PSRCHIVE Python interface activities

Paul Demorest

June 13, 2018

1 Introduction

The PSRCHIVE Python interface is installed as part of the IPTA 2018 Docker image. A couple quick notes about using it:

1. This currently only works with Python 2. After logging into a linux shell in Docker, run `source activate python2` to get a python2.7-based environment.
2. I usually prefer to work in ipython (rather than notebooks), so the examples in the slides all use this. You can start ipython with `ipython --pylab=tk` in order to get matplotlib commands automatically imported. This should all work fine in notebooks as well. If needed, you can `import matplotlib.pyplot as plt` and prefix all plotting commands with `plt.`
3. Some useful information can be found at `http://psrchive.sourceforge.net/manuals/python`

2 examples

1. Load any example archive from this morning's data set using `psrchive.Archive_load`. Use ipython tab or read the C++ reference guide to see what `get_` methods are available for retrieving metadata.
2. For all `*.rf` files in the `Part1/archive_files` directory, write a Python script that will print the filename, number of subintegrations, and number of channels. What is the corresponding `vap` or `psredit` command? (Tip: Check out the python `glob` module for getting lists of filenames.)
3. Pick one file. For all subintegrations in the file, see if you can figure out how to print the start time as MJD.

3 Basic plotting

1. Run some of the simple examples from the presentation.

2. Load an archive from the data set and create an image plot of profiles as a function of frequency, integrated over time. This is similar to `psrplot -jT -pG` or `pav -GT`. See if you can make both dispersed and dedispersed versions. (Tip: See the matplotlib `imshow()` command.)

3. Do the same thing for profiles versus time (subintegration), integrated over frequency channel.

4 Extracting data from an archive

1. Use the data file `t121208_041217.rf`. From the baseline-removed data array, find the maximum value in each profile. (Tip: see the numpy `max()` function.)

2. The result of this is a function of time (subintegration) and frequency. This is a basic pulsar dynamic spectrum – plot it as an image.

3. Try using the `Archive.bscrunch(n)` method to reduce the number of profile bins by factors of $n = 2, 4, 8, \ldots$ See how this affects the S/N in your dynamic spectrum.

4. Advanced version: Rather than taking the profile maximum, see if you can use the `ProfileShiftFit` class to determine the profile scale factor versus time and frequency for your dynamic spectrum. Compare with the results from the simpler method.

5 Ideas for additional work

1. Refer to the examples from the previous session's PSRCHIVE activity. See how many could be implemented in Python.

2. Take `t121208_041217.rf` and tscrunch it. Use `ProfileShiftFit` to subtract an appropriately scaled/shifted version of the standard profile from each channel. Plot the difference (e.g., profile residuals) versus frequency. How consistent are the data with a single template shape? (Tip: Check out the `ProfileShiftFit.apply_scale_and_shift()` method.

3. Repeat using the calibrated version of the same archive. How do these results compare?