PSRCHIVE tutorial - IPTA student workshop, 2018

Aditya Parthasarathy and Ryan Shannon

June 10, 2018

Abstract

In this tutorial, you'll learn to use the PSRCHIVE pulsar data analysis software. We shall learn how to view, edit and pre-process pulsar data for timing analysis. PSRCHIVE is an open-source, object oriented data analysis software that implements a wide range of data analysis algorithms for use in data calibration, statistical analysis and visualization. Please refer to van Straten et al. (2012) or visit *http://psrchive.sourceforge.net/* for a comprehensive description of the PSRCHIVE software suite.

1 General Introduction

The data that we are going to process today are commonly referred to as **archive data**. These files contain astronomical data recorded during a pulsar observation and are typical stored as a three-dimensional array of pulse profiles, the axes being time (sub-integration), frequency (channel) and polarization. The data have attributes called **metadata** that describe the various physical attributes of the observation.

A useful point to note here is that, many of the PSRCHIVE programs print out a brief help message when the *-h* command line option is used,
for e.g. *psrstat -h*. Descriptive usage instructions are also available online.

The data for this tutorial are stored in docker images. See Ryan's slide for the path to the data.

This tutorial is split into **4** parts. Each part accomplishes a particular processing task and produces a set of data products. In case you are held back in a particular part of the exercise but want to proceed ahead, do not worry (!), each directory is self-consistent and has everything that you need for that exercise.

Let's get started!

2 Viewing and evaluating pulsar metadata with *psredit* and *psrstat*

The directory **Part1** contains *archive files* that we will be processing during this tutorial.

The first step towards analyzing pulsar data is learning how to view, edit and evaluate pulsar metadata. Each archive file (in the *archive_files* directory) contain metadata that provide important and relevant information about the pulsar observation and the data recording.
The naming convention is pretty straightforward.

```
Filename: t120106_011259.rf
"t" denotes the pulsar backend used for the observation.
"120106_011259" denotes the yymmdd_hhmmss of the observation.
```

Let's start by querying the metadata of the archive files using *psredit*.

Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/psredit/
```

```
psredit <filename>.rf
```

Use *psredit* to explore the metadata of the archive files. Remember that you can find helpful instructions with the *-h* command line option. Try to find out the following:

- What are the dimensions of the data? How many polarizations, frequency channels and sub-integrations are present? Are these the same for all the observations?
- What type of observations are these?
- Which telescope was used to record these observations?
- What can you find out about the pulsar backend used for these observations?
- What receiver was used during these observations?

```
Write down your answers here (or edit the PDF!)
```

Now let's try using *psrstat* for evaluating the data.
Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/psrstat/
```

Try running *psrstat* on a single archive file. What difference do you notice between the outputs of *psrstat* and *psredit*?

```
Write down your answers here (or edit the PDF!)
```

Now, let's try to look at the signal-to-noise distribution of these observations.

```
In the archive_files directory (Part1),
psrstat -c snr *.rf
```

The above command will print out the S/N value of each archive file. Note that *psrstat* computes the S/N of the profile in the first sub-integration (subint), frequency channel (chan) and polarization (pol). We now know that each archive file has *nchan* number of frequency channels. Let's try using the *loop* functionality to print out the S/N values of a single archive file as a function of frequency channel.

```
In the archive_files directory (Part1),
psrstat -Q -l chan=0-nchan -c snr <filename>.rf
```

The -*Q* option prints out only the value, instead of key=value. The -*l* command loops over the specified parameter, in this case, *chan*. Redirect the output of the above command to a text file and try plotting (HINT: use *gnuplot*) the S/N distribution as a function of the frequency channel. Try playing around with the other parameters as well. The *snr_distribution* directory in *Part1* has the results to the above exercises including a *bonus* one, if you are interested! How do the S/N distributions look like? What other parameters did you use to plot the S/N?

```
Write down your answers here (or edit the PDF!)
```

psrstat is a very useful program that can be used in combination with other tools to diagnose problems in the data.

3 Using *psrsh* commands to pre-process the data

In the last section, we saw that the S/N value is computed from the first sub-integration, frequency channel and polarization for each archive file. We can use pre-processing commands to integrate different dimensions of the data before computing the value of any specified parameter.

Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/psrsh/
```

For example, if we want to compute the S/N values for each polarization but integrate the frequency channels prior to that:

```
In the archive_files directory (Part1),
psrstat -Q -l pol=0- -j fscrunch -c snr <filename>.rf
```

Note that this is very similar to a previous command where we computed the S/N ratios of a single archive file as a function of frequency channel. In this case, we are doing the same but with two main changes:

- We are looping over polarization rather than frequency channels
- We are using a pre-processing command *-j* to integrate all the frequency channels.

A list of pre-processing command can be viewed by running:

```
psrsh -H
```

This prints out three columns. The first column is the command name, the second column is a single-letter shortcut key and the third column is a short description of the command. So, for example if you want to print the S/N of the total intensity profile (p) integrated across the entire bandwidth (F) and across every sub-integration (T):

```
In the archive_files directory (Part1),
psrstat -Q -j FTp -c snr *.rf
```

Redirect the output of the above command to a text file and plot the distribution of the S/N ratios for all the fully integrated archive files.

- What difference do you see in S/N ratios before and after integration?
- Does the distribution look different?

Note: The above command might take some time to finish executing. Look for a text file titled *FTp_snr.txt* in the *snr_distribution* directory for answering the above questions.

```
Write down your answers here (or edit the PDF!)
```

4 Using *psrplot* to visualize data

The *psrplot* program can be used to visualize the archive data.
Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/psrplot/
```

The *psrplot* help instructions inform us that *psrplot -P* lists the available plot types. Use this information to plot the phase-vs-frequency image of the total intensity of the pulsar signal in each file. Remember that the total intensity profile is formed by using the *pscrunch (p)* pre-processing command.

Let us use the files in the *processed* directory in *Part2*. These are archive files produced after integrating in time (sub-integration).

```
In the processed directory (Part2),
psrplot -j p -p freq *.T
```

You can also specify the plotting device directly from the command line using the *-D* option.

```
In the processed directory (Part2),
psrplot -j p -p freq -D 1/xs *.T
```

What do you notice in these plots? How do you correct for that effect?

```
Write down your answers here (or edit the PDF!)
```

Try using *psrplot* to:

- Plot the total intensity profile as a function of frequency (HINT: loop over chan) for a single archive file.
- Plot the Stokes parameters after integrating over all the frequency channels for a single archive file.
- Plot the pulse phase vs time image of an archive file.

5 Excising radio frequency interference (RFI)

5.1 Automatic RFI excision using *paz*

The *paz* program can be used to automatically detect and excise narrow-band and impulsive RFI from the archive data files. Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/paz/
```

There are a couple of commonly used pre-processing commands that *PSRCHIVE* implements to do this:

```
zap median
zap mow
```

Let us use *psrplot* and *psrsh* job pre-processor to view an example of data corrupted by RFI and then use the *zap median* pre-processing command to automatically detect and zap it.

```
In the processed directory (Part2),
psrplot -p freq -jDpC -D 1/xs t120408_221525.T
and then use,
psrplot -p freq -jDpC,"zap median" -D 1/xs t120408_221525.T
```

In the first case, we use the commands *D,p and C* to de-disperse, integrate the polarizations and finally center the profile. You can see that there is a strong RFI signal present at around 1320 MHz. The second command includes the *zap median* pre-processing command. Do you see a difference in the result?
Try it with other archive files. How effective is the RFI mitigation?

```
Write down your answers here (or edit the PDF!)
```

You might notice that there is still some residual RFI left. To better characterize subtle RFI, the *zap median* algorithm can be configured to use any expression that is interpreted by *psrstat*.

Similar to *zap median*, impulsive RFI can be mitigated using *zap mow*. Take a look at the script *zap.psh* in the *Part2* directory.

```
#! /usr/bin/env psrsh

zap median window=24

zap median cutoff=3

zap median exp={$all:max-$all:min}

zap median

zap mow robust

zap mow window=0.1

zap mow cutoff=4.0

zap mow
```

Please read the online documentation or feel free to ask questions, if you want to know more about the above pre-processing commands.

You can use the *paz* program to load the above pre-processing script to automatically detect and mitigate RFI from all of the archive files. Create a new directory and store the cleaned version of the archive files output by *paz*

```
In the Part2 directory,
mkdir <new_directory>;
paz -J zap.psh -O zapped_profiles -e <ext> <path_to_data>
```

Note that *-J* is used to load the pre-processing script. The cleaned profiles are output to the directory *zapped_profiles* in this case. It will take a while to excise RFI from all of the archive files. Try running it for a few profiles and terminate the program.

Compare the RFI excised profiles with the original archive files. How different are they?

```
Write down your answers here (or edit the PDF!)
```

5.2 Interactive RFI excision using *pazi*

Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/pazi/
```

In the *processed* directory (Part2), you might have noticed archive files that are formed after integrating over time and polarisation. These files are given an extension: **.Tp**. Use these files for interactive RFI excision.

```
In the processed directory (Part2),
pazi <filename>.Tp
```

This will load two plotting windows. Use the *-h* command to learn how to use the tool.
Load a single archive file using *pazi* and try to:

- mitigate RFI in sub-integrations in the pulse phase vs time plot
- mitigate RFI in frequency channels in the pulse phase vs frequency plot
- repeat the above process until the archive file is clean of most RFI
- save the file

6 Calibrating data with *pac*

Let us now try to calibrate our data set. Prior to that, let us use *psrstat*, *psredit* and *psrplot* to evaluate and view the calibrated files, present in *Part3* directory (calibration_files).

What do you understand about the calibrated files? How are they different from the archive files?

```
Write down your answers here (or edit the PDF!)
```

Descriptive usage instructions can be found online:

```
http://psrchive.sourceforge.net/manuals/pac/
```

A simple first-order calibration is done in two stages:

- Create a database of calibrators
- Perform the calibration

```
In the calibration_files directory (Part3),
pac -w -u cf -k calib_db
```

The above command produces a database file named *calib_db*, specified using the *-k* option. The *-w* option enables creation a new database. The *-u* option specifies the extension of the calibration archive files.

Once a database is created, it can be used to calibrate the cleaned archive files. You can access the cleaned archive files from the cleaned directory in *Part3*. We will be using these files for the calibration. Create a new directory to store the cleaned, calibrated profiles.

```
In the calibration_files directory (Part3),
pac -d calib_db -O new_calib/ ../cleaned/*.zap
```

This uses the created database (calib_db) to create calibrated profiles (with .calibP extension) that are stored in the directory *new_calib*. Please note that there is already a directory titled *calibrated_cleaned* in the *Part3* directory that contains the calibrated profiles.

What differences do you notice between the calibrated and uncalibrated files? How do the Stokes parameters of the calibrated files look like? Hint: You can plot 2 files in the same PGPLOT window using the *-N* option in *psrplot*.

```
Write down your answers here (or edit the PDF!)
```

7 Generating arrival time estimates with *pat*

Now that we have calibrated and cleaned the data and have gone through how archive data is pre-processed, let's go ahead and generate arrival time estimates using *pat*.
The arrival time estimates are derived using a previously created standard template profile. You can take a look at this in the *standard* directory, in *Part4*. Are you able to view and evaluate the template profile using *psrplot*, *psrstat*?

7.1 Estimation of arrival times

Before creating the arrival time estimates, we can use psrsmooth to smoothen the standard profile

```
In the standard profiles directory (Part4),
psrsmooth -W J1744-1134_20cm_ana_PDFB4.std
```

This generates a new file (.sm extension) after using a Wavelet smoothing (default Sinc) routine on the input standard template profile.

Once we have a smoothed template and calibrated profiles, we can use *pat* to generate pulse arrival times.

```
In the standard profiles directory (Part4),
pat -FT -jp -A FDM -s <standard> -f ‘tempo2’ <calibrated_data>
```

The above command produces arrival times after frequency, time integration (-*FT*), and polarization integration (-*jp*). It uses a Fourier domain with Markov chain Monte Carlo algorithm (-*A FDM*) to compute the ToAs. The -*s* flag is used to specify the standard and -f, the output format.
Redirect the output of this command to a text file (J1744.tim). We will be using this to compute the timing residuals.

7.2 Using tempo2: a brief introduction

Once we have the ToAs, we can use *tempo2* to generate the timing residuals for this pulsar. In the *timing* directory (Part4), you will see that there are a couple of files: J1744-1134.par and *J1744-1134.toa*.
You will learn more about these in the **pulsar timing** tutorial sessions. For now, let us use the following command to generate the timing residuals.

```
In the timing directory (Part4),
tempo2 -gr plk -f J1744-1134.par ../
```

What do you see? Do the timing residuals look good? Are there any obvious problems that you can see?

```
Write down your answers here (or edit the PDF!)
```

References

van Straten W., Demorest P., Oslowski S., 2012, Astronomical Research and Technology, http://adsabs.harvard.edu/abs/2012AR9, 237