ASTR4004/ASTR8004
Astronomical Computing
Lecture 05

Christoph Federrath

17 August 2017

Version control, Plotting, Movies

1 Version control

1.1 Basics of version control

1. Imagine you work on a code development project or you write a paper and you'd like to keep track of changes and earlier versions of your code/paper. A neat way to achieve this is to use version control software/tools.

2. Popular version control frontends (partially free or commercial, if you want repositories to be private or shared by a large number of developers) are provided by services such as `http://www.bitbucket.org` or `http://www.github.com`. These are primarily webpages that allow you to share your code with others, browse the source code and keep track of changes. For bigger projects, you can also establish teams that work on the different pieces/modules of the same code.

3. A key element of these version control systems is that they keep their own files inside the directory(ies) of your code, in order to store and update changes – basically to keep an entire history of what's been going on with the code; who made changes, what changes, and when. They also allow you to revert to previous versions in case some bugs slipped into the code or something broke at some stage.

4. In order to start a versioned code, you will need to install a version control system or version control software. Some of the first bigger ones were Concurrent Versions System (`cvs`) and Subversion (`svn`). Nowadays Mercurial (`hg`) and Git (`git`) are popular version control systems. Here we will focus on `git` with some examples, but Mercurial is very similar.

1.2 Starting a `git` repository

1. To get started with `git`, you have to install it on your computer. Then we pick a directory with source code or any files that we want to version-control and change into it.

2. This is how we start a Git repository in that directory (say `mycode/`):
```
> cd mycode/
> git init .
```

3. Then type `> git add [file1] [file2] [...]` to add all of the relevant files that you want to keep under version control.

4. Now we can check the status of the files by typing `> git status`. This brings up a list of changes and a list of files that are in the directory (and subdirectories), but that are not under version control. You can create a hidden file called `.gitignore` and add all of the files that you want git to ignore, so they don't annoy you every time you type `> git status`.

5. Finally, once you are happy with the changes (say you added all files or if they were added earlier you may have modified them when you develop your code further), you type
   `> git commit -m 'message describing change(s)' [file_to_be_commited]`.

6. This last command commits the change to the file and creates a new version in the system, which you can revert to later. Or when you make further changes to the same file, you can compare it to the previously committed version by typing:
   `> git diff [file_to_check_changes_since_previous_commit]`

1.3 Uploading/Communicating repository to server

1. Up to this point, the version-controlled code just lives on your own computer, but we also want to upload the code to a safe location on a server, in case something happens to your own computer, but also in case we want to share the code with specific people or with the entire public.

2. To do this, we can create an account on `bitbucket.org` and start a new repository or import the existing repository from the previous steps.

3. Once configured correctly, i.e., providing the correct URL and paths, such that your computer knows that it should upload changes in the local repository to the bitbucket or github server, you can simply type
   `> git push`
   to push all the changes in the local copy to the server. This will then allow you to browse the code online, to share it and to view changes to the version-controlled code in an internet browser (basically, it's just a nicer view of what you can get with `> git status` and `> git diff` on your local working copy).

4. However, having the code on the server will also allow you to share it with others. There are different options to do this, for example, cloning a repository, or forking or branching a repository. We won't go into details of that, but essentially, this is allows one to get a copy of the repository and continue working on it within their own local copy, such that the global source repository is not damaged. However, changes by another user that are deemed useful for the global copy of the repository can be merged in by issuing a so-called *pull-request*.

5. In summary, when you develop code, you should always use a version control system. What seems awkward at first is actually extremely useful once you get into trouble with keeping track of changes based on your own strategies. Version control software provides a standardised way of keeping different versions of a code or simply a bunch of files that undergo regular changes.

2 Plotting data with gnuplot

2.1 Simple plots of data from ASCII text files

1. Lets make some simple plots first. Change into your `$HOME/astro_comp/` on the MSO server (e.g., misfit). 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.

2.2 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_8004/04/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.

2.3 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`

3 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 setting (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 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.

6. 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.

7. 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.

8. 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.