ASTR4004/ASTR8004 Astronomical Computing Lecture 05

Christoph Federrath

9 August 2019

Plotting, Movies

1 Plotting data with gnuplot

1.1 Simple plots of data from ASCII text files

Gnuplot is a powerful plotting tools that can create plots from analytic functions and from tabulated data (e.g., from ASCII text files). Gnuplot is very useful for exploring data, plotting them in different ways, etc., even if that data is on a remote server. It can also be used to produce publication-quality figures, however, that requires tweaking a lot and making scripts, some of which we'll do here. Ultimately, IDL or python produce the 'nicest' plots with somewhat less effort, but making plots look 'perfect' always takes some fiddling with plot command options, etc., whichever programming language or package you are using.

1. Let's make some simple plots first. Change into your `$HOME/astro_comp/` on the MSO server (e.g., malice). Now look into `EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data` with `more` or `less` or `cat` (the file should still be there from the previous lecture or you can download it again). You will see that there is a header in the file with 10 lines (showing the mean, standard deviation and other statistical moments of the distribution function), followed by an empty line and then 4 columns with data in them.

2. Now go back to the shell and start gnuplot. First, plot column 1 against column 3 in the data file. Use this command:
   ```
   gnuplot> plot "EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data" using 1:3
   ```
   Note that you can also use auto-completion of the file name in gnuplot by hitting the tab key, just the same as in the shell. Ok, so this should show the density PDF in that file, which should look pretty close to a Gaussian.

3. Now replace "plot" with "p" and "using" with "u". This should do exactly the same as before, but is a bit more compact, i.e., no need to write out each gnuplot command, you can simply use the first letter in most cases and it will do the job.

4. Now plot the y-axis logarithmically. Do this with
   ```
   gnuplot> set log y
   ```
   and next enter the plot command from before again. Note that you can bring up the most recent last commands again, simply by using the up-arrow key; this should bring up the last few commands used. The same works in the Bash shell, which is very useful in

case you made a mistake when typing the command, you can bring up the previous one instantly and correct only the bits of the command that didn't work or that you want to change—for example if you wanted to keep everything the same, but instead plot data from one of the other files in the directory.

5. Now plot on top of the previous plot, the respective data columns from file with number `*0060*`. Do this simply by adding to the end of the previous gnuplot command:
   `, "EXTREME_hdf5_plt_cnt_0060_dens.pdf_ln_data" using 1:3`
   This should now display two curves (or set of points) plotted on top of one another. You can add more lines/curves simply by appending more to the plot command. For example, if we wanted to plot now also a constant line at $y = 10^{-3}$, we'd simply add '`, 1e-3`' to the end of the previous gnuplot command line and we should see the horizontal line at $y = 10^{-3}$.

6. You can change the x and y axis labels with:
   ```
   gnuplot> set xlabel "log-density contrast"
   gnuplot> set ylabel "PDF"
   ```
   ...then plot again and see if the axis labels have changed.

7. Ok, so this is all fine, but it usually doesn't look very nice. Gnuplot is really good for making a quick plot of some data, but making it pretty needs a bit more tweaking. Now lets see what can be done to make nicer figures, generate gnuplot scripts and automatically write postscript figures to a file.

1.2 Fitting data with gnuplot

Gnuplot can be used to fit model functions to data.

1. Look at the data file http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/05/RSAA_publications.txt, which contains RSAA's number of publications from year 2011 to 2018 and the cumulative number of citations extracted from https://rsaa.anu.edu.au/research/publications.

2. Plot the number of publications as a function of year. Then shift the data in the x-axis by -2011, such that year 2011 becomes 0.

3. Define a linear function $f(x) = ax + b$.

4. Now fit the RSAA publications data (note the shift in years, such that 2011 becomes 0):
   ```
   fit f(x) "RSAA_publications.txt" u ($1-2011):2 via a,b
   ```

5. Plot the fit on top of the data.

1.3 Making scripts for gnuplot and generating postscript figures

1. Download the gnuplot script template `gnuplot.p` from http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/05/gnuplot.p (or via the link on the course web page).

2. Look into the file and go through each line step-by-step and see if you can make sense of the commands. Similar to Bash scripts, gnuplot just goes through the script line-by-line and executes the commands as if you entered them one-by-one in the interactive gnuplot

mode. The advantage of the script is obvious: you don't have to write it all to the gnuplot prompt and you can very quickly regenerate the same plot and/or make little changes easily. So it's usually a good idea to script a plot that you make, in order to come back to it any time later, in order to reproduce the same plot. Running the script is then achieved simply by loading the script in gnuplot:
```
gnuplot> load "gnuplot.p"
```
which executes all the commands in the script line-by-line.

3. In this particular script, we have already switched to a different output type (or terminal); in this case to 'postscript' (`set term post eps`), which generates `.eps` figures. Such figure files may contain vector graphics (much preferred, because scalable), but can also contain pixel graphics (for example maps). Note that the graphics editor `inkscape` is very useful, because it can handle vector graphics and can be used to modify figures, keeping the advantages of vector graphics. In contrast, the popular `gimp` graphics editor can only handle pixel graphics. So while you can load an image containing vector graphics in `gimp`, it will be automatically converted to a pixel graphics image and thus loses all the advantages of scalable vector graphics.

1.4 3D plots with gnuplot

1. Lets make a simple 3D plot of the previous figure in gnuplot. Use:
```
splot "EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data" u 1:2:3
```
This shows the same density PDF as before, but now as a 3D plot. Note that you can turn the plot around by dragging it with your mouse.

2. Note that the 2nd column in the data file does not contain any useful data (it's all zeros), so it does not contain any information in the 2nd axis of the plot. Lets replace the 2nd column with the 4th column of the data file (which contains the cumulative distribution function):
```
splot "EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data" u 1:4:3
```
This will now show a true 3D plot; i.e., the data in the 4th column of the file is now plotting along the 2nd axis of the 3D plot.

3. One can do more advanced things, for example changing the point types and adding a colour bar, e.g., by appending to the end of the previous line:
```
splot ... u 1:4:3 with points palette pointsize 1 pointtype 6
```

2 Making movies from a time series of plots/images

1. First we have to write a gnuplot script that writes out `.png` figures for each of the PDF data files (the time sequence of the PDF) in the tarball from last the lecture.

2. We can use the basic gnuplot script from before, but in order to make png figures, lets change the gnuplot term to png:
```
set term png size 800,480 enhanced font "Helvetica,14"
```

3. Then after some other formatting definitions and settings (e.g., line types, log, key position, and axes labels; see script template from earlier) we make a gnuplot loop like this:
```
do for [i=20:90] {
    infile = sprintf('EXTREME_hdf5_plt_cnt_%04d_dens.pdf_ln_data',i)
    outfile = sprintf('frame_%04d.png',i-20)
    set output outfile
    p [-10:10] [1e-5:1] infile u 1:3 w lp ls 3 t sprintf('time=%04.0f',i)
    print outfile." created"
}
```

4. When you run the script, it should generate a list of figures `frame_0000.png`, `frame_0001.png`, ..., `frame_0070.png` and print to the screen that those files were created.

5. Now we need to use `ffmpeg`. You might want to install this on your own computer to be able to make the movie, or you can use it on malice. At the moment `ffmpeg` is located in `/pkg/linux/anaconda3/bin/`. You can add that directory to your PATH, by adding to your `.bashrc`:
   ```
   export PATH=$PATH:/pkg/linux/anaconda3/bin
   ```

6. Now that we have the still frames, we can make a movie. The simplest `ffmpeg` command to make a movie from a bunch of still frames is:
   ```
   > ffmpeg -i frame_%04d.png movie.mpeg
   ```
   where you have files `frame_0000.png`, `frame_0001.png`, etc. previously generated with gnuplot.

7. You can play the movie file with `ffplay`. However, it won't play well over the network, so I recommend to copy it to your local computer (`scp`) and play it directly on your computer rather than in a window over the network.

8. So this is ok, but looks a bit pixelated. For a nicer movie, we have to increase the bit rate by adding the option `-b:v 5000k` to `ffmpeg`. This will greatly increase the bit rate and thus the quality of the movie output.

9. There are lots more advanced options in `ffmpeg`, for example cropping or extracting frames from movies and changing the encoder. It is one of the most powerful movie conversion/processing tools.