Introduction to PSRCHIVE

Willem van Straten

Swinburne University of Technology

10 May 2011 - NAOC, Beijing, China

What is PSRCHIVE ?

PSRCHIVE is:

- **not** a single monolithic program
- a suite of programs
  - integrated with the UNIX environment
- a C++ development library
  - python bindings also available
- a mature work in progress
  - development and testing are ongoing

PSRCHIVE is:

- Open Source
- widely used
  - Africa, Australia, Canada, China, Germany, Netherlands, United Kingdom, United States, …
- relatively well documented
  - http://psrchive.sourceforge.net

PSRCHIVE is:

- powerful!
  - sophisticated calibration
  - matrix template matching
  - advanced rotation measure estimation
  - unique rotating vector model fitting
  - digitization distortion corrections
  - custom virtual memory management
  - etc.

Why use PSRCHIVE ?

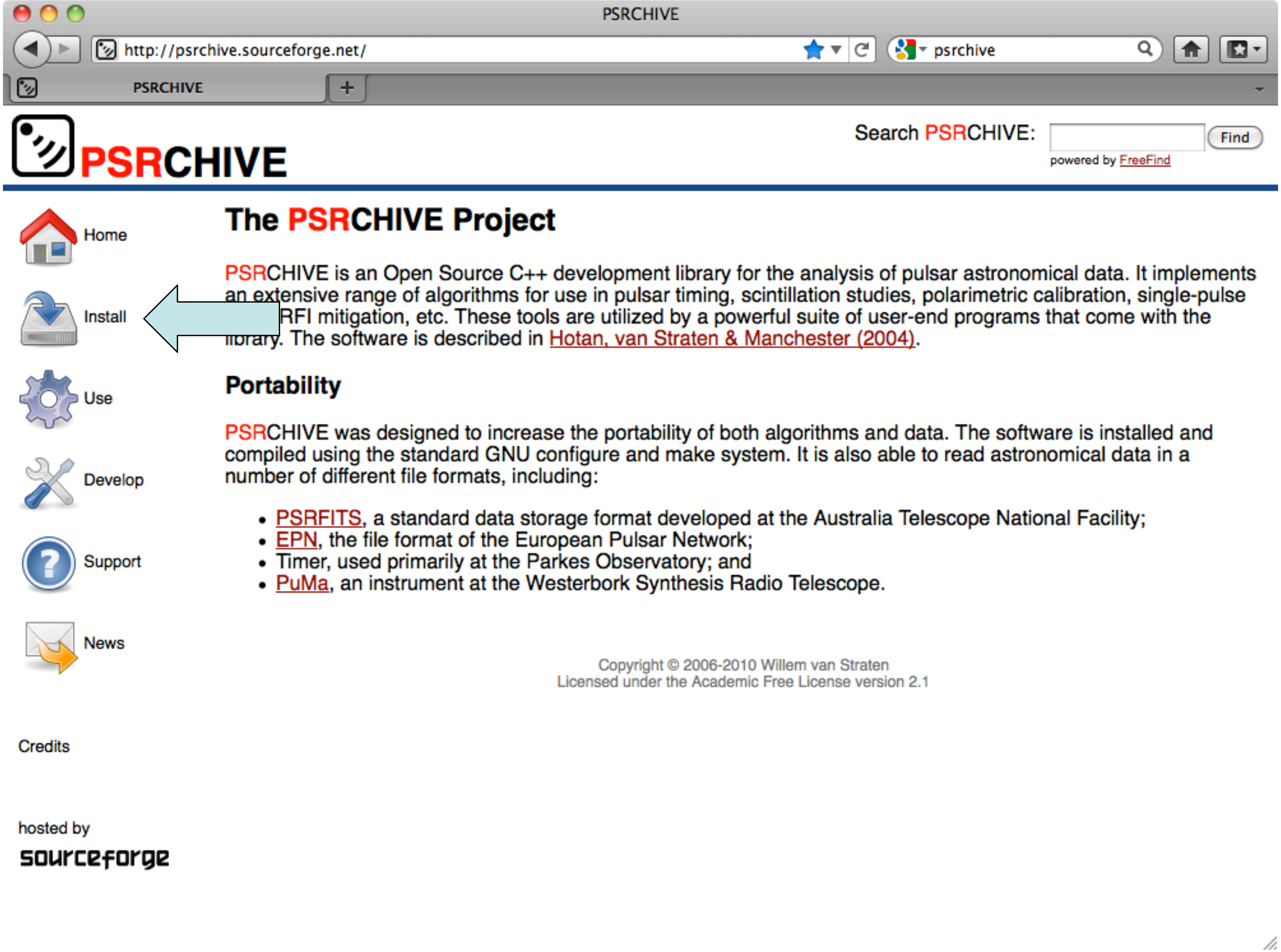
PSRCHIVE can:

- read/write many folded data formats:
  - PSRFITS, EPN, PRESTO, ASP, WAPP …
- perform many common tasks:
  - correct dispersion and Faraday rotation
  - calibrate instrumental polarization
  - excise corrupted data (e.g. RFI)
  - calculate arrival times
  - produce various publication quality plots

PSRCHIVE cannot:

- search for new pulsars:
  - sigproc, presto, etc. do this (used to refine S/N of survey candidates)
- reduce/fold time series data:
  - dspsr, sigproc, presto, etc. do this (dspsr uses psrchive)

Where is PSRCHIVE ?



PSRCHIVE

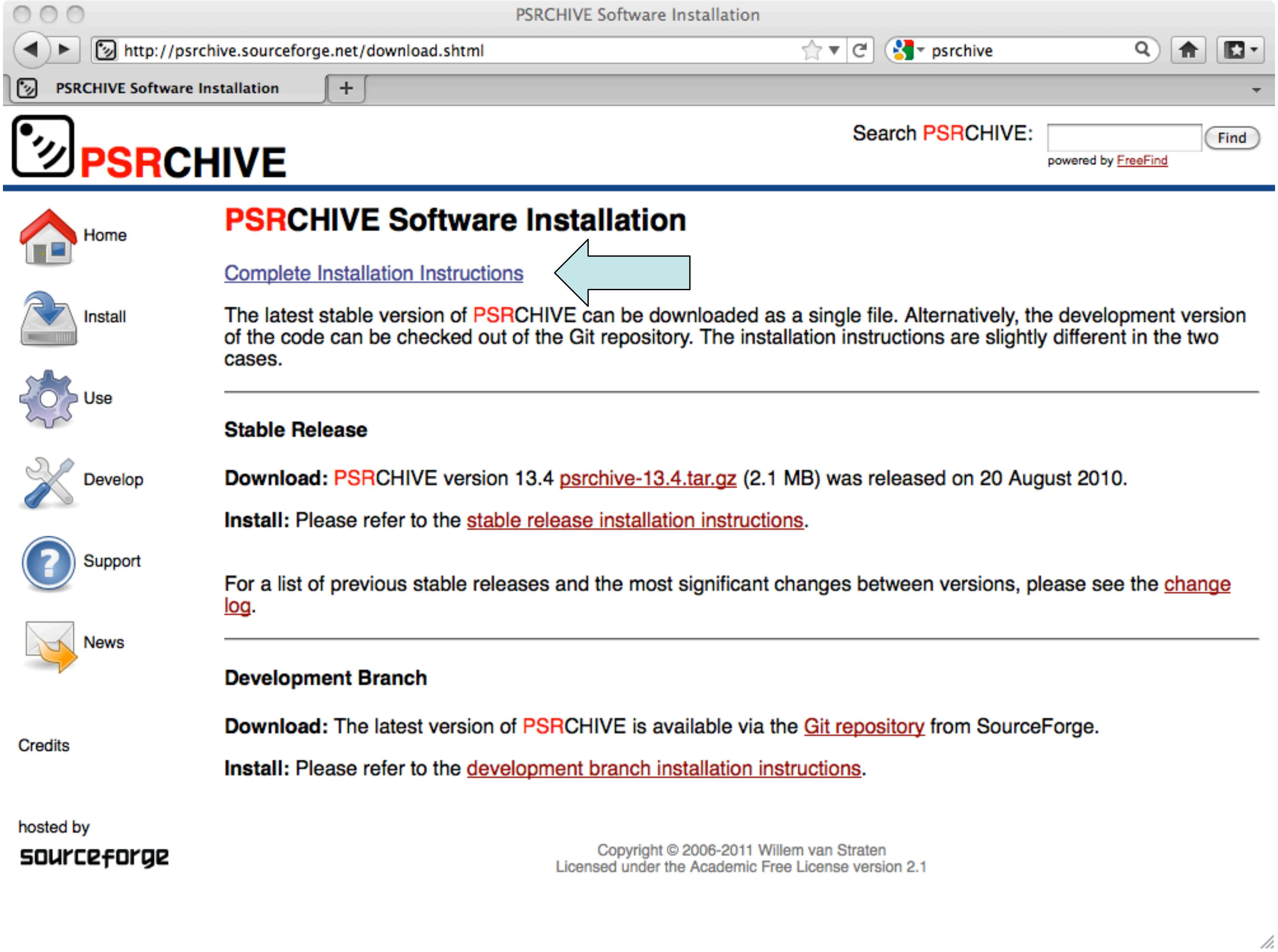
The PSRCHIVE Project

PSRCHIVE is an Open Source C++ development library for the analysis of pulsar astronomical data. It implements an extensive range of algorithms for use in pulsar timing, scintillation studies, polarimetric calibration, single-pulse RFI mitigation, etc. These tools are utilized by a powerful suite of user-end programs that come with the library. The software is described in Hotan, van Straten & Manchester (2004).

Portability

PSRCHIVE was designed to increase the portability of both algorithms and data. The software is installed and compiled using the standard GNU configure and make system. It is also able to read astronomical data in a number of different file formats, including:

- PSRFITS, a standard data storage format developed at the Australia Telescope National Facility;
- EPN, the file format of the European Pulsar Network;
- Timer, used primarily at the Parkes Observatory; and
- PuMa, an instrument at the Westerbork Synthesis Radio Telescope.



PSRCHIVE Software Installation

Complete Installation Instructions

The latest stable version of PSRCHIVE can be downloaded as a single file. Alternatively, the development version of the code can be checked out of the Git repository. The installation instructions are slightly different in the two cases.

Stable Release

Download: PSRCHIVE version 13.4 psrchive-13.4.tar.gz (2.1 MB) was released on 20 August 2010.

Install: Please refer to the stable release installation instructions.

For a list of previous stable releases and the most significant changes between versions, please see the change log.

Development Branch

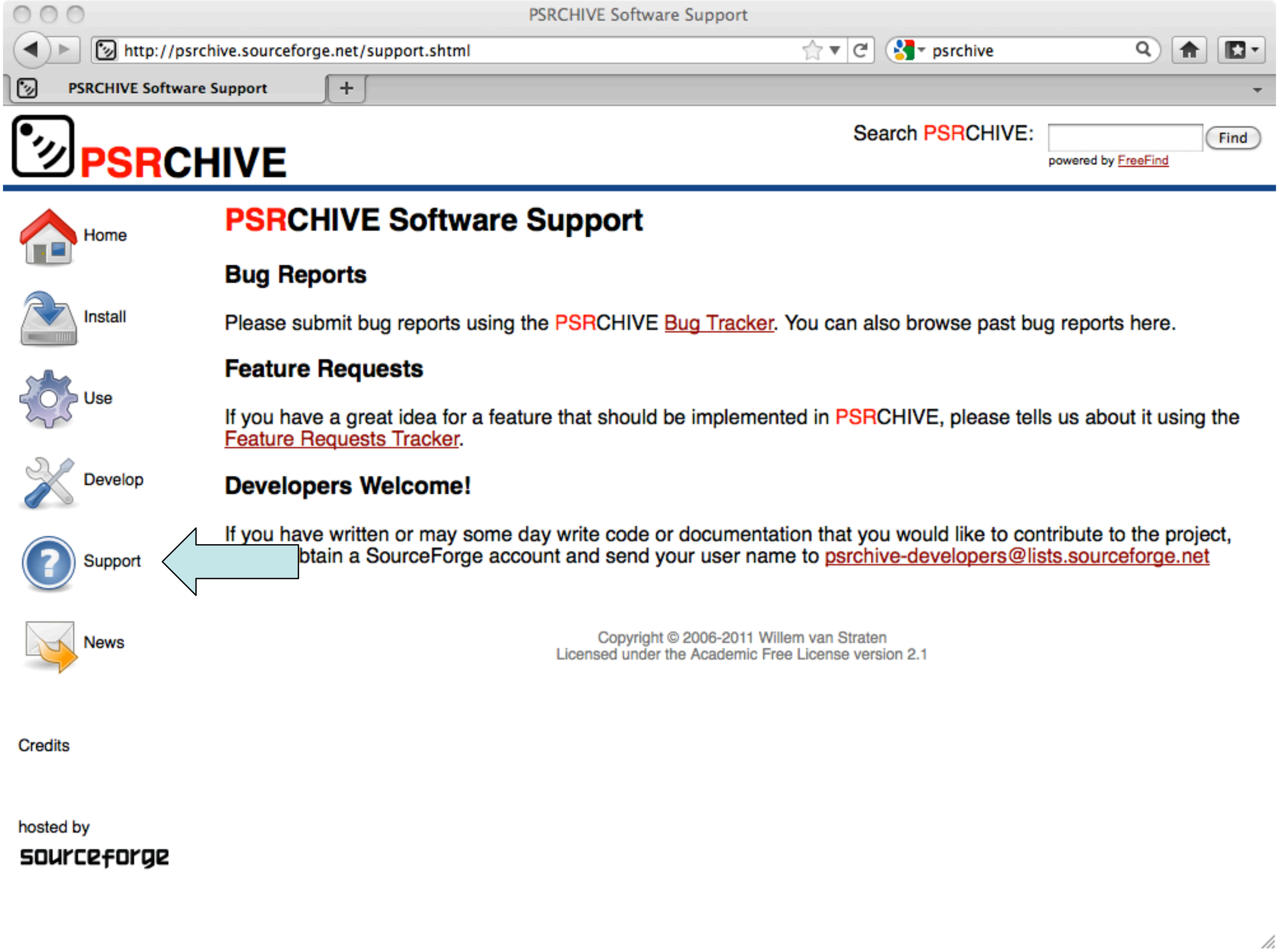
Download: The latest version of PSRCHIVE is available via the Git repository from SourceForge.

Install: Please refer to the development branch installation instructions.

Who is PSRCHIVE ?

PSRCHIVE Team:

- Cees Bassa
- Paul Demorest
- Aidan Hotan
- Andrew Jameson
- Mike Keith
- Jonathan Khoo
- Aris Noutsos
- Willem van Straten
- …



PSRCHIVE Software Support

Bug Reports

Please submit bug reports using the PSRCHIVE Bug Tracker. You can also browse past bug reports here.

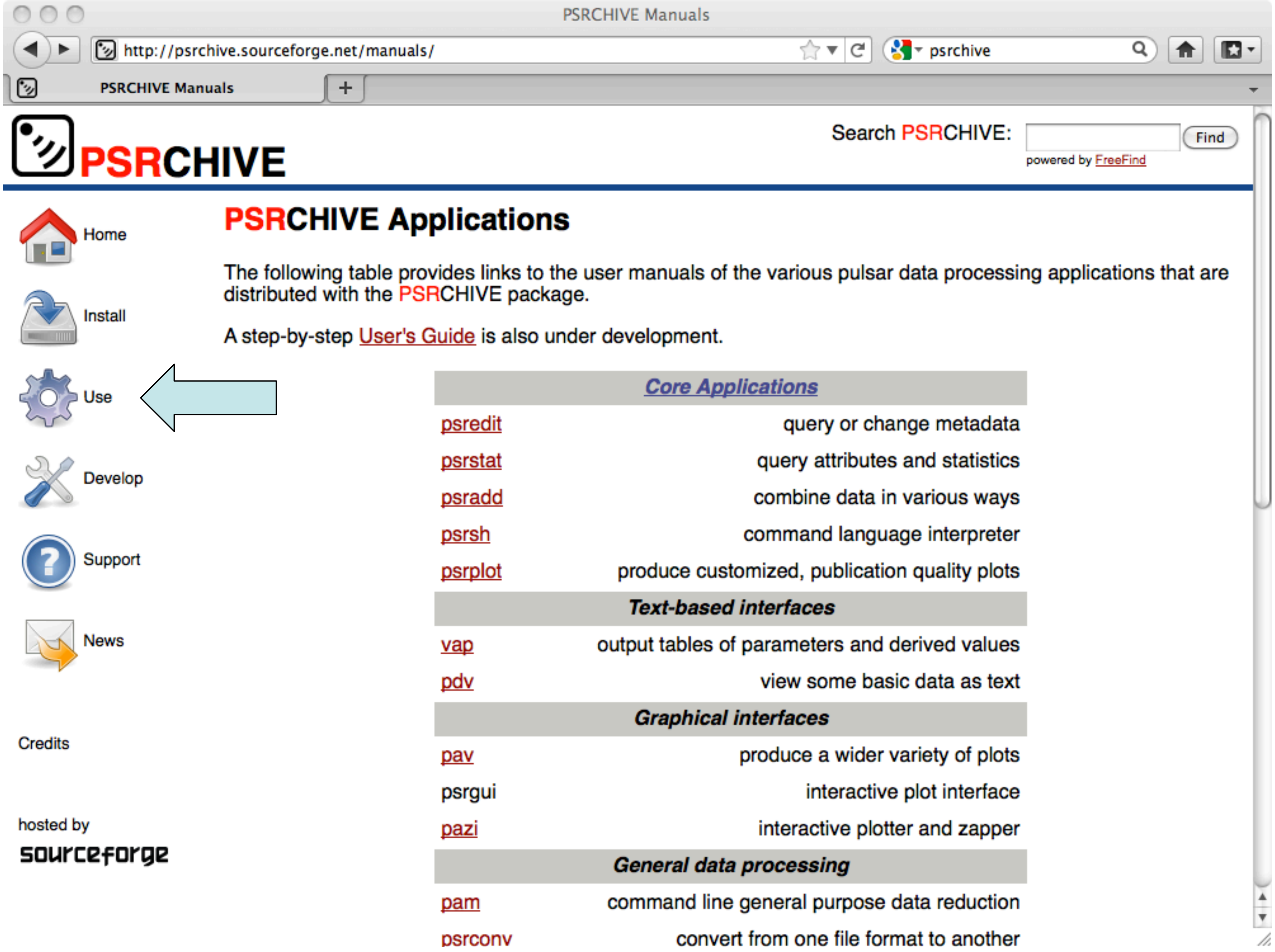
Feature Requests

If you have a great idea for a feature that should be implemented in PSRCHIVE, please tells us about it using the Feature Requests Tracker.

Developers Welcome!

If you have written or may some day write code or documentation that you would like to contribute to the project, btain a SourceForge account and send your user name to psrchive-developers@lists.sourceforge.net

How to use PSRCHIVE



PSRCHIVE Applications

The following table provides links to the user manuals of the various pulsar data processing applications that are distributed with the PSRCHIVE package.

A step-by-step User's Guide is also under development.

Core Applications	
psredit	query or change metadata
psrstat	query attributes and statistics
psradd	combine data in various ways
psrsh	command language interpreter
psrplot	produce customized, publication quality plots
Text-based interfaces	
vap	output tables of parameters and derived values
pdv	view some basic data as text
Graphical interfaces	
pav	produce a wider variety of plots
psrgui	interactive plot interface
pazi	interactive plotter and zapper
General data processing	
pam	command line general purpose data reduction
psrconv	convert from one file format to another

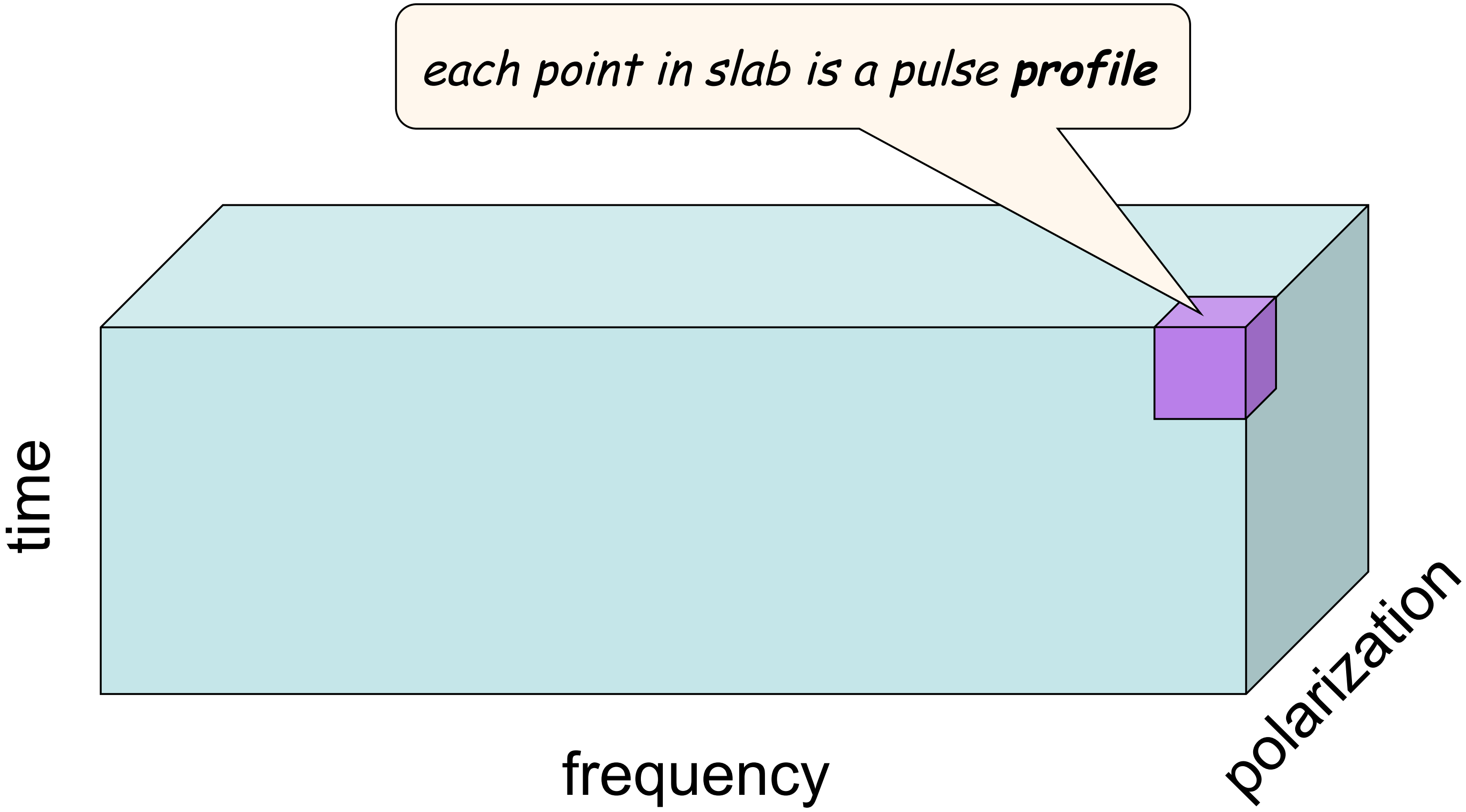
PSRCHIVE Core Applications

- standard command line options
  - remember once, use often
- powerful command language
  - full functionality in every program
- evaluation of mathematical expressions
  - variety of statistical tools

Use PSRCHIVE to …

- get to know your data
  - query and edit: `psredit`
  - evaluate: `psrstat`
  - plot: `psrplot`
- modify your data
  - command: `psrsh`
- combine your data
  - integrate: `psradd`

PSRCHIVE Data Slab



Query your data

print every attribute of file

```
> psredit filename.ar
Attribute Name   Description                               Value
----------------------------------------------------------------
nbin             Number of pulse phase bins                 1024
nchan            Number of frequency channels                128
npol             Number of polarizations                       4
nsubint          Number of sub-integrations                    1
...
freq             Centre frequency (MHz)                     1341
bw               Bandwidth (MHz)                             -64
dm               Dispersion measure (pc/cm^3)               2.46
...
```

Query your data

print selected attributes of files

```
> psredit -c rcvr:name,freq *.ar
n2003200174919.ar rcvr:name=unknown freq=1341
n2003200180318.ar rcvr:name=unknown freq=1341
n2003200180804.ar rcvr:name=unknown freq=1341
n2003200181319.ar rcvr:name=unknown freq=1341
n2003200181821.ar rcvr:name=unknown freq=1341
n2003200182323.ar rcvr:name=unknown freq=1341
...
```

Edit your data

```
> psredit -c rcvr:name=MULT_1 -m *.ar
n2003200174919.ar
Updating n2003200174919.ar ... done
n2003200180318.ar
Updating n2003200180318.ar ... done
...
> psredit -c rcvr:name *.ar
n2003200174919.ar rcvr:name=MULT_1
n2003200180318.ar rcvr:name=MULT_1
n2003200180804.ar rcvr:name=MULT_1
...
```

modify the original files

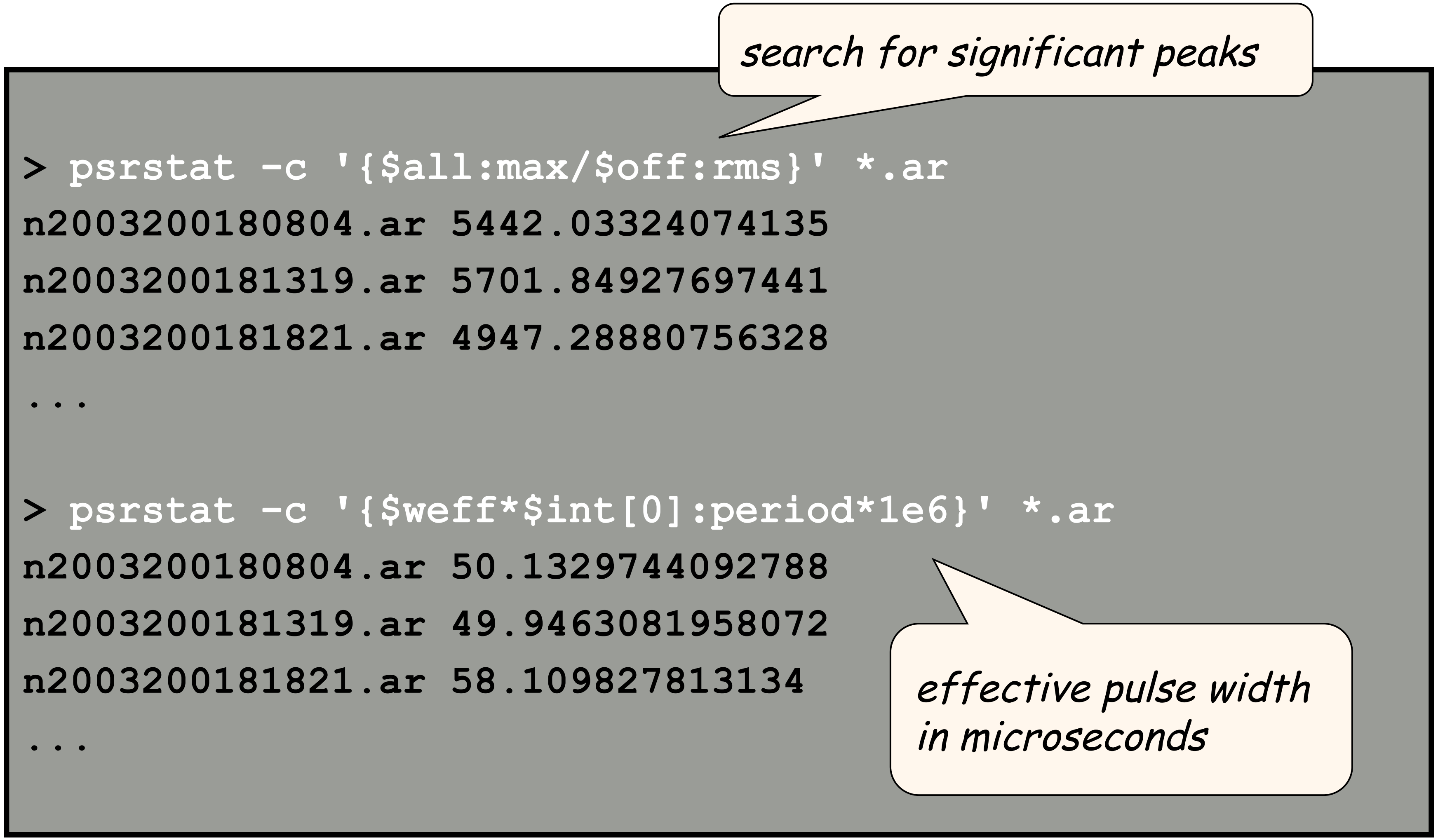
Evaluate your data

```
> psrstat -c snr *.ar
n2003200180804.ar snr=1659.31103515625
n2003200181319.ar snr=1579.50610351562
n2003200181821.ar snr=1188.38513183594
...
> psrstat -c snr *.ar -Q
n2003200180804.ar 1659.31103515625
n2003200181319.ar 1579.50610351562
n2003200181821.ar 1188.38513183594
...
> psrstat -c snr *.ar -Q | sort -nk 2
...
n2003200215839.ar 2393.0625
n2003200214835.ar 2419.32006835938
n2003200215337.ar 2512.64477539062
```

don't print label = value

combine with UNIX sort to find file with highest S/N

Evaluate your data



```
> psrstat -c '{$all:max/$off:rms}' *.ar
n2003200180804.ar 5442.03324074135
n2003200181319.ar 5701.84927697441
n2003200181821.ar 4947.28880756328
...

> psrstat -c '{$weff*$int[0]:period*1e6}' *.ar
n2003200180804.ar 50.1329744092788
n2003200181319.ar 49.9463081958072
n2003200181821.ar 58.109827813134
...
```

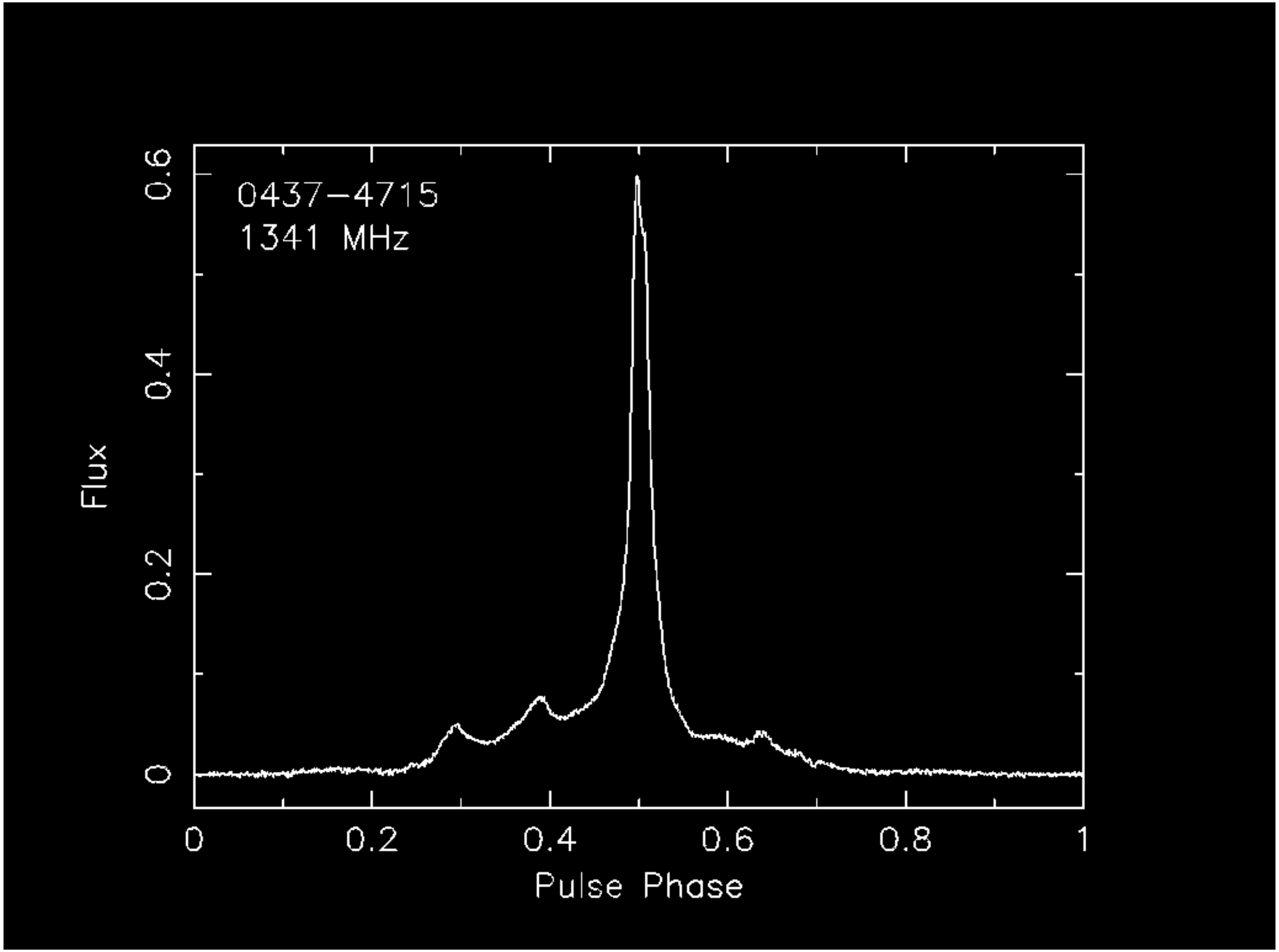
search for significant peaks

effective pulse width in microseconds

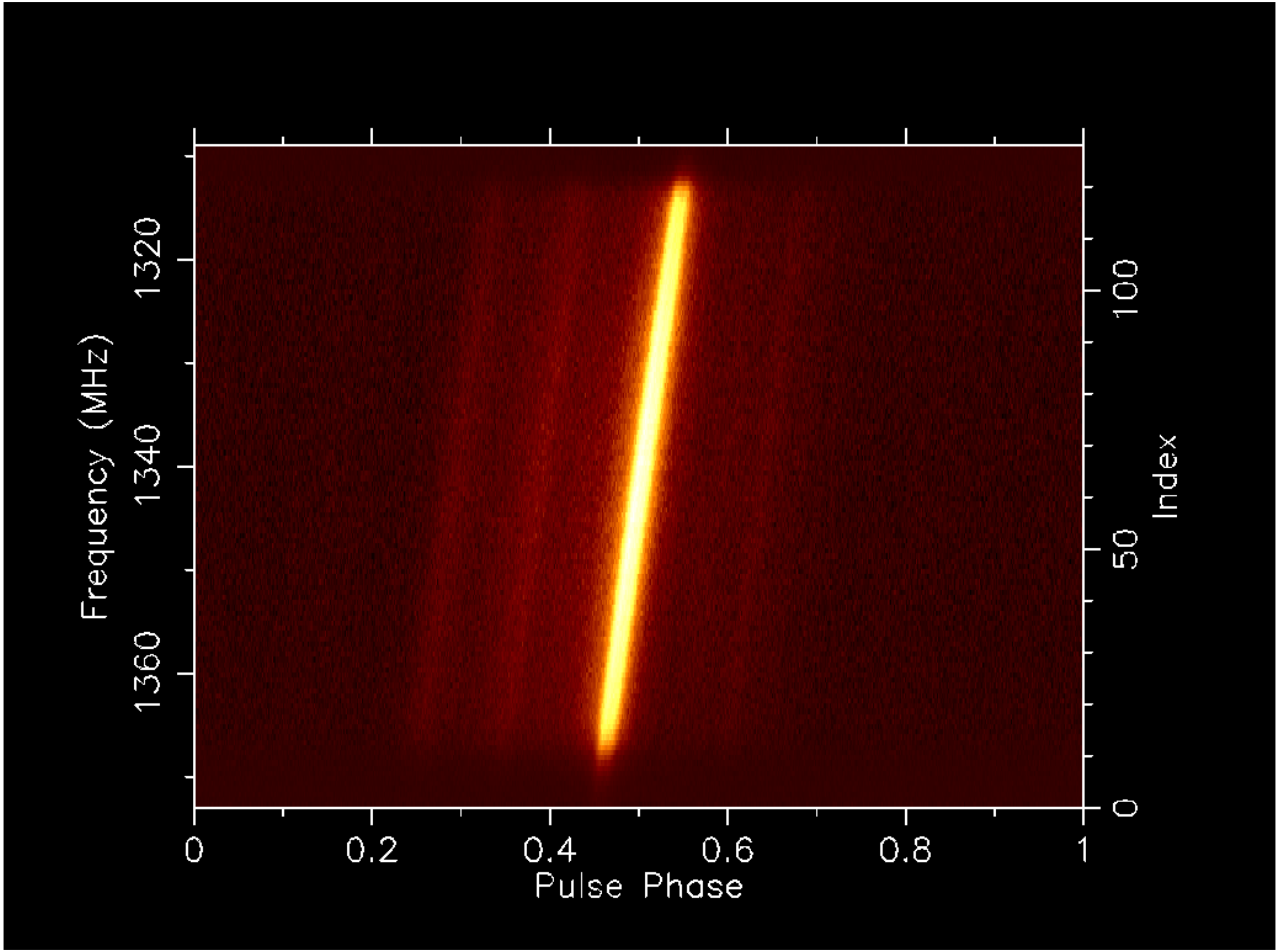
Plot your data

```
> psrplot -P
Available Plots:
flux       [D]  Single plot of flux
stokes     [s]  Stokes parameters
Scyl       [S]  Stokes; vector in cylindrical
...
freq       [G]  Phase vs. frequency image of flux
freq+      [F]  freq + integrated profile and spectrum
time       [Y]  Phase vs. time image of flux
...
psd        [b]  Pulsed power spectrum
...
dspec      [j]  Dynamic S/N spectrum
...
```

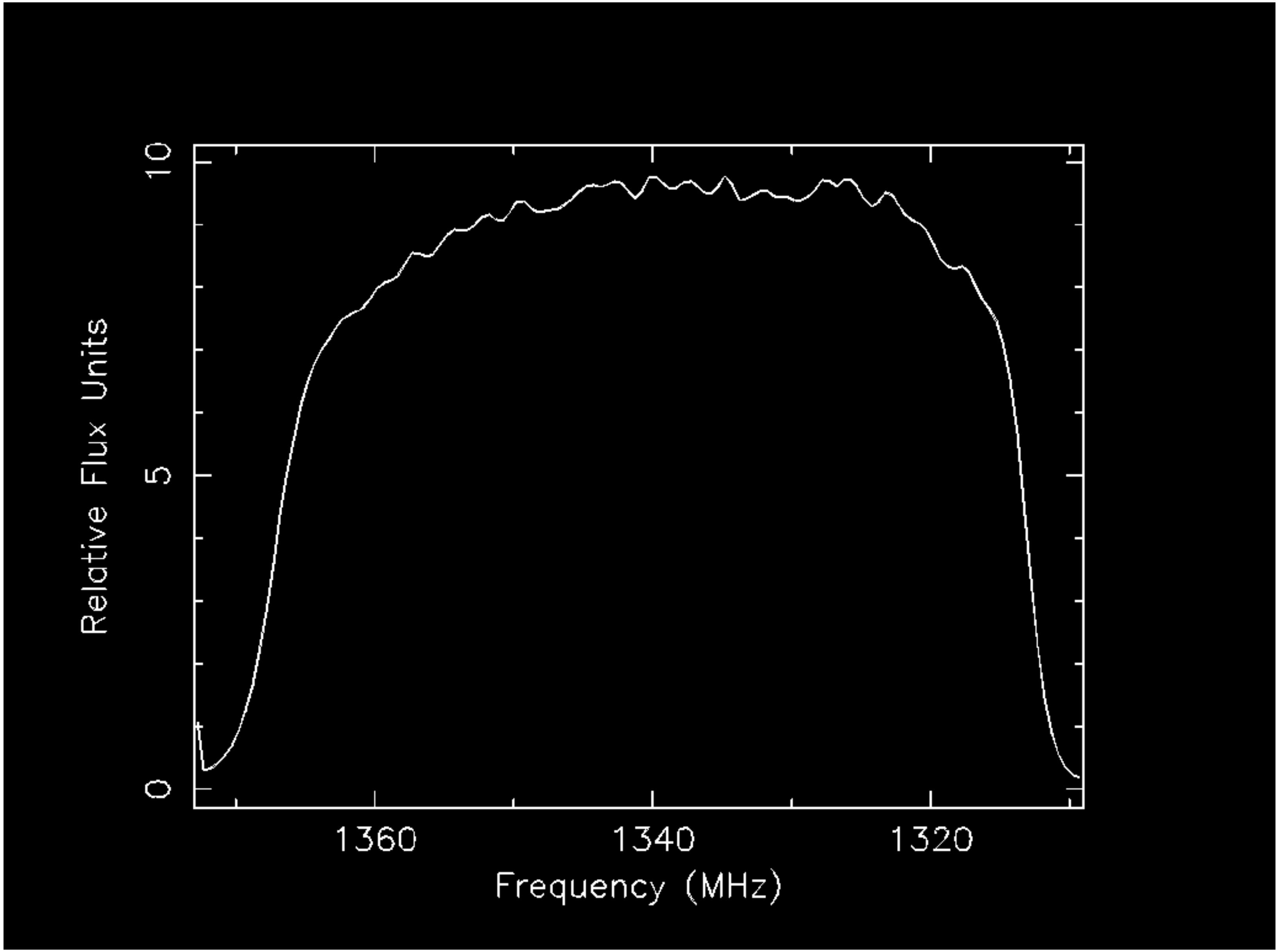
most commonly used plots



```
> psrplot -p flux file.ar
```



```
> psrplot -p freq file.ar
```



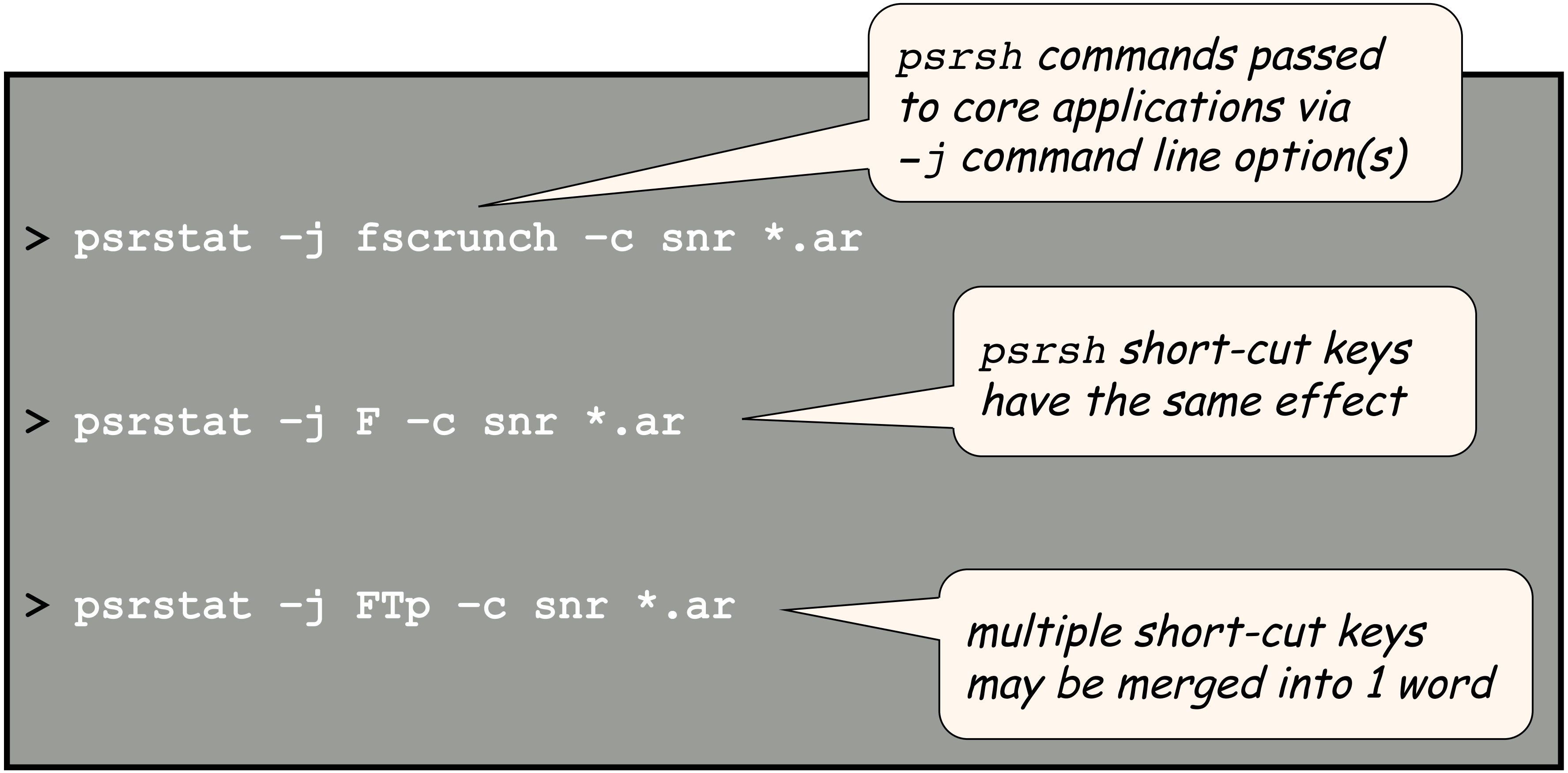
```
> psrplot -p psd file.ar
```

Modify your data

```
> psrsh -H
Available commands (shortcut keys in [] brackets):

...
edit        [e] edit archive parameters
...
fscrunch    [F] integrate archive in frequency
tscrunch    [T] integrate archive in time
pscrunch    [p] integrate archive to produce total intensity
bscrunch    [B] integrate archive in phase bins
centre      [C] centre profiles using polyco or max
dedisperse  [D] dedisperse all profiles in an archive
...
zap             zap data using the specified method
...
```

most commonly used commands

Modify your data



```
> psrstat -j fscrunch -c snr *.ar
```

`psrsh` *commands passed to core applications via* `-j` *command line option(s)*

```
> psrstat -j F -c snr *.ar
```

`psrsh` *short-cut keys have the same effect*

```
> psrstat -j FTp -c snr *.ar
```

multiple short-cut keys may be merged into 1 word

Combine your data

```
> psredit -c int:duration *.ar
n2003200180804.ar int:duration=315.620255999998
n2003200181319.ar int:duration=301.988687999998
n2003200181821.ar int:duration=301.989311999998
...
```

5-minute integrations

```
> psradd -I 3600 *.ar
```

create 1-hour integrations

```
> ls *.it
n2003200180804.it  n2003200201716.it  n2003200224809.it
n2003201010209.it  n2003200191239.it  n2003200214333.it
n2003200235426.it  n2003201021553.it
```

```
> psradd -o total.ar -j F *.it
```

join into one file and integrate all frequency channels together

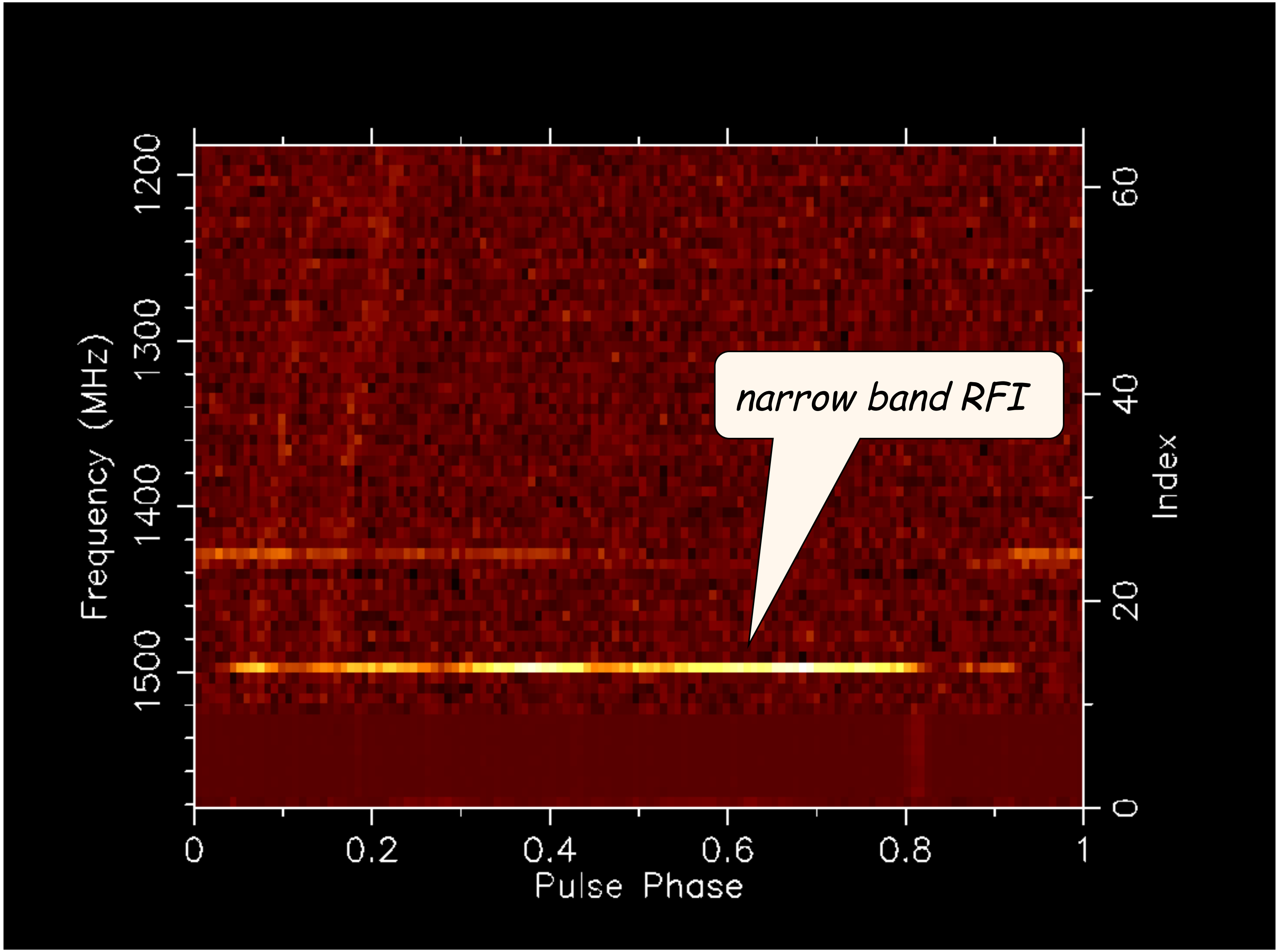
```
> psredit -c nsubint,nchan total.ar
total.ar nsubint=8 nchan=1
```

Combine your data

```
> psrstat -c '{$snr/sqrt($int:duration)}' -j pFT *.??
total.ar 83.0168392699134
n2003200180804.ar 93.3996755467036
n2003200180804.it 79.3453513397579
n2003200181319.ar 90.8920650853029
n2003200181821.ar 68.3850900692238
n2003200182323.ar 73.1501502112451
n2003200182825.ar 70.2737382706604
n2003200183327.ar 92.3652659044368
n2003200183859.ar 69.243181171337
...
```

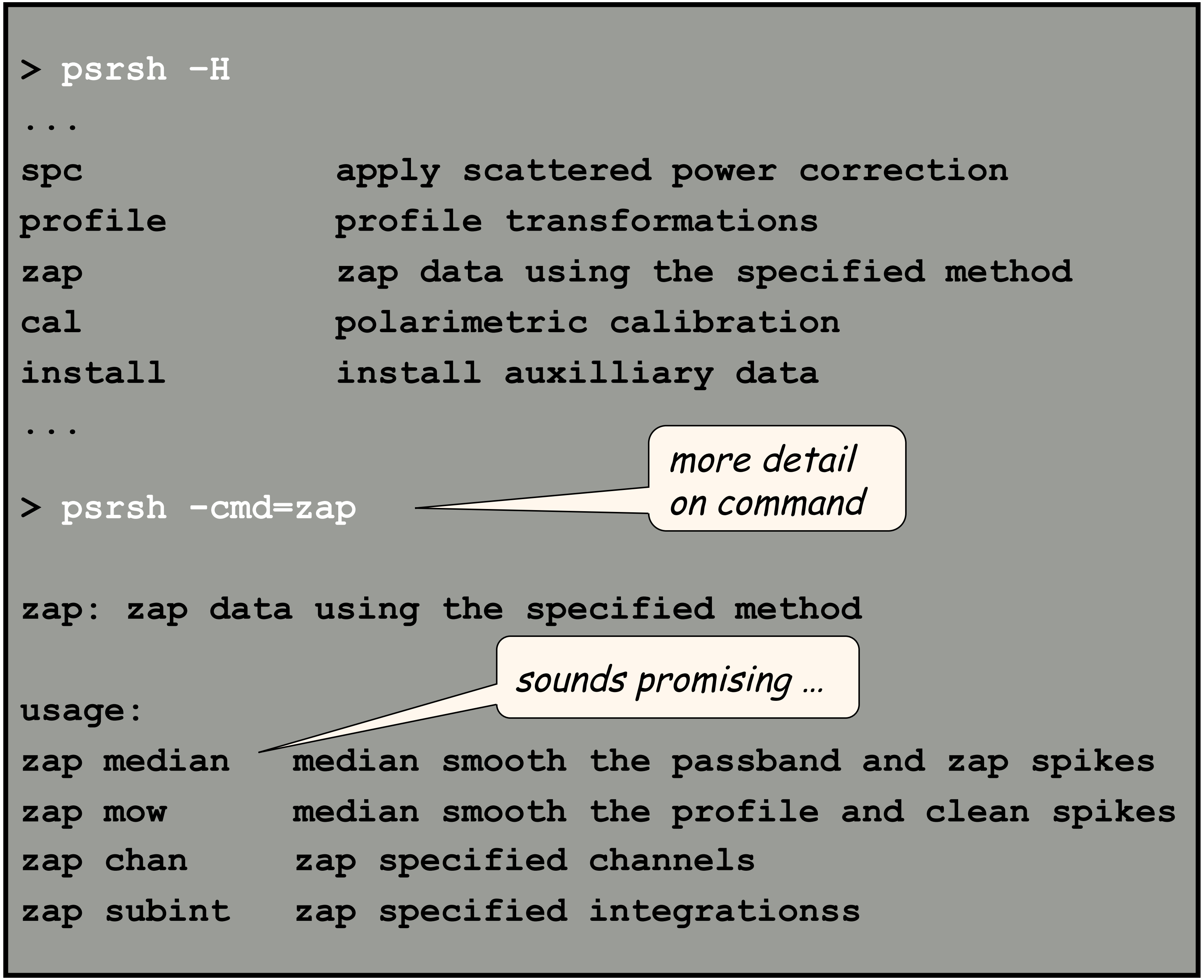
double-check results against expectations

Radio Frequency Interference mitigation with PSRCHIVE



narrow band RFI

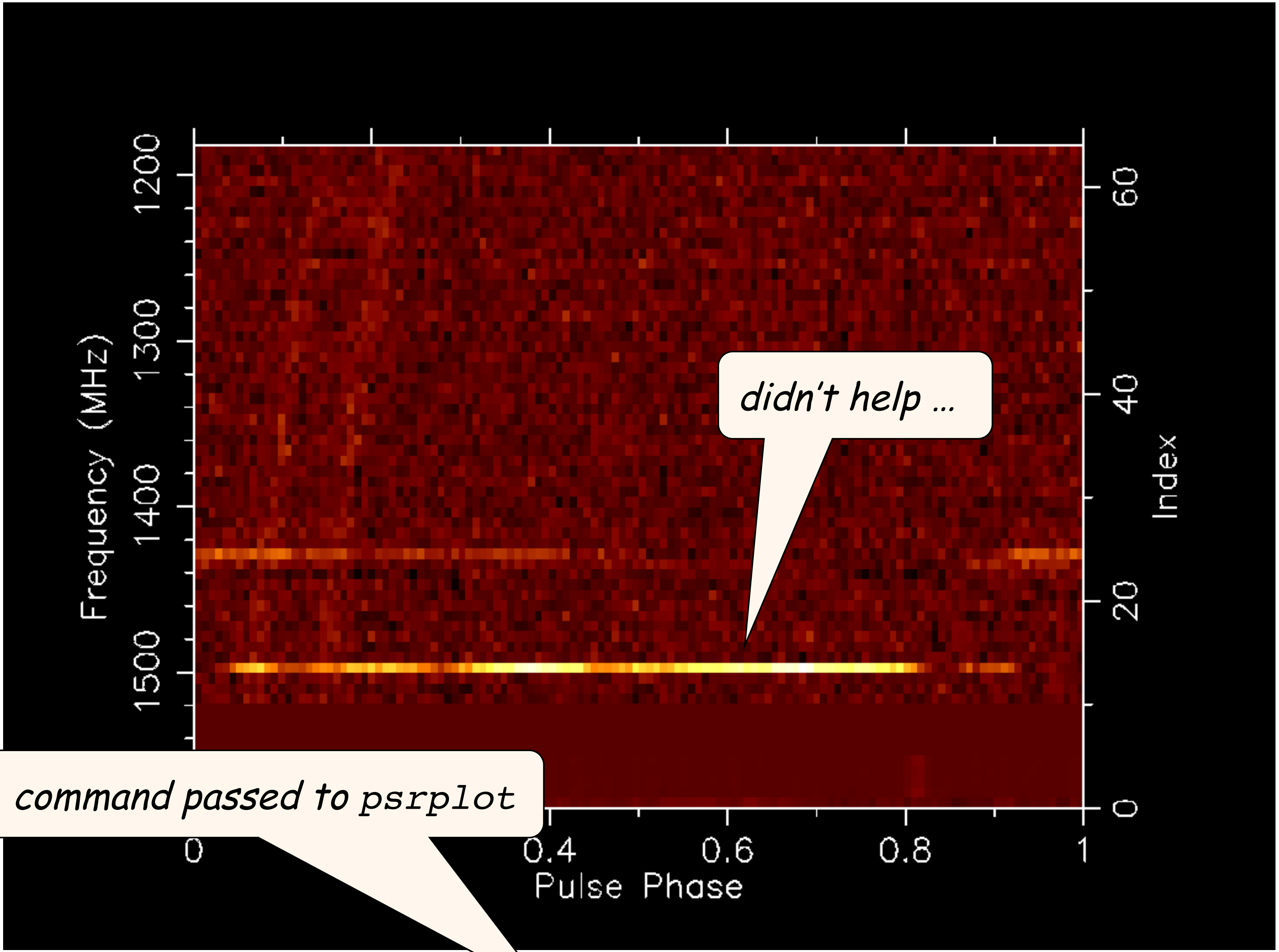
```
> psrplot -p freq example.ar
```



```
> psrsh -H
...
spc              apply scattered power correction
profile          profile transformations
zap              zap data using the specified method
cal              polarimetric calibration
install          install auxilliary data
...

> psrsh -cmd=zap

zap: zap data using the specified method

usage:
zap median    median smooth the passband and zap spikes
zap mow       median smooth the profile and clean spikes
zap chan      zap specified channels
zap subint    zap specified integrationss
```

more detail on command

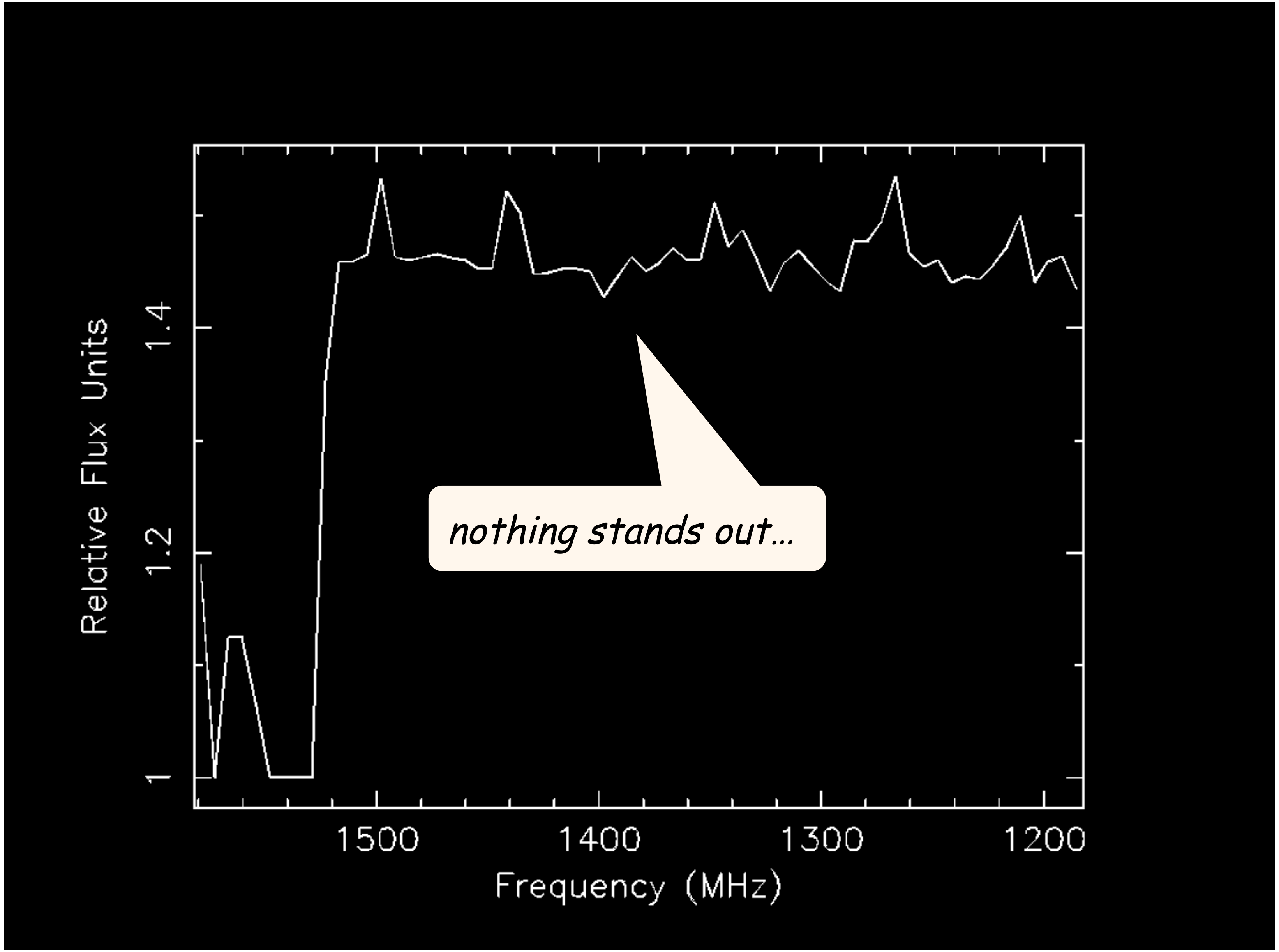
sounds promising ...



didn't help …

`psrsh` *command passed to* `psrplot`

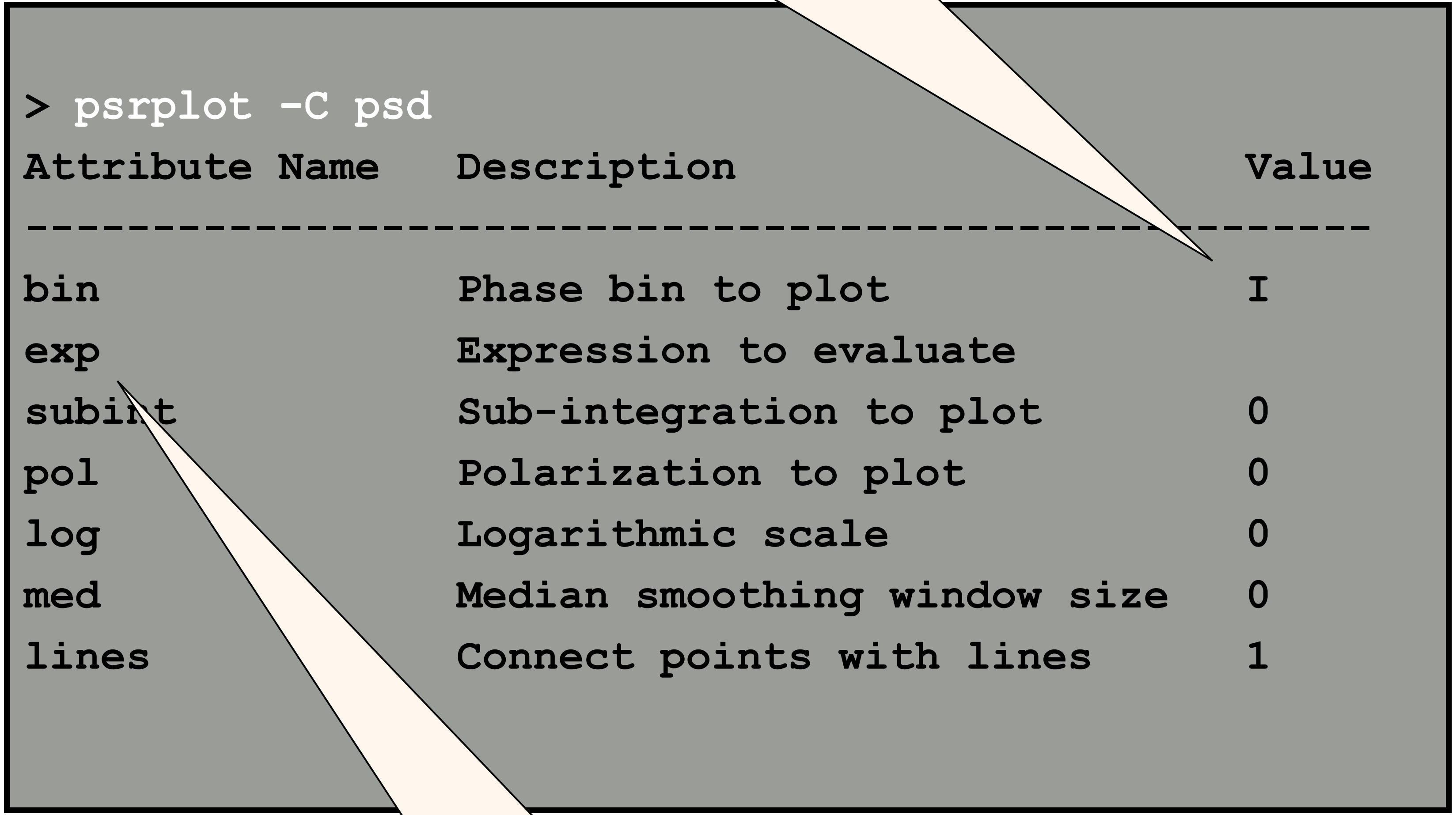
```
> psrplot -p freq example.ar -j “zap median”
```



nothing stands out…

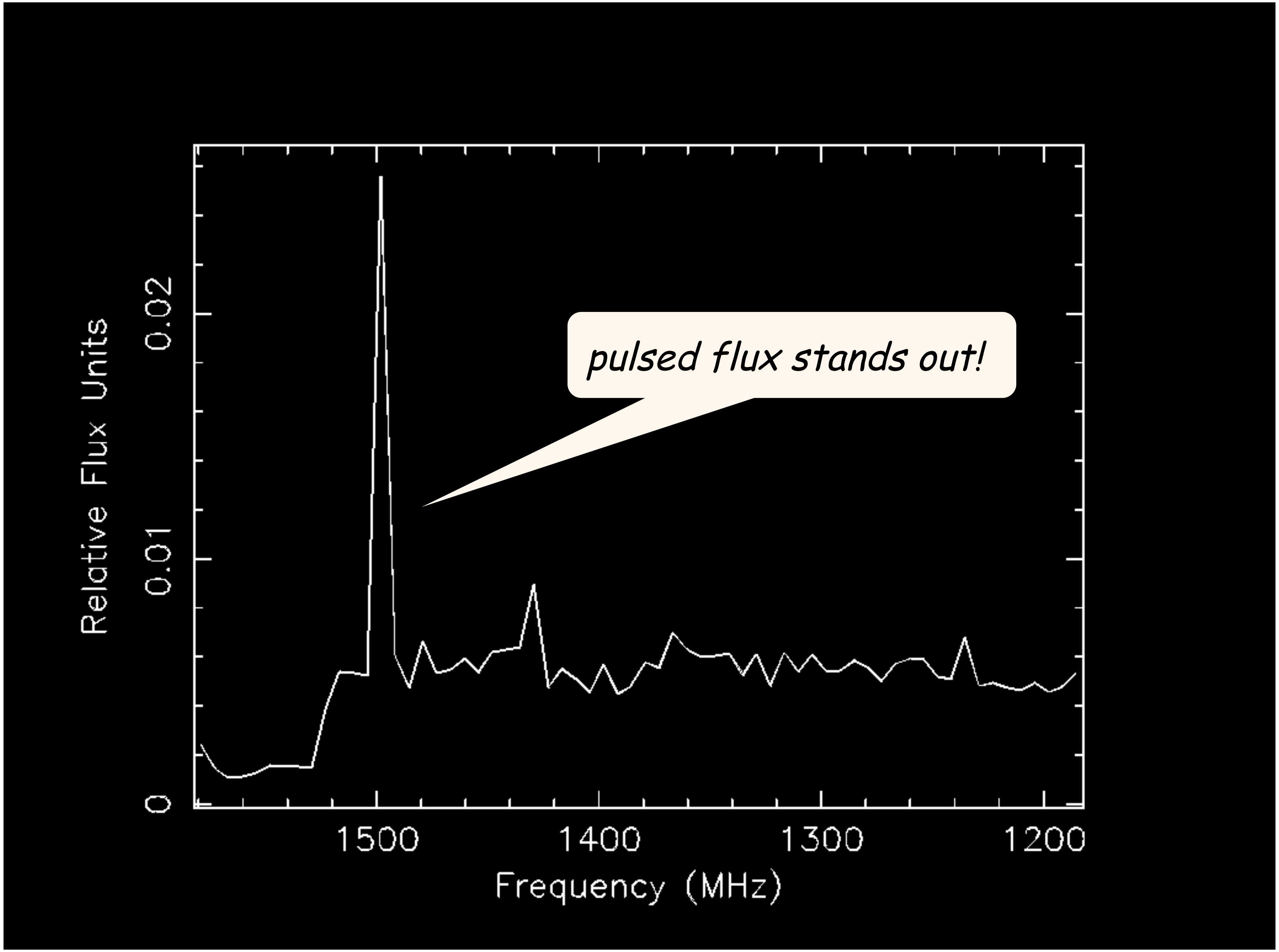
```
> psrplot -p psd example.ar
```

integrate flux over all phase bins



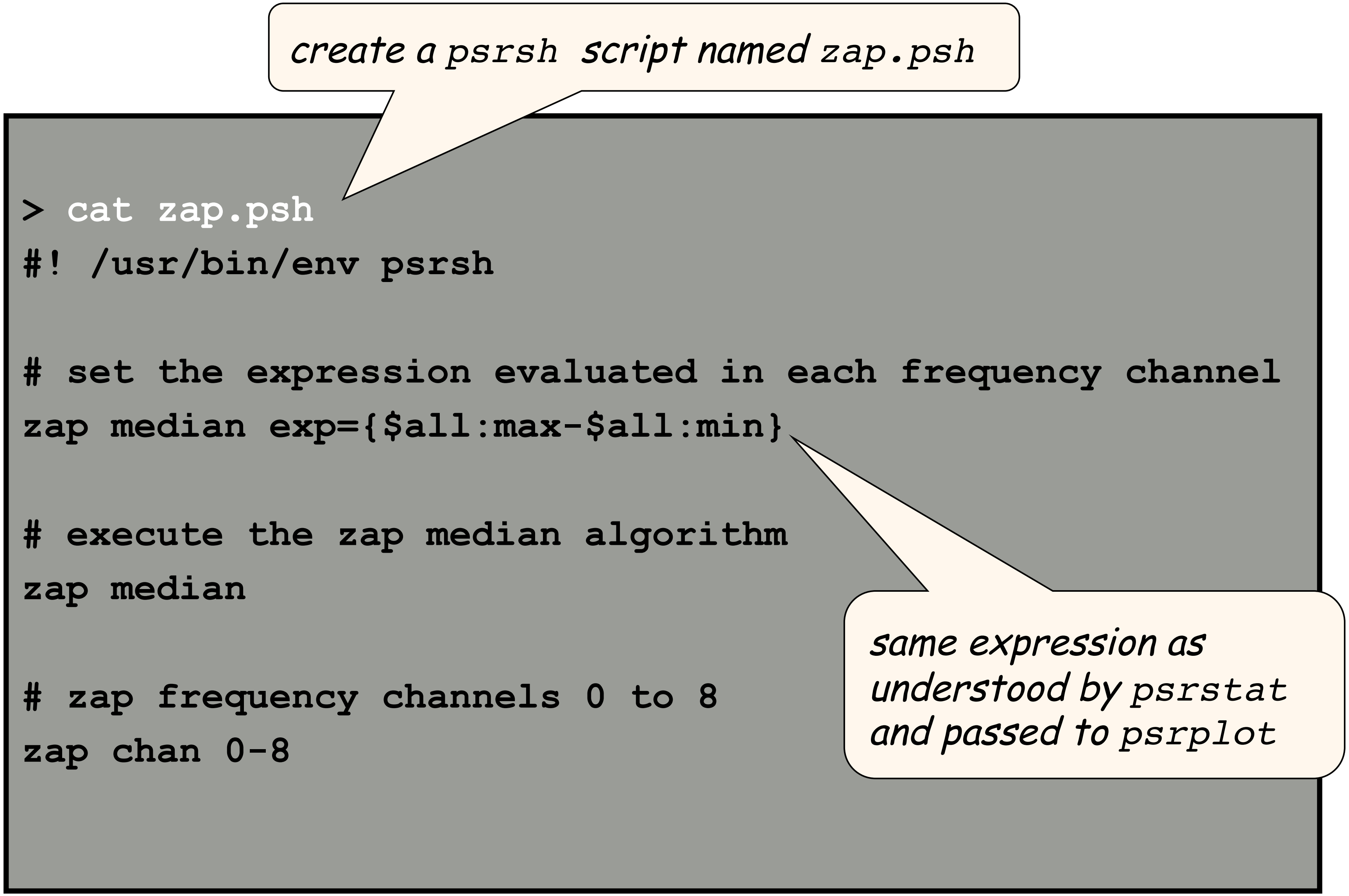
```
> psrplot -C psd
Attribute Name   Description                  Value
-------------------------------------------------------
bin              Phase bin to plot            I
exp              Expression to evaluate
subint           Sub-integration to plot      0
pol              Polarization to plot         0
log              Logarithmic scale            0
med              Median smoothing window size 0
lines            Connect points with lines    1
```

evaluate expression as understood by `psrstat`



pulsed flux stands out!

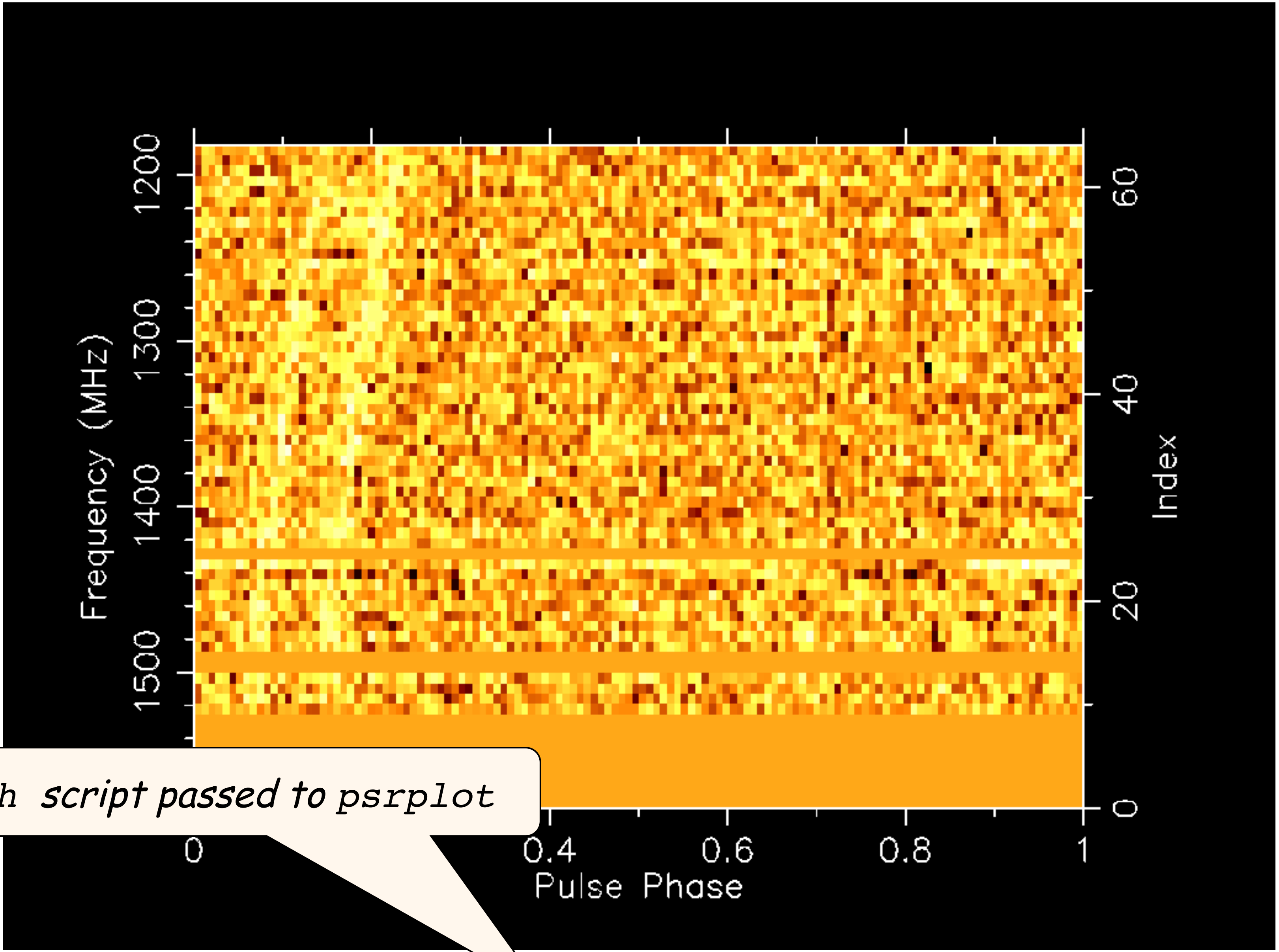
```
> psrplot -p psd example.ar -c 'exp={$all:max-$all:min}'
```



create a psrsh script named zap.psh

```
> cat zap.psh
#! /usr/bin/env psrsh

# set the expression evaluated in each frequency channel
zap median exp={$all:max-$all:min}

# execute the zap median algorithm
zap median

# zap frequency channels 0 to 8
zap chan 0-8
```

same expression as understood by psrstat and passed to psrplot



psrsh script passed to psrplot

```
> psrplot -p freq example.ar -J zap.psh
```

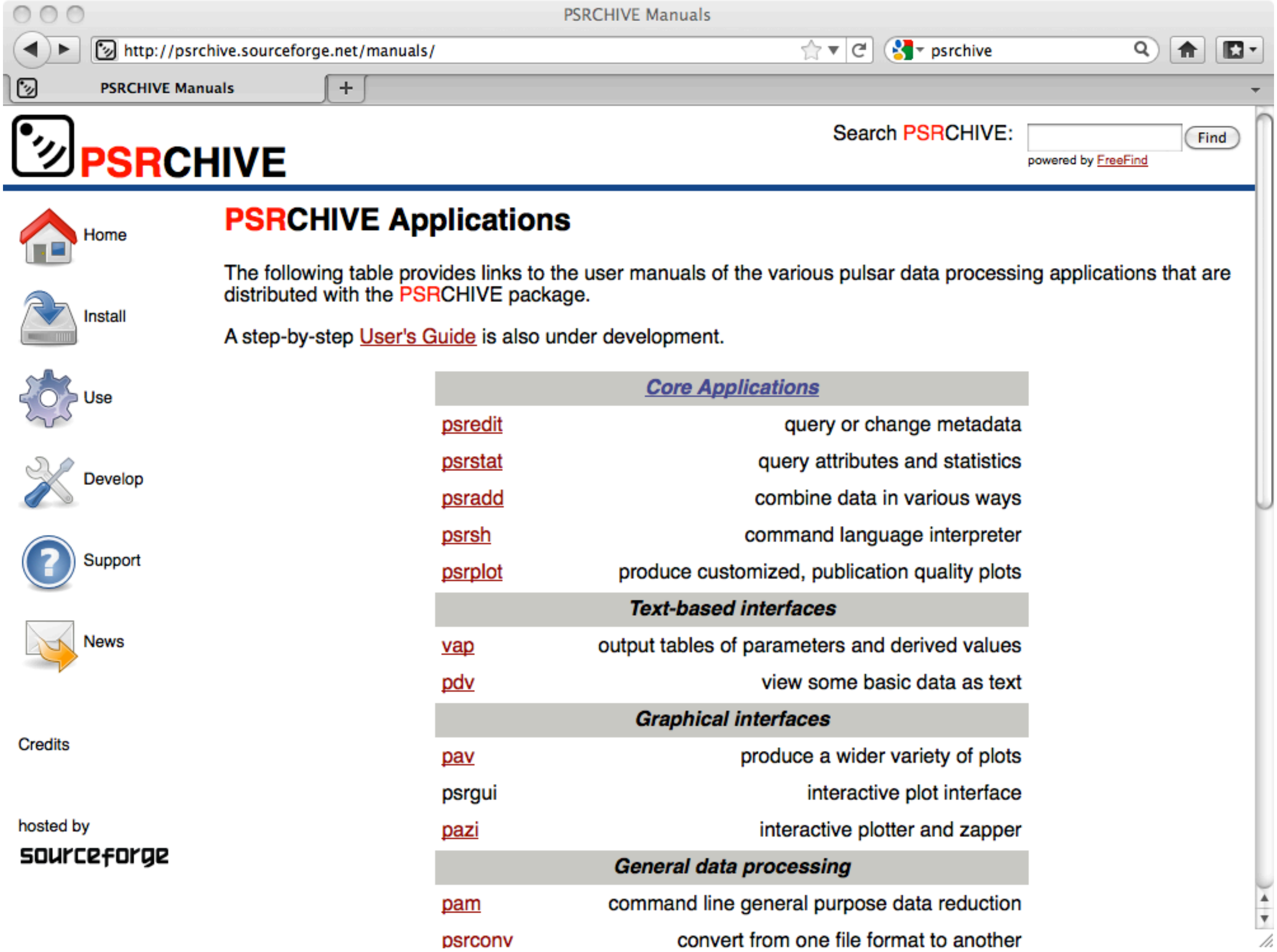
`psrsh` *scripts can be executed like a standard psrchive program*

```
> chmod a+x zap.psh
> zap.psh -e zz example.TT
Unloading example.zz ... done
```

Conclusion

- PSRCHIVE Core Applications:
  - general data analysis tools
  - tightly integrated interfaces

- PSRCHIVE Advanced Applications:
  - `pac` and `pcm`: polarization calibration
  - `pat`: arrival time estimation
  - `pdmp`: survey candidate refinement
  - etc.



PSRCHIVE Applications

The following table provides links to the user manuals of the various pulsar data processing applications that are distributed with the PSRCHIVE package.

A step-by-step User's Guide is also under development.

Core Applications	
psredit	query or change metadata
psrstat	query attributes and statistics
psradd	combine data in various ways
psrsh	command language interpreter
psrplot	produce customized, publication quality plots
Text-based interfaces	
vap	output tables of parameters and derived values
pdv	view some basic data as text
Graphical interfaces	
pav	produce a wider variety of plots
psrgui	interactive plot interface
pazi	interactive plotter and zapper
General data processing	
pam	command line general purpose data reduction
psrconv	convert from one file format to another

Conclusion

- PSRCHIVE New Applications:
  - `psrzap`
    - interactive RFI mitigation tool
    - understands `psrstat` expressions
    - outputs `psrsh` script (`zap such` commands)
  - `psrmodel`:
    - Rotating Vector Model (RVM) fits
    - statistically rigorous error analysis

Conclusion

- PSRCHIVE Future Applications:
  - C++ and/or python
  - developers welcome!

Thank you!