Searching for Radio Pulsars

Michael Keith
CSIRO Astronomy & Space Science -- ATNF

How do we find radio pulsars?

- Finding radio pulsars is conceptually simple, however in practice there are a lot of details.
- Unfortunately, there is a large number of different software packages involved
- There are many telescope specific details that are not covered here

What do the raw data look like

- In this talk we assume you have data which is:
  - **Regularly sampled** and continuous (i.e. there are no gaps.)
  - **Multiple frequency channels** (also regularly spaced)
  - Single polarisation, pointed at a single location on the sky
- Luckily this covers 99% of all pulsar search data.

The process

RAW DATA FILE(S)

DATA EXTRACTION

1 FILE - MANY CHANNELS

DE-DISPERSION

MANY 1-CH TIME SERIES

PERIODICITY SEARCH

THOUSANDS OF PERIODICITIES

COLLATION OF CANDIDATES

10-100S OF CANDIDATES

FOLDING & OPTIMISING

OPTIMISED CAND PLOTS

CANDIDATE SELECTION

0-1 POSSIBLE PULSARS

Also “optional” items...

INTERFERENCE MITIGATION

ACCELERATION SEARCHES

SINGLE PULSE SEARCH

RAW DATA → **DATA EXTRACTION** → 1 FILE, MANY CHANNELS

- This is the name for a process by which we take the data as we have obtained it (from an archive, or direct from a telescope) and prepare it for our search.
- The data may be in a number of formats, so we may have to convert it before it can be used for our search. (We will talk about this more later!)
- We may have to extract a single observation from a file that contains many observations (e.g. multi-beam data or novel scanning modes).
- At this point, we have a file containing a time-series for each frequency channel for a single observation.

1 FILE, MANY CHANNELS → **DE-DISPERSION** → MANY 1-CH TIME-SERIES

- We have to sum the channels together to get enough S/N to be able to detect any pulsars, however since we need to know the dispersion measure.

- For unknown pulsars (i.e. when we are searching) we have to try many ‘trial’ DM values.

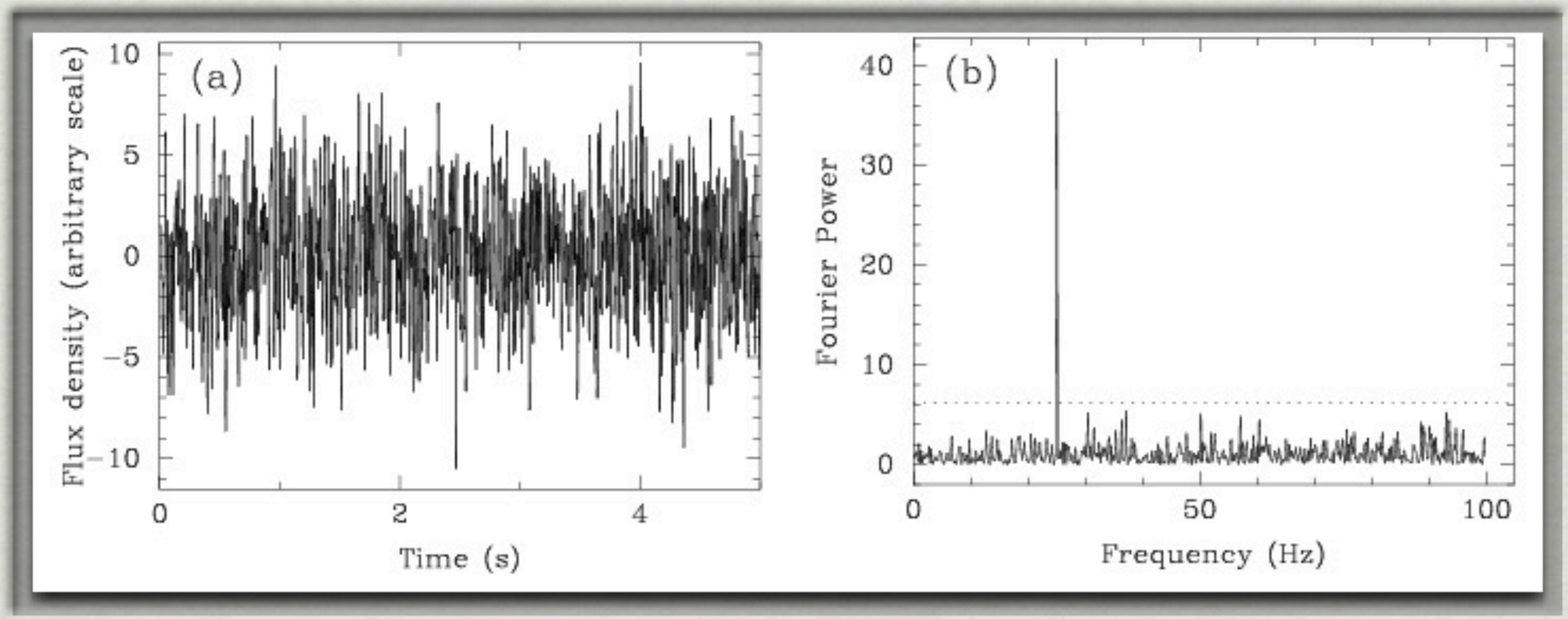
- For example, for the HTRU survey we have 1200 trials with DM values between 0 and 1000 $cm^{-3}pc$.

- So we end up with one single-channel time-series per DM trial.

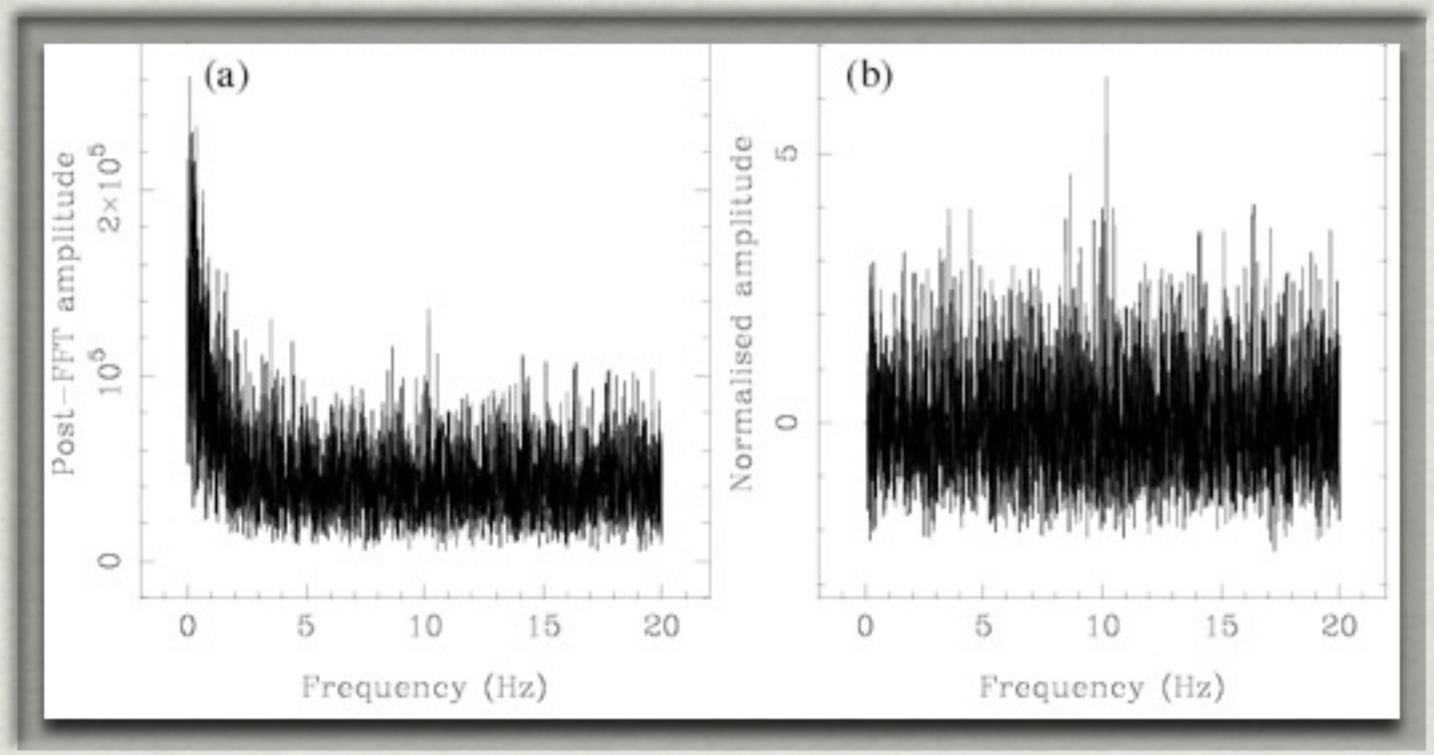
MANY 1-CH TIME-SERIES → **PERIODICITY SEARCH** → 1000S OF PERIODICITIES

- For the majority of pulsars, the *individual pulses are too weak to detect*.

- The pulses are periodic however, so we use a Fourier transform to detect the periodic signal.



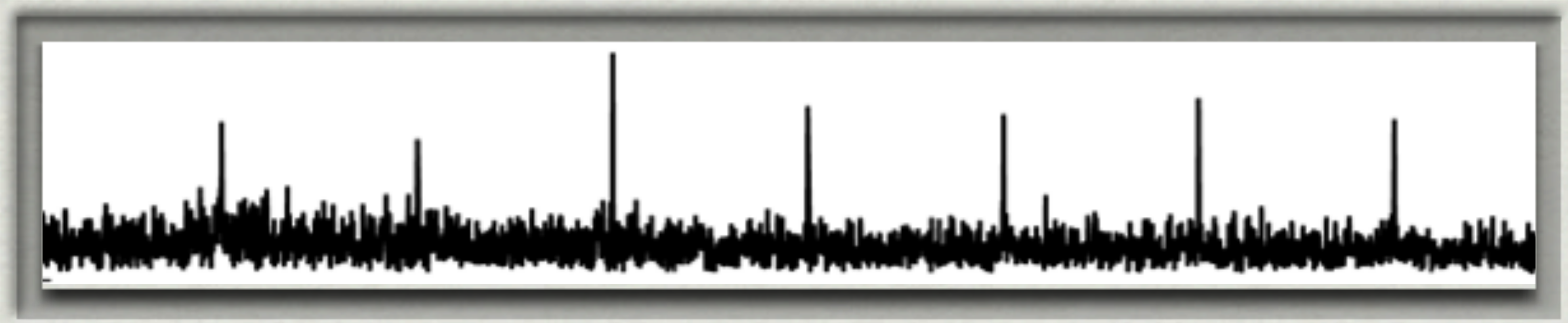
MANY 1-CH TIME-SERIES → **PERIODICITY SEARCH** → 1000S OF PERIODICITIES



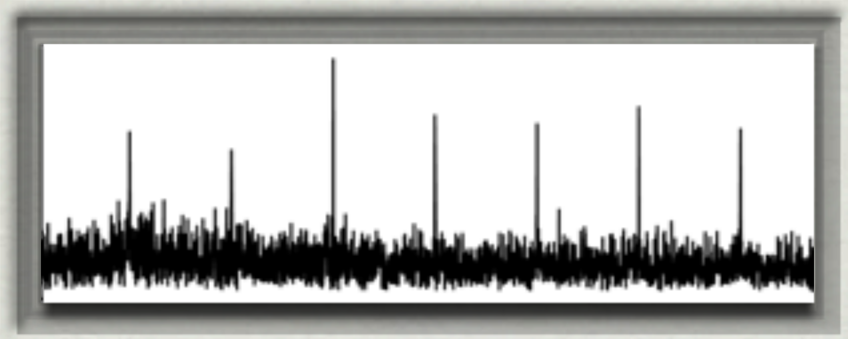
- Real data tend to be dominated by “red noise”, which must be removed in order to detect signals < ~10 Hz (i.e. most pulsars).
- Typical approaches subtract a running mean (or median) and divide by a running RMS to whiten and normalise the data.
- The whitening process is often complicated by RFI which can easily throw off the normalisation algorithm. The most common cause for failure to detect long period signals is incorrect red-noise compensation.

MANY 1-CH TIME-SERIES → **PERIODICITY SEARCH** → 1000S OF PERIODICITIES

- Signals from pulsars tend to have a fairly narrow duty cycle. This means that their spectral power gets distributed into many harmonics and so sensitivity is lost compared to an equivalent sinusoidal signal.
    - Remember: a train of δ-functions transforms to a train of δ-functions!
- We can recover this power by incoherent “harmonic summing”. Here we take the amplitude spectrum, stretch it by a factor of 2 and add it to the original amplitudes. The fundamental and 1st harmonic are added together and so produce a more significant spike.

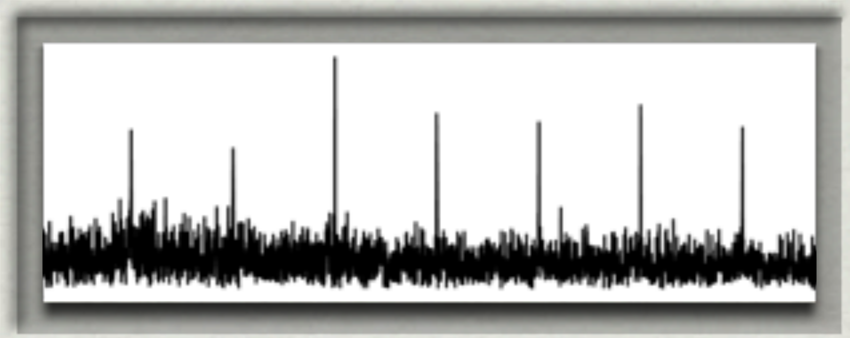


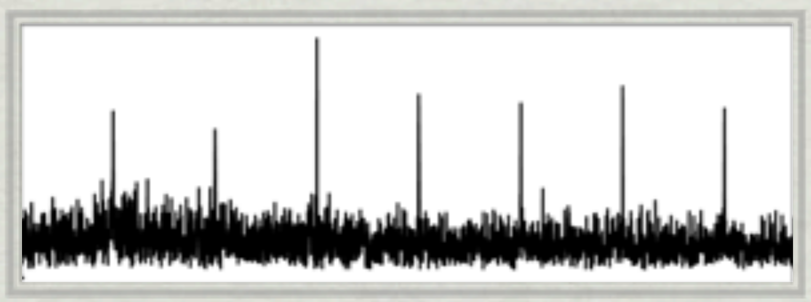
Harmonic Summing



F

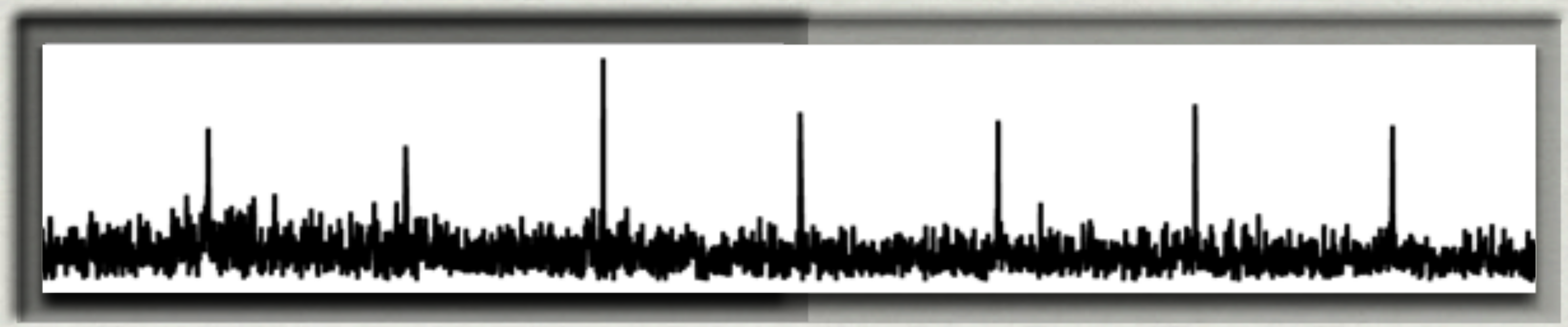
Harmonic Summing



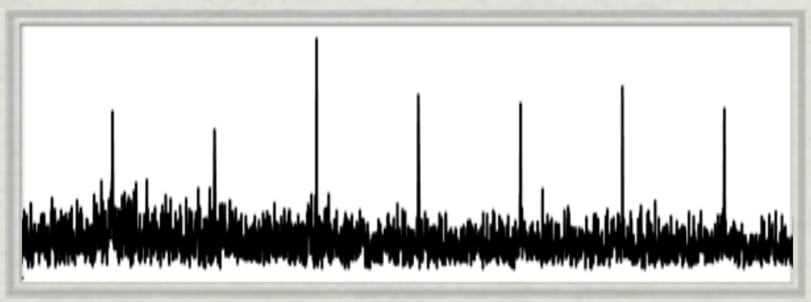


F

Harmonic Summing

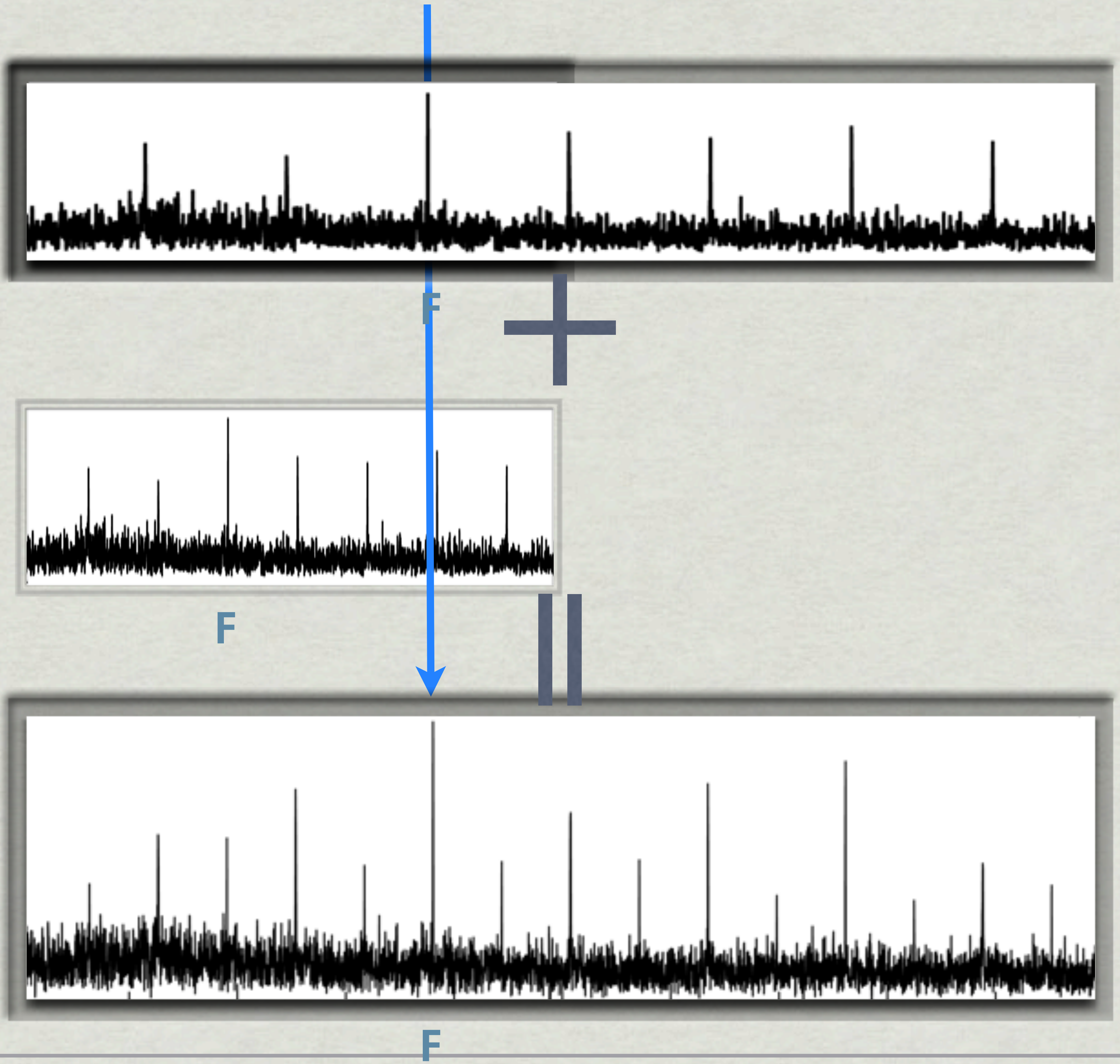


F



F

Harmonic Summing



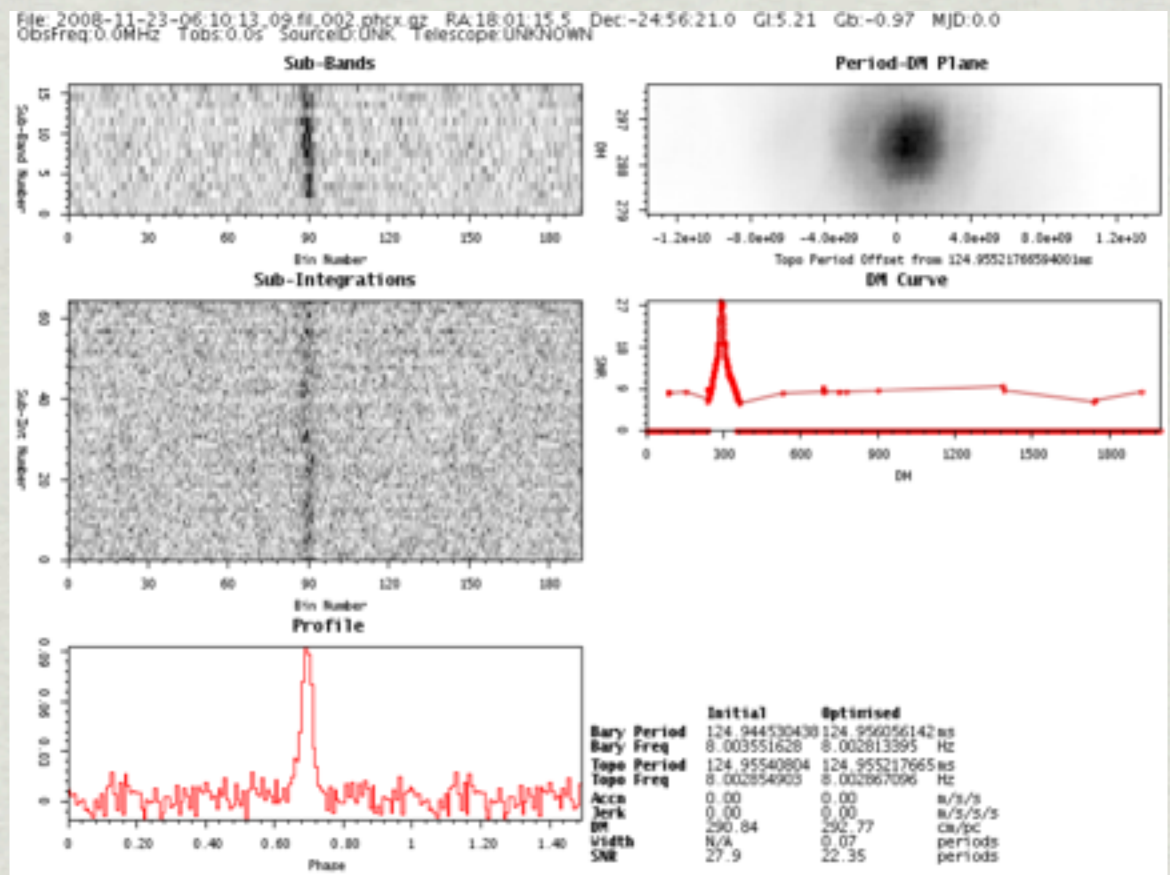
1000S OF PERIODICITIES → **COLLATION OF CANDIDATES** → 10-100S OF CANDIDATES

- Once we have done our Fourier search, and found all the significant signals in the spectrum *of each DM trial*, we have a list of many thousands of periods, and their corresponding DM values.
- Many of these signals will be either the same signal detected in several DM trials.
- Some of these signals will be harmonically related to each other.
- In this collation stage, we reduce all related periodicities into a single candidate, represented by its best period and DM.
- Hopefully this is now a more manageable number of candidates, of the order of 100.

10-100S OF CANDIDATES → **FOLDING & OPTIMISATION** → OPTIMISED CAND PLOTS

- Although we now have a relatively small number of candidates, it is still not clear which are the most likely to be pulsars.

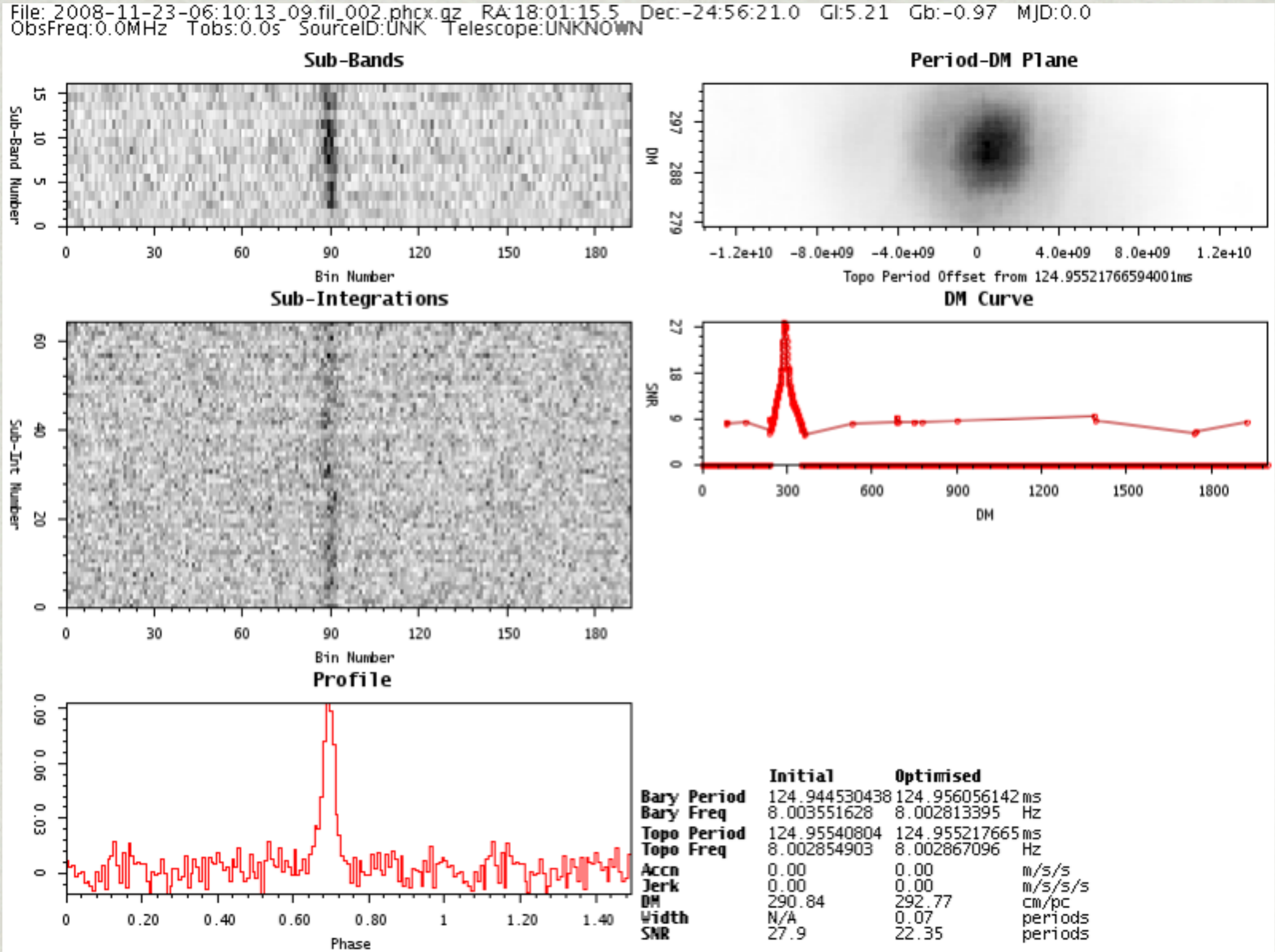
- To help with this, and to optimise the S/N, period and DM, we fold the original time-series with the candidate period.



10-100S OF CANDIDATES

FOLDING & OPTIMISATION

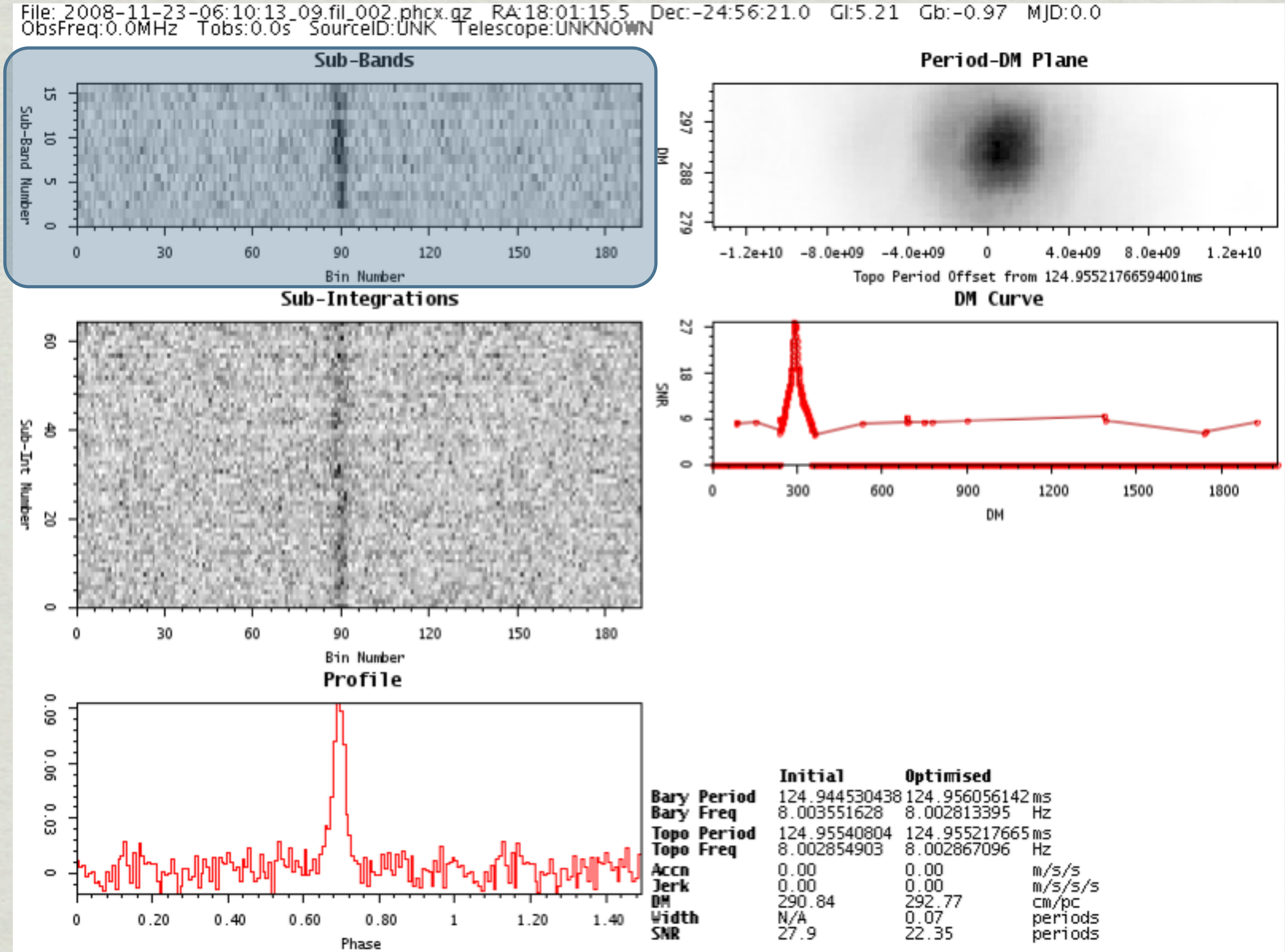
OPTIMISED CAND PLOTS



10-100S OF CANDIDATES

FOLDING & OPTIMISATION

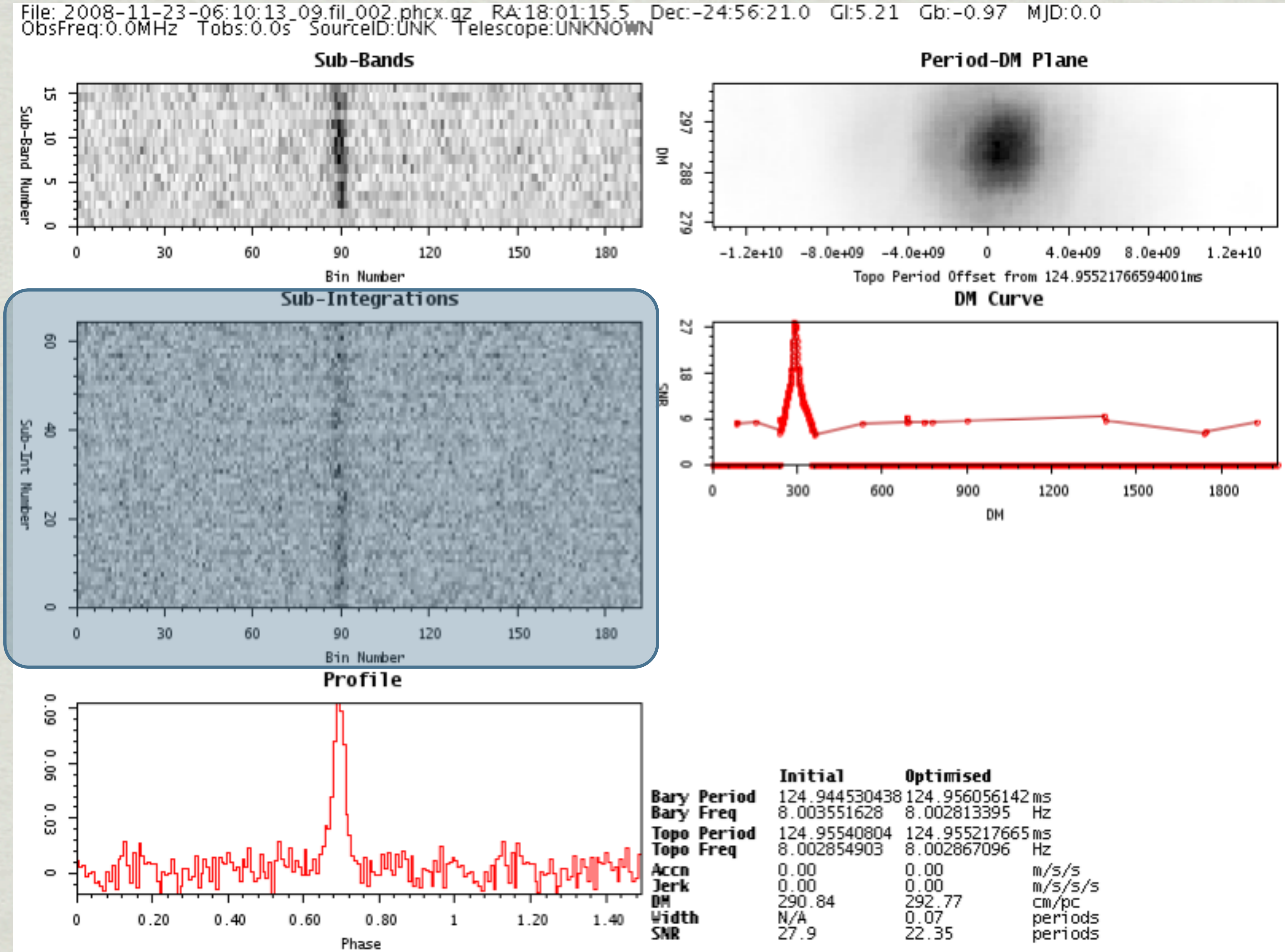
OPTIMISED CAND PLOTS



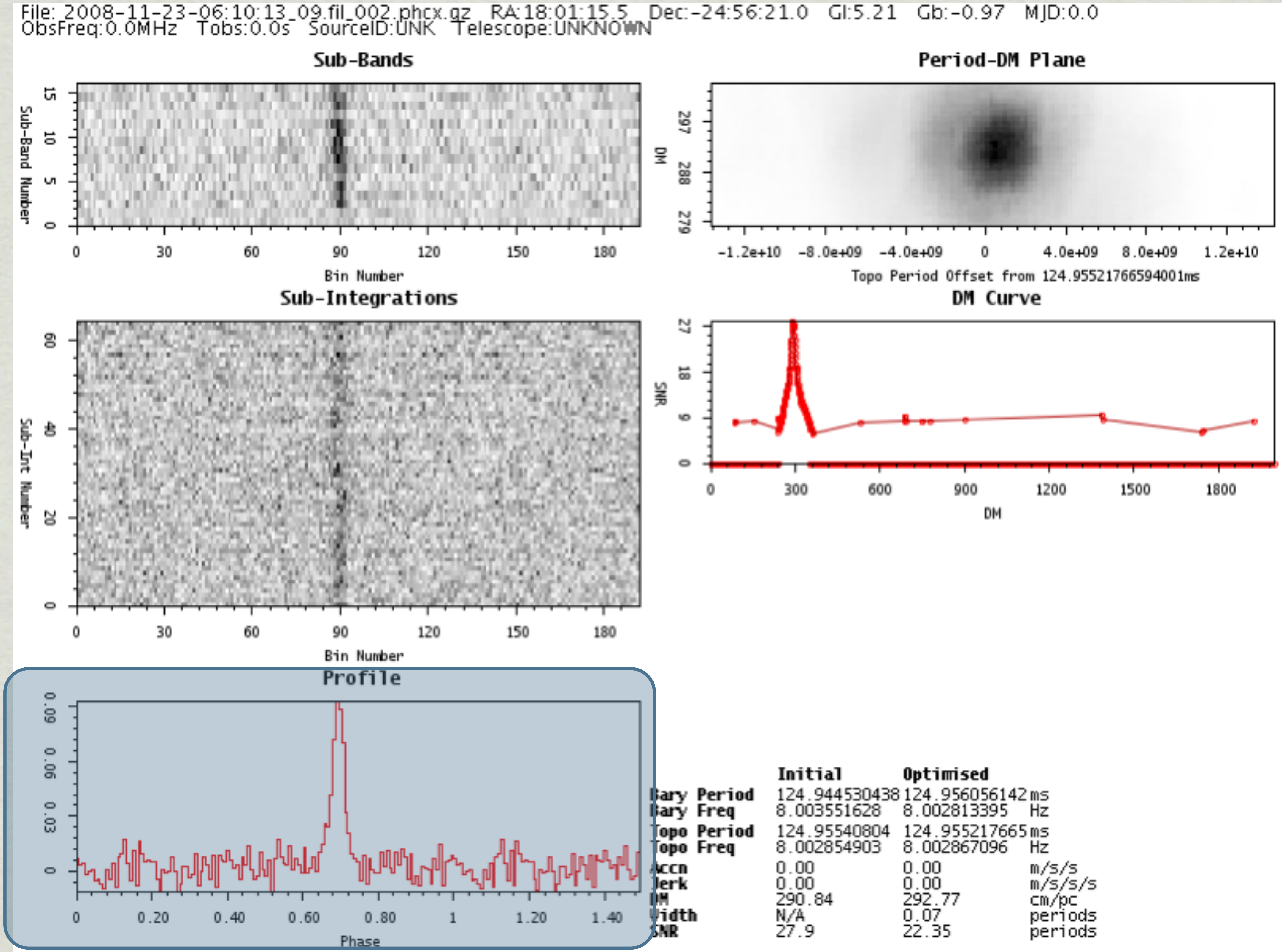
10-100S OF CANDIDATES

FOLDING & OPTIMISATION

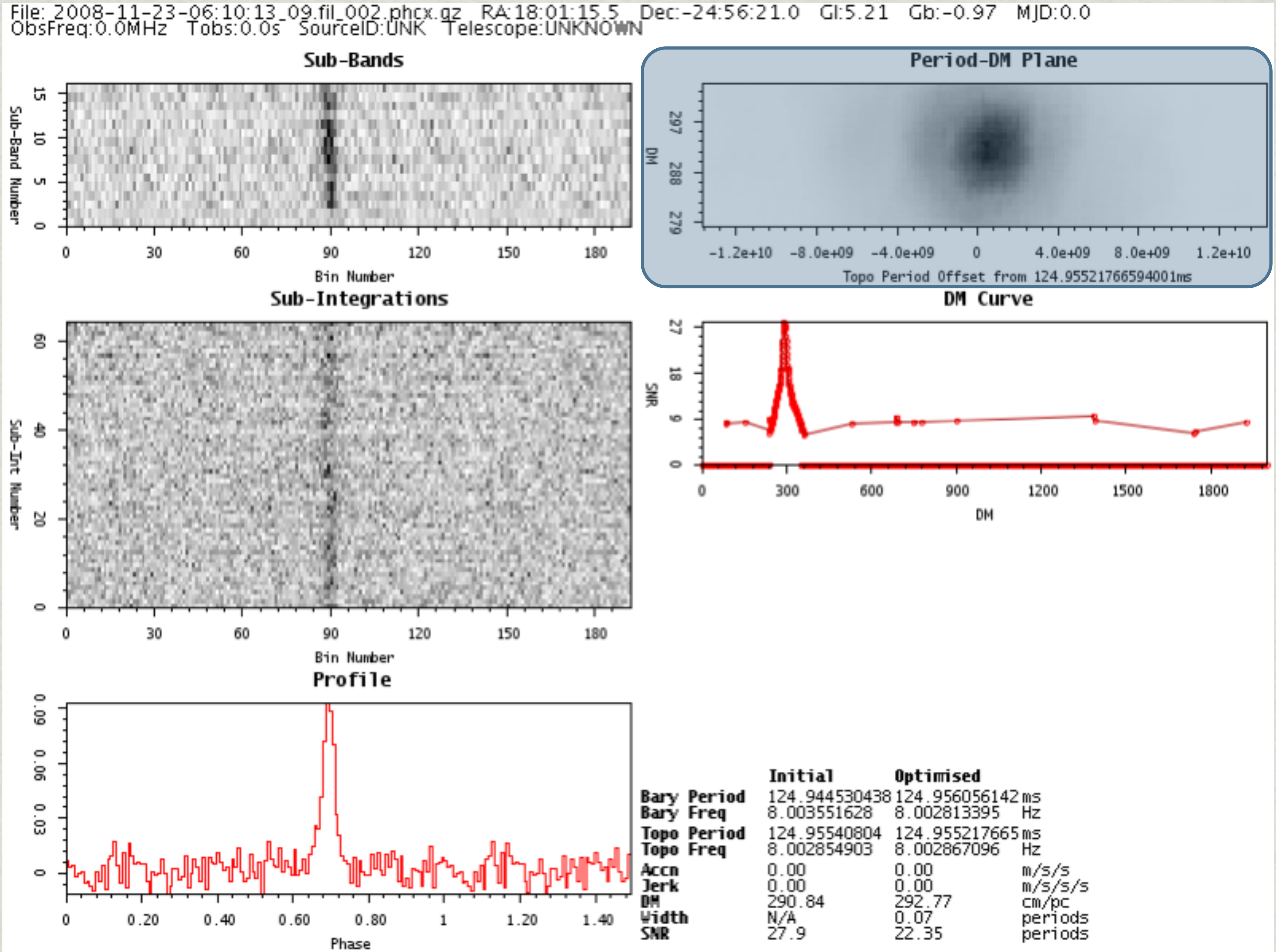
OPTIMISED CAND PLOTS



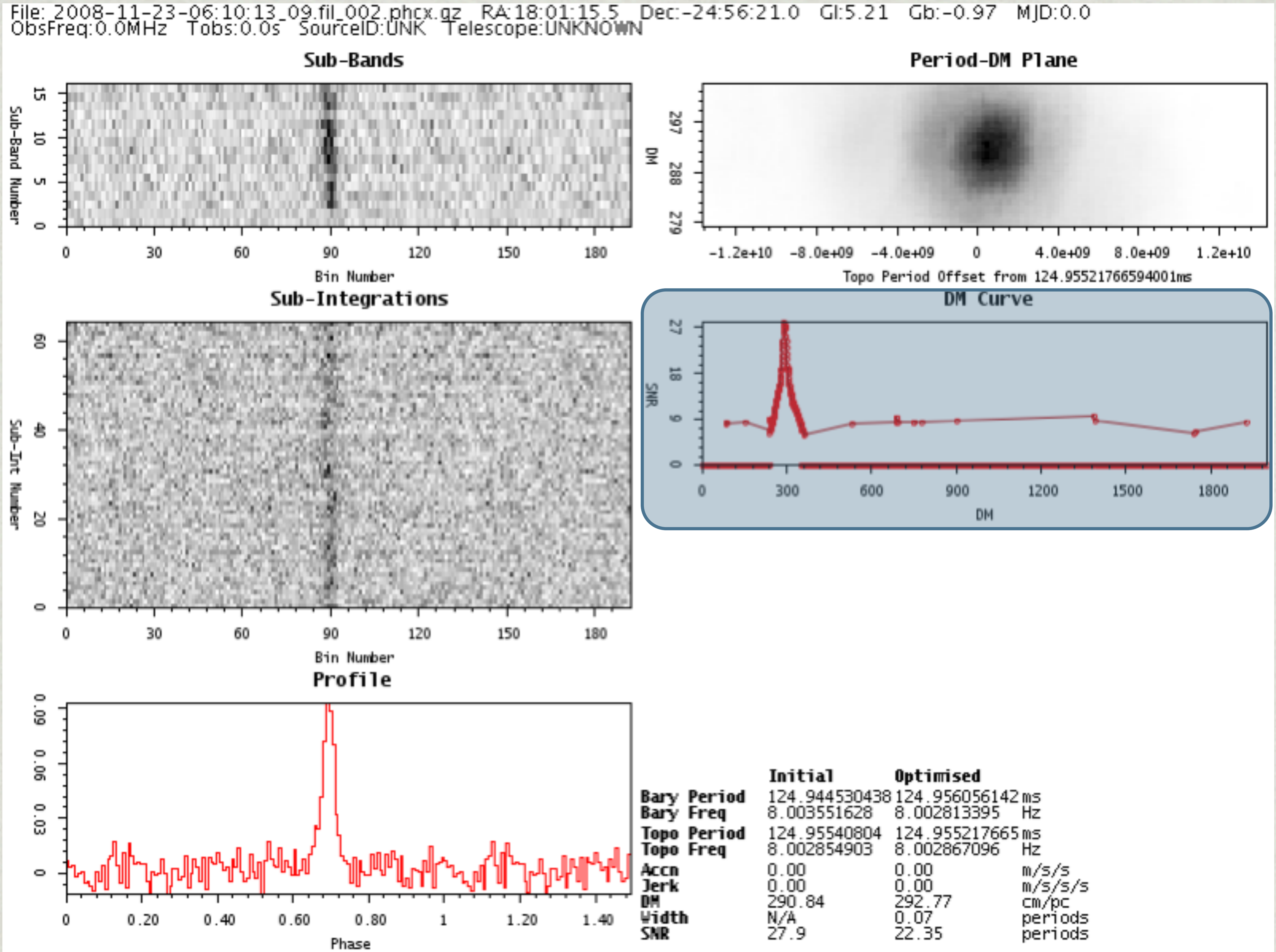
10-100S OF CANDIDATES → **FOLDING & OPTIMISATION** → OPTIMISED CAND PLOTS



10-100S OF CANDIDATES → FOLDING & OPTIMISATION → OPTIMISED CAND PLOTS



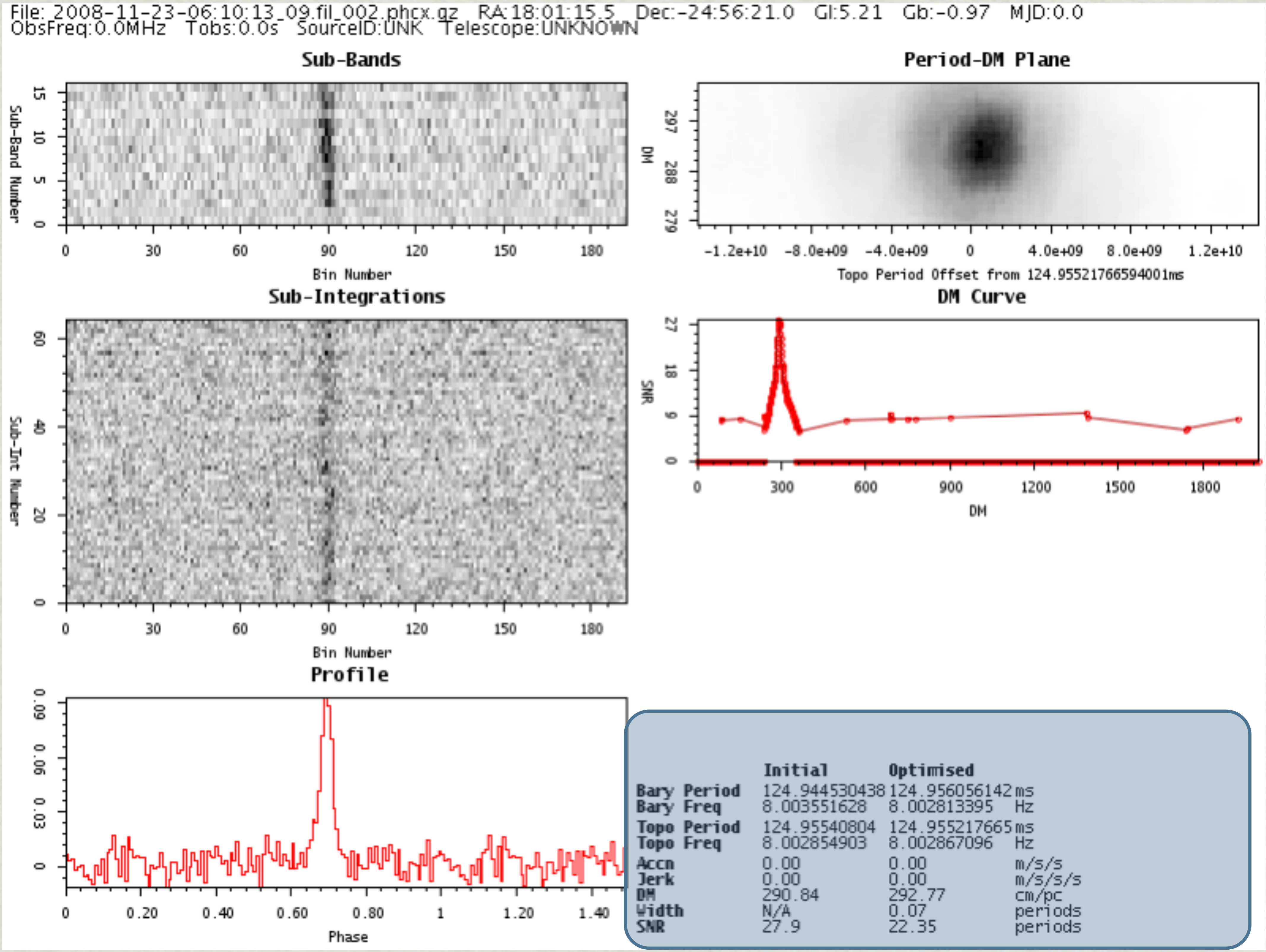
10-100S OF CANDIDATES → **FOLDING & OPTIMISATION** → OPTIMISED CAND PLOTS



10-100S OF CANDIDATES

FOLDING & OPTIMISATION

OPTIMISED CAND PLOTS



	Initial	Optimised	
Bary Period	124.944530438	124.956056142	ms
Bary Freq	8.003551628	8.002813395	Hz
Topo Period	124.95540804	124.955217665	ms
Topo Freq	8.002854903	8.002867096	Hz
Accn	0.00	0.00	m/s/s
Jerk	0.00	0.00	m/s/s/s
DM	290.84	292.77	cm/pc
Width	N/A	0.07	periods
SNR	27.9	22.35	periods

Installing required software

Required Software

- There are a variety of different software pipelines that can be used to search for pulsars. Here we will introduce the pipeline used for the HTRU pulsar survey.

- For this, we require a large number of software packages...

  `FFTW, cfitsio, pgplot, psrxml, pulsarchunter, pulsarhunter, presto, sigproc*, tempo2, psrchive & dspsr.`

- To help install all this, we can use a package installer such as “psrsoft”. Alternatively, all code can be downloaded and built by hand.

* note that we require a ‘non-standard’ version of sigproc, developed by the HTRU team and available at: https://github.com/SixByNine/sigproc

By hand method...

- Code can be downloaded from:

  FFTW:
  http://www.fftw.org/
  pgplot:
  http://www.astro.caltech.edu/~tjp/pgplot/
  cfitsio:
  http://heasarc.gsfc.nasa.gov/fitsio/
  sigproc, **psrxml**, **pulsarhunter**, **pulsarChunter**:
  https://github.com/SixByNine
  presto:
  https://github.com/scottransom/presto
  psrchive:
  http://psrchive.sf.net/
  dspsr:
  http://dspsr.sf.net/
  tempo2:
  http://tempo2.sf.net/

- Follow the build instructions...

PSRSoft method...

- PSRSoft is a automated installer system for compiling various pulsar software packages. It can help you get the software required for pulsar searching. It can be downloaded from:
  http://www.pulsarastronomy.net/wiki/Software/PSRSoft

```
Terminal — bash — 80×25
mkeith@Exit> ls
psrsoft.tar.gz
mkeith@Exit> tar -xvzf psrsoft.tar.gz
psrsoft/bin/
psrsoft/bin/psrsoft-selfupdate
psrsoft/bin/psrsoft-update-index
psrsoft/bin/shell_files/
psrsoft/bin/shell_files/psrsoft.tcshrc
psrsoft/bin/shell_files/psrsoft.bashrc
psrsoft/bin/shell_files/.tcshrc
psrsoft/bin/psrsoft.py
psrsoft/bin/psrsoft-shell
psrsoft/bin/psrsoft
psrsoft/config/profile.example
mkeith@Exit>
```

Terminal — bash — 80×25

```
mkeith@Exit> cd psrsoft
mkeith@Exit> ls
bin     config
mkeith@Exit> cd config
mkeith@Exit> ls
profile.example
mkeith@Exit> cp profile.example profile
mkeith@Exit> vim profile
```

Terminal — bash — 80×25

```
# Now the standard compiler options.

export LDFLAGS="-L$PSRSOFT_USR/lib $LDFLAGS"
export CFLAGS="-I$PSRSOFT_USR/include $CFLAGS"
export CXXFLAGS="-I$PSRSOFT_USR/include $CXXFLAGS"
export CC=gcc  # Make sure to set this to your C compiler
export CXX=g++ # And for C++
export F77=gfortran # A compatible F77 compiler (prefer gfortran)
export FC=$F77
export FLIBS=-lgfortran # The flags required to link fortran to C.
export PYTHON=/usr/bin/python

# this is the path to your java install root.
# The java binary must be at $JAVA_HOME/bin/java
# The javac binary must be at $JDK_HOME/bin/javac
# Common values are:
# MacOSX -- /usr/
# Linux  -- /usr/local/jdk*/  (or similar)
export JAVA_HOME=/usr/
export JDK_HOME=$JAVA_HOME

# make sure the java binaries are in the path
export PATH=$JDK_HOME/bin:${PATH}
```

MAKE SURE TO SET THESE VALUES!

Terminal — Python — 80×31

```
mkeith@Exit> ./bin/psrsoft hitrun --stable
==== PSRSOFT version 1.5 ====
 Pkg Index: 'stable'
==============================
Updating package index
Getting latest package descriptions... done
Searching for package hitrun in stable tree
1) hitrun 2.7

Analysing dependancies


Packages to be installed...
============================


  1            fftw |N | (3.1.2 2011-01-18 14:39)
  2          psrxml |NO| (1.05 2010-02-04 12:30)
  3    pulsarchunter |NO| (1.10 2010-11-08 16:24)
  4          pgplot |N | (5.2 2010-09-09 11:32)
  5         cfitsio |NO| (3090 2009-10-06 14:47)
  6     presto-core |NO| (1.01 2011-02-08 17:42)
  7     pulsarhunter |N | (1.3r79 2011-02-23 15:21)
  8         sixproc |NO| (5.0.5 2011-04-07 09:49)
  9          tempo2 |NO| (1.10 2010-12-09 11:17)
 10           tempo |N | (11.010_sf 2011-02-12 12:30)
 11        psrchive |N | (13.4+ 2011-02-12 11:41)
 12           dspsr |N | (2.0 2011-02-01 10:59)
 13          hitrun |N | (2.7 2010-12-20 15:58)
Install 13 packages into /Users/mkeith/tmp/psrsoft_test/psrsoft/usr? (y/n)
```

THIS “HITRUN” PACKAGE INSTALLS ALL CODE REQUIRED FOR THE HTRU SEARCH PIPELINE

Installing fftw — bash — 80×31

```
 10          tempo |N | (11.010_sf 2011-02-12 12:30)
 11       psrchive |N | (13.4+ 2011-02-12 11:41)
 12          dspsr |N | (2.0 2011-02-01 10:59)
 13         hitrun |N | (2.7 2010-12-20 15:58)
Install 13 packages into /Users/mkeith/tmp/psrsoft_test/psrsoft/usr? (y/n)y
Installing...

/bin/bash -v ./var/psrsoft/uninstallers/fftw
/bin/bash: ./var/psrsoft/uninstallers/fftw: No such file or directory
wget --no-cache  -N http://www.atnf.csiro.au/people/Michael.Keith/psrsoft//stabl
e/fftw.tar.gz
--13:52:41--  http://www.atnf.csiro.au/people/Michael.Keith/psrsoft//stable/fftw
.tar.gz
           => `fftw.tar.gz'
Resolving www.atnf.csiro.au... 150.229.106.28
Connecting to www.atnf.csiro.au|150.229.106.28|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2,901,866 (2.8M) [application/x-tar]

100%[====================================>] 2,901,866      1.84M/s

13:52:42 (1.84 MB/s) - `fftw.tar.gz' saved [2901866/2901866]

Un-taring package tarball
x ./
x ./var/
x ./var/psrsoft/
x ./var/psrsoft/file_md5/
x ./var/psrsoft/file_md5/fftw
x ./var/psrsoft/uninstallers/
```

A specific example...

Data formats...

- **PSRFITS** (search mode). A newly developed FITS based format for storing time-series data. The ATNF Pulsar Data Archive uses this format, as do the PDFB instruments at Parkes.
- SigProc "**Filterbank**" format files. These files are used as input to the SigProc software package. This talk will assume we are going to convert to this format. The BPSR instrument at Parkes records in this format.
- **Presto** format files. Presto is another software package with a simple data format.
- There are also many many instrument specific data formats, however most can be converted to one of the above formats.

PSRFITS

- “Search mode” PSRFITS files are much more complex than most data formats, however they contain many header parameters that are missing from older formats.
- For this talk, let’s assume you have downloaded a **search-mode** PSRFITS format data file from e.g. the ATNF Parkes Pulsar Data Archive:
  - http://atoa.atnf.csiro.au/parkesdata.jsp
  - I will use S00314_1.sf as an example...
  - (This happens to be an observation of the Vela pulsar, but it will be useful as a test!)

Converting

- The methods that I introduce here require conversion to "sigproc filterbank" (.fil) format files before starting.
- Convert from any compatible format, including PSRFITS with e.g.

```
Done hitrun — bash — 80×17
mkeith@Exit> :psrsoft: ls
PulsarObs.tar
mkeith@Exit> :psrsoft: tar -xvf PulsarObs.tar
atnf-attribution.txt
S00314_1.sf
mkeith@Exit> :psrsoft: filterbank S00314_1.sf > S00314_1.fil
mkeith@Exit> :psrsoft: ls
PulsarObs.tar           S00314_1.sf
S00314_1.fil            atnf-attribution.txt
mkeith@Exit> :psrsoft:
```

Check the file

- We can check what we have with the “header” program...

Done hitrun — bash — 80×23

```
mkeith@Exit> :psrsoft: header S00314_1.fil
Data file                        : S00314_1.fil
Header size (bytes)              : 355
Data size (bytes)                : 6389760
Data type                        : filterbank (topocentric)
Telescope                        : Parkes
Datataking Machine               : ?????
Source Name                      : J0835-4510
Source RA (J2000)                : 08:35:20.7
Source DEC (J2000)               : -45:10:35.7
Frequency of channel 1 (MHz)     : 446.062500
Channel bandwidth      (MHz)     : -0.125000
Number of channels               : 256
Number of beams                  : 1
Beam number                      : 1
Time stamp of first sample (MJD) : 48381.368807870371
Gregorian date (YYYY/MM/DD)      : 1991/05/05
Sample time (us)                 : 300.00000
Number of samples                : 199680
Observation length (seconds)     : 59.9
Number of bits per sample        : 1
Number of IFs                    : 1
mkeith@Exit> :psrsoft:
```

PSRSoft & hitrun script

- The quickest way to set your environment to have the psrsoft software:

Done hitrun — bash — 80×7

```
mkeith@Exit> ./bin/psrsoft-shell
mkeith@Exit> :psrsoft: 
```

- The “hitrun” packages actually provides a handy script that does a search for you.

Done hitrun — tcsh — 80×8

```
mkeith@Exit> ./bin/psrsoft-shell
mkeith@Exit> :psrsoft: hitrun
HitRun Processing script
========================
Software Versions
/Users/mkeith/Code/psrsoft/usr/bin/hitrun [sigproc .fil file]

Create a new hitrun settings file? (y/n)
```

* Generate a new "hitrun_settings" file to customise the way the script works. Answer the questions or accept defaults

```
Done hitrun — bash — 108×41
mkeith@Exit> :psrsoft: hitrun
HitRun Processing script
========================
Software Versions
/Users/mkeith/Code/psrsoft/usr/bin/hitrun [sigproc .fil file]

Create a new hitrun settings file? (y/n)y

CREATE hitrun_settings file
===========================

This program will help you configure the 'hitrun' script
We will ask you a series of questions to customise the processing
routines that are in use.

Press return to conitnue





$HITRUN_FFT_MODE - Choose which FFT search routines to use.
 * For very large data file >2^26 points, use Presto only

   0) DISABLE FFT search
   1) Sigproc seek
   2) Presto accelsearch
   3) Both Presto and Sigproc
   4) Pulsarchunter pch-seek (experimental)
Choose FFT search mode [1]:
$HITRUN_FFT_MODE=SIGPROC
```

- Now, process a ".fil" file with: hitrun $filfile $setings file
- e.g.:

Done hitrun — bash — 108×41

```
mkeith@Exit> :psrsoft: cat hitrun_settings
#HITRUN SETTINGS - autogenerated on Sun 8 May 2011 20:40:43 EST
setenv HITRUN_FFT_MODE 'SIGPROC'
setenv HITRUN_BEST 'MJK'
setenv HITRUN_SP_SEARCH 'ONPLOT'
setenv HITRUN_KILLFILE_MODE 'SWIN'
setenv HITRUN_TIMEZAP_MODE 'GIANT'
setenv HITRUN_NTHREADS 2
set sigproc_snrlimit=4.0
set mindm=0
set maxdm=1000
set max_tim=99999
set giant_settings=''
set start_ch=0
set end_ch=255
set ddgulp=100000
mkeith@Exit> :psrsoft: hitrun S00314_1.fil ./hitrun_settings
HitRun Processing script
========================
Software Versions
HITRUN Processing S00314_1.fil (6.1M S00314_1.fil)
====================================================
Config options: (./hitrun_settings)
HITRUN                 = /Users/mkeith/Code/psrsoft/usr/share/hitrun
HITRUN_NTHREADS        = 2
HITRUN_KILLFILE_MODE = SWIN
HITRUN_FFT_MODE        = SIGPROC
HITRUN_TIMEZAP_MODE  = GIANT
HITRUN_SP_SEARCH       = ONPLOT ()
dm range               = 0 -> 1000
max_tim files at once= 99999
```

The Details!

- Running the “hitrun” script might get you the results you want, however...
- Things often go wrong, or not how you expect, therefore I’ll show you the particular bits of software that are used.
- Feel free to ignore a lot of the details and read up on it later!

De-dispersion

- We don't know the DM of any pulsar that is in this file. We can de-disperse our file using the "dedisperse_all" program (this is part of the HTRU version of SigProc).
- We specify the file, a range of DMs (here we do 0 to 1000) and the number of samples to load at once (here we choose 100000)
- This program is multi-threaded.

Done hitrun — bash — 80×12

```
mkeith@Exit> :psrsoft: dedisperse_all S00314_1.fil -d 0 1000 -g 1000000
maxdelay = 11161
1 subbands from 256 chans
Dividing output by 2 to scale to 1 byte per sample per subband
Gulp 0 Loading 199680 samples, i. e. 6389760 bytes of Raw Data

Skipping back 357152 bytes
Gulp 1 Loading 11161 samples, i. e. 357152 bytes of Raw Data

mkeith@Exit> :psrsoft:
```

De-dispersion

- Ok, so what did we produce?
- This produced 389 ".tim" files, containing a single time-series at DMs ranging from 0 to 1000.

Done hitrun — bash — 80×23

```
mkeith@Exit> :psrsoft: ls | head -n 5 && echo ...SNIP... && ls | tail -n 5
PulsarObs.tar
S00314_1.fil
S00314_1.fil.0000.00.tim
S00314_1.fil.0000.27.tim
S00314_1.fil.0000.54.tim
...SNIP...
S00314_1.fil.0967.02.tim
S00314_1.fil.0978.12.tim
S00314_1.fil.0989.35.tim
S00314_1.sf
atnf-attribution.txt
mkeith@Exit> :psrsoft:
```

Seek

- We can search these time-series with the “seek” program from sigproc.
- This does FFT, whitening, harmonic summing, etc.
- Creates a list of S/N and period in “.prd” file.
- Useful options are:

  -head -- put a header in the .prd file This is required for later use.

  -recon -- compute the “reconstructed S/N”.

  -fftw -- Use the FFTW library for a speed enhancement.

Done hitrun — bash — 80×41

```
mkeith@Exit> :psrsoft: seek S00314_1.fil.0064.13.tim -fftw -recon -head
 SEEK: is part of SIGPROC version: 4.3
 Timer is up and running...
 Working with time series data...
 Read       188522  samples...
 Reference DM:    64.125870
 Sampling time:    300.00000       us
 Nearest power of 2:              18
 Padding time series with additional zeros...
 Data length:            1  min           18  sec
 FFT: (fftw-3.1.2)...
 Forming amplitude spectrum. (Pmax=   10.000000000000000       s!)
 Raw spectral resolution:    12.715657      mHz)
 Nyquist frequency:   1666.6666       Hz
 Whitening spectrum...
 Calculating AGL mean and rms every          128  bins...   1.6276040
 Resulting spectral RMS:   1.0606314
 Harmonic sums are:      1    2    4    8   16
 Doing harmonic summing...
 Lyne-Ashworth harmonic summing
 Doing harmonic searching...
 SNR threshold for fold            1 is   5.0000000
 SNR threshold for fold            2 is   4.7500000
 SNR threshold for fold            3 is   4.5500002
 SNR threshold for fold            4 is   4.3699999
 SNR threshold for fold            5 is   4.3000002
 Best suspect:   89.294344801165593       ms
 S/N: 224.5
 Found peak at:   11.198917492778856       Hz
 Number of harmonics:            8
 Timer clocked            1  s for this job.
mkeith@Exit> :psrsoft: ls S00314_1.fil.0064.13.*
S00314_1.fil.0064.13.prd          S00314_1.fil.0064.13.top
S00314_1.fil.0064.13.tim
mkeith@Exit> :psrsoft: 
```

Seek

- We need to seek every ".tim" file.
- Note:
  - seek is single-threaded, so you might want to run several at once!

Done hitrun — bash — 80×41

```
mkeith@Exit> :psrsoft: for t in *.tim ; do seek $t -fftw -recon; done > /de
l
mkeith@Exit> :psrsoft: seek S00314_1.fil.0000.00.tim -fftw -recon -head > /
ull
mkeith@Exit> :psrsoft: cat S00314_1.*.prd > all.prd
mkeith@Exit> :psrsoft: head -n 30 all.prd
##BEGIN HEADER##
SOURCEID = J0835-4510
FREF =     446.1 MHz
TSTART =    48381.3688
TELESCOPE = Parkes
RAJ = 08:35:20.7
DECJ = -45:10:35.7
TSAMP =     300.0 us
PROGRAM = SEEK
VERSION = 4.3
HARM_FOLDS =    1   2   4   8  16
BARYCENTRIC = F
COLS = SNR_SPEC SNR_RECON PERIOD
##END HEADER##
 DM:    0.0000000      AC:    0.0000000      AD:    0.0000000      TSAMP:   3.0
14249235392E-004
     38.1     40.1   89.29117654     40.7     39.7    4.80003685     46.3     41.
4.80001854     45.5     37.1   19.20014740     43.5     33.2   19.20007415
     30.4     35.4    4.80007347     37.8     38.5   89.27850573     36.9     29.
9.29751329     29.4     21.5   89.29434480     25.1     13.7  153.62344837
     30.3     34.6    9.99818247     27.0     31.1   19.99636493     29.3     22.
6.78594372     26.2     21.5  153.57188744     22.7     16.4   89.29276064
     25.8      3.9   89.18991067     26.1     22.4   38.39297186     24.4     23.
9.99477618     22.7     18.6   19.99525278     19.7     13.5   19.99517334
     25.8     31.1   89.39267262     19.1      5.2   89.17726859     21.0     19.
8.00665810     20.7     15.4  267.97811943     17.0     11.1  357.29147718
     17.0     12.5   38.38594447     18.8      8.0   89.43079316     16.2     20.
8.45571689     17.2     22.7  178.49368888     16.2     16.8   24.00002403
     16.3     20.6   76.81875823     15.5     18.7  153.63751645     15.2      4.
9.14567905     13.0     16.0  624.92737013     16.1     10.8  307.06882064
     15.6     13.1   16.93892709     14.1     12.0   16.94029537     14.3     19.
7.57069047     12.6     15.1  614.55006582     15.3     14.7   33.60035408
      8.1     10.2   14.94052790     12.9      8.7   76.73443467     14.1     16.
7.27503291     11.6     14.2   33.59968116     14.9     12.4  267.67882992
```

ph-best

- The next step is to collect together the many many periods that are in the .prd file
- We use “ph-best” from pulsarhunter. It requires the .prd file name and a file stem for the candidate files.
- e.g. `ph-best all.prd candidate`
- here we created 59 candidates.

Done hitrun — bash — 116×42

Spec SNR	Recon SNR	Period	DM	Accel	Jerk	#Harm	Fold
392.800000	389.600000	89.299098	68.129417	0.000000e+00	0.000000e+00	154	16
81.500000	89.600000	89.392673	52.657421	0.000000e+00	0.000000e+00	45	1
81.200000	19.900000	89.189911	77.737404	0.000000e+00	0.000000e+00	59	1
72.000000	65.800000	4.800019	44.650597	0.000000e+00	0.000000e+00	96	4
64.600000	69.000000	55.806243	66.502129	0.000000e+00	0.000000e+00	51	8
52.000000	40.500000	145.102345	68.129417	0.000000e+00	0.000000e+00	54	16
46.100000	48.600000	238.154871	63.350616	0.000000e+00	0.000000e+00	49	8
40.000000	31.500000	153.571887	923.861820	0.000000e+00	0.000000e+00	86	8
39.100000	35.900000	14.883623	60.331085	0.000000e+00	0.000000e+00	33	2
37.800000	26.900000	196.431528	70.637154	0.000000e+00	0.000000e+00	35	16
36.000000	27.600000	232.167768	70.637154	0.000000e+00	0.000000e+00	25	16
32.700000	34.100000	1338.963831	77.737404	0.000000e+00	0.000000e+00	23	16
32.700000	32.600000	114.810216	69.792191	0.000000e+00	0.000000e+00	31	16
31.200000	32.500000	59.516189	37.123585	0.000000e+00	0.000000e+00	31	2
30.300000	34.600000	9.998182	0.000000	0.000000e+00	0.000000e+00	82	1
30.200000	35.800000	715.141381	52.657421	0.000000e+00	0.000000e+00	19	8
28.800000	32.400000	290.213057	64.909470	0.000000e+00	0.000000e+00	9	16

ph-best

- The next step is to collect together the many many periods that are in the .prd file
- We use “ph-best” from pulsarhunter. It requires the .prd file name and a file stem for the candidate files.
- e.g. `ph-best all.prd candidate`
- here we created 59 candidates.

Done hitrun — bash — 116×42

Spec SNR	Recon SNR	Period	DM	Accel	Jerk	#Harm	Fold
392.800000	389.600000	89.299098	68.129417	e+00	0.000000e+00	154	16
81.500000	89.600000	89.392673	52.657421	e+00	0.000000e+00	45	1
81.200000	19.900000	89.189911	77.737404	0.000000e+00	0.000000e+00	59	1
72.000000	65.800000	4.800019	44.650597	0.000000e+00	0.000000e+00	96	4
64.600000	69.000000	55.806243	66.502129	0.000000e+00	0.000000e+00	51	8
52.000000	40.500000	145.102345	68.129417	0.000000e+00	0.000000e+00	54	16
46.100000	48.600000	238.154871	63.350616	0.000000e+00	0.000000e+00	49	8
40.000000	31.500000	153.571887	923.861820	0.000000e+00	0.000000e+00	86	8
39.100000	35.900000	14.883623	60.331085	0.000000e+00	0.000000e+00	33	2
37.800000	26.900000	196.431528	70.637154	0.000000e+00	0.000000e+00	35	16
36.000000	27.600000	232.167768	70.637154	0.000000e+00	0.000000e+00	25	16
32.700000	34.100000	1338.963831	77.737404	0.000000e+00	0.000000e+00	23	16
32.700000	32.600000	114.810216	69.792191	0.000000e+00	0.000000e+00	31	16
31.200000	32.500000	59.516189	37.123585	00e+00	0.000000e+00	31	2
30.300000	34.600000	9.998182	0.000000	00e+00	0.000000e+00	82	1
30.200000	35.800000	715.141381	52.657421	00e+00	0.000000e+00	19	8
28.800000	32.400000	290.213057	64.909470	0.000000e+00	0.000000e+00	9	16

VELA

RFI

ph-best

- ph-best writes out a single ".lis" file that summarises the candidates, and a ".phcx.gz" file for each candidate.
- The ".phcx.gz" file is an XML file containing details of the candidate that will be used later.

Done hitrun — bash — 80×42

```
mkeith@Exit> :psrsoft: ls candidate*
candidate.lis           candidate_020.phcx.gz   candidate_040.phcx.gz
candidate_001.phcx.gz   candidate_021.phcx.gz   candidate_041.phcx.gz
candidate_002.phcx.gz   candidate_022.phcx.gz   candidate_042.phcx.gz
candidate_003.phcx.gz   candidate_023.phcx.gz   candidate_043.phcx.gz
candidate_004.phcx.gz   candidate_024.phcx.gz   candidate_044.phcx.gz
candidate_005.phcx.gz   candidate_025.phcx.gz   candidate_045.phcx.gz
candidate_006.phcx.gz   candidate_026.phcx.gz   candidate_046.phcx.gz
candidate_007.phcx.gz   candidate_027.phcx.gz   candidate_047.phcx.gz
candidate_008.phcx.gz   candidate_028.phcx.gz   candidate_048.phcx.gz
candidate_009.phcx.gz   candidate_029.phcx.gz   candidate_049.phcx.gz
candidate_010.phcx.gz   candidate_030.phcx.gz   candidate_050.phcx.gz
candidate_011.phcx.gz   candidate_031.phcx.gz   candidate_051.phcx.gz
candidate_012.phcx.gz   candidate_032.phcx.gz   candidate_052.phcx.gz
candidate_013.phcx.gz   candidate_033.phcx.gz   candidate_053.phcx.gz
candidate_014.phcx.gz   candidate_034.phcx.gz   candidate_054.phcx.gz
candidate_015.phcx.gz   candidate_035.phcx.gz   candidate_055.phcx.gz
candidate_016.phcx.gz   candidate_036.phcx.gz   candidate_056.phcx.gz
candidate_017.phcx.gz   candidate_037.phcx.gz   candidate_057.phcx.gz
candidate_018.phcx.gz   candidate_038.phcx.gz   candidate_058.phcx.gz
candidate_019.phcx.gz   candidate_039.phcx.gz   candidate_059.phcx.gz
mkeith@Exit> :psrsoft:
```

dspsr

- Now we have some candidates, we can fold the raw data using the candidate period/DM.
- To do this we can use "dspsr".

- dspsr can fold using a pulsar ephemeris, or we can provide a simple file containing period, dm and acceleration, in the format below:

```
Done hitrun — bash — 80×42
mkeith@Exit> :psrsoft: cat candidate_001.dat
SOURCE: candidate_001_ars
PERIOD:   0.089299097620
ACC: 0.000000e+00
DM: 68.129417
EPOCH: 48381.3691545370
mkeith@Exit> :psrsoft:
```

dspsr

- We can make these files using ph-lis2dspsr, which also prints the dspsr command for folding the data...
- We use -L10 -t2 to make 10s sub-integrations and use 2 threads.

Done hitrun — dspsr — 130×42

```
mkeith@Exit> :psrsoft: ph-lis2dspsr S00314_1.fil candidate.lis 'dspsr -L10 -t2' > do_dspsr
mkeith@Exit> :psrsoft: cat do_dspsr
dspsr -L10 -t2 -P candidate_001.dat -P candidate_002.dat -P candidate_003.dat -P candidate_004.dat -P candidate_005.dat -P candida
te_006.dat -P candidate_007.dat -P candidate_008.dat -P candidate_009.dat -P candidate_010.dat S00314_1.fil
dspsr -L10 -t2 -P candidate_011.dat -P candidate_012.dat -P candidate_013.dat -P candidate_014.dat -P candidate_015.dat -P candida
te_016.dat -P candidate_017.dat -P candidate_018.dat -P candidate_019.dat -P candidate_020.dat S00314_1.fil
dspsr -L10 -t2 -P candidate_021.dat -P candidate_022.dat -P candidate_023.dat -P candidate_024.dat -P candidate_025.dat -P candida
te_026.dat -P candidate_027.dat -P candidate_028.dat -P candidate_029.dat -P candidate_030.dat S00314_1.fil
dspsr -L10 -t2 -P candidate_031.dat -P candidate_032.dat -P candidate_033.dat -P candidate_034.dat -P candidate_035.dat -P candida
te_036.dat -P candidate_037.dat -P candidate_038.dat -P candidate_039.dat -P candidate_040.dat S00314_1.fil
dspsr -L10 -t2 -P candidate_041.dat -P candidate_042.dat -P candidate_043.dat -P candidate_044.dat -P candidate_045.dat -P candida
te_046.dat -P candidate_047.dat -P candidate_048.dat -P candidate_049.dat -P candidate_050.dat S00314_1.fil
dspsr -L10 -t2 -P candidate_051.dat -P candidate_052.dat -P candidate_053.dat -P candidate_054.dat -P candidate_055.dat -P candida
te_056.dat -P candidate_057.dat -P candidate_058.dat -P candidate_059.dat S00314_1.fil
mkeith@Exit> :psrsoft: sh do_dspsr
dspsr: Loading phase model from candidate_001.dat
dspsr: Loading phase model from candidate_002.dat
dspsr: Loading phase model from candidate_003.dat
dspsr: Loading phase model from candidate_004.dat
dspsr: Loading phase model from candidate_005.dat
dspsr: Loading phase model from candidate_006.dat
dspsr: Loading phase model from candidate_007.dat
dspsr: Loading phase model from candidate_008.dat
dspsr: Loading phase model from candidate_009.dat
dspsr: Loading phase model from candidate_010.dat
dspsr: blocksize=254200 samples or 256 MB
unloading 5 seconds: 1991-05-05-08:51:00
unloading 10 seconds: 1991-05-05-08:51:10
unloading 10 seconds: 1991-05-05-08:51:20
```

psrchive

- After dspsr, we end up with a directory per candidate (in our example, this is called candidate_???_ars).
- This directory contains 1 “.ar” file per sub-integration.
- Now we have done folded the data, we can use psrchive to do a variety of things (see talk by Willem van Straten).

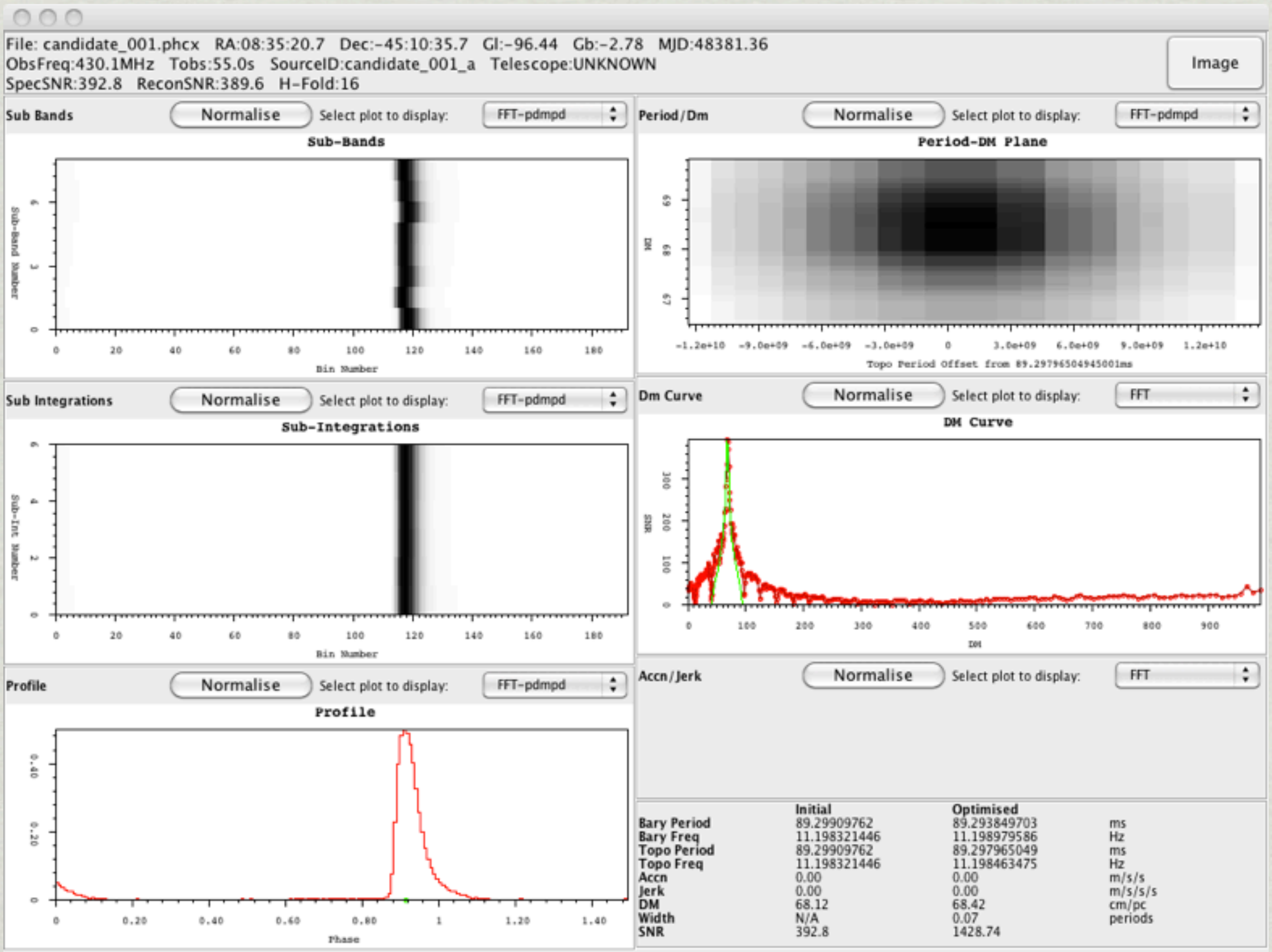
```
Done hitrun — bash — 134×42
mkeith@Exit> :psrsoft: ls candidate_001_ars/
1991-05-05-08:51:00.ar  1991-05-05-08:51:20.ar  1991-05-05-08:51:40.ar
1991-05-05-08:51:10.ar  1991-05-05-08:51:30.ar  1991-05-05-08:51:50.ar
mkeith@Exit> :psrsoft: psradd candidate_001_ars/*.ar -o candidate_001.ar
mkeith@Exit> :psrsoft: pam --setnchn 8 -m candidate_001.ar
candidate_001.ar written to disk
mkeith@Exit> :psrsoft: pdmp -g candidate_001.ps/ps -input-phcx candidate_001.phcx.gz -output-phcx candidate_001.phcx candidate_001.ar

Working on archive candidate_001_a: candidate_001.ar
Searching for optimum DM and Period...
 DM: 34   P1: 0   P0: 23
Outputting candidate to candidate_001.phcx

Best S/N = 1428.75
BC MJD = 48381.370729
BC Period (ms) = 89.29498222  TC Period (ms) =  89.29909762  DM = 68.1
Best BC Period (ms) = 89.2938497  Correction (ms) = -0.001132518354  Error (ms) = 0.001868773683
Best TC Period (ms) = 89.29796505  Correction (ms) = -0.001132570549  Error (ms) = 0.001868773683
Best DM = 68.4  Correction = 0.295  Error = 0.358
Best BC Frequency (Hz) = 11.19897959  Error (Hz) = 0.0002343762577
Pulse width (bins) = 9
Best Pdot =   0 s/s
Best Accn =  -0 m/s

pdmp took 0.26 seconds
mkeith@Exit> :psrsoft: ph-view-phcx candidate_001.phcx
mkeith@Exit> :psrsoft: 
```

psrchive


File: candidate_001.phcx RA:08:35:20.7 Dec:-45:10:35.7 Gl:-96.44 Gb:-2.78 MJD:48381.36
ObsFreq:430.1MHz Tobs:55.0s SourceID:candidate_001_a Telescope:UNKNOWN
SpecSNR:392.8 ReconSNR:389.6 H-Fold:16
Image
Sub Bands
Normalise
Select plot to display:
FFT-pdmpd
Sub-Bands
Sub-Band Number
Bin Number
Period/Dm
Period-DM Plane
DM
Topo Period Offset from 89.29796504945001ms
Sub Integrations
Sub-Integrations
Sub-Int Number
Dm Curve
FFT
DM Curve
SNR
Profile
Phase
Accn/Jerk
Initial
Optimised
Bary Period 89.29909762 89.293849703 ms
Bary Freq 11.198321446 11.198979586 Hz
Topo Period 89.29909762 89.297965049 ms
Topo Freq 11.198321446 11.198463475 Hz
Accn 0.00 0.00 m/s/s
Jerk 0.00 0.00 m/s/s/s
DM 68.12 68.42 cm/pc
Width N/A 0.07 periods
SNR 392.8 1428.74

Finished

- The .ar archives and the phcx files are the end result of either the “by hand” method or the “hitrun” script.
- If you have a few beams, you can view all the plots by hand, however if there are lots, you will probably need a tool like “jreaper” or similar.

Alternatives/Options

- This basic search can be expanded with acceleration searching and various RFI removal techniques.

presto

- The signal from pulsars in tight binaries will be affected by the acceleration around the orbit. To correct for this we do “acceleration searches”
- Although we can do acceleration searching with sigproc, my preferred way to do acceleration searching is to use the “accelsearch” algorithm from the **presto** software package.
- We can use this as an alternative, or in addition to, seek.
- This method uses an efficient “matched filter” technique for finding accelerated signals. See Ransom et al. (2002) for details!

presto

Done hitrun — vim — 108×41

```
#!/bin/bash
filfile=*.fil
psrxml=`basename $filfile .fil`.psrxml

zmax=10  # this is the amount of acceleration searching to do. See Ransom et al. 2002

#if you don't have a psrxml file already...
makePsrXml $filfile > $psrxml

for timfile in *.tim; do
        dm=`header $timfile -dm`
        val=`pch-dmcomp $psrxml -d $dm -S -F`
        nscr=`echo $val | cut -d " " -f 1`
        nsamp=`echo $val | cut -d " " -f 2`
        prepdata $timfile -o ${timfile}_presto -filterbank -numout $nsamp -downsamp $nscr -dm $dm
        accelsearch -zmax $zmax -flo 1.0 -numharm 16 -locpow -harmpolish  ${timfile}_presto.dat
done
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Getting the dm, sensible FFT size, etc...

the actual presto search

presto

- Our presto search creates files with names similar to:
- S00314_1.fil.0000.00.tim_presto_accel_10
- we can also read these into ph-best with a little fiddling:

```
Done hitrun — bash — 80×7
mkeith@Exit> :psrsoft: ls *ACCEL*_1 > presto.accelsearch
mkeith@Exit> :psrsoft: ph-best presto.accelsearch prestocand
```

- And then continue as normal, except that the candidates may have a non-zero acceleration value!

Interference (RFI)

- The "hitrun" script has a couple of RFI removal techniques in it, though they may not work for all data.
- The first is to FFT each channel and then ignore any that have strong signals in them.
- The second is to remove any bright, narrow, spikes from the zero-DM time series. Care must be taken to avoid removing low DM pulsars!
- We are also working on multi-beam RFI rejection schemes that look for signals that are correlated across all 13 beams of the Parkes multibeam system

- RFI is always an issue, and getting rid of it may require spending time getting to know your particular data!

Conclusion

- Installing the required pulsar search software can be made easier with package managers such as “psrsoft”
- The “hitrun” script allows you to use the search system that has been developed for the HTRU survey.
- Any data can be used that can be converted to sigproc filterbank format (including PSRFITS).
- Results come as in the form of candidate files and psrchive archives. These can be either:
  - Plotted using ph-view-phcx or psrchive tools; or
  - loaded into “jreaper” to view many at once.
- You usually have to look through many many candidate plots before you find something that looks like a pulsar. Being able to spot the real signals is not easy and takes practice!