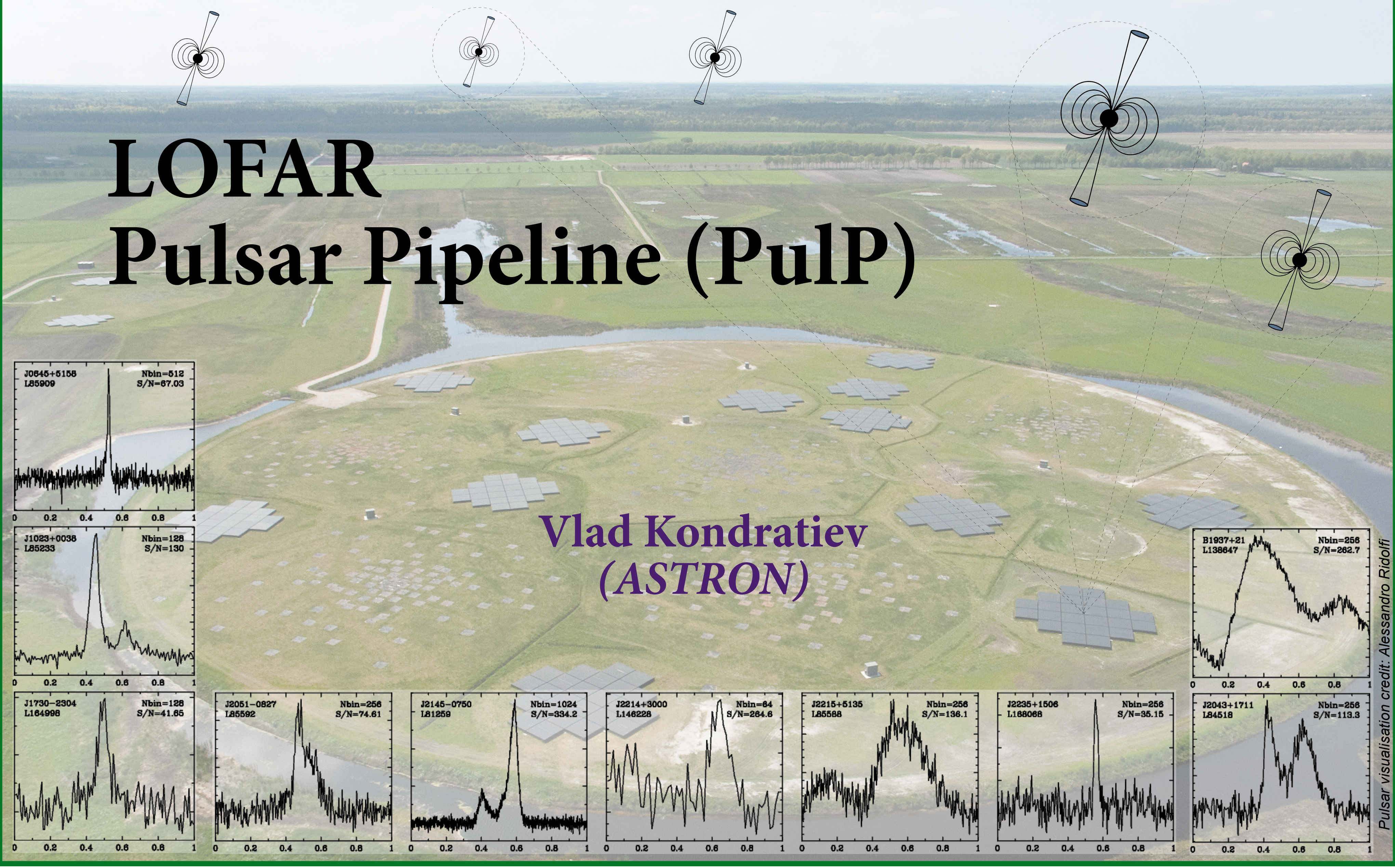
LOFAR Pulsar Pipeline (PulP)

Vlad Kondratiev
(ASTRON)



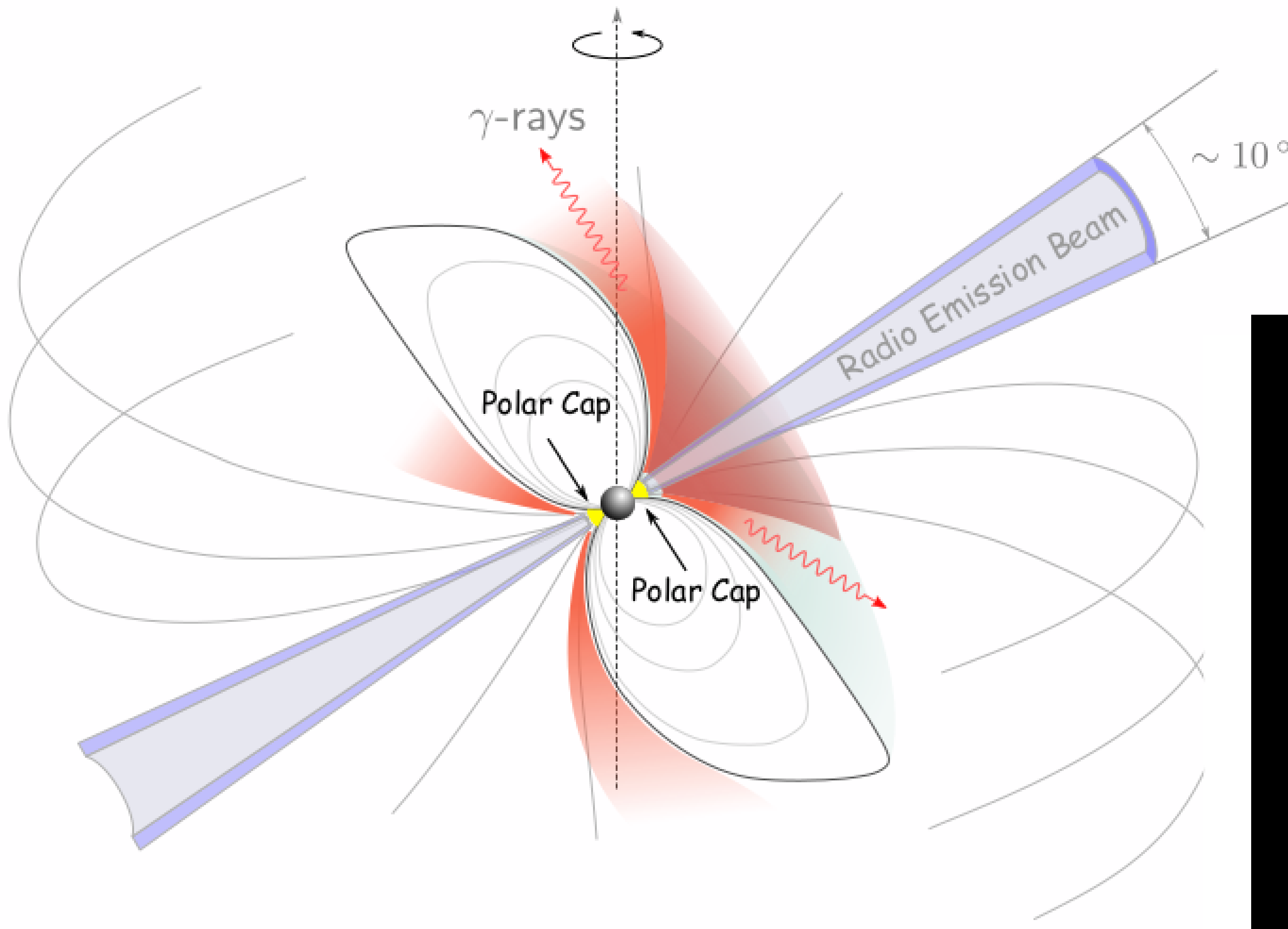
What is PulP?

PulP is LOFAR **Pul**sar **P**ipeline for *known pulsars*. The essential goal of the PulP is to get the average profile of the pulsar(s) and provide a user with freq/time/phase/pol data cubes for further analysis. It is ***not*** the *search* pipeline, i.e. you can not do periodicity and single-pulse searches for a large range of dispersion measure trials. However, PulP can provide both PSRFITS/filterbank data and raw data converted to 8-bit for further searches.

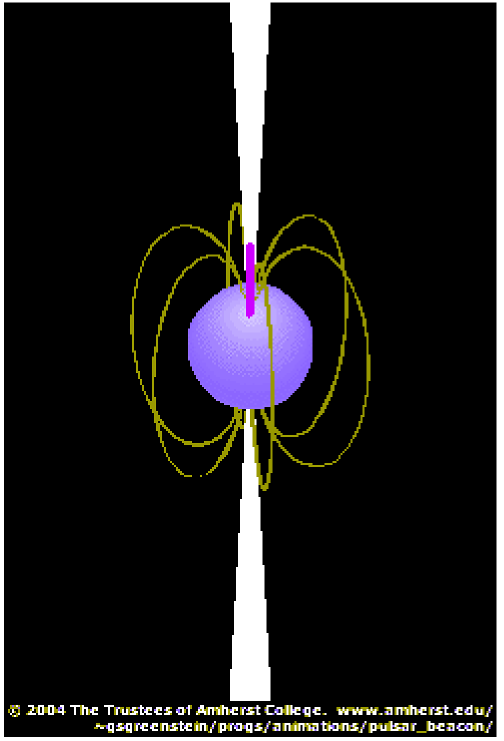
Disclaimer: In no way PulP can guarantee the optimal use of all the processing parameters for ***every*** observation/pulsar. The default parameters are only meant to provide the good results for most observations based on our current knowledge in the automated way. And serves to give the first diagnostics of the quality of the data.

What is pulsar?

rapidly rotating
highly-magnetised neutron star,
«electric lighthouse»



Credit: Andrey Timokhin



Outline:

- PulP overview
  - → briefly: PulP implementation
- In a nutshell about (de-)dispersion, folding
- PulP flowchart
  - → DSPSR pipeline
  - → PRESTO pipeline
- PulP output data
- PulP in the NorthStar
- PulP options

PulP overview

- Bookkeeping, service functions
  - → Logging
  - → Cluster configuration/settings
  - → User options
  - → Where input data are?
  - → Observing setup (HDF5 metadata / *parset*)
  - → Coordination of processing data for different TABs/frequency parts
  - → Feedback files for LTA ingest
- The actual data processing
- Diagnostic summaries and pipeline output data products

PulP overview

- Bookkeeping, service functions
  - → Logging
  - → Cluster configuration/settings
  - → User options
  - → Where input data are?
  - → Observing setup (HDF5 metadata / *parset*)
  - → Coordination of processing data for different TABs/frequency parts
  - → Feedback files for LTA ingest
- **The actual data processing**
- **Diagnostic summaries and pipeline output data products**

PulP overview

- Bookkeeping, service functions

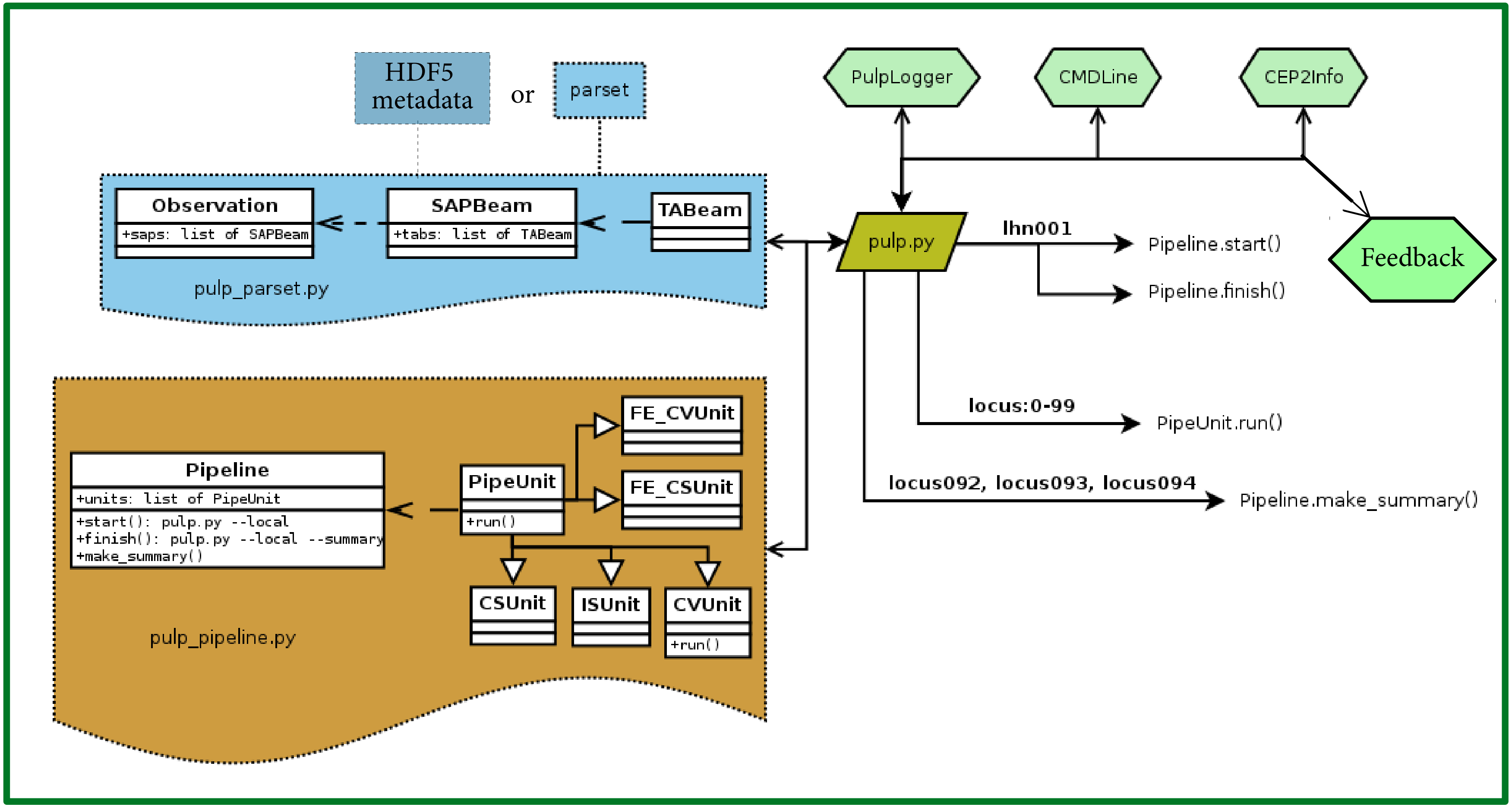
  → Logging
  → Cluster configuration/settings
  → User options
  → Where input data are?
  → Observing setup (HDF5 metadata / *parset*)
  → Coordination of processing data for different TABs/frequency parts
  → Feedback files for LTA ingest

- **The actual data processing**

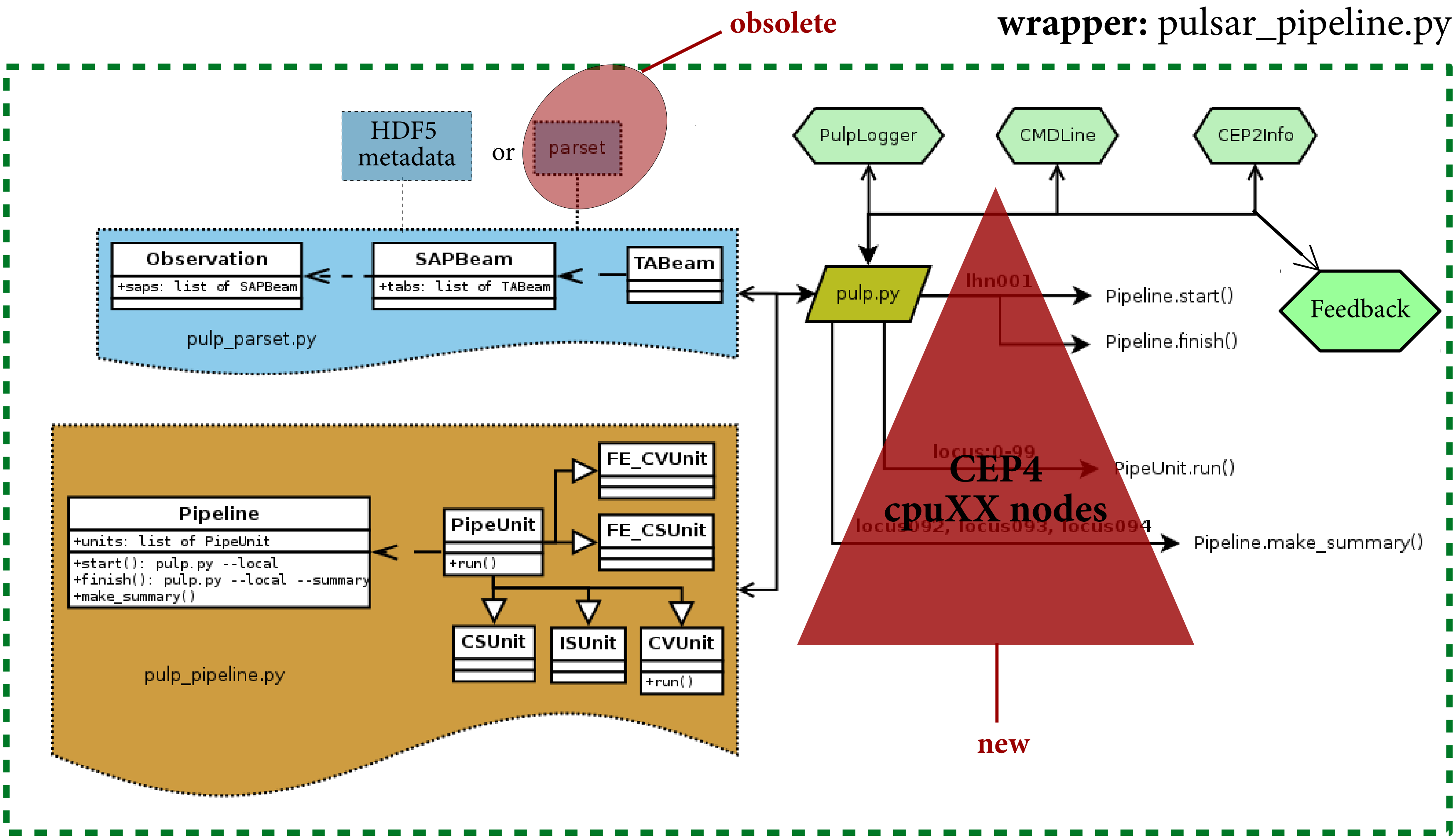
- **Diagnostic summaries and pipeline output data products**

NB: PulP is still implemented in the LOFAR system as a «**black box**» via the *wrapper* that calls the original PulP itself. This slows down bug fixing and further improvements quite significantly, and the plan is to truly incorporate Pulp into the central system as it is the case for imaging pipeline(s).

PulP implementation

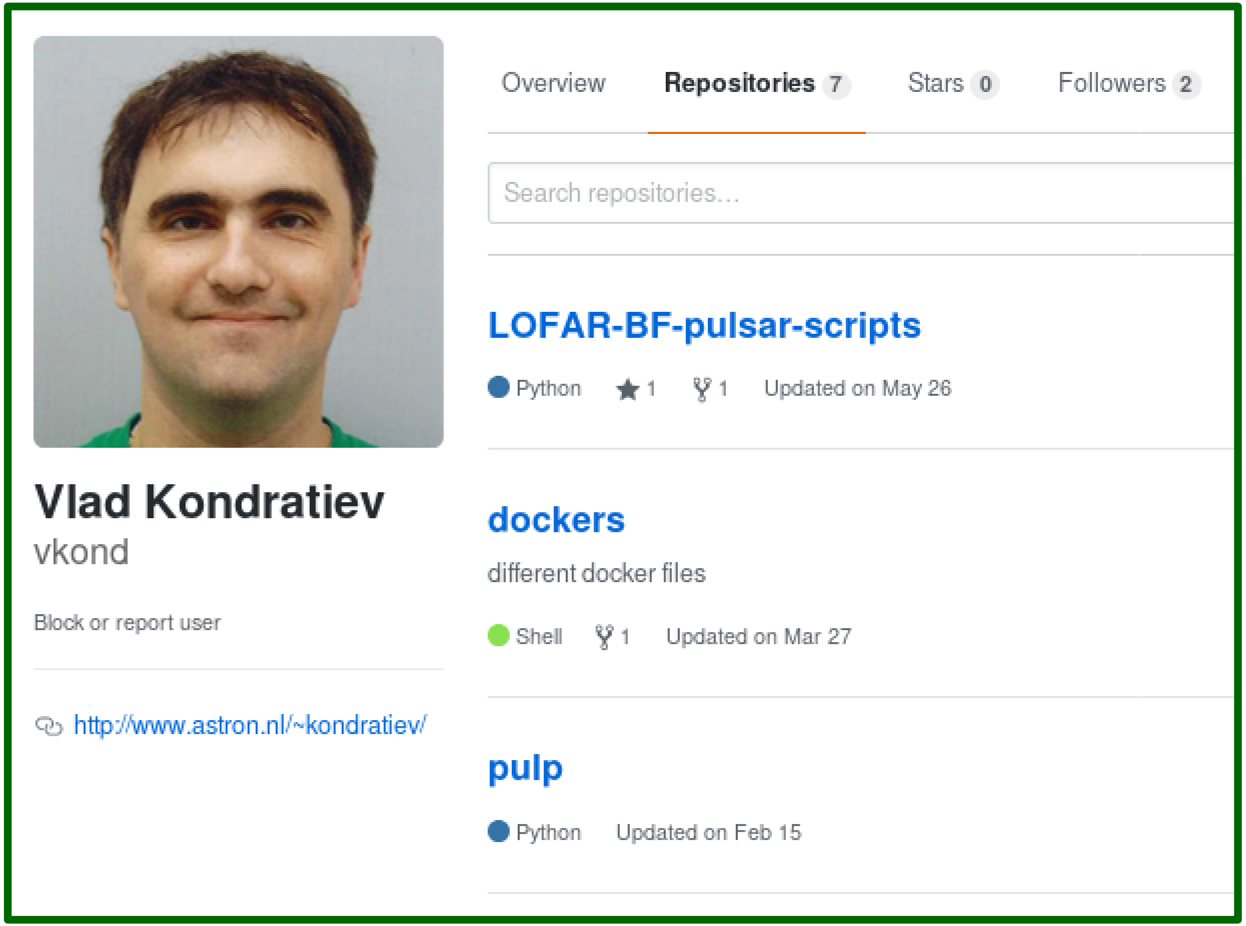


PulP implementation



@github

https://github.com/vkond



Various scripts

https://github.com/vkond/LOFAR-BF-pulsar-scripts

Dockerfile for LOFAR

https://github.com/vkond/dockers

PulP:
https://github.com/vkond/pulp

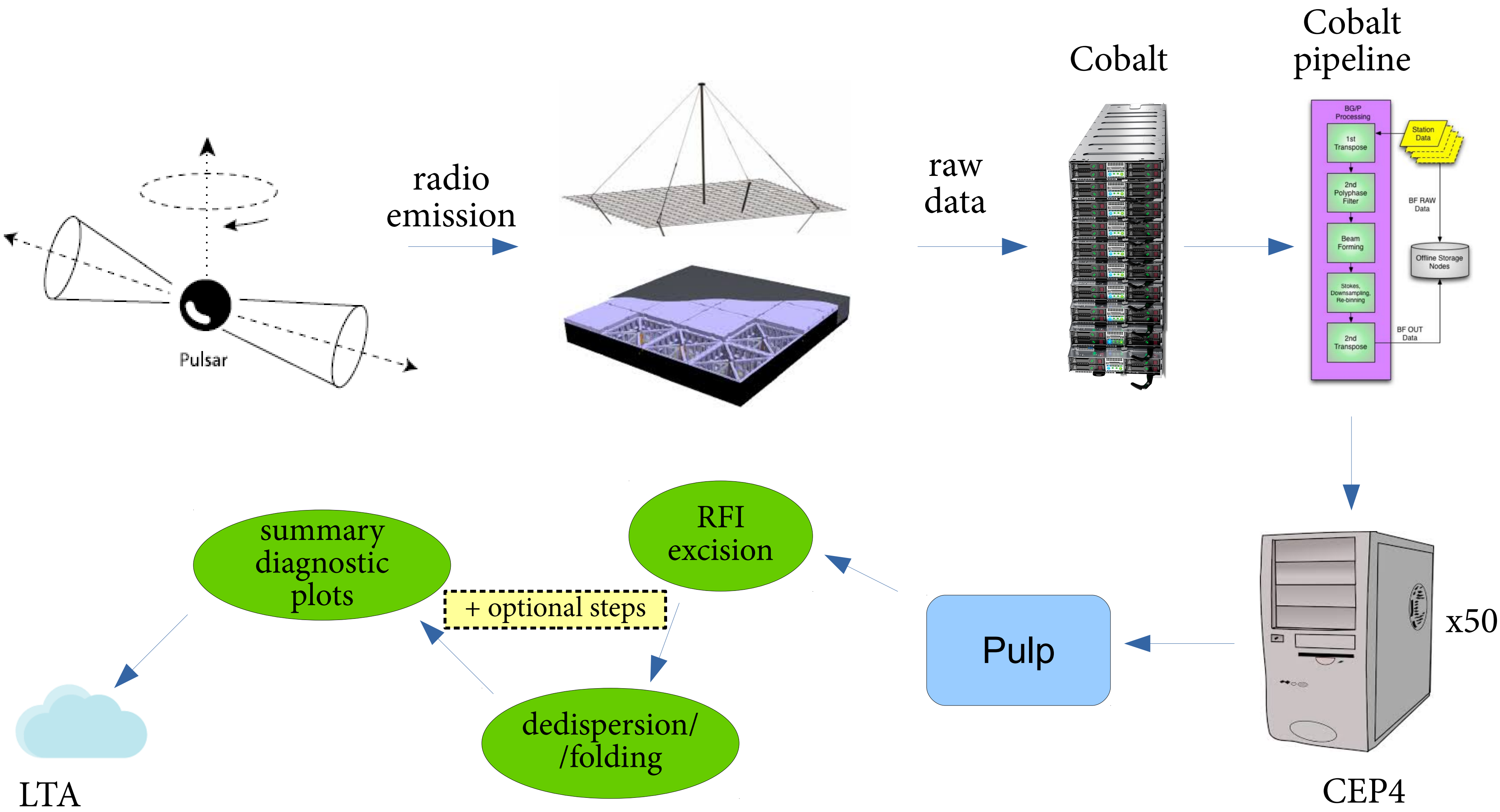
Pulsar software (needed by PulP)

- FFTW
- PGPLOT, + python bindings
- TEMPO
- TEMPO2
- psrcat
- Sigproc
- **PRESTO**
- psrdada
- **PSRCHIVE**
- DAL
- **DSPSR**
- COAST_GUARD (written by Patrick Lazarus, for RFI excision)
- LOFAR-BF-pulsar-scripts

- in the future (needed for pulsar flux calibration):
  - → casacore
  - → python-casacore
  - → mscorpol

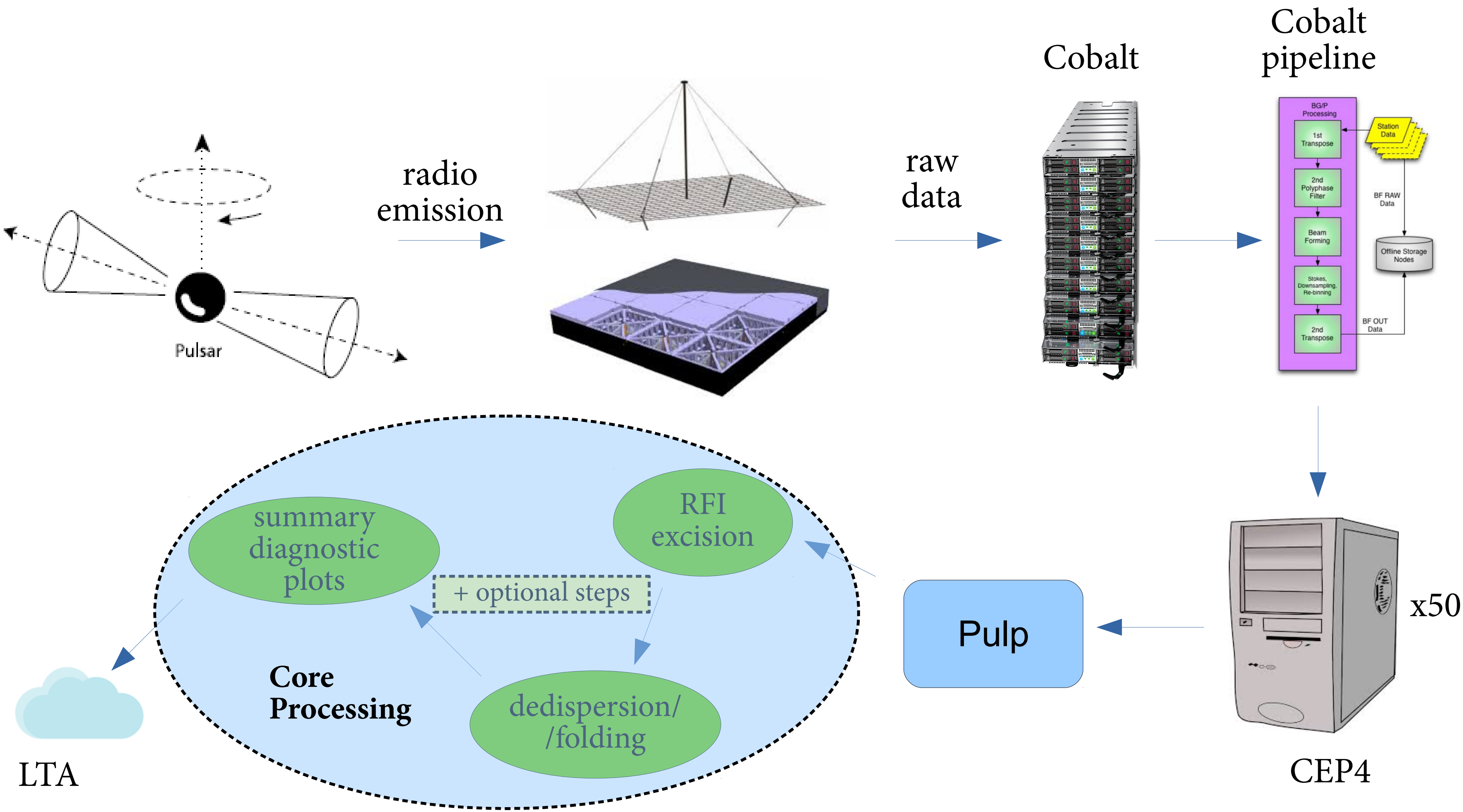
Lofar-pulp Docker container has all this software installed:
https://github.com/vkond/dockers

Data flow



Pulsar
radio emission
raw data
Cobalt
Cobalt pipeline
BG/P Processing
1st Transpose
2nd Polyphase Filter
Beam Forming
Stokes, Downsampling, Re-binning
2nd Transpose
Station Data
BF RAW Data
Offline Storage Nodes
BF OUT Data
x50
CEP4
Pulp
RFI excision
dedispersion/ /folding
+ optional steps
summary diagnostic plots
LTA

Data flow



Pulsar
radio emission
raw data
Cobalt
Cobalt pipeline
BG/P Processing
1st Transpose
2nd Polyphase Filter
Beam Forming
Stokes, Downsampling, Re-binning
2nd Transpose
Station Data
BF RAW Data
Offline Storage Nodes
BF OUT Data
x50
CEP4
Pulp
RFI excision
dedispersion/ /folding
summary diagnostic plots
+ optional steps
Core Processing
LTA

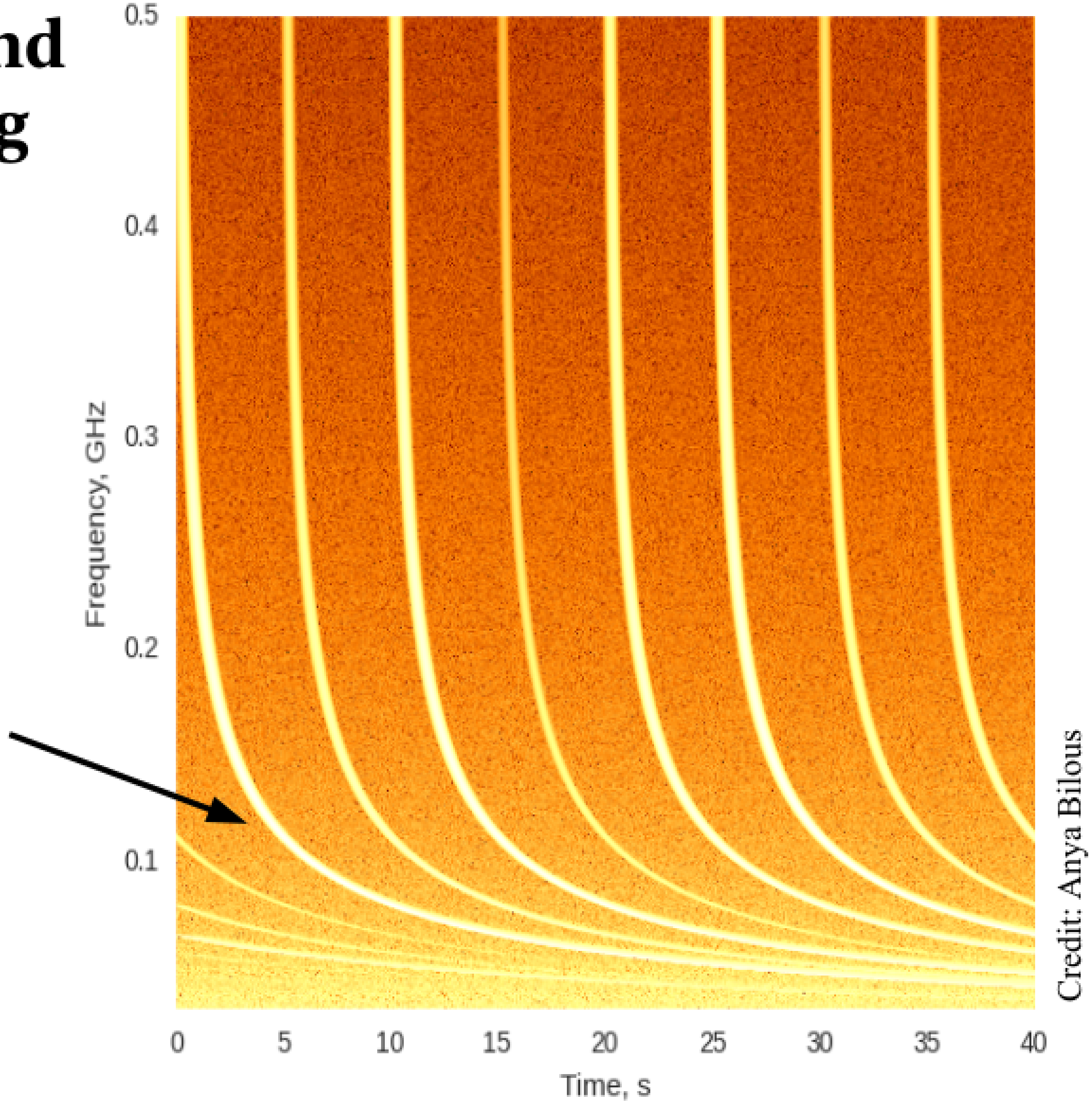
Dispersion

Simulated ultra-broadband pulse recording

DM = 15 pc cm^{-3}
P = 5 s

Dispersive delay

$$\delta t \sim DM/\nu^2$$



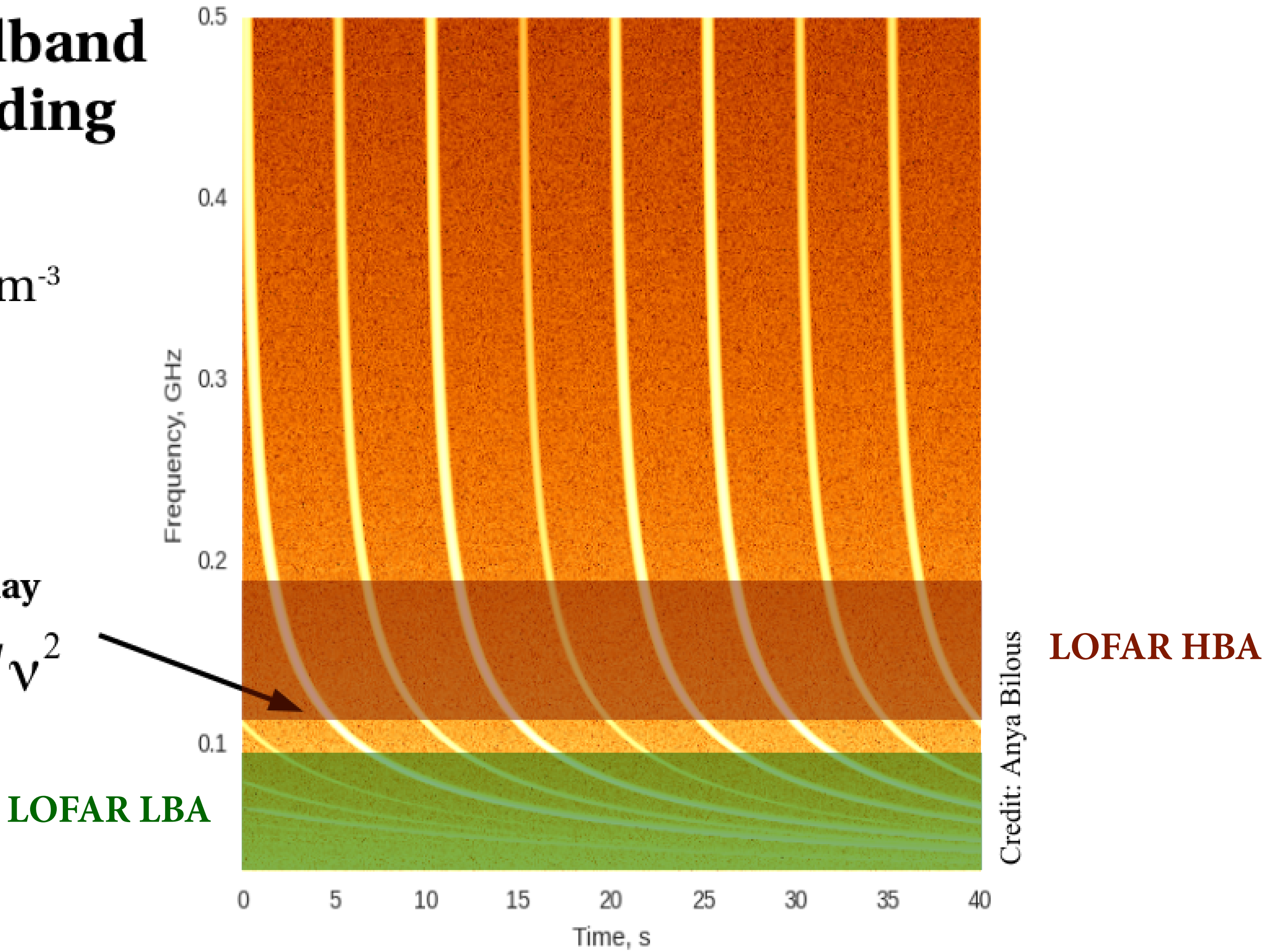
Dispersion

Simulated ultra-broadband pulse recording

DM = 15 pc cm^{-3}
P = 5 s

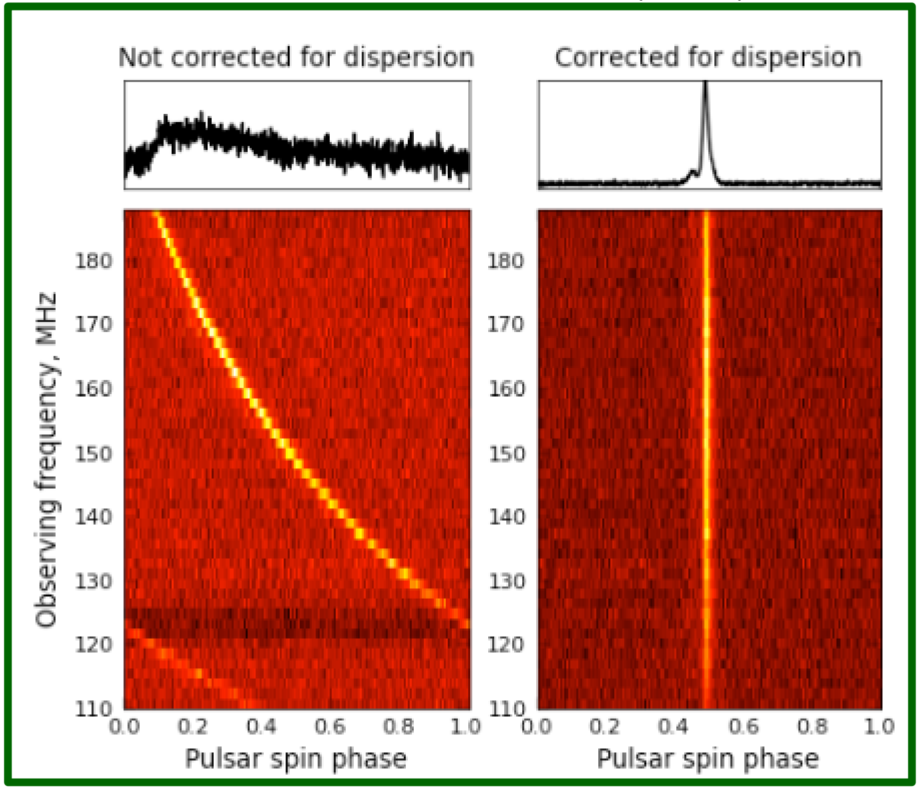
Dispersive delay

$$\delta t \sim DM / \nu^2$$

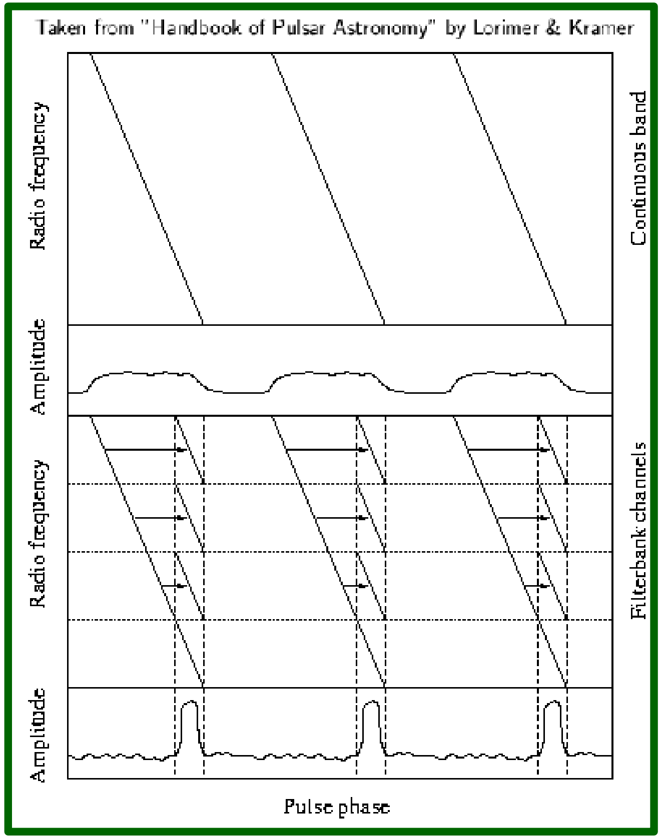


Dispersion

PSR B2021+51 **DM is off by only 3 pc/cc!**



Credit: Anya Bilous

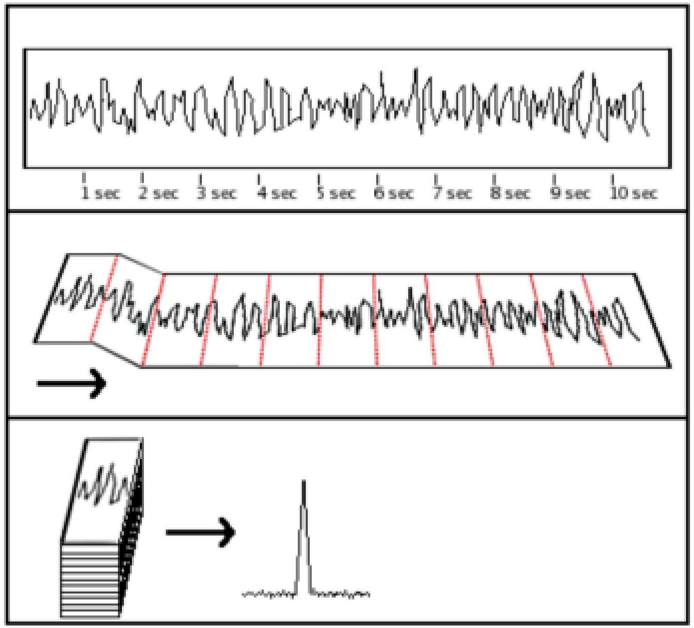


- DM [pc cm^{-3}] measures the integrated column density of free electrons along the line of sight
- Can be corrected using (in)coherent dedispersion

- Incoherent dedispersion – shifting channels in time
- Coherent dedispersion – requires complex-voltage data and is more computationally expensive

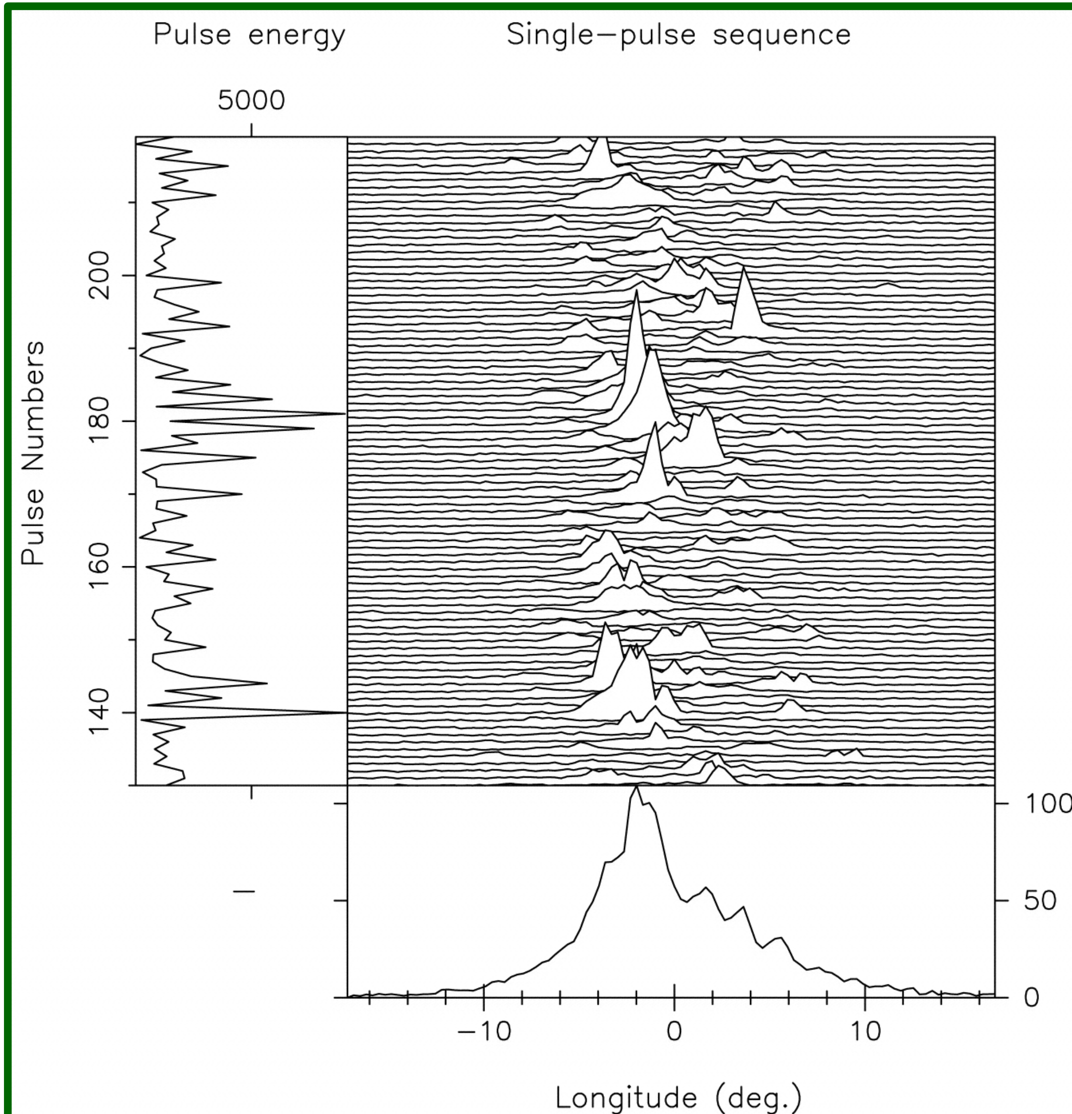
Folding

in a nutshell

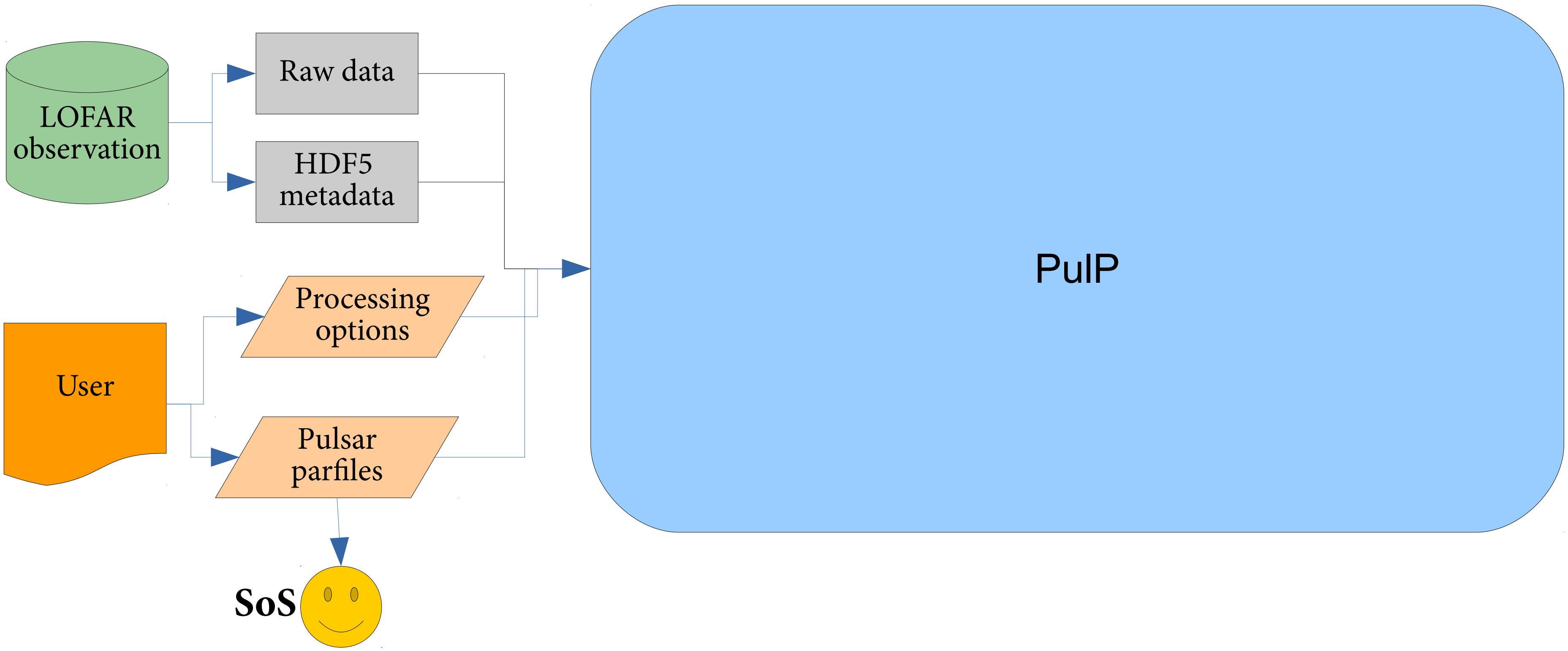


Deshpande & Rankin (1999)

PSR B0943+10



PulP flowchart (1)



Parfiles: if parfiles are not given toPulP, then based on the target name it will try to find the corresponding pulsar in the ATNF catalog. If no pulsar is found in the catalog, PulP will look for the brightest pulsar in a given SAP and fold it.

Pulsar ephemeris (parfiles)

```
PSRJ           J0034-0534
RAJ             00:34:21.8320019          1   0.00068844071120754594
DECJ           -05:34:36.81231            1   0.02161483083208828989
F0             532.71342977772821597      1   0.00000002836577022925
F1             -9.33246370730386516 3e-16 1   4.92605919201583764 76e-17
PEPOCH         49550.037311801294202
POSEPOCH       49550.037311801294202
DMEPOCH        49550
DM             13.764894275846959064      1   0.00004430172861405973
DM1            0
PMRA           8.0823671616462304792          0.13169130726699274092
PMDEC          -9.5157740417196312044         0.30750778304651565920
BINARY         ELL1
PB             1.5892817926966151351      1   0.00000000792094121129
A1             1.4377774324431148653      1   0.00000359435416977026
TASC           49550.704855759820283      1   0.00003326831506895045
EPS1           6.4089927497823510916e-05  1   0.00000873904414835617
EPS2           -3.0385531645805153 1e-05  1   0.00001349161624088706
START          55959.632675467299123      1
FINISH         56448.301487586140865      1
TZRMJD         56190.012138005647902
TZRFRQ         137.15199999999998681
TZRSITE        t
TRES           31.090
EPHVER         5
CLK            TT(TAI)
MODE 1
EPHEM          DE421
NITS           1
NTOA           218
CHI2R          132.9672 207
```

can be as simple as this:

```
PSR       J1706+35
RAJ       17:07:03.61
DECJ      +35:55:54.5
P0        0.159764851
P1        0.0
PEPOCH    58244.04308936660
DM        19.240
EPHEM     DE405
CLK       UNCORR
```

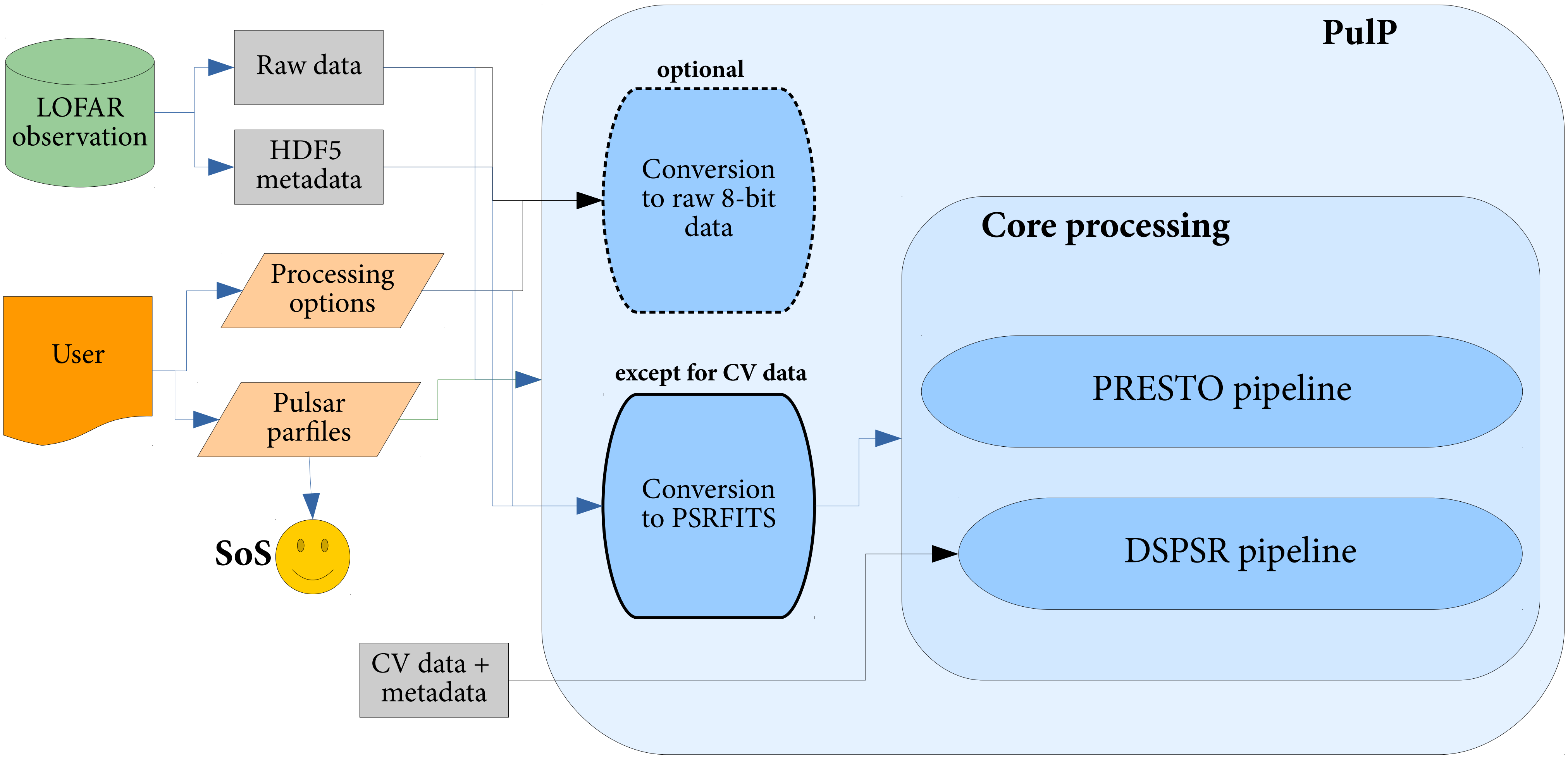
Input raw data

- HDF5 format
- Header information (metadata) is stored in **_bf.h5* file
- The raw data itself is stored in **_bf.raw* file
- This *.raw* file is linked from within *.h5* file and can be accessed directly via opened *.h5* file

- Filename structure:
  → Lnnnnnn_SAPxxx_Byyy_Sz_Pmmm_bf.h5

    - Lnnnnnn – LOFAR observation ID (ObsID)
    - xxx – Sub-array pointing (SAP) number
    - yyy – Tied-array beam (TAB) number
    - z – Stokes parameter, can only take values 0,1,2,3
        - Stokes I observation – have only S0 files
        - Stokes IQUV observation: S0 – I, S1 – Q, S2 – U, S3 – V
        - Complex-voltage data: S0 – Xreal, S1 – Ximag, S2 – Yreal, S3 – Yimag
    - mmm – Frequency part, i.e. when every file has only fraction of subbands

PulP flowchart (2)

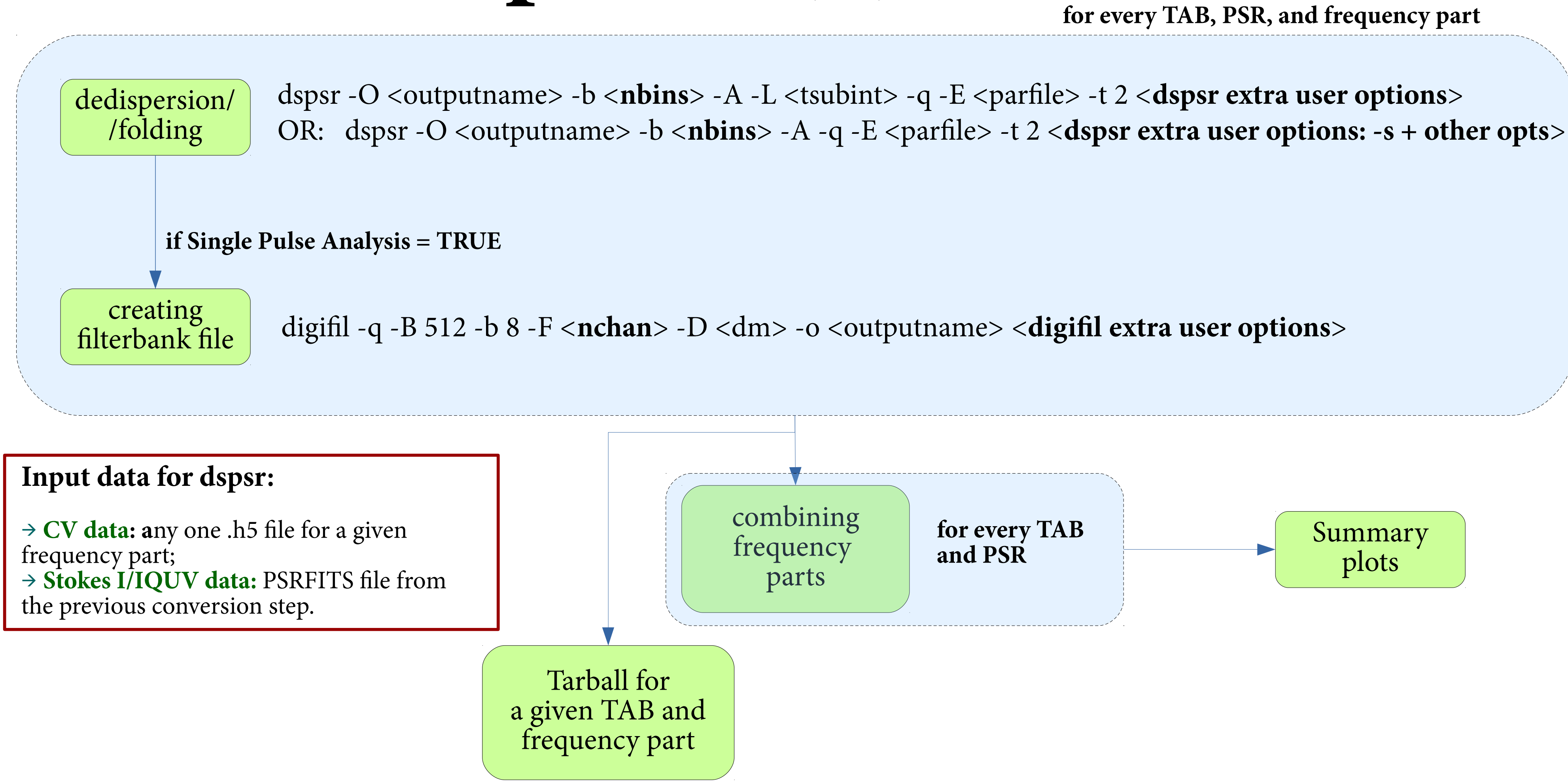


Data conversion

- Conversion to raw 8-bit data (optional)
  - → *digitize.py*
  - → written by Marten van Kerkwijk
  - → available at: *https://github.com/mhvk/scintellometry/blob/master/scintellometry/lofar/digitize.py*
  - → *digitize.py -s 5 -o <output dir> <input .h5>*

- Conversion from raw 32-bit data to PSRFITS (for non-CV data)
  - → custom-made program *2bf2fits*
  - → written by Tom Hassall, Patrick Weltevrede, with contribution from Vlad Kondratiev
  - → currently available at LOFAR Users Software Repository
  - → will make it available at Github as well
  - → does not save scales/offsets in PSRFITS
  - → needs major revisiting…
  - → Command example (very detailed input):
    - *2bf2fits -CS -H -append -nbits 8 -A 100 -sigma 3 -nsubs 400 -sap 0 -tab 0 -stokes 0 -o L667444_SAP0_BEAM0 -nsamples 24 -nchans 16 -ra 2.15980858832 -dec 1.30000703891 -psr B0809+74 -clock 200 -band HBA_110_190 -startdate 2018-09-12 -starttime 20:17:00.000000000 -samptime 0.0104858 -duration 299.977 -subs 54..453 -obsid L667444 -observer Pizzo /data/projects/PipelineTests/L667444/cs/L667444_SAP000_B000_S0_P000_bf.raw*

DSPSR Pipeline (1)

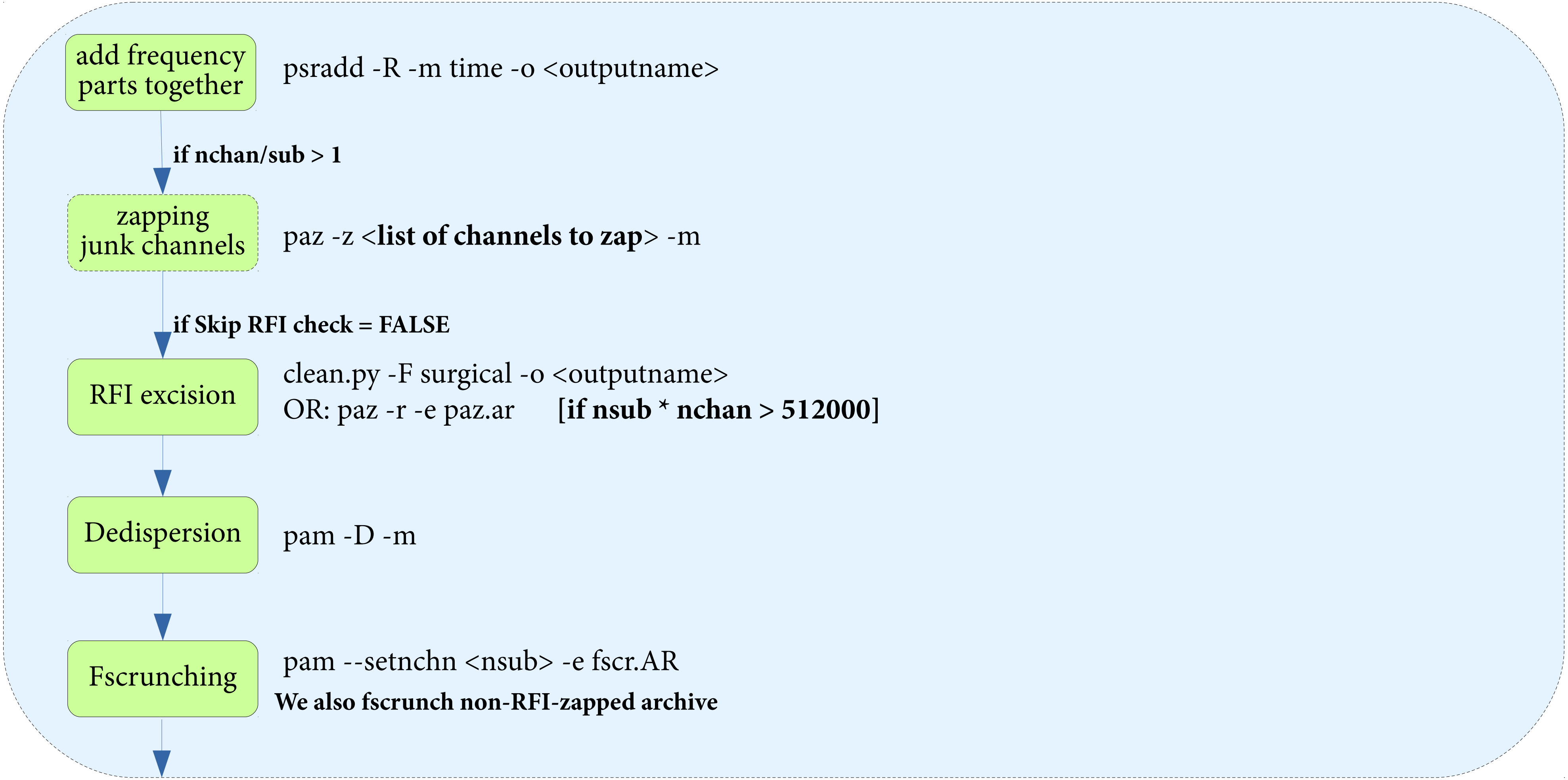


<**nbins**> - calculated automatically based on the sampling time and F0/P0 from the parfile.
Maximum possible <nbins>=1024

<**nchan**> - number of channels in a given frequency part. If number of channels = 1, then <nchan>=2

DSPSR Pipeline (2)

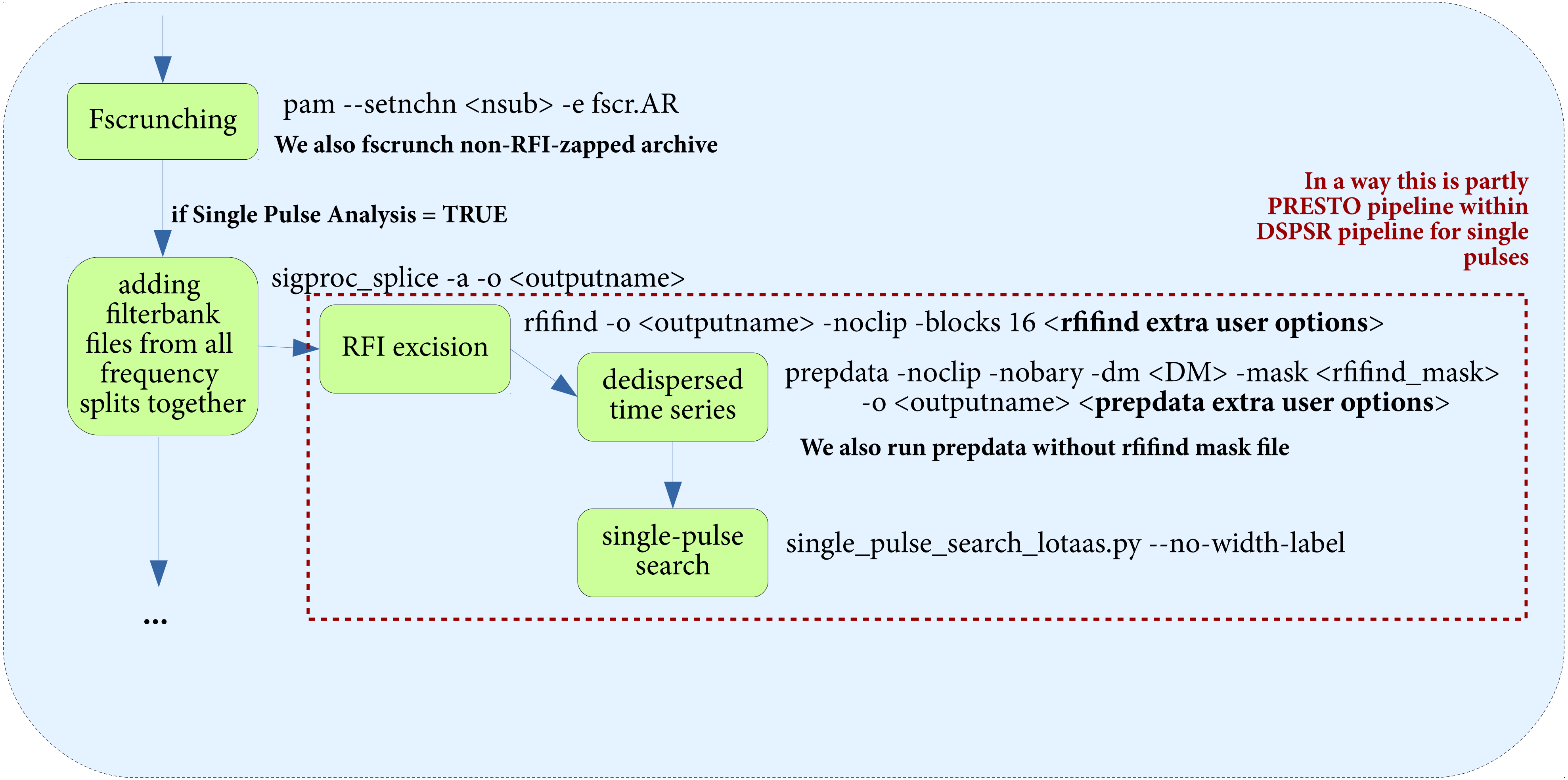
**combining frequency parts;
for every TAB, PSR**



<**list of channels to zap**> - if there are 16 chan/sub, we need to zap every 16th channel, then list becomes «0 15 31 47...»
This is necessary, as when 2nd PPF is used, the first channel in each subband gets corrupted

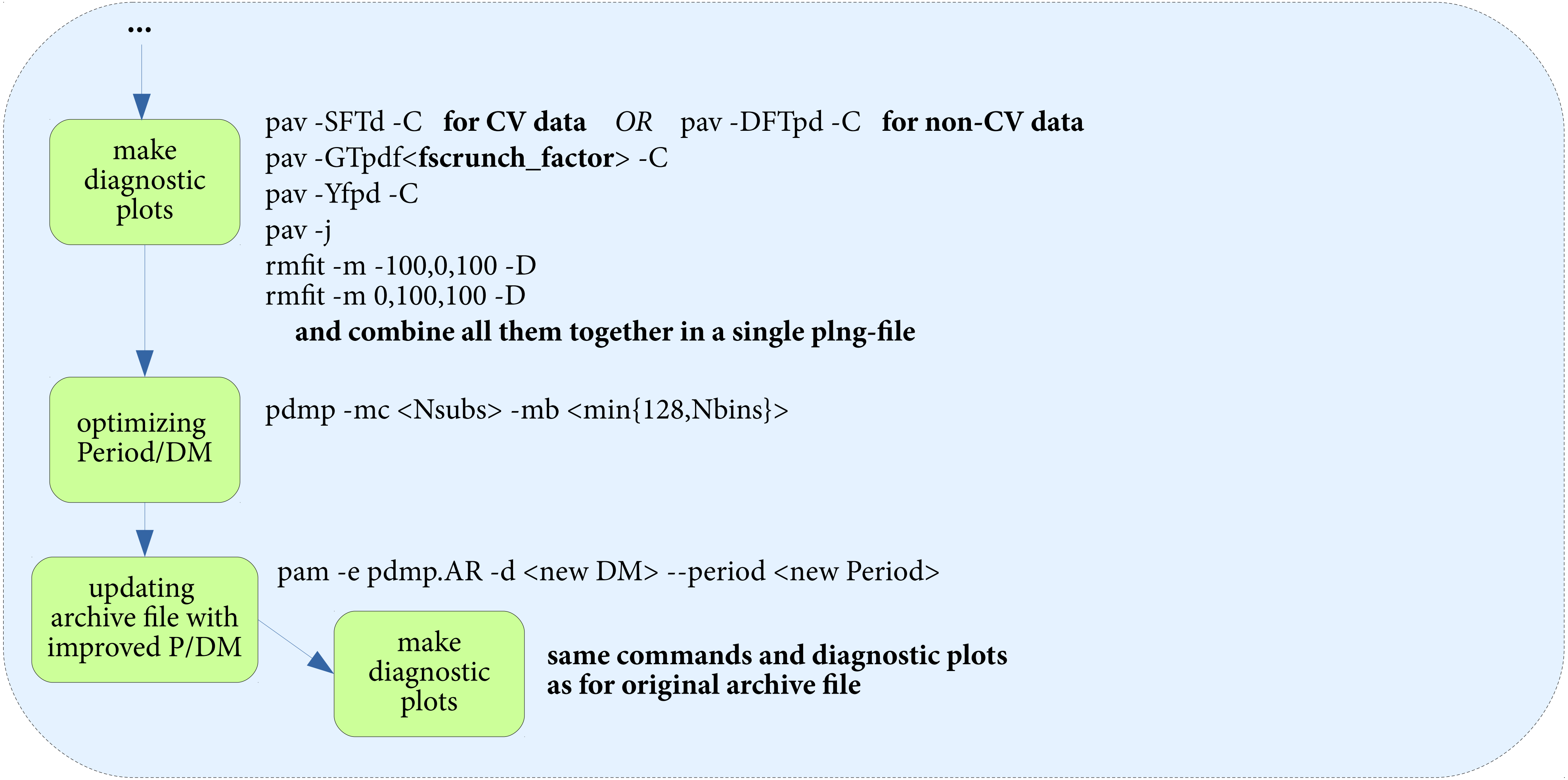
DSPSR Pipeline (2, cont.)

combining frequency parts; for every TAB, PSR



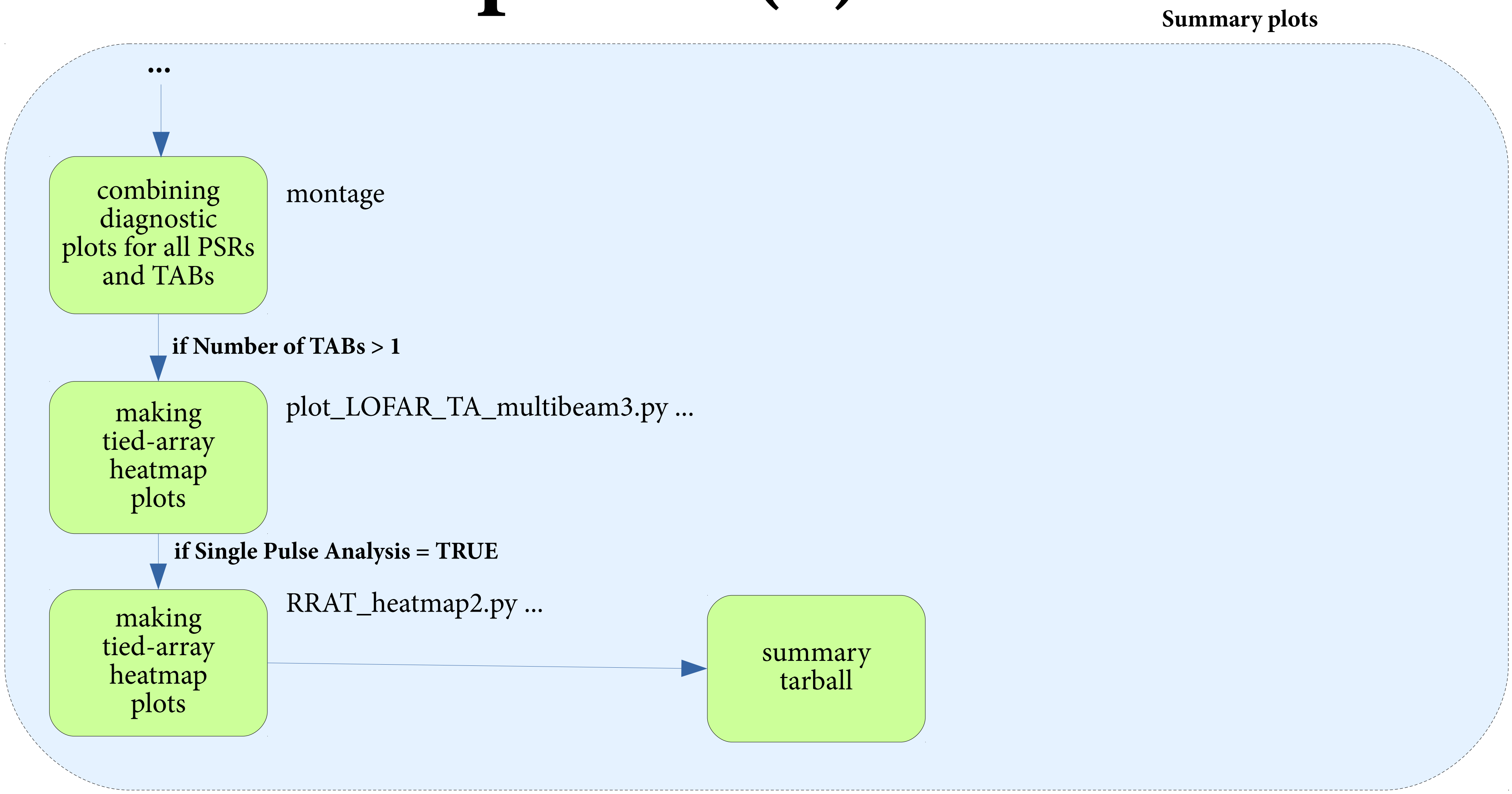
DSPSR Pipeline (3, cont.)

combining frequency parts;
for every TAB, PSR



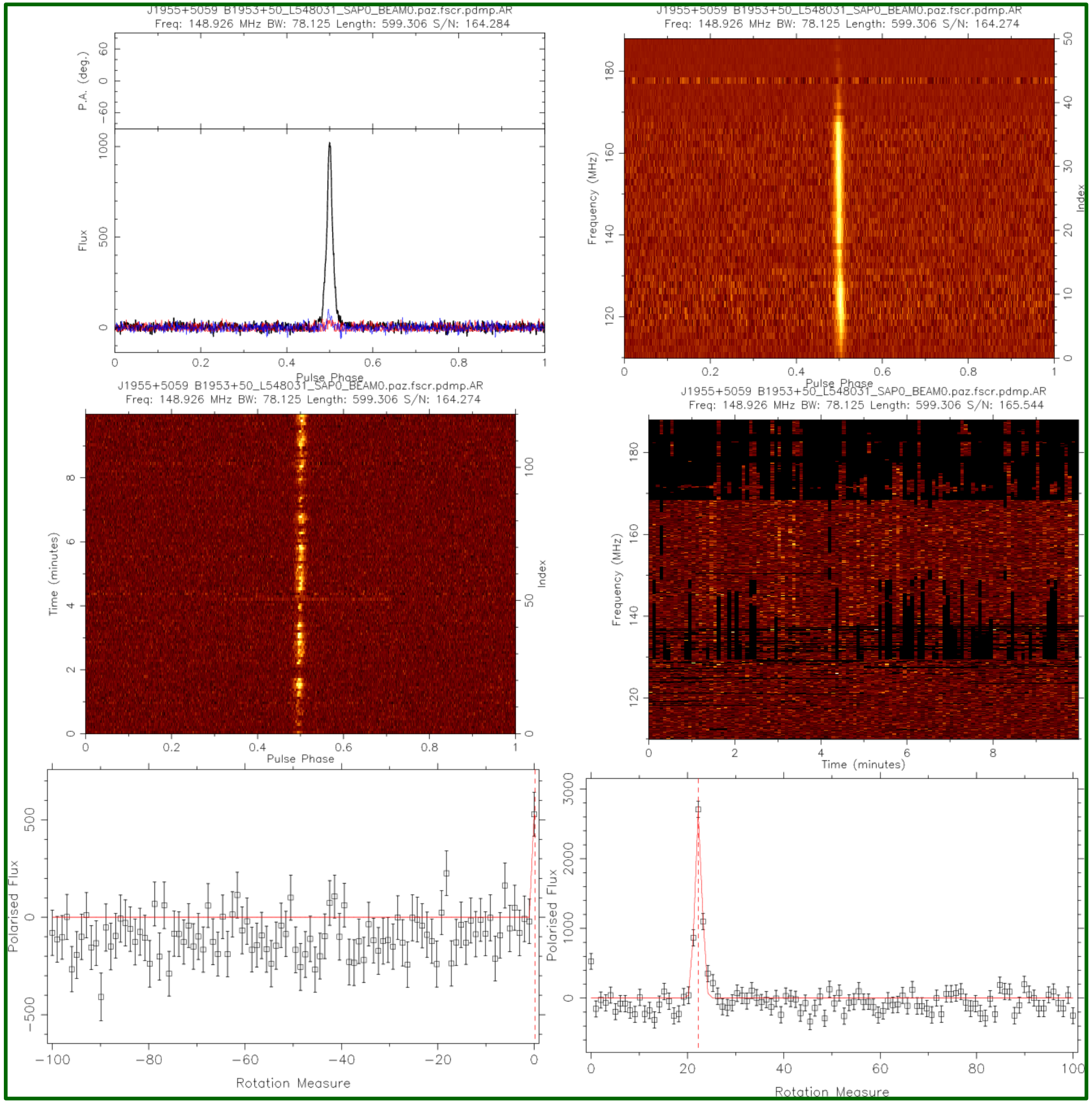
<**fscrunch_factor**> = Nsubs / X, where X = highest common denominator of Nsubs between 1 and min{Nsubs, 63}

DSPSR Pipeline (4)



Diagnostic plots (1)

pav
rmfit

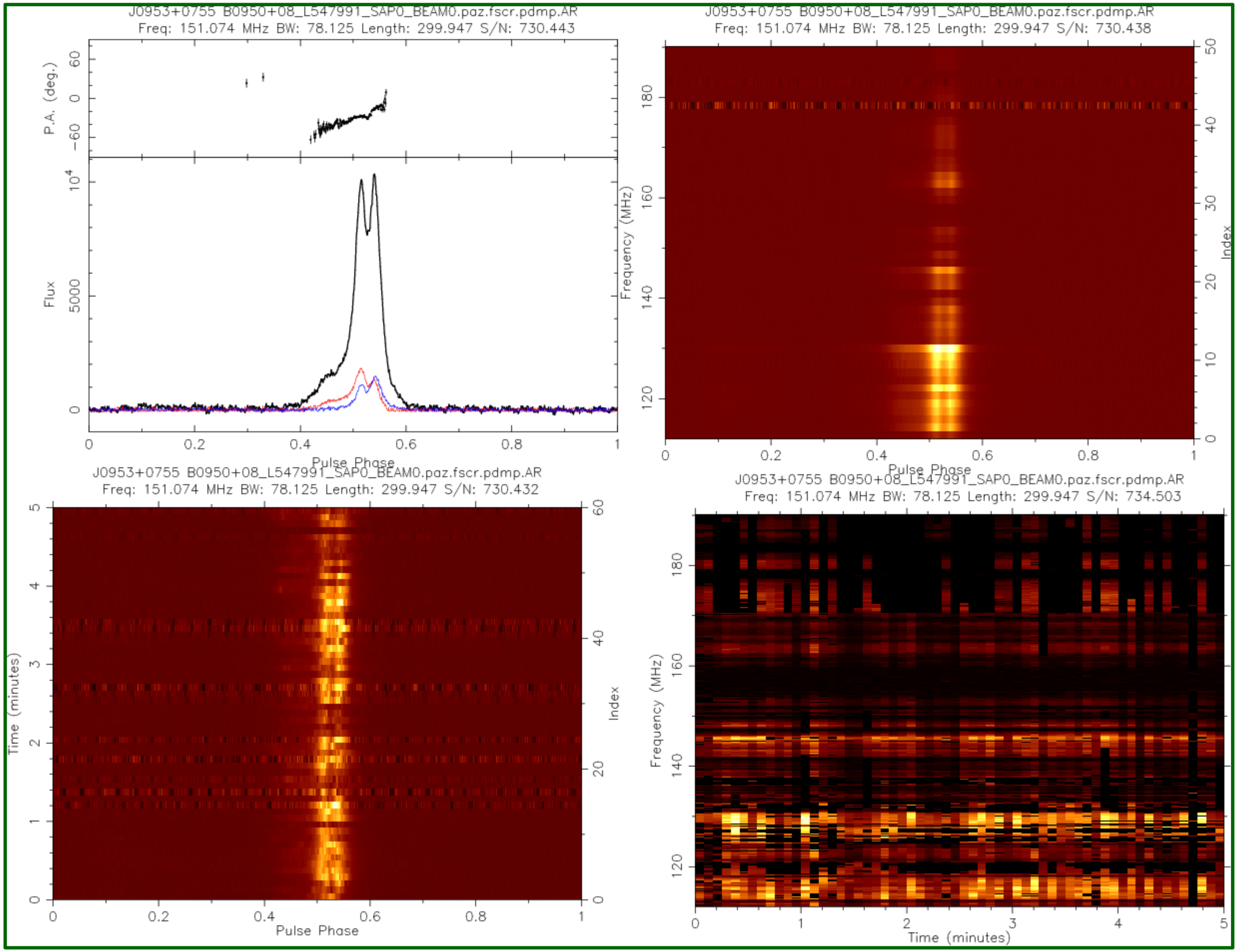


status.png

*_diag.png

*_diag_pdmp.png

Diagnostic plots (2)

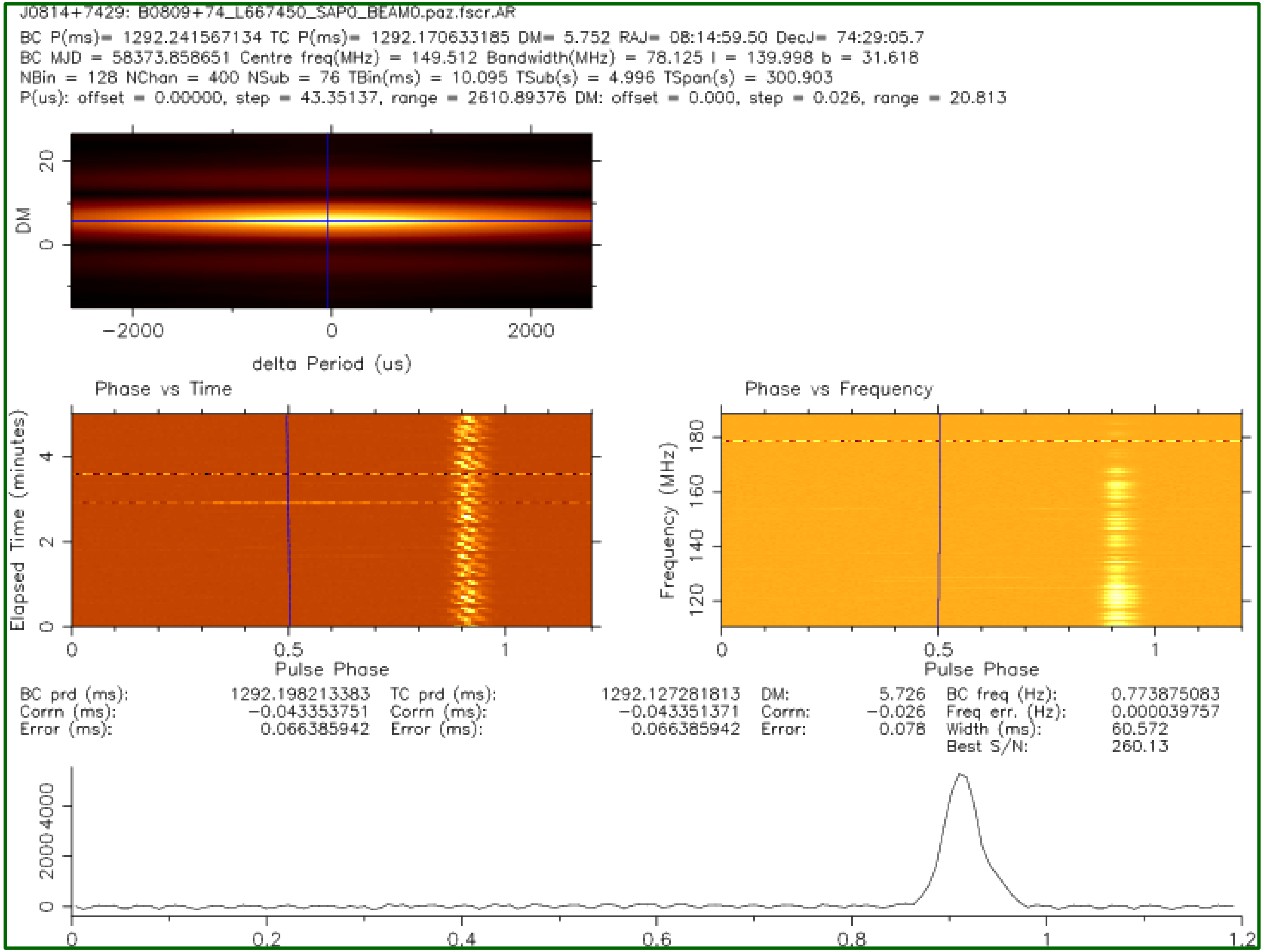


status.png

*_diag.png

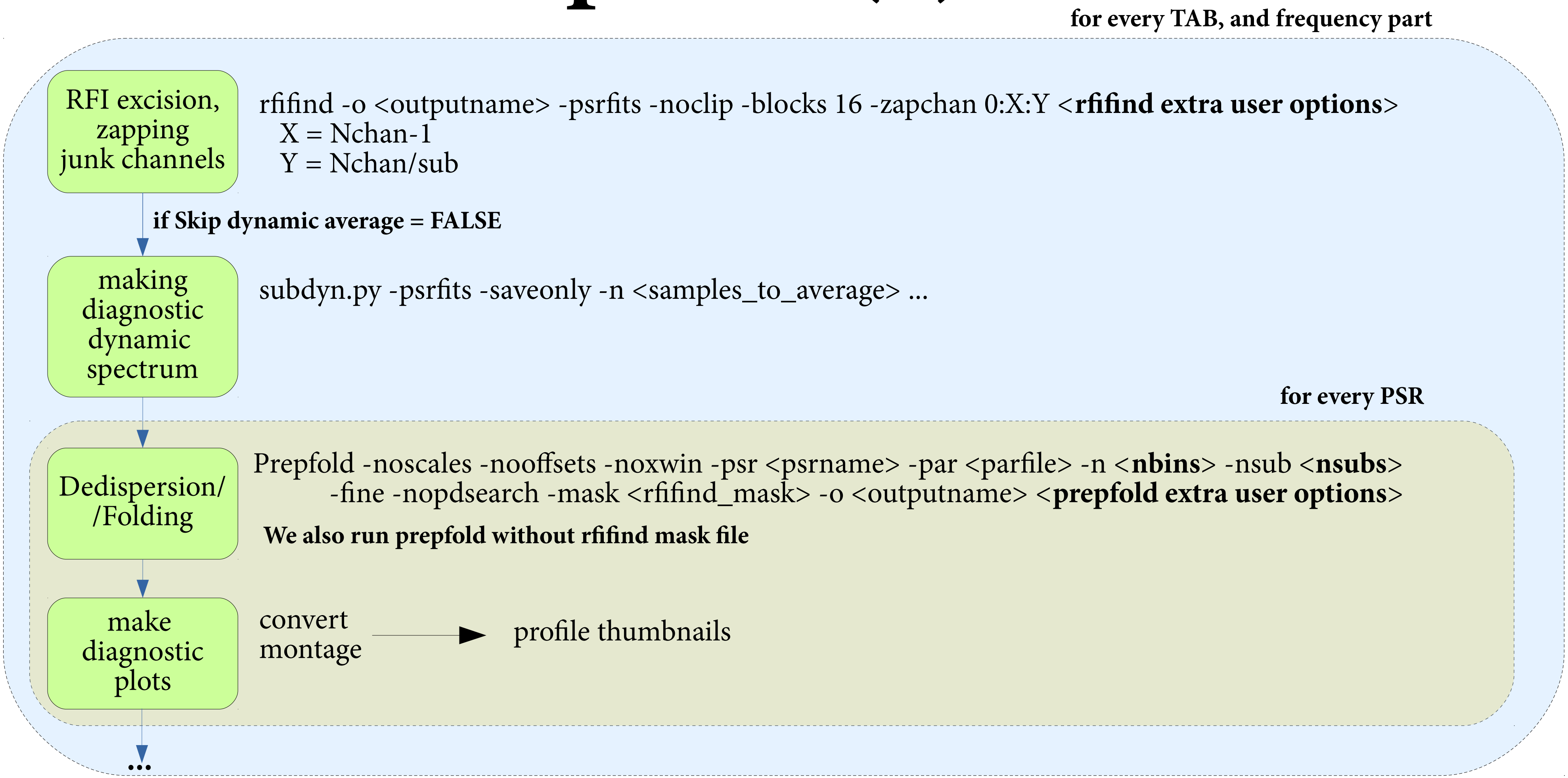
*_diag_pdmp.png

Diagnostic plots (3)



*_pdmp.ps

PRESTO Pipeline (1)



<**nbins**> - calculated automatically based on the sampling time and F0/P0 from the parfile.
Maximum possible <nbins>=1024

<**nsubs**> - if Nchan > 512, nsubs = 512. Otherwise, nsubs = Nchan

PRESTO Pipeline (2, cont.)

for every TAB, and frequency part

...

if Single Pulse Analysis = TRUE

for every PSR

dedispersed time series

prepdata -noscales -nooffsets -noclip -nobary -dm <DM> -mask <rfifind_mask>
-o <outputname> <**prepdata extra user options**>

We also run prepdata without rfifind mask file

single-pulse search

single_pulse_search_lotaas.py --no-width-label

<**nsubs**> – greatest common denominator of Nchan between 1 and 1024
<**lodm**> – DM - 0.5*dmstep*numdms. If lodm<=0, then lodm=0.01

if RRATs analysis = TRUE

dedispersed time series for a range of DMs

prepdata -noscales -nooffsets -noclip -nobary -dm 0.0 -mask <rfifind_mask> -o <outputname>
<**prepdata extra user options**>

We also run prepdata for DM=0.0 without rfifind mask file

prepsubband -noscales -nooffsets -noclip -nobary -nsub <**nsubs**> -lodm <**lodm**> -dmstep 0.01
-numdms 1000 -mask <rfifind_mask> -o <outputname> <**prepsubband extra user options**>

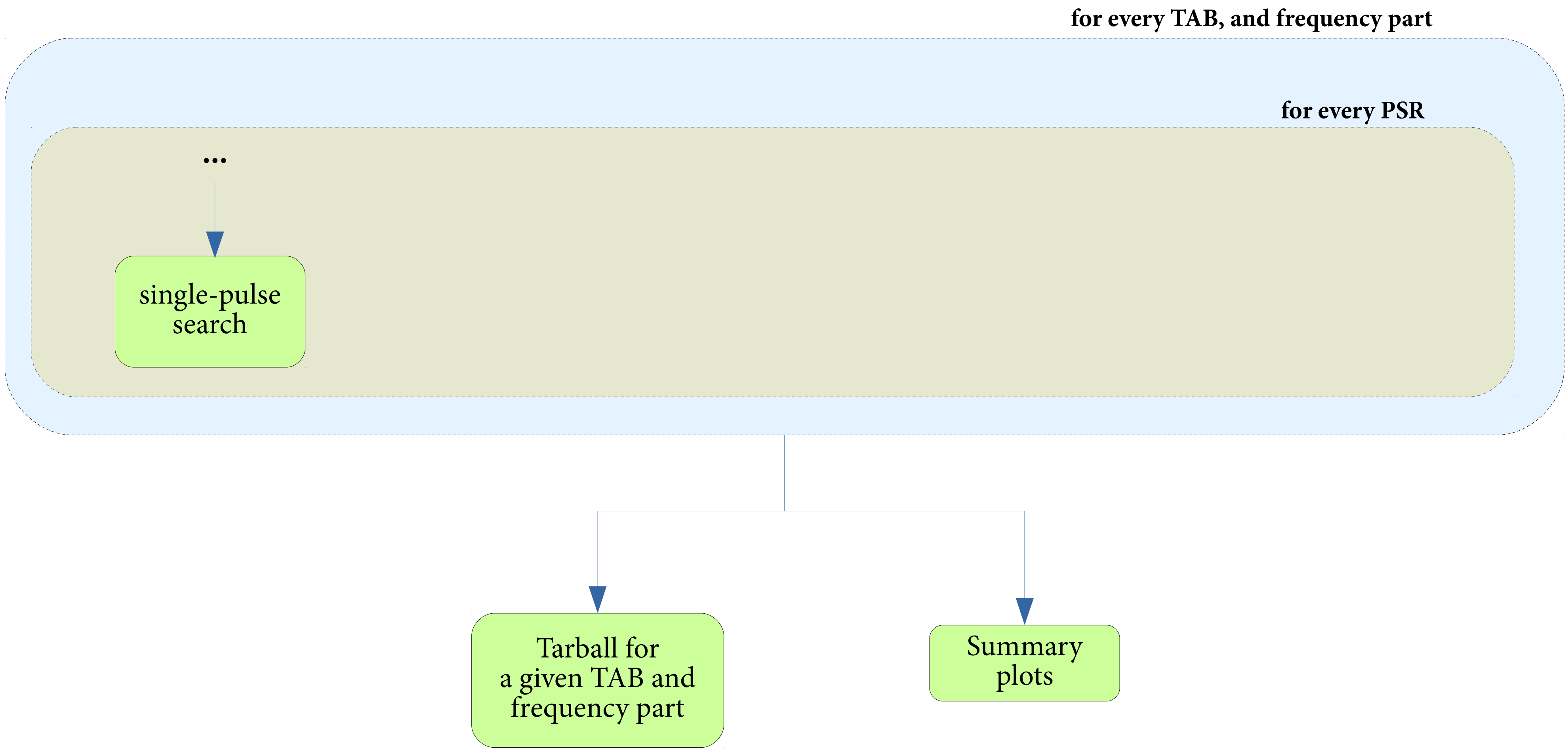
We also run prepsubband without rfifind mask file

single-pulse search

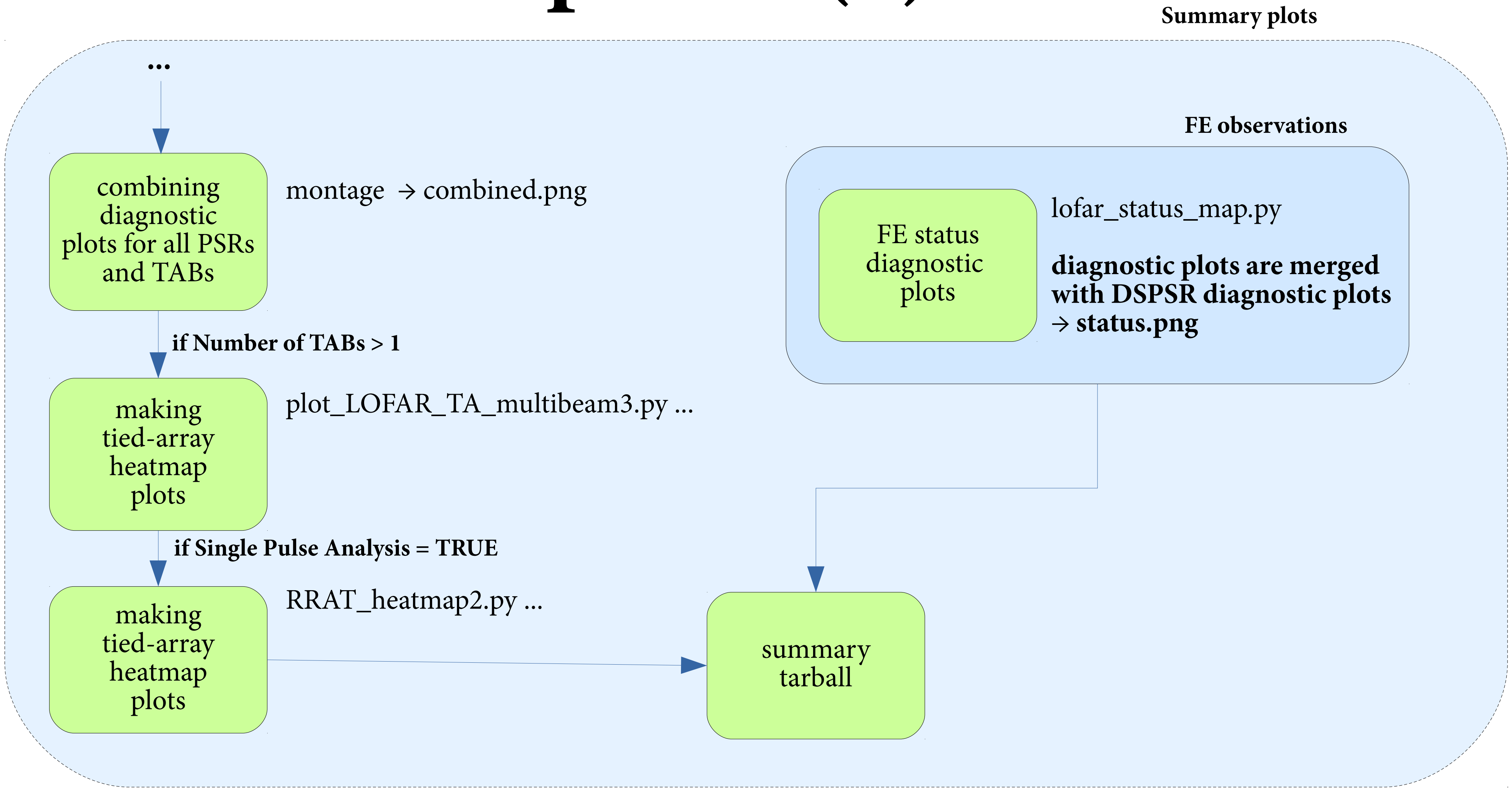
single_pulse_search_lotaas.py -p -g *.dat
single_pulse_search_lotaas.py -t 5.5 —no-width-label -g *.singlepulse

Also running similar command but excluding DM=0

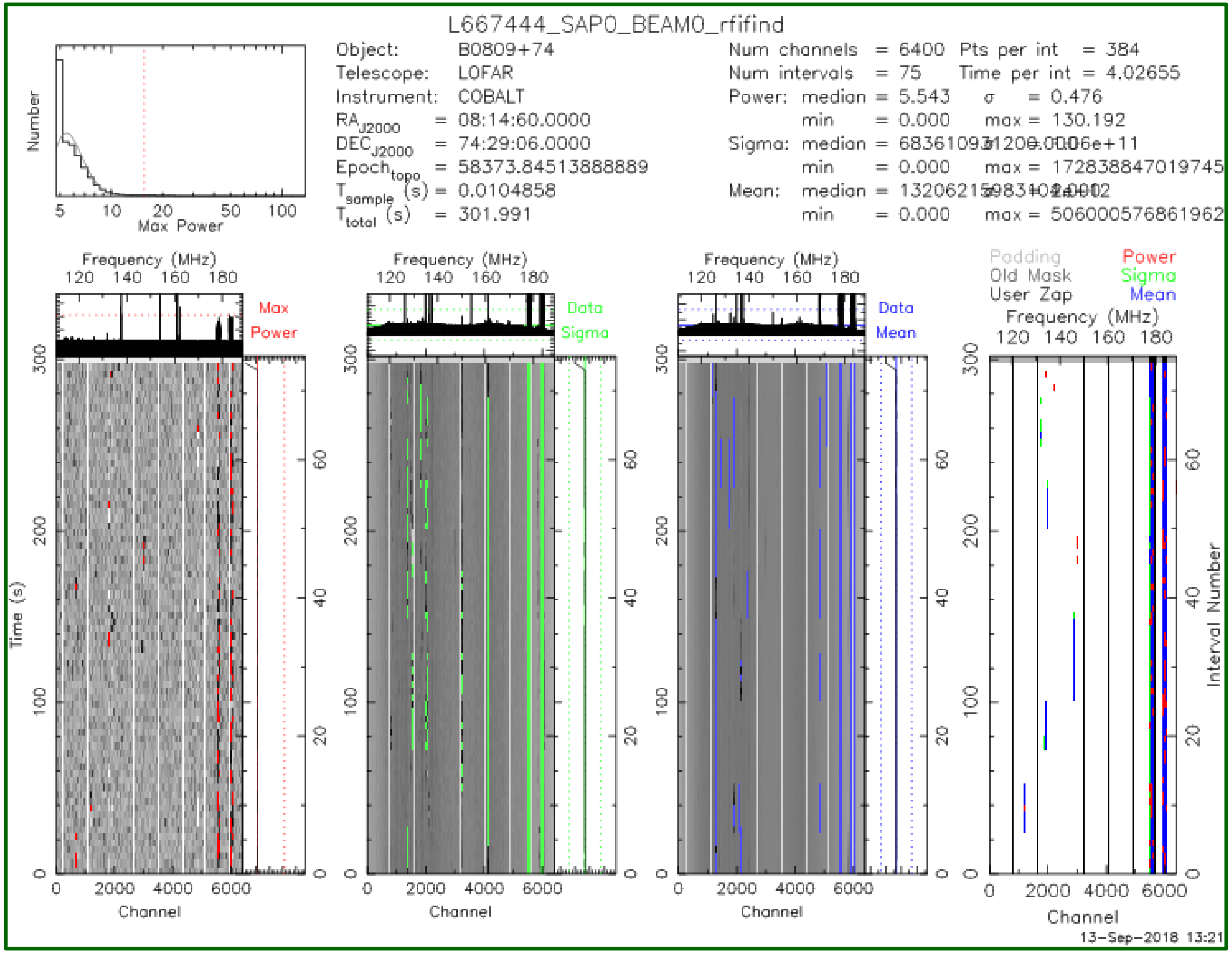
PRESTO Pipeline (3)



PRESTO Pipeline (4)

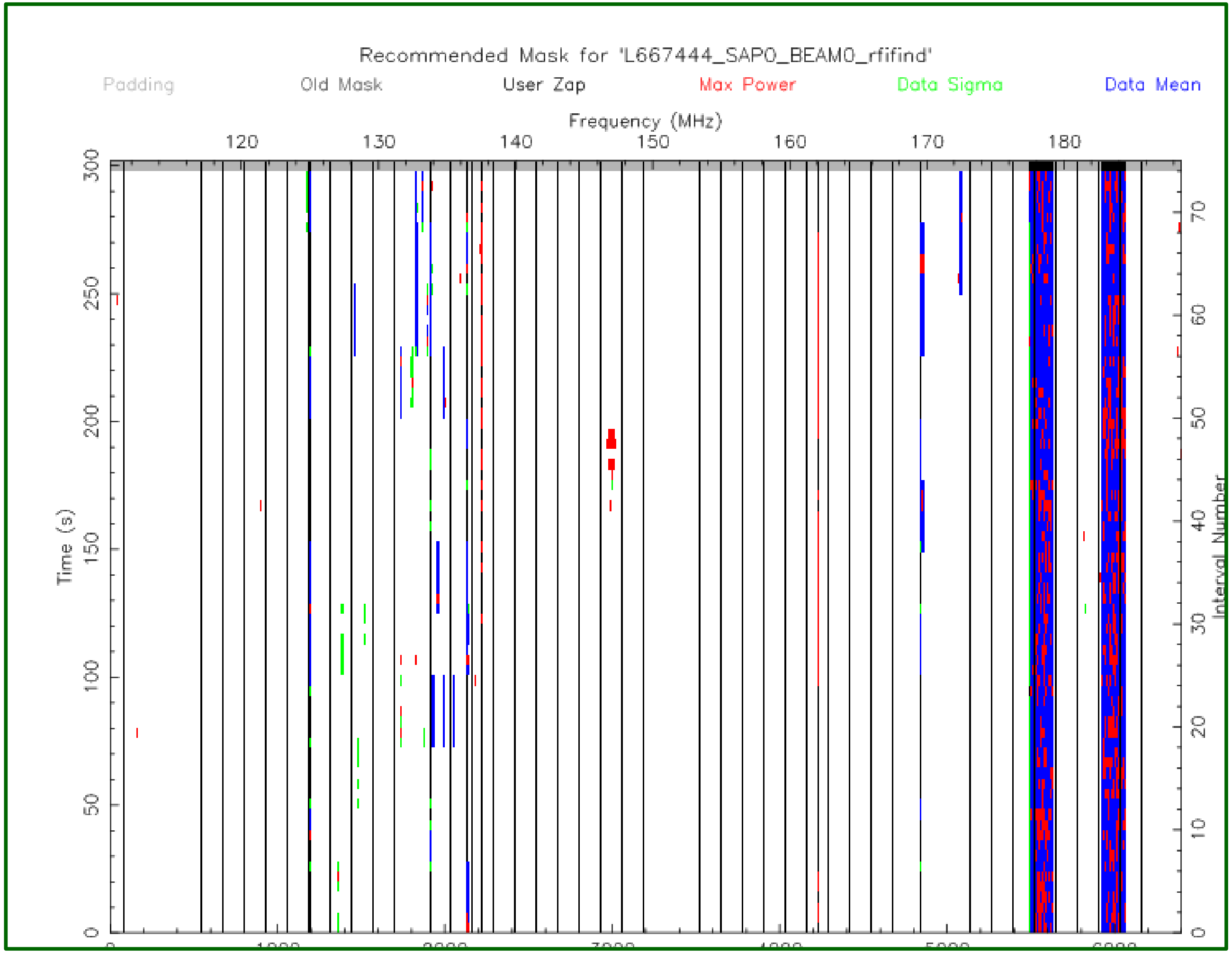


Diagnostic plots (1)



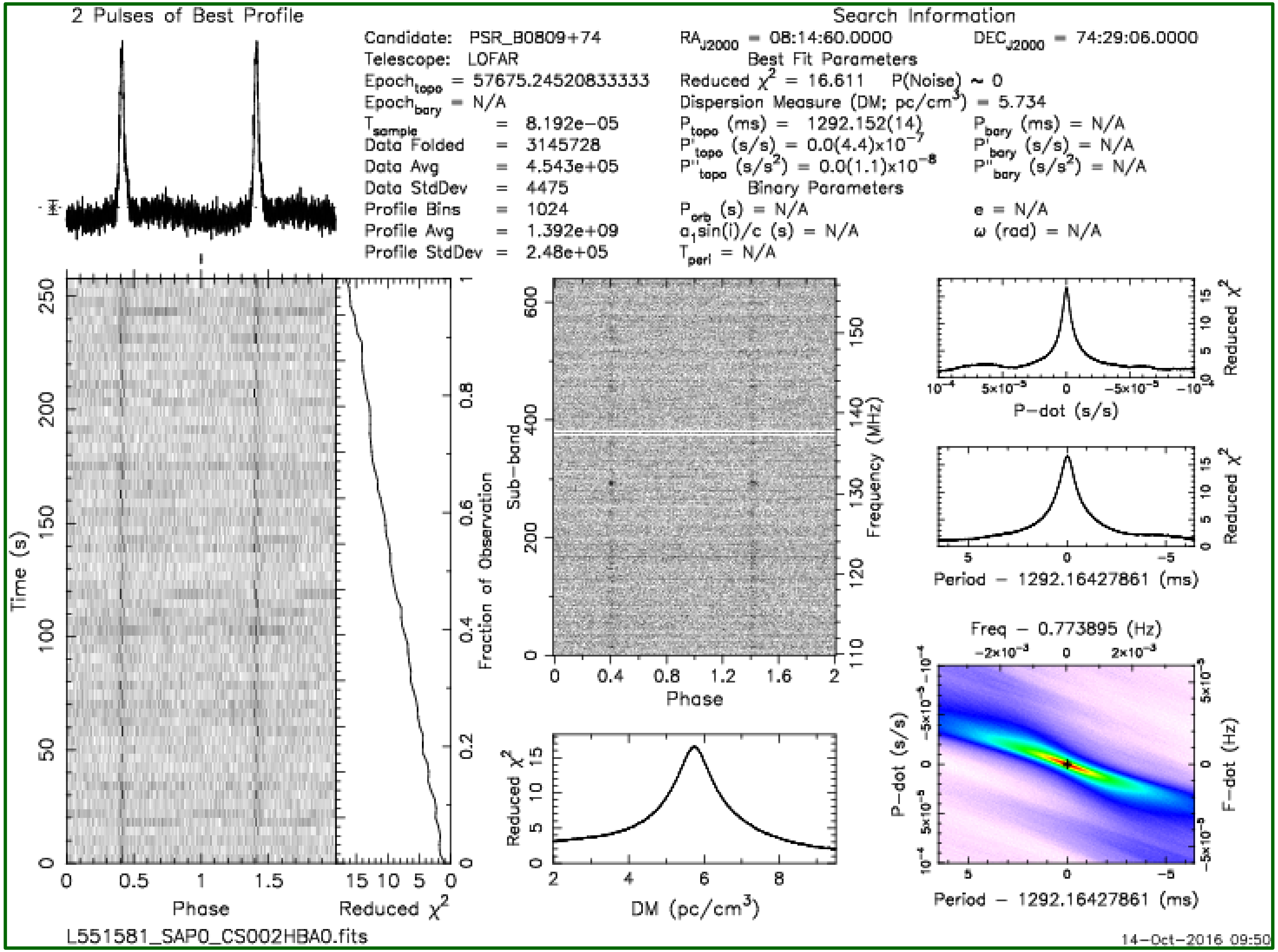
*_rfifind.ps

Diagnostic plots (1, cont.)



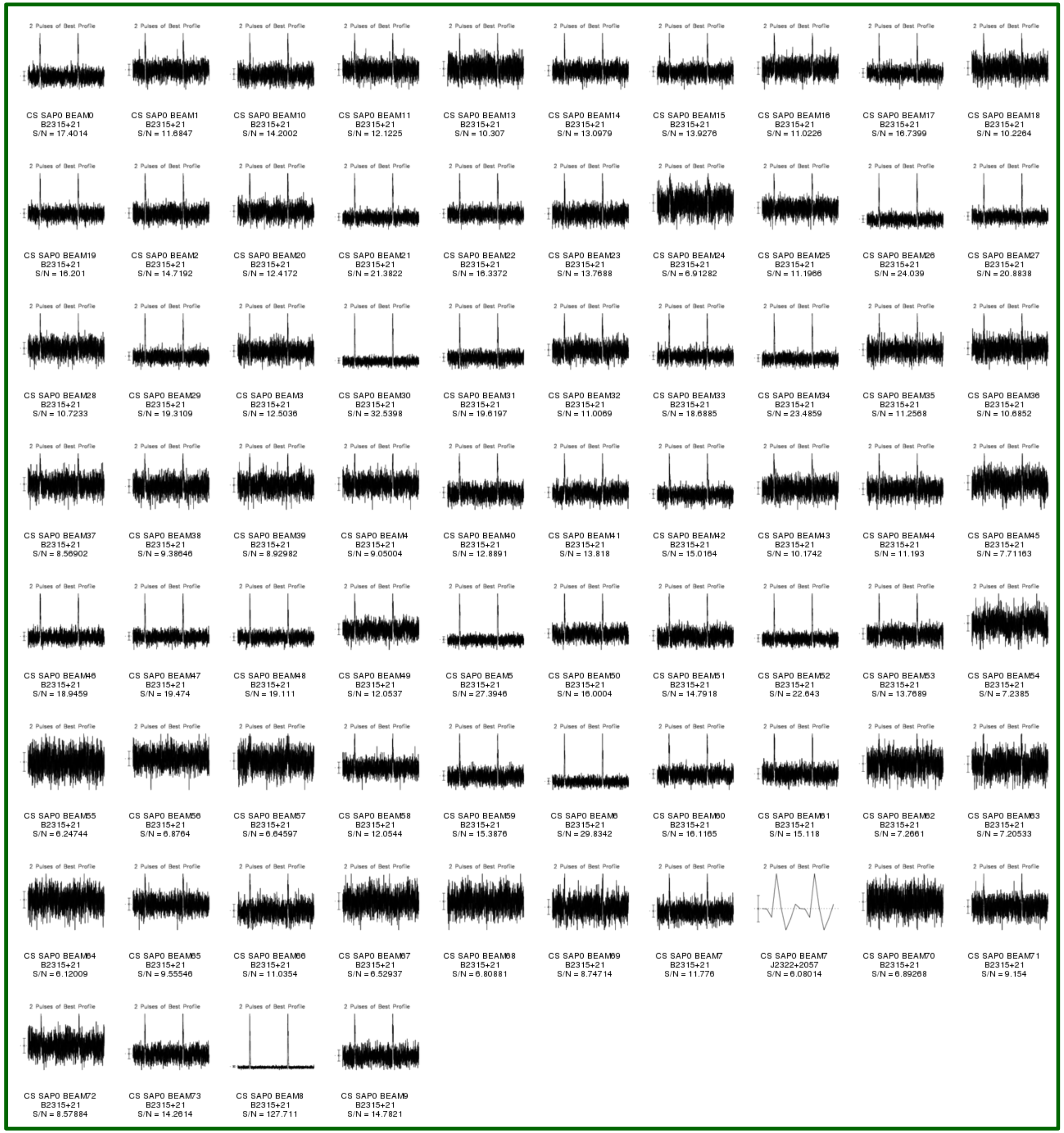
*_rfifind.ps

Diagnostic plots (2)



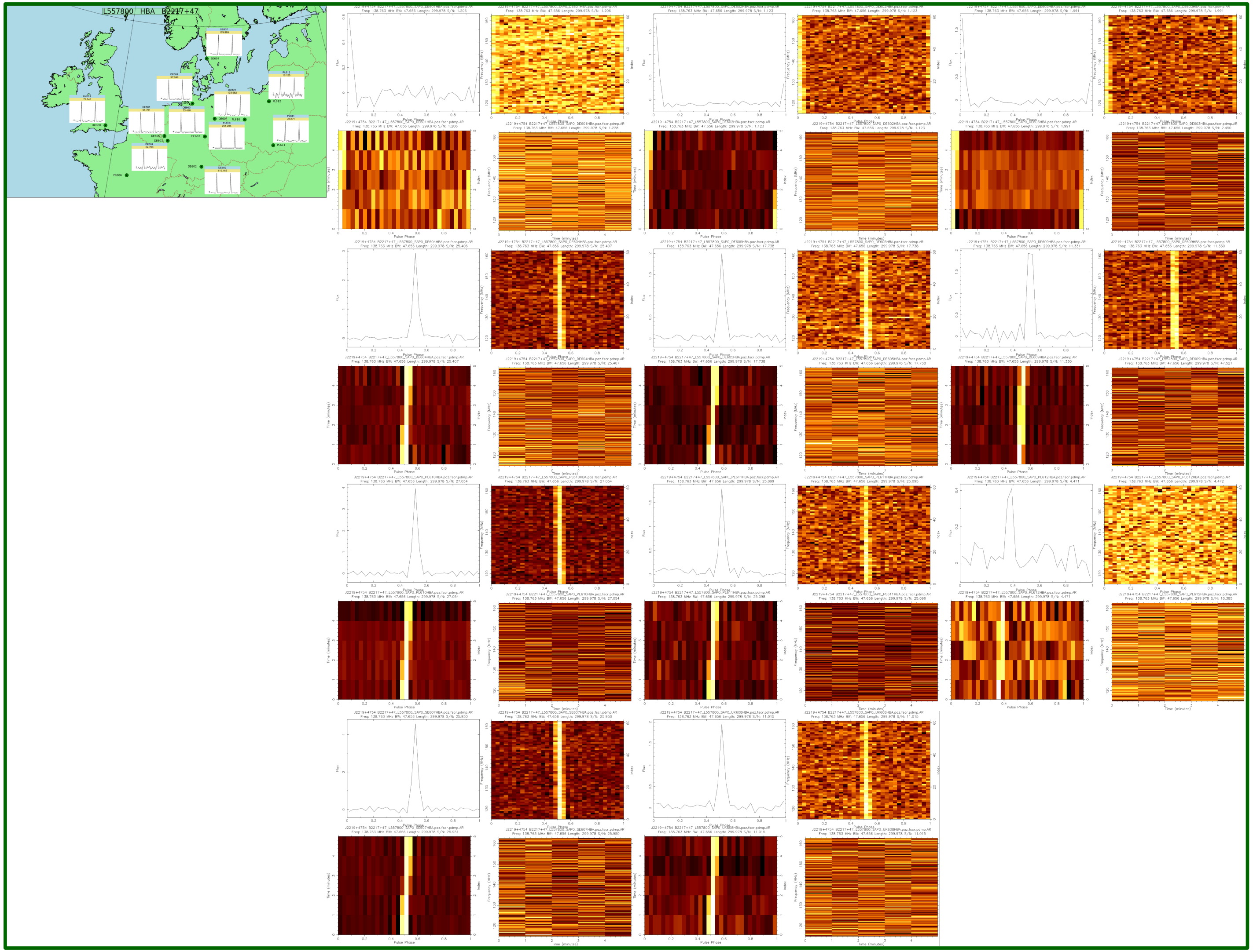
*_pfd.png

Diagnostic plots (3)



combined.png

Diagnostic plots (4)

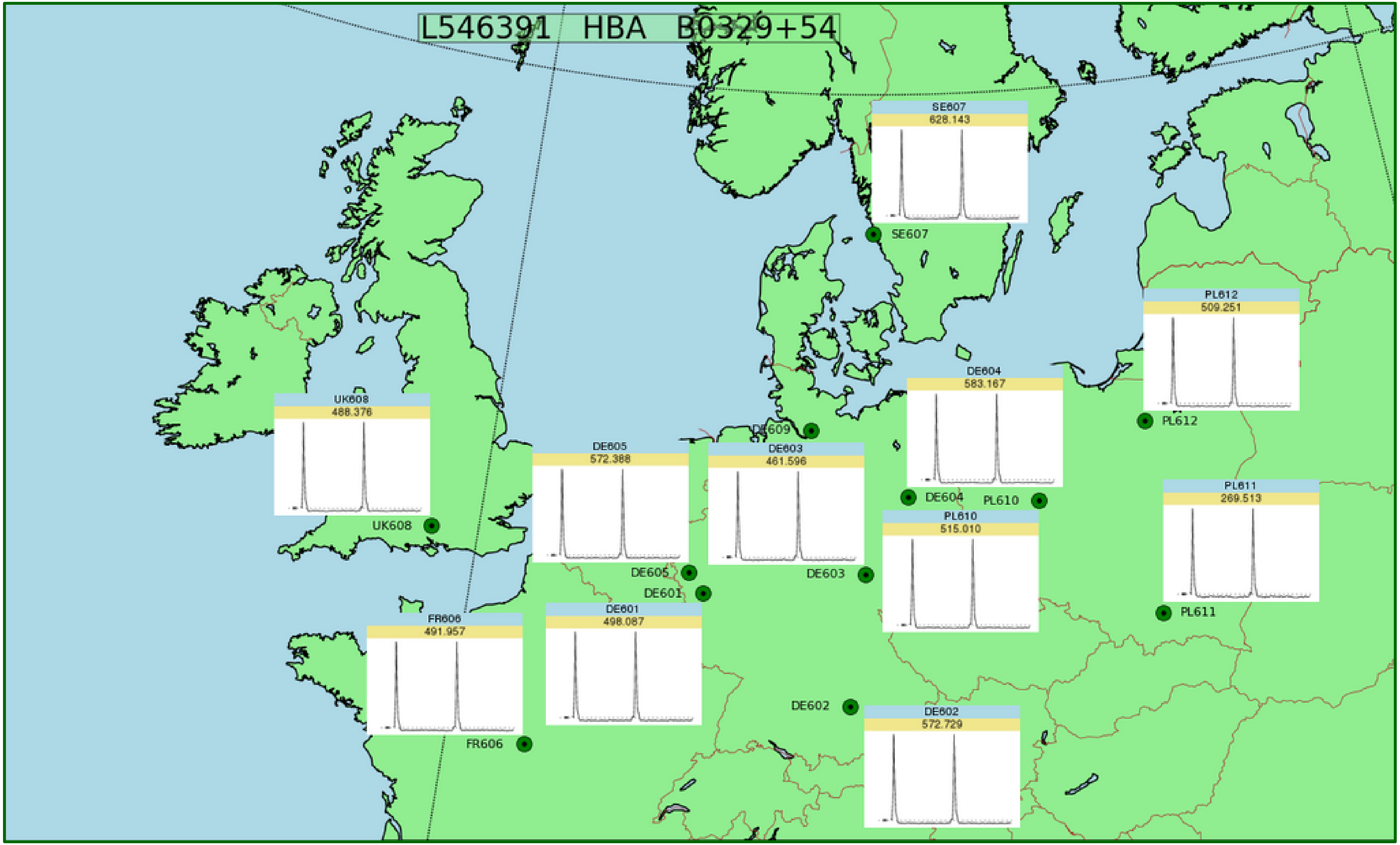


status.png

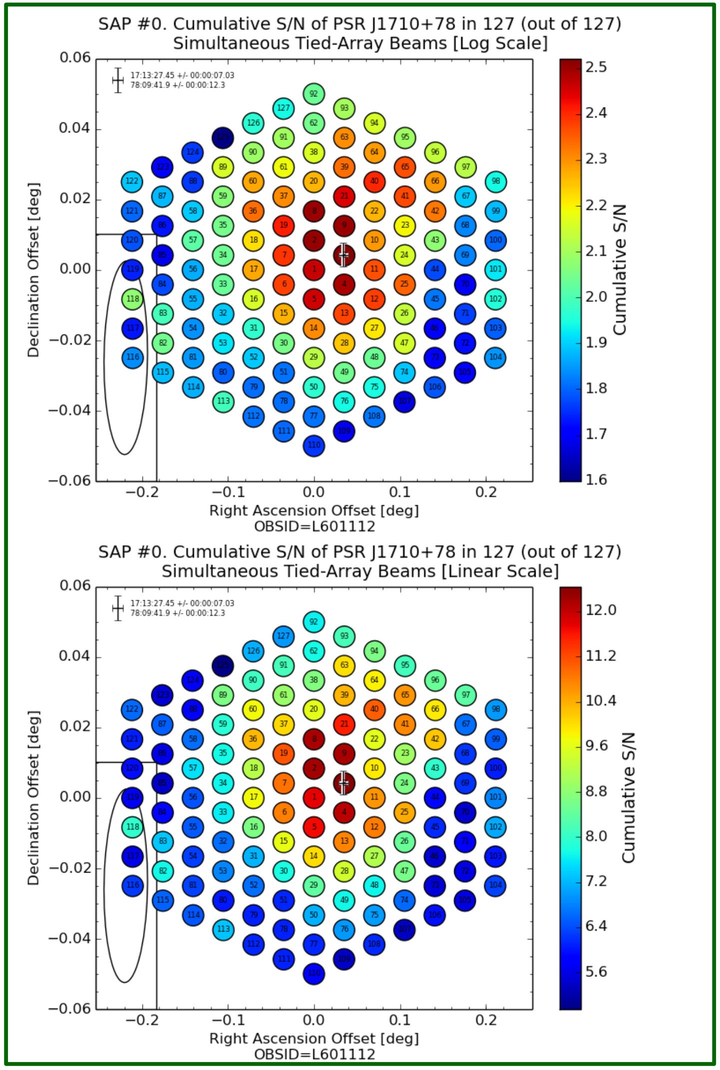
for FE observations

Diagnostic plots (4, cont.)

status.png



Diagnostic plots (5)

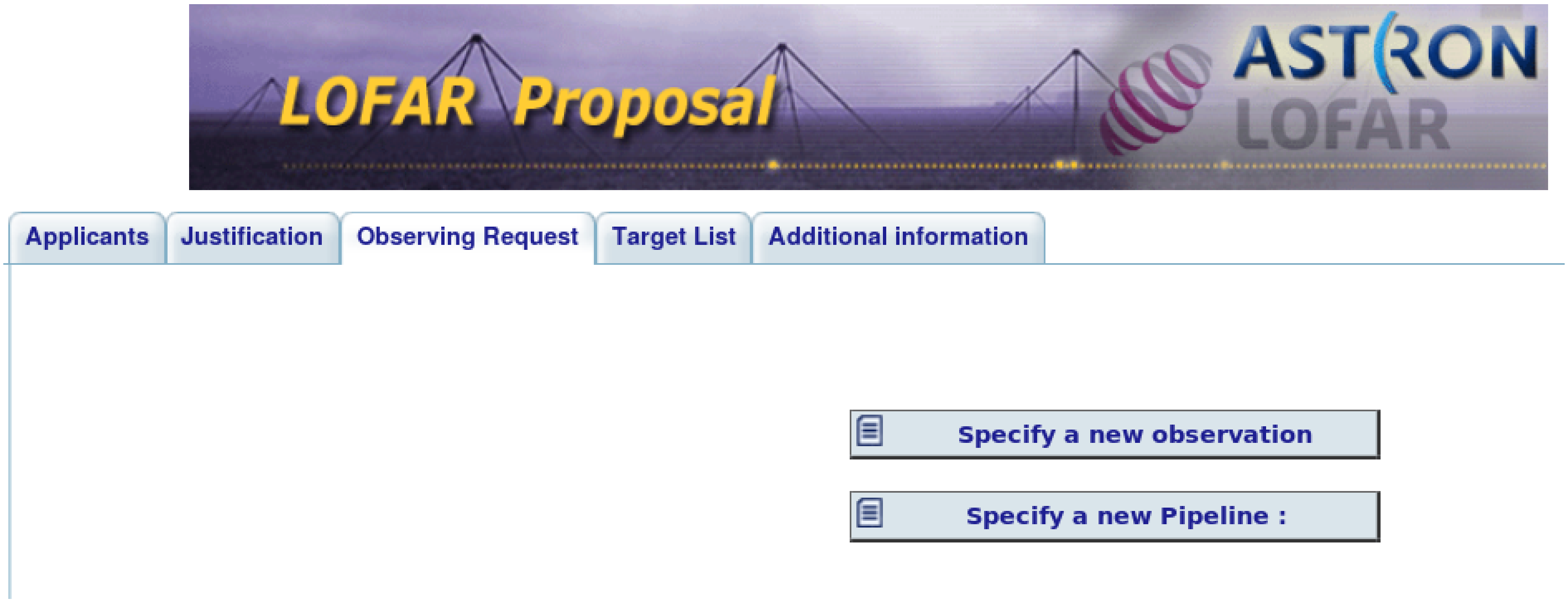


TAheatmap_*.png

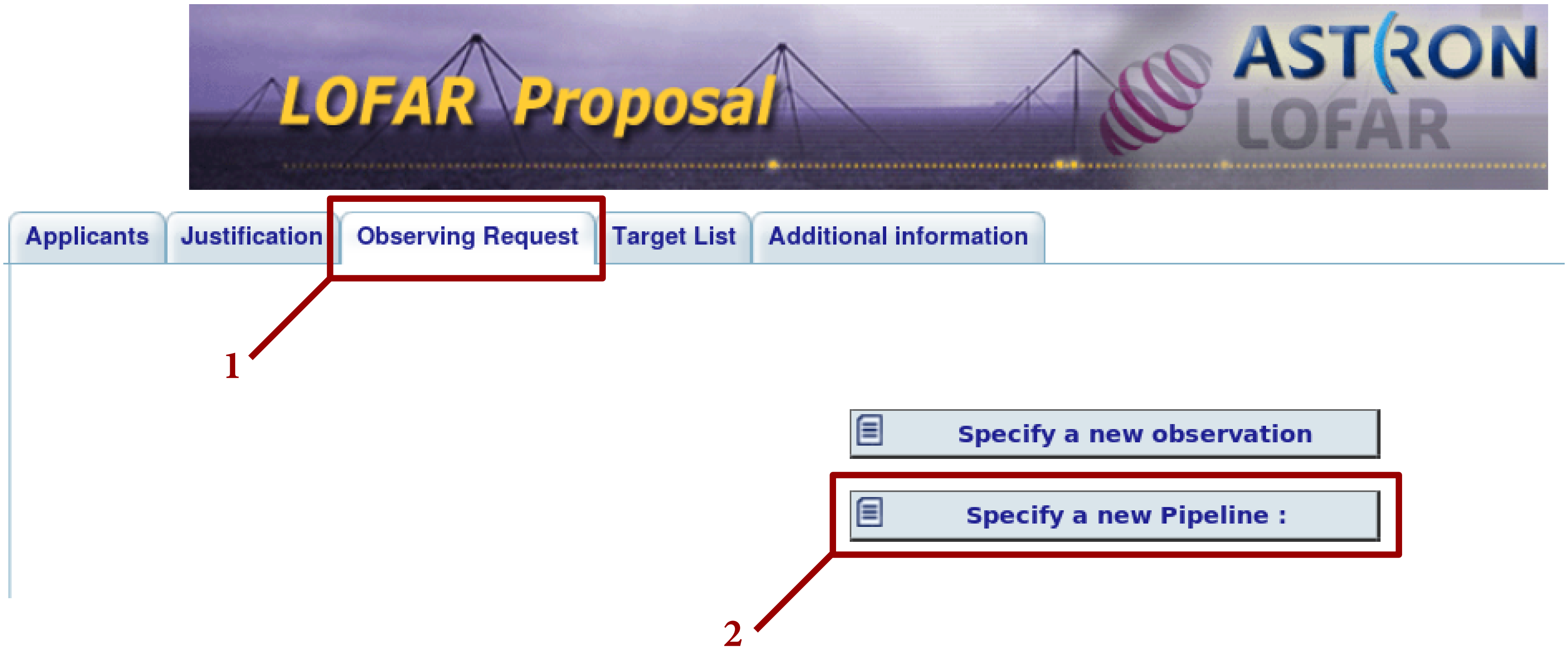
PulP output data

- Raw data in 8-bit format (optional)
- Raw .h5 metadata files
- Pulsar data cubes (both from PRESTO and DSPSR pipelines)
- PRESTO pipeline:
  → rfifind mask
  → PSRFITS filterbank data
- DSPSR pipeline:
  → filterbank file(s) when SP analysis was done (optional)
- Single-pulse data (optional)
  → .singlepulse
  → Single-pulse plots
- Diagnostic plots
  → Plot with multiple profiles (multiple TABs, etc.) – *combined.png*
  → DSPSR diagnostic plots – *status.png*
  → Localization maps – *TAheatmap_*.png*

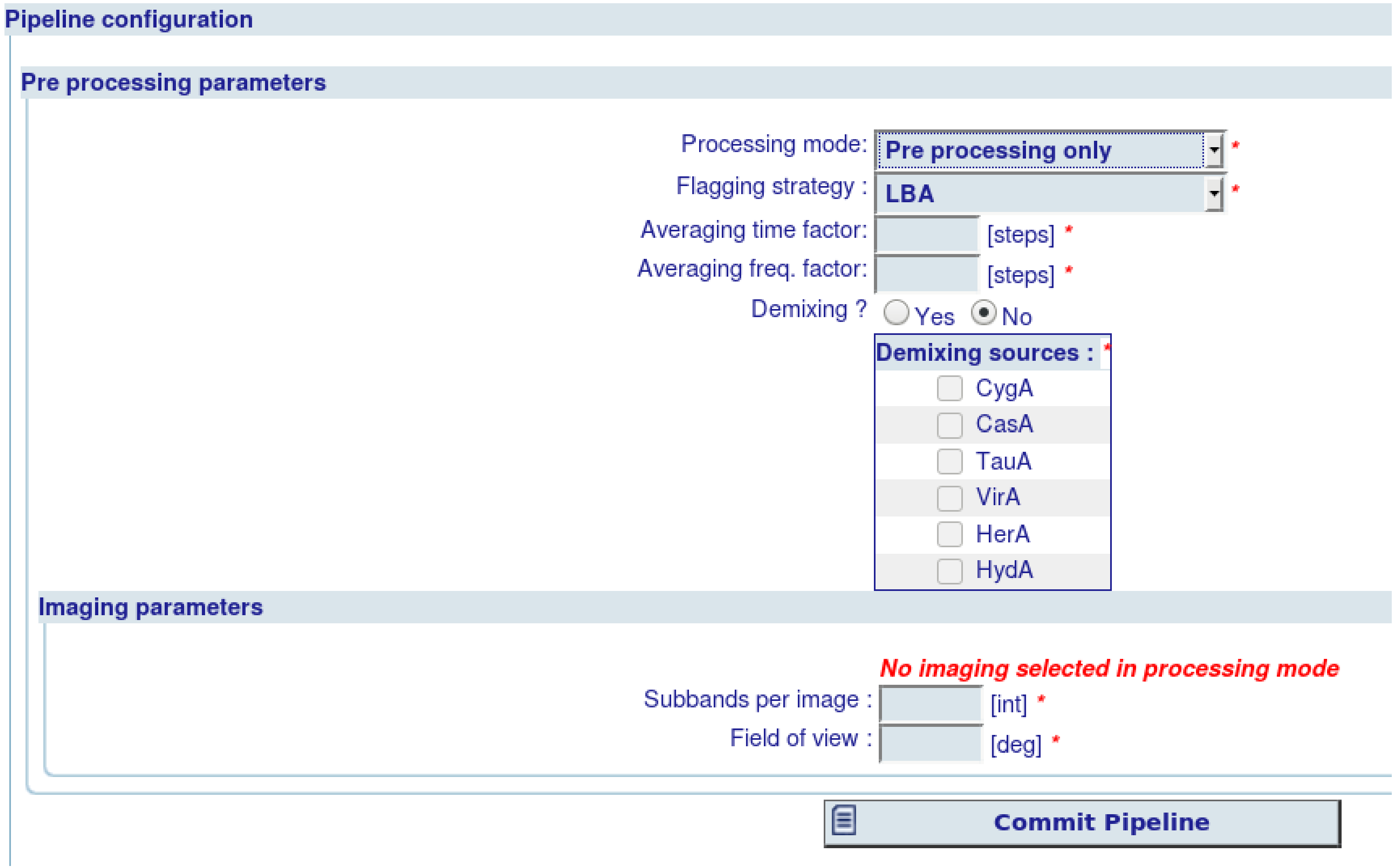
NorthStar: PulP (1)



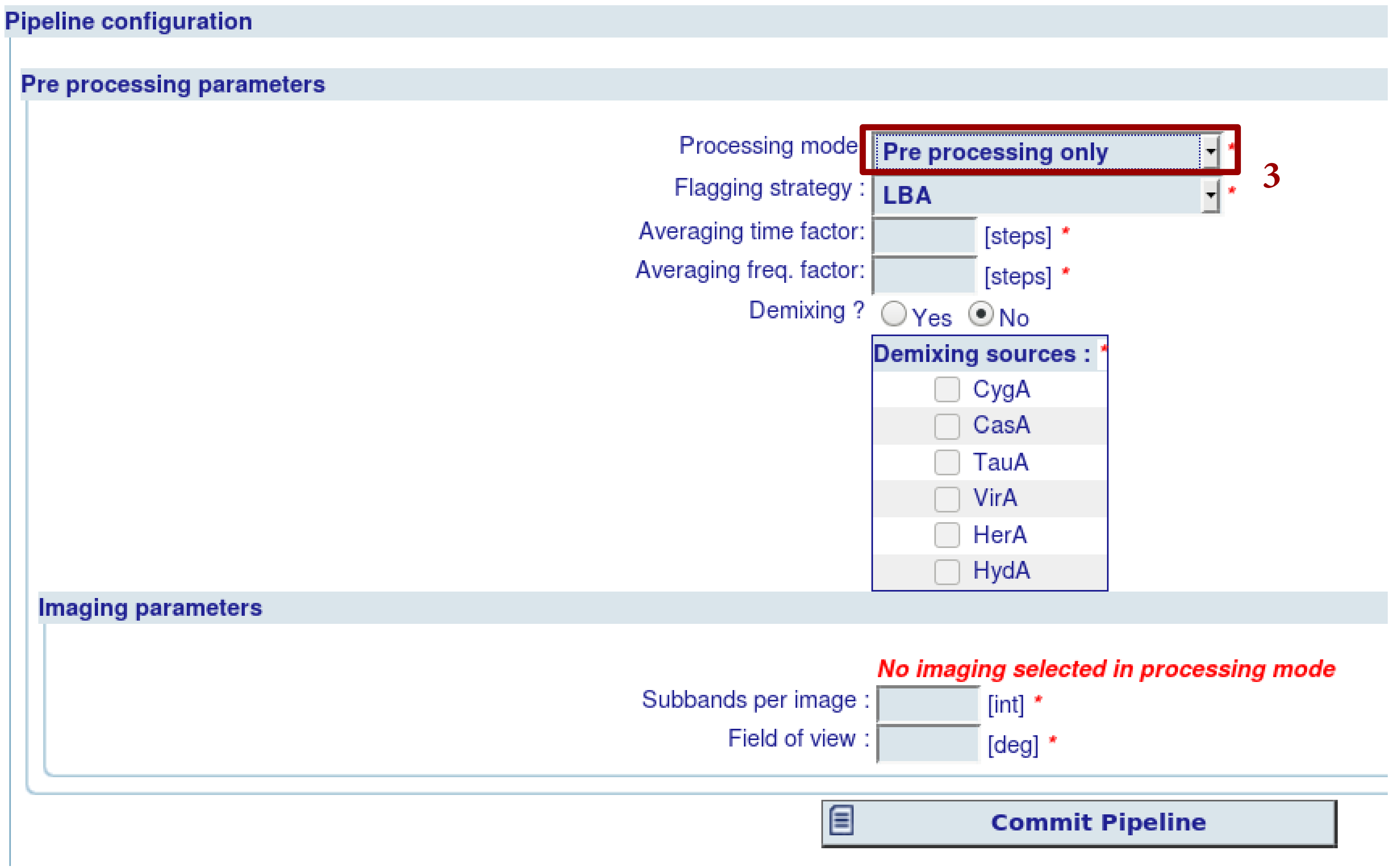
NorthStar: PulP (1)



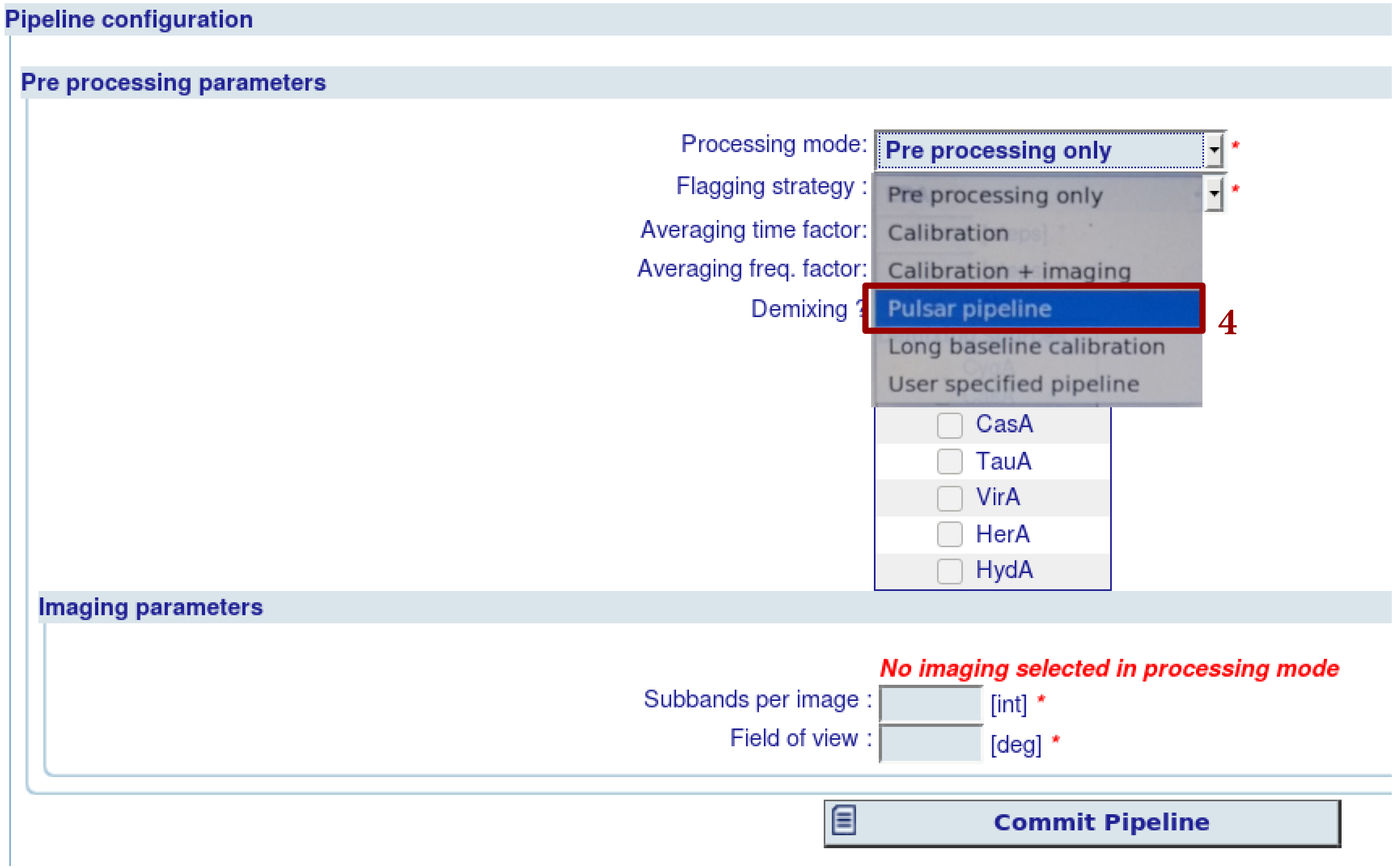
NorthStar: PulP (2)



NorthStar: PulP (2)



NorthStar: PulP (3)



NorthStar: PulP (4)

Pipeline configuration

Pre processing parameters

Processing mode: Pulsar pipeline *

Skip RFI check : ◯ Yes ◉ No

Skip folding : ◯ Yes ◉ No

Skip pdmp : ◯ Yes ◉ No

Skip dspsr : ◯ Yes ◉ No

Skip prepfold : ◯ Yes ◉ No

Single pulse analysis : ◯ Yes ◉ No

RRATs analysis : ◯ Yes ◉ No

Skip dynamic average : ◯ Yes ◉ No

Length of subintegration (sec) : 0

Convert HDF5 32-bit raw data to 8-bit : ◯ Yes ◉ No

Clip threshold (in units of sigma) : 5

Sigma limit in conversion from raw HDF5 to PSRFITS : 3

Number of blocks read at once in conversion to PSRFITS : 100

Prepfold options :

Prepsubband options :

RFIfind options :

*Expert settings: not all options may result in a successful pipeline, therefore the Observatory may decline setting certain parameters.

Dspsr options :

Digifil options :

Prepdata options :

Extra options to convert from raw HDF5 to PSRFITS :

Pulsar :

Configuration comments :

5

Commit Pipeline

6

NorthStar: PulP options (1)

- Skip RFI check
  - → all RFI excision commands will be skipped (*rfifind*, *paz*, *clean.py*, *subdyn.py*)
- Skip folding
  - → No folding will be done
  - → For CV data, nothing will happen unless Single Pulse Analysis = TRUE, or conversion to 8-bit is requested
  - → For non-CV data, *prepfold* will not run, only *2bf2fits* to create PSRFITS file and *rfifind*
- Skip pdmp
  - → *pdmp* will not run in DSPSR pipeline
- Skip dspsr
  - → DSPSR pipeline will not run
- Skip prepfold
  - → *prepfold* will not run
- Single pulse analysis
  - → Single pulse search part of both PRESTO and DSPSR pipelines will not be carried out
- RRATs analysis
  - → RRATs part of the PRESTO pipeline will not be carried out

NorthStar: PulP options (2)

- Skip dynamic average
  - instead «average» should read «spectrum»!
  - *subdyn.py* will not run

- Length of subintegration (sec)
  - Specify sub-integration length for *dspsr* command (that goes with -*L* option)
  - By default it is 5 sec for CV data and 60 sec for non-CV data
  - If -*s* option is specified by a user in Dspsr extra options, then this option is ignored

- Convert HDF5 32-bit raw data to 8-bit
  - Self-explanatory, to run *digitize.py*

- Clip thershold (in units of sigma)
  - Clip threshold for *digitize.py*. Above this threshold raw data will be clipped. Default value is 5.0

- Sigma limit in conversion from raw HDF5 to PSRFITS
  - Same as for converting from raw 32-bit to raw HDF5 8-bit, but rather to PSRFITS data using *2bf2fits* command. Default value is 3

- Number of blocks read at once in conversion to PSRFITS
  - Determines the length of data where the running mean/rms is calculated when converting to PSRFITS using *2bf2fits*. The larger the number, the longer the time window for calculation. Default is 100

NorthStar: PulP options (3)

- Prepfold options
  - *prepfold* extra user options
- Prepsubband options
  - *prepsubband* extra user options
- RFIfind options
  - *rfifind* extra user options
- Dspsr options
  - *dspsr* extra user options
- Digifil options
  - *digifil* extra user options
- Prepdata options
  - *prepdata* extra user options
- Extra options to convert from raw HDF5 to PSRFITS
  - *2bf2fits* extra user options

NorthStar: PulP options (4)

- Pulsar
  - → Pulsar name for folding *without* «PSR» prefix
  - → Several (max 3) pulsar names can be specified separated by commas, no spaces
  - → Folding will be done for the first 3 pulsars
  - → If no pulsar name is given, then target name from the observation specification will be used.
  - → User must provide Science Support with the necessary ephemeris files (parfiles)
  - → If parfiles are not given, PulP will use ephemeris from the ATNF catalog
  - → If given pulsar is not known by ATNF and parfile is not provided, PulP will search for the brightest pulsar in the FoV of the SAP and fold the data for this brightest pulsar
  - → Instead of pulsar name, special words can be used, such as:
    - «meta» – use target name for each SAP separately in the HDF5 metadata
    - «parset» - use target name from the observational parset file for each SAP separately (obsolete)
    - «sapfind» – find brightest pulsar in the FoV for each SAP
    - «sapfind3» – find 3 brightest pulsars in the FoV for each SAP
    - «tabfind» – find brightest pulsar in the FoV for each TAB
    - «tabfind+» – look for pulsars first following «meta». If no legitimate pulsar is found, then look for pulsar following logic of «sapfind». If no pulsar found then look for it following logic of «tabfind»
  - → «NONE» as a pulsar name is ignored

PulP options (advice from the experience)

- For CV data:
  - → Dspsr options: *-U minX1 -t 1*
  - → Default subint time is 5 s, so for pulsars with P>5 s, consider using larger Length of subintegration, or make subintegrations equal to pulsar period (add option *-s* to Dspsr options)

- For non-CV data:
  - → Extra options to convert from raw HDF5 to PSRFITS: *-nsamples 8192*
  - → Number of blocks read at once in conversion to PSRFITS: *6*
  - → RFIfind options: *-blocks 1*
  - → For short observations (<~ 3 min), change the default Length of subintegration of 60 s to a smaller number (5 — 10 s)

PSRCHIVE config file

- To process low frequency observations it is very important to have good parameters in your PSRCHIVE configuration file *psrchive.cfg*

  - → Possible locations of this file:
    - ~/.psrchive.cfg
    - echo $PSRCHIVE_CONFIG
  - → If you do not have it, then you can run *psrchive_config* which will print the contents of configuration to stdout. Redirect the input to the file, and inspect it to have proper values.
    - *psrchive_config > ~/.psrchive.cfg*

- What to pay attention to:

  - → Dispersion::barycentric_correction. Should be ON
    - Dispersion::barycentric_correction = 1
  - → FrequencyAppend::weight = none
  - → WeightedFrequency::round_to_kHz. Should be OFF (we need maximum precision!)
    - WeightedFrequency::round_to_kHz = 0

  - → Also possibly consider:
    - FrequencyAppend::force_new_predictor = 1
    - Predictor::default = tempo2
    - Predictor::policy = default

Lunch Time!