Chapter 13

Plotting LOTAAS survey results (Group B, week 4)

In this exercise you will take the LOTAAS data set used in week 3 to create some plots. Unlike some other data sets that you may encounter the data are spread over a few hundred small files spread over a directory tree. The goal for this exercise is to write a program that gathers the necessary data and creates plots for these data. Goals for this week are to:

- write a small program split into several modules,
- create properly structured modules,
- document code properly, and
- create the basic plots (scatter plot, line plot, histogram).

Note: properly label your plots also have a look at the hints section below.

13.1 Parsing `.bestprof` files

As you have seen before the `.bestprof` files contain easily extracted data about each pulsar candidate. Before we start creating plots we first have to write a parser for these data files. As last week, use should write the parser yourself. Do not use Pandas, Numpy or Scipy data reading functions or `csv` module.

- Create a Python module called `bestprof.py`, this is where the `.bestprof` file handling will go.
- Have a look through a `.bestprof` file with a text editor. Note that the file consists of two parts, a header with key-value pairs and a simple two-column list of values. The latter is the pulse profile split into a number of bins (how many is a parameter chosen by those who run the LOTAAS survey).
- Create a function `parse_bestprof` that accepts a filename (as a string) and that returns a dictionary for the key-value part of the header and a list of values for the list part of the `.bestprof` file (i.e. for the pulse profile). *Note: Python allows multiple returns from functions.*
- Note: turn the interesting parts of the header into the appropriate data types. At least do this for the Dispersion Measure DM, the pulse period P, pulse period derivative $\dot{P}$ and reduced chi-squared χ_r. Do the same thing for all values in the pulse profile.
- Extra credit: split the file parsing into two functions, `parse_header` and `parse_profile`, that get called by `parse_bestprof`.

13.2 Walking a directory tree

Now that you are able to parse individual `.bestprof files` you must must write some code to be able to find all relevant files.

- Create a Python module called `finder.py` that will contain the code that searches for `.bestprof` files.
- Look up the function `os.wallk`, as it will allow you to search for file much like the BASH `find` command does.
- Write a function called `find_bestprof` that takes a directory (as a string) and returns a list of `.bestprof` files names. *This involves writing a loop over all subdirectories using the `os.walk` function, checking the filenames and accumulating the relevant filenames in a list that you return.* Make sure that you return not just sub directories but the full paths to the files.
- *You can check your code by comparing the output of the BASH `find` command with the output of your code. Do both find the same number of files?*

13.3 Plotting survey results

You will now tie together all the code you wrote before and create plots for the whole set of survey data.

- Create a Python module `lotaas_plotter.py` and make sure that it is saved in the same directory as the `bestprof.py` and `finder.py` modules.
- Create a `main` function function that accepts a directory name (as string) and that will contain all functionality of this program.
- Confirm that you can `import` the functions defined in the modules you created earlier on. Fix it if this does not work.
- Call the `find_bestprof` function to gather a list of `.bestprof` filenames. Loop over this list and call the `parse_bestprof` for each of the file names. You should create a list of tuples for the output of all these functions.
- Below are three plotting functions that should all be called from the `main` function.
- Create a function `plot_p_pdot` that uses Matplotlib to create a scatter plot of logarithm of P versus the logarithm of $\dot{P}$. You are allowed to throw away values for which you cannot take the logarithm. If you are unsure about what these plots look like use the internet to find some examples of P-Pdot diagrams. Save the resulting plot in a file called `ppdot.pdf`.
- Create a function function `plot_profile` that creates a line plot of the pulse profile value versus bin number. Make sure to include the period and reduced chi-squared values in the title of the plot. Save the profile of the brightest signal (or of the pulsar signal if you can find it) to a file called `pulseprofile.pdf`. Extra credit: create a plot of value versus time (you will need to grab a number from the header to make this work).
- Create a function called `plot-histogram-p` that creates a histogram of all the pulse periods in the suvey data. Save the resulting plot to a file called `p_histogram.pdf`.
- If you did not do so already: Add a `if __name__ == '__main__':` line to your program and call the main function from the block of code behind it.
- Using the `sys.argv` variable it is possible to create a script that accepts command line arguments — but doing so, is the quick and dirty way. If you want you can change your `lotaas_plotter.py` to accept the directory where it needs to search as a command line argument. *The "proper" way of creating command line arguments is to use a module like `argparse`.*

13.4 Hints

- If you are working on university computers, copy your data to the scratch disk in a directory with your name. Work in that directory and only save your scripts to your home directory (because the network is quite slow).
- To do path manipulations in Python (like creating or concatenating full paths) look at the `os.path` module of Python. Interesting functions in that module are `os.path.join`, `os.path.split`, `os.path.abspath`, and `os.path.realpath`.
- If you are seeing old plots when you are creating new ones with Matplotlib you should probably use the `matplotlib.pyplot.clf` function to clear the old figures.
- A good rule of thumb for the number of bins in a histogram is the square root of the number of items in the data.

13.5 Handing in

You should make a gzipped tarball (called `week4.tar.gz`) with the following files:

- `bestprof.py`
- `finder.py`
- `lotaas_plotter.py`
- `ppdot.pdf`
- `pulseprofile.pdf`
- `p_histogram.pdf`
- `report-week-4.pdf` (Only if you want to explain your methods.)