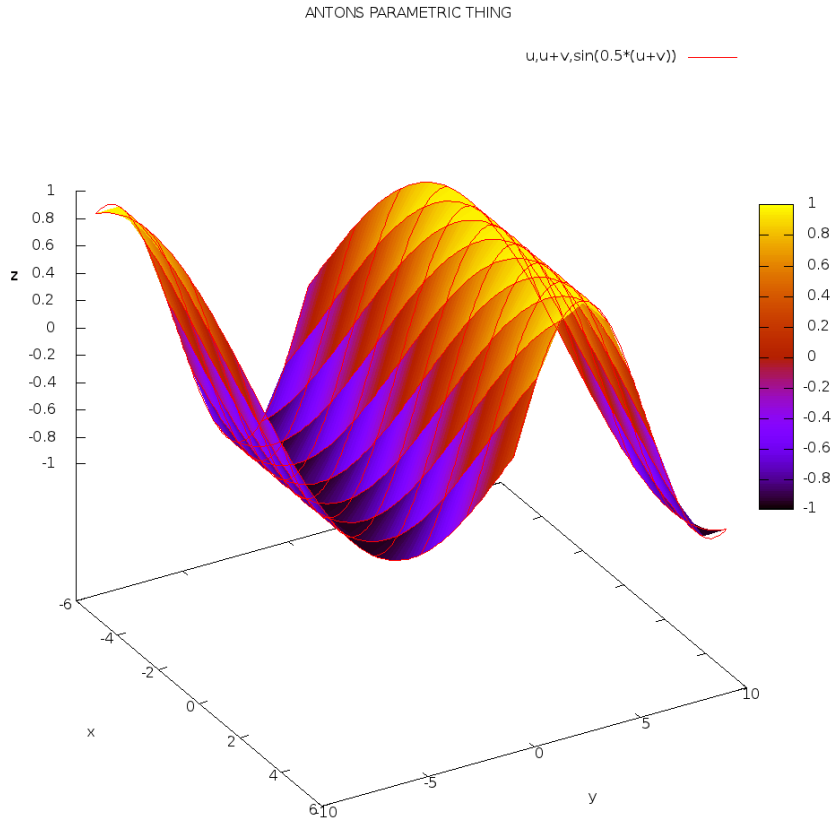

Figure 1: a parametric mode "splot" with hidden3d option (depth sorting) disabled

to get the options of the command.

```
splot sin(x * y)
```

You can use the mouse to move the plot around and find the ideal viewing angle. To save these settings to a gnuplot script:

```
save set "myfile.gp"
```
If you want to read about parametric plots you can do something like Figure 1.

A