Data processing with ASKAPsoft

The ASKAP Central Processor

Daniel Mitchell

2018 ICRAR/CASS Radio School

Before we start...

- What is the purpose of this lecture?
  - To know what ASKAPsoft is
  - To know how to use it
  - To know how ASKAP pipeline images are made
  - To prepare for the tutorials
- What is the purpose of the tutorials?
  - To gain experience with calibration and imaging
  - To gain experience with ASKAPsoft
  - To know what ASKAP pipelines do and what to look for

Overview

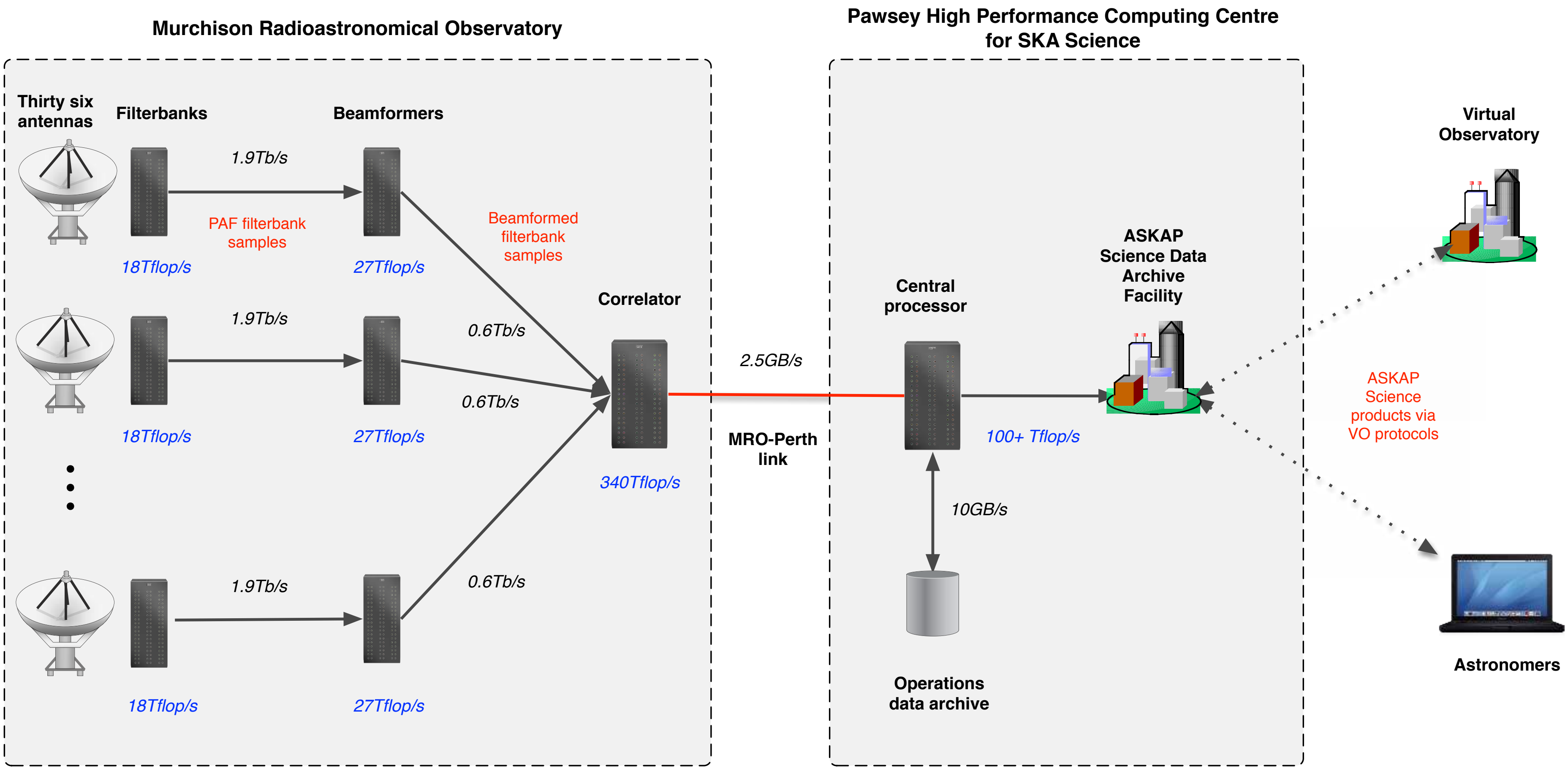
- A brief overview of ASKAP
- A brief look at the ASKAPsoft pipeline
- A closer look at ASKAPsoft
  - Some of the things ASKAPsoft does differently
  - Common ASKAPsoft tasks
- A walk through the tutorial

A brief overview of ASKAP

Key Features

- 36 x 12m antennas forming a radio synthesis array (< 6 km)
- Use of phased array feeds for wide field of view (30 square degrees)
- Operation at centimetre wavelengths @ 300 MHz bandwidth
- Situated in an isolated location, operated remotely, without a dedicated maintenance staff on-site
- Large data volumes (2.8 GB/s) which will require distributed processing
- Fully automated pipelined processing for many use cases
- Primarily survey instrument, service (queue) observing & dynamic scheduling

ASKAP Data Flow



T. Cornwell, February 22 2010

Central Processor (CP)

- CP Hardware
  - Ingest nodes (16 x node cluster)
  - 1 PB fast storage for scratch space + fast interconnect (> 10 GB/s)
  - 472 CPU nodes, 64 GB/node (Galaxy CPU nodes)
  - Head nodes and external access nodes
- CP Software (ASKAPsoft)
  - Test Framework (incl. simulators)
  - Real-time Services (manager, ingest, RFI, sky model, calibration data)
  - Calibration and Imaging Pipelines (data distribution framework, calibration, continuum, spectral line, source finder, slow transient, continuum-10”, postage stamps, zoom modes, pipeline scripts)
  - CASDA upload

A brief look at the ASKAPsoft pipeline

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required: Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution

4 – Get science data → Apply calibration → Apply flags → Average to 1MHz → Image via MSMFS and continuum cube → Form sky model for self cal → Perform – self cal loops as required

5 – Mosaick all beams together

6 – Run spectral line pipeline → Apply calibration to full resolution data → Subtract best model → Image each channel parallel by frequency → Image domain continuum subtraction

7 – Detect sources → Validation → Archiving

1 - Fetch metadata – footprint, frequencies

The Current Pipeline

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:

Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution

The Current Pipeline

- 1 - Fetch metadata – footprint, frequencies
- 2 – What tasks are required Calibration? Continuum, Spectral line?
- 3 – Assuming bandpass calibration is required:
  - Fetch calibration observation of 1934-638
  - Match the relevant beam
  - Flag the calibration data as requested / required
  - Generate bandpass solution

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:

Fetch calibration observation of 1934-638

Match the relevant beam

Flag the calibration data as requested / required

Generate bandpass solution

The Current Pipeline

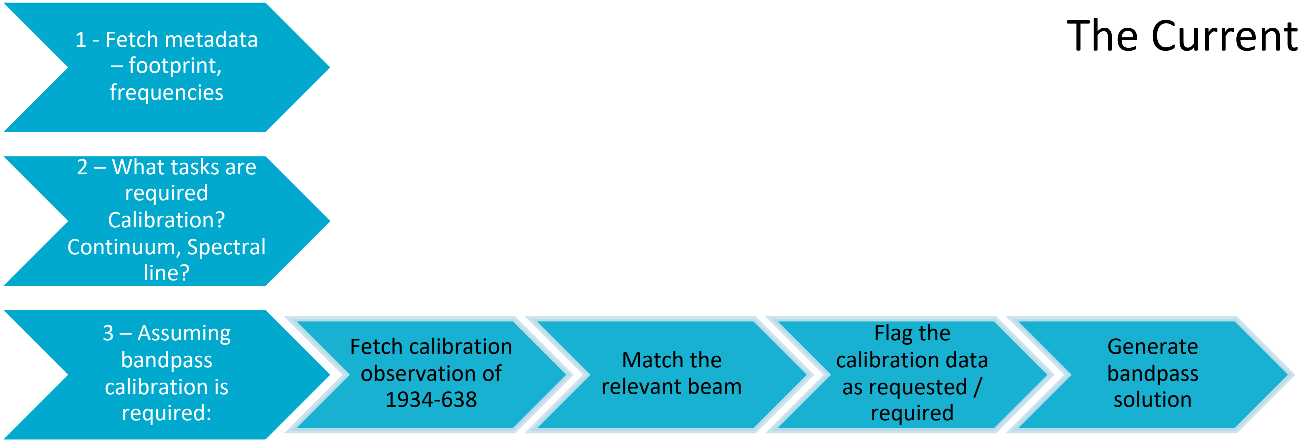
1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:

- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

The Current Pipeline



The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

CSIRO

The Current Pipeline

- 1 - Fetch metadata – footprint, frequencies
- 2 – What tasks are required Calibration? Continuum, Spectral line?
- 3 – Assuming bandpass calibration is required:
  - Fetch calibration observation of 1934-638
  - Match the relevant beam
  - Flag the calibration data as requested / required
  - Generate bandpass solution
- 4 – Get science data
  - Apply calibration
  - Apply flags
  - Average to 1MHz
  - Image via MSMFS and continuum cube
  - Form sky model for self cal
  - Perform – self cal loops as required

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required: → Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution

4 – Get science data → Apply calibration → Apply flags → Average to 1MHz → Image via MSMFS and continuum cube → Form sky model for self cal → Perform – self cal loops as required

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

The Current Pipeline

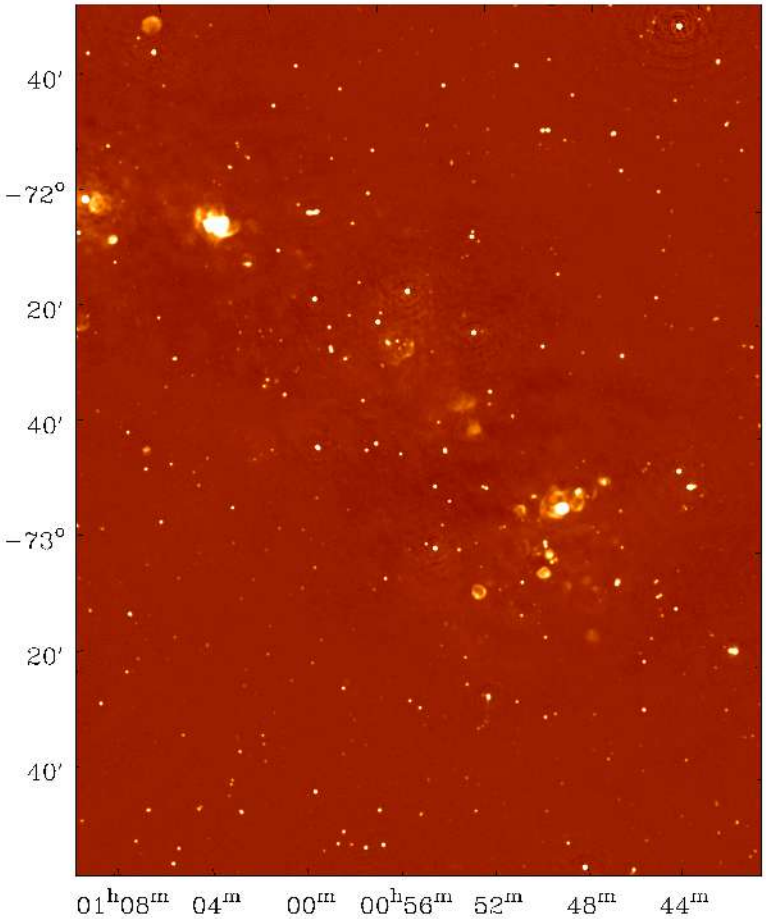
1 - Fetch metadata – footprint, frequencies

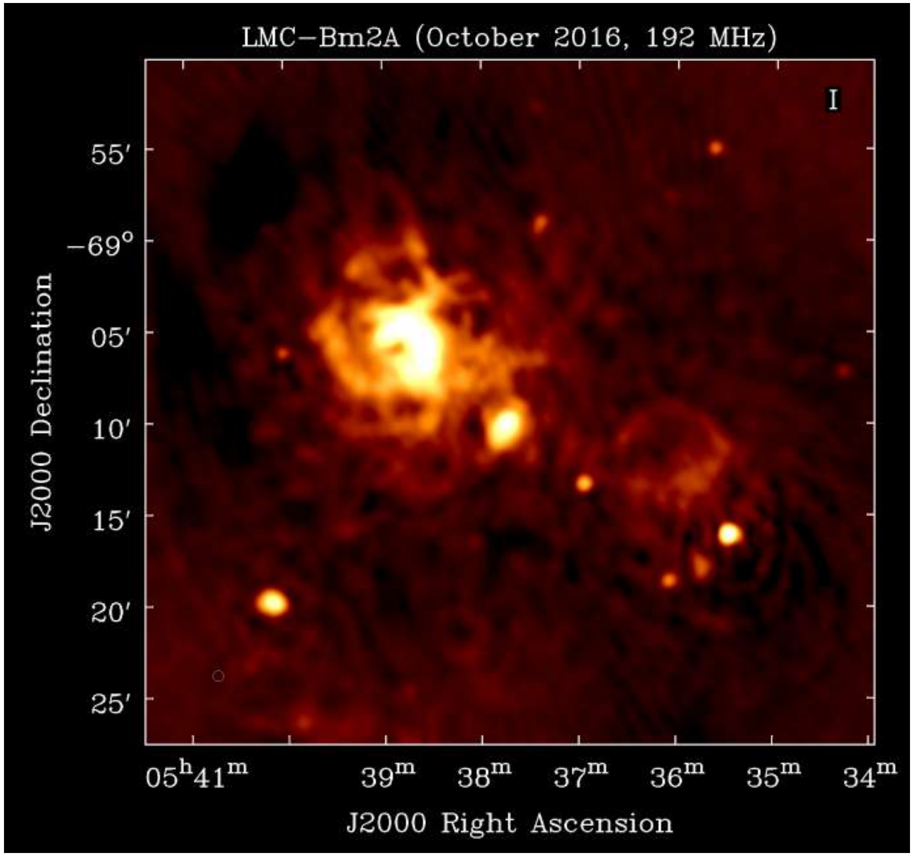
2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required: → Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution

4 – Get science data → Apply calibration → Apply flags → Average to 1MHz → Image via MSMFS and continuum cube → Form sky model for self cal → Perform – self cal loops as required

LMC and SMC





Thanks to Wasim Raja

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required: → Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution

4 – Get science data → Apply calibration → Apply flags → Average to 1MHz → Image via MSMFS and continuum cube → Form sky model for self cal → Perform – self cal loops as required

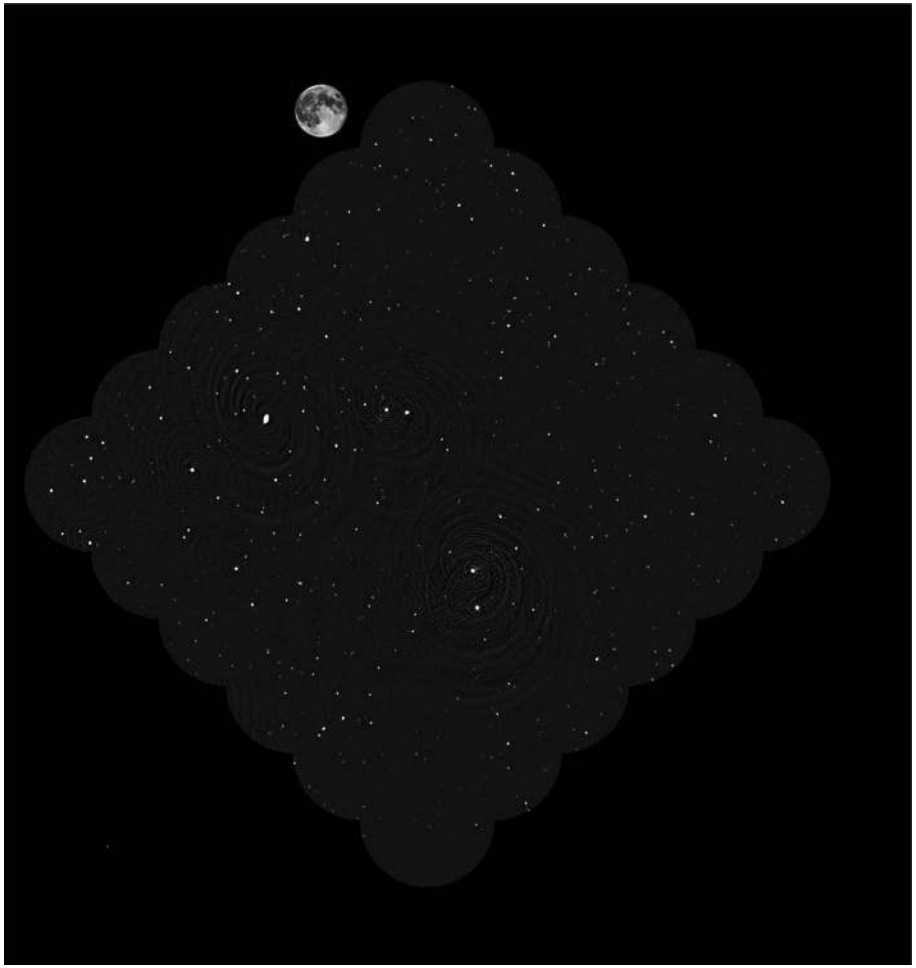
5 – Mosaick all beams together

11 hr observations of the APUS field – Thanks to Wasim Raja

30+ sq. degrees
f_c = 939.5MHz
BW = 48MHz

RMS noise in image: 300µJy/Beam (Theoretically ~150-200µJy/Beam)
Dynamic range: 1.4 X 10^4

CROSS Matching: (courtesy, Martin Bell, using the VAST pipeline)
No. of sources detected: 1380 (above 7σ)
RMS Pointing error: ~7’’ (In comparison, Synthesised Beam: ~60’’X60’’)"



The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

5 – Mosaick all beams together

6 – Run spectral line pipeline
- Apply calibration to full resolution data
- Subtract best model
- Image each channel parallel by frequency
- Image domain continuum subtraction

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required: → Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution

4 – Get science data → Apply calibration → Apply flags → Average to 1MHz → Image via MSMFS and continuum cube → Form sky model for self cal → Perform – self cal loops as required

5 – Mosaick all beams together

6 – Run spectral line pipeline → Apply calibration to full resolution data → Subtract best model → Image each channel parallel by frequency → Image domain continuum subtraction

The Current Pipeline

- 1 - Fetch metadata – footprint, frequencies
- 2 – What tasks are required Calibration? Continuum, Spectral line?
- 3 – Assuming bandpass calibration is required: → Fetch calibration observation of 1934-638 → Match the relevant beam → Flag the calibration data as requested / required → Generate bandpass solution
- 4 – Get science data → Apply calibration → Apply flags → Average to 1MHz → Image via MSMFS and continuum cube → Form sky model for self cal → Perform – self cal loops as required
- 5 – Mosaick all beams together
- 6 – Run spectral line pipeline → Apply calibration to full resolution data → Subtract best model → Image each channel parallel by frequency → Image domain continuum subtraction

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

5 – Mosaick all beams together

6 – Run spectral line pipeline
- Apply calibration to full resolution data
- Subtract best model
- Image each channel parallel by frequency
- Image domain continuum subtraction

The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

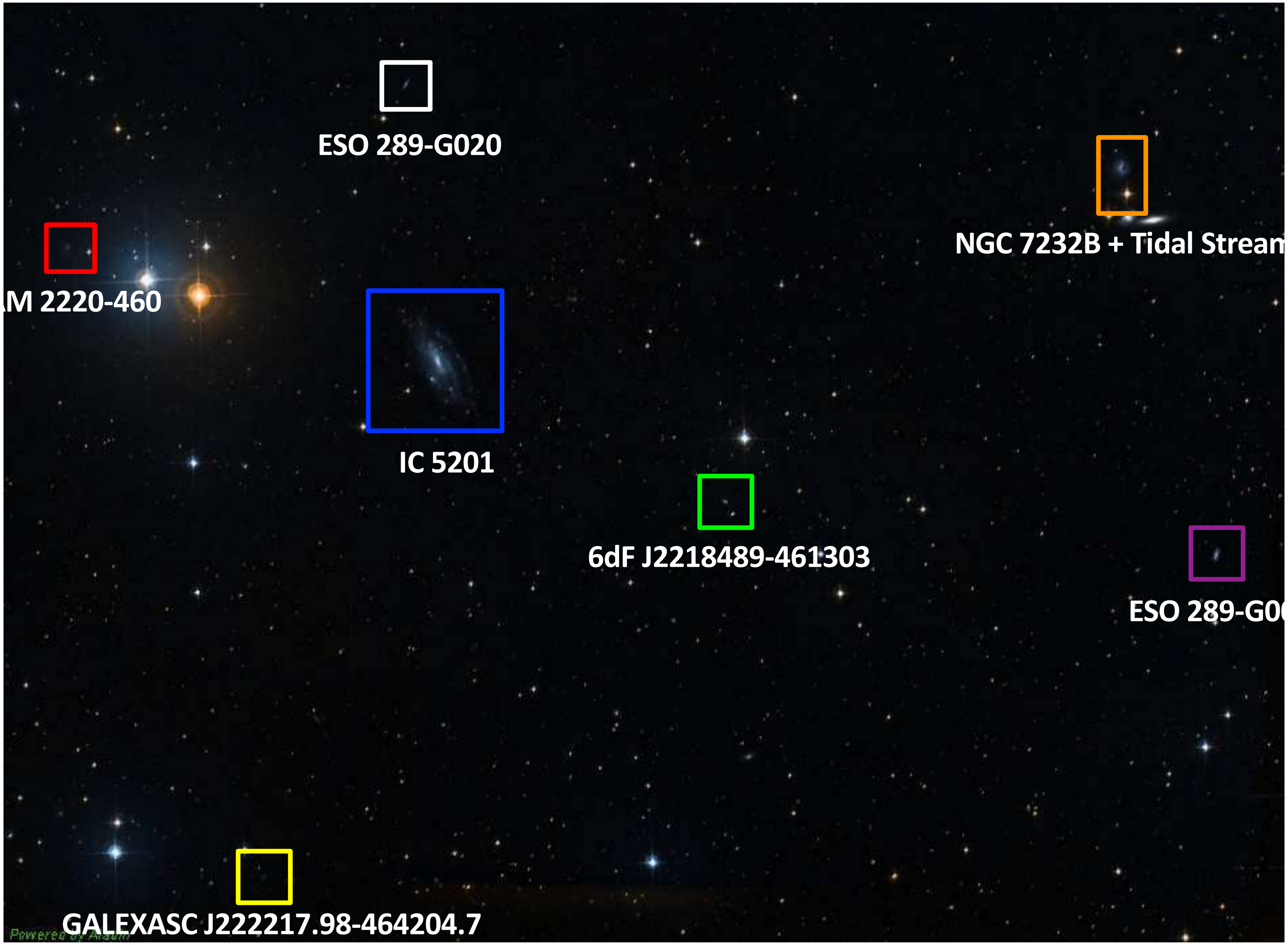
3 – Assuming bandpass calibration is required:
- Fetch calibration observation of 1934-638
- Match the relevant beam
- Flag the calibration data as requested / required
- Generate bandpass solution

4 – Get science data
- Apply calibration
- Apply flags
- Average to 1MHz
- Image via MSMFS and continuum cube
- Form sky model for self cal
- Perform – self cal loops as required

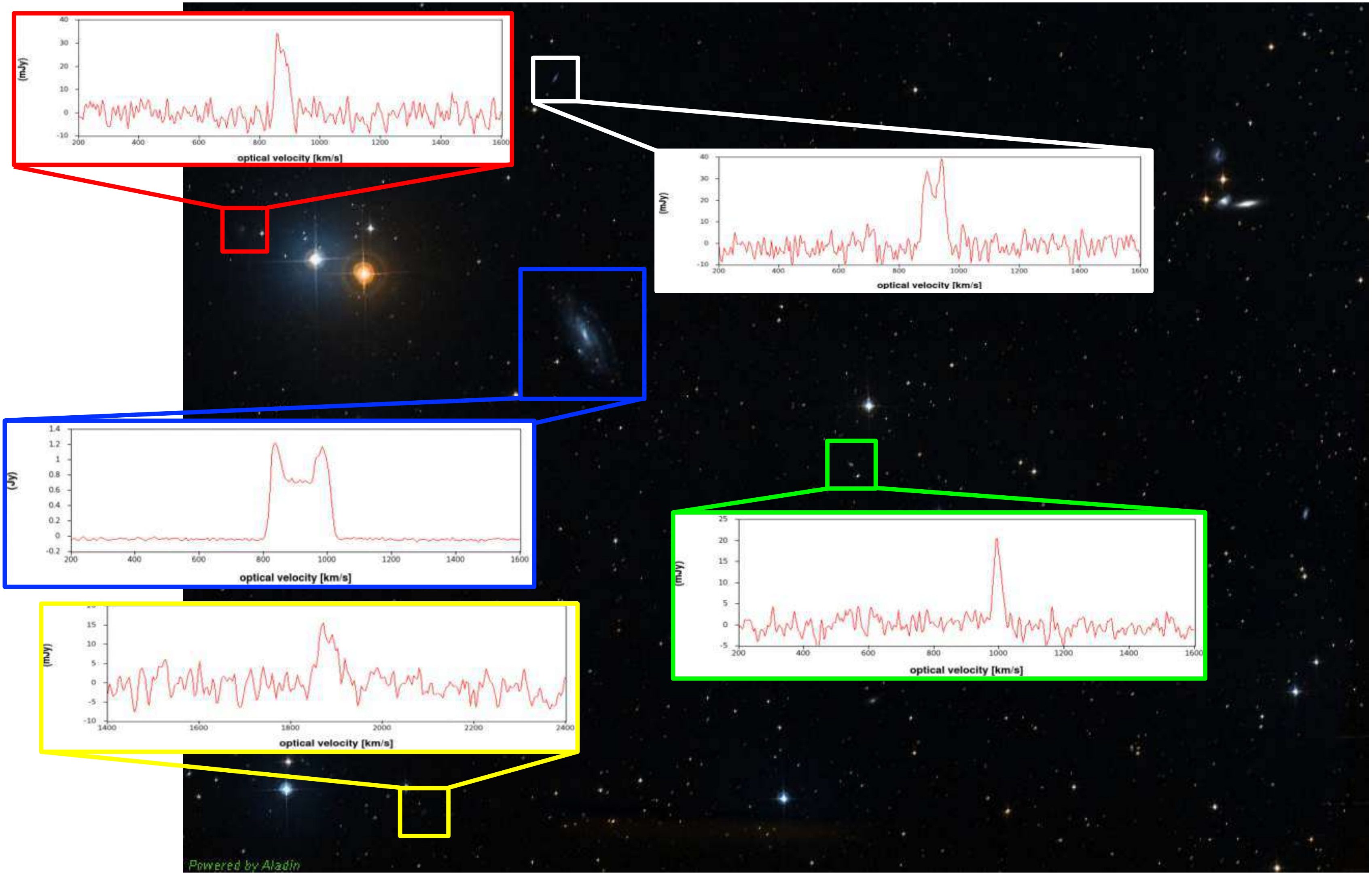
5 – Mosaick all beams together

6 – Run spectral line pipeline
- Apply calibration to full resolution data
- Subtract best model
- Image each channel parallel by frequency
- Image domain continuum subtraction

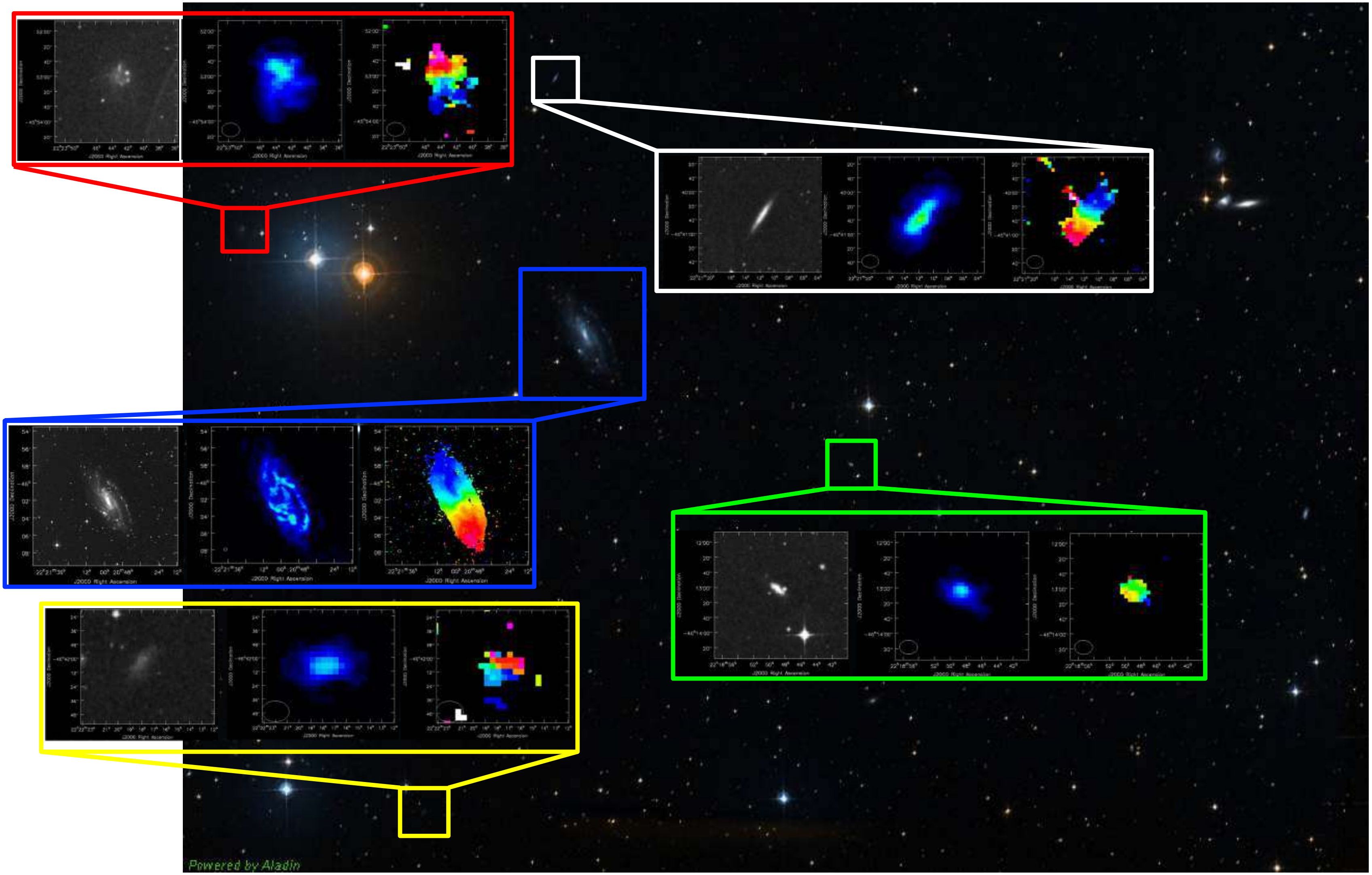
NGC 7232 Field. Thanks to Dane Kleiner and the WALLABY team



NGC 7232 Field. Thanks to Dane Kleiner and the WALLABY team



NGC 7232 Field. Thanks to Dane Kleiner and the WALLABY team



The Current Pipeline

1 - Fetch metadata – footprint, frequencies

2 – What tasks are required Calibration? Continuum, Spectral line?

3 – Assuming bandpass calibration is required: | Fetch calibration observation of 1934-638 | Match the relevant beam | Flag the calibration data as requested / required | Generate bandpass solution

4 – Get science data | Apply calibration | Apply flags | Average to 1MHz | Image via MSMFS and continuum cube | Form sky model for self cal | Perform – self cal loops as required

5 – Mosaick all beams together

6 – Run spectral line pipeline | Apply calibration to full resolution data | Subtract best model | Image each channel parallel by frequency | Image domain continuum subtraction

7 – Detect sources | Validation | Archiving

The Current Pipeline

- 1 - Fetch metadata – footprint, frequencies
- 2 – What tasks are required Calibration? Continuum, Spectral line?
- 3 – Assuming bandpass calibration is required:
  - Fetch calibration observation of 1934-638
  - Match the relevant beam
  - Flag the calibration data as requested / required
  - Generate bandpass solution
- 4 – Get science data
  - Apply calibration
  - Apply flags
  - Average to 1MHz
  - Image via MSMFS and continuum cube
  - Form sky model for self cal
  - Perform – self cal loops as required
- 5 – Mosaick all beams together
- 6 – Run spectral line pipeline
  - Apply calibration to full resolution data
  - Subtract best model
  - Image each channel parallel by frequency
  - Image domain continuum subtraction
- 7 – Detect sources
  - Validation
  - Archiving

A closer look at ASKAPsoft

ASKAPsoft

- A suite of calibration and imaging tasks built primarily for ASKAP pipelines.
- Built from the start for large-scale high-performance computing.
- Has a range of calibration tasks
- Has a range of imaging tasks and algorithms:
  - gridders: Box, SphFunc, WProject, WStack, AWProject, AProjectWStack
  - solvers: Hogbom, MultiScale, MultiScaleMFS and BasisfunctionMFS
  - preconditioners: Wiener, GaussianTaper
- Has a linear mosaicking task
- Has a source finding task
- Has tasks for vis simulation, flagging, splitting, merging, etc.

ASKAPsoft User Documentation

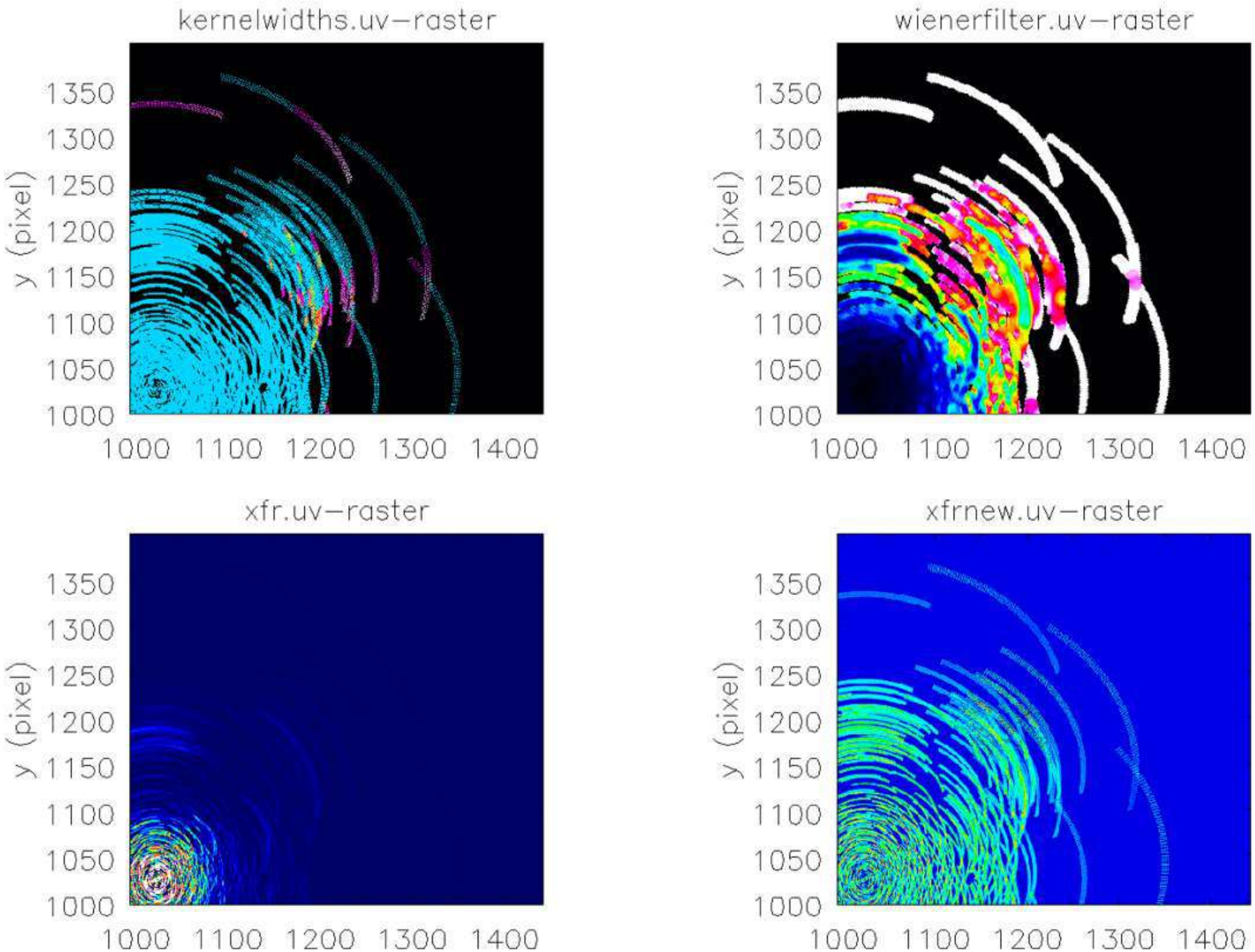
- https://www.atnf.csiro.au/computing/software/askapsoft/sdp/docs/current/
- https://www.atnf.csiro.au/computing/software/askapsoft/sdp/docs/nightly/
- Introduction
- Platform Documentation
- General Documentation
- Calibration and Imaging Documentation
- Source-Finding Documentation
- Utilities
- Services Documentation
- ASKAP Processing Pipelines
- Tutorials
- Recipes

Some of the things ASKAPsoft does differently

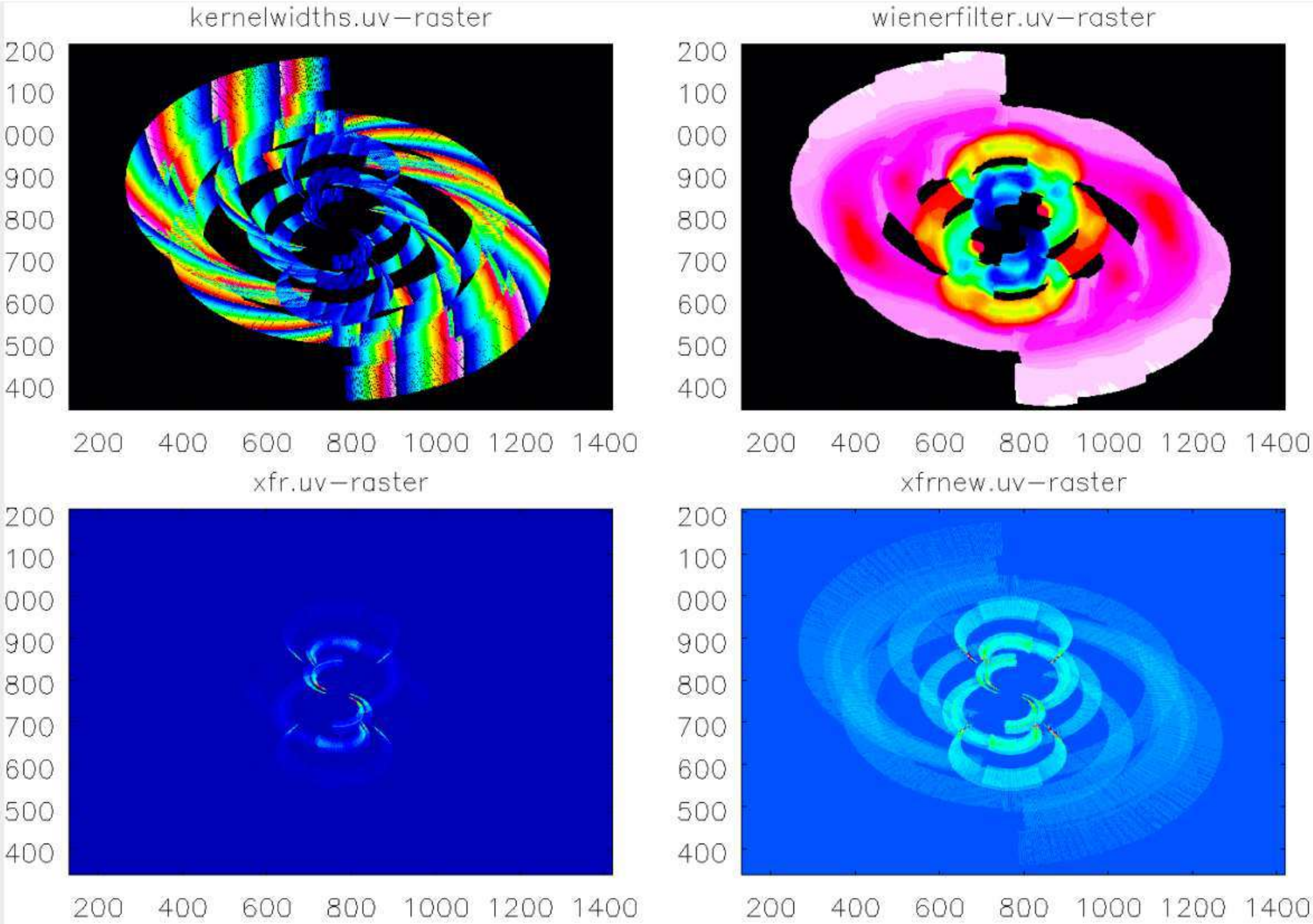
Preconditioning

- "Precondition" dirty images and PSFs for deconvolution
- Like standard weighting and tapering, but applied *after* gridding to uv cells rather than visibilities
  - e.g. uniform weight, robust weighting, Gaussian uv tapers
  - Uniform weighting is basically a Wiener filter
- Done to avoid multiple reads of very large visibility data sets
- Various complications when it comes to wide-field imaging
- Is it still needed for ASKAP processing?
  - Continuum visibilities: *relatively* small → use standard vis weighting?
  - Spectral line visibilities: separate weights per channel, and single-channel visibilities are *relatively* small → use standard vis weighting?

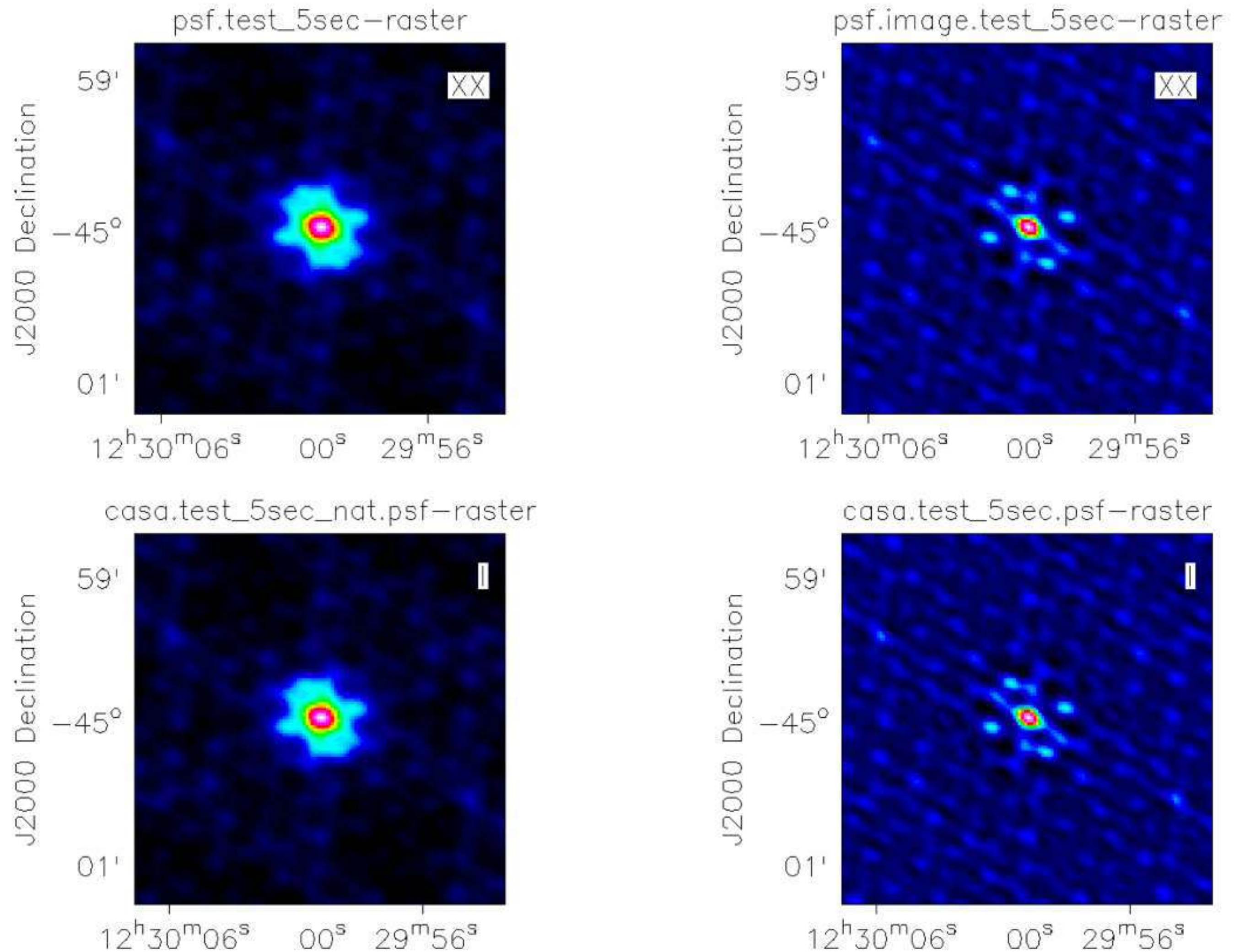
Preconditioning



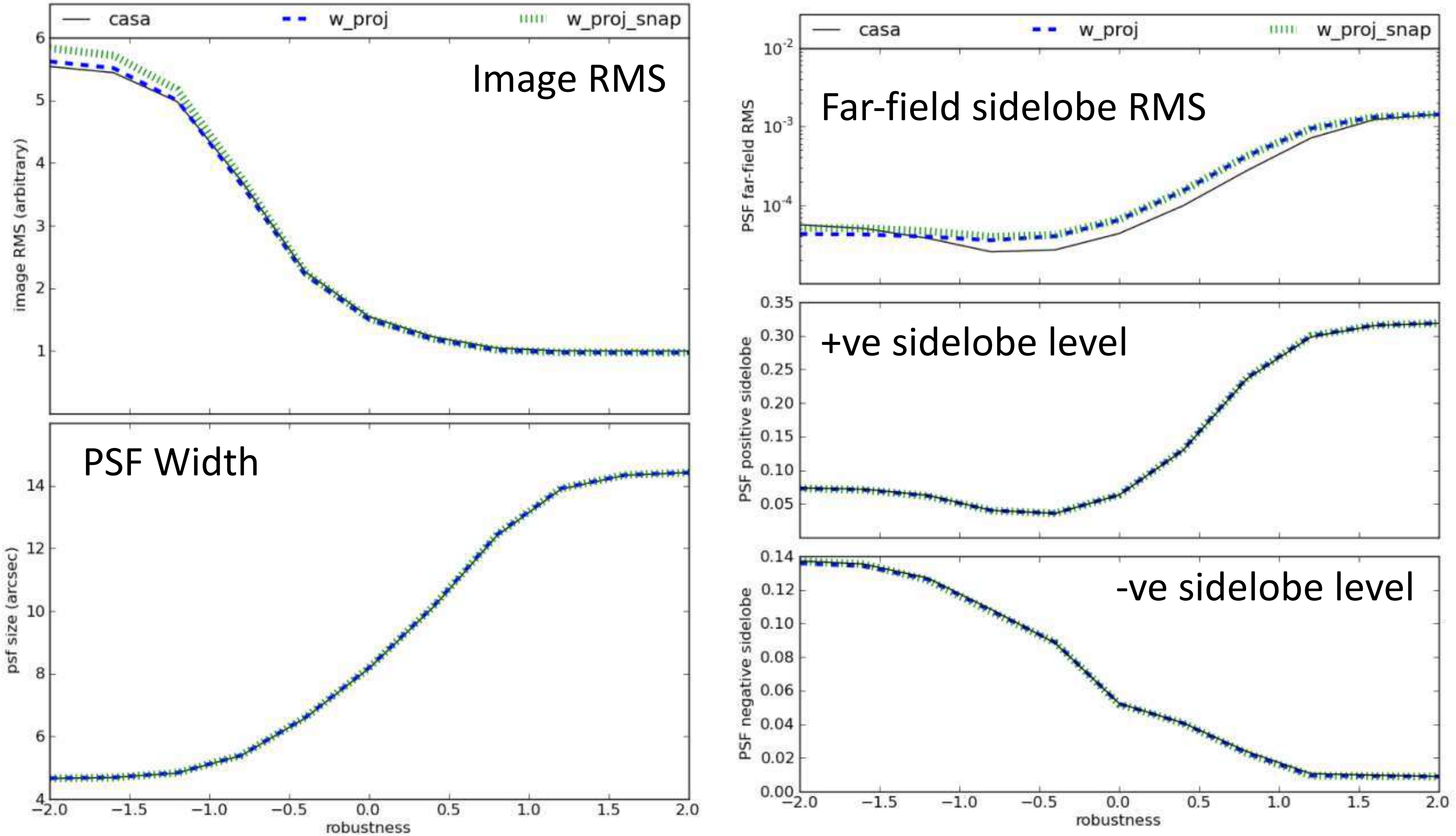
Preconditioning



Preconditioning



Preconditioning



Courtesy of Josh Marvil

Multi-beam imaging

- Joint gridding
  - Original design was to grid all beams onto the same large uv plane
  - uv grid: $12{,}000^2$ pixels × 4 pol × 8 bytes (5 GB) × N_{freq} × N_{tt} × overhead
  - Kernel grids: $w_{max}\theta^2 \approx 400$ pixels across
  - With 8×8 oversampling & 3000 w-planes: ≈ 90 GB
  - Memory-bandwidth limited

- Separate gridding and joint deconvolution
  - Grid each beam separately
  - Do linear mosaic before deconvolution
  - Kernel grids: $w_{max}\theta^2 \approx 45$ pixels across (2° image width)
  - With 8×8 oversampling & 360 planes: ≈ 0.1 GB

- For early science, mostly do separate gridding and separate deconvolution

Common ASKAPsoft tasks

Cflag

- Visibility flagger. Limited in scope for HPC and eventual inclusion in *Ingest*
- Example 1

$ cflag -c config.in

```
Cflag.dataset                                = target.ms

# Enable Stokes V flagging flagger with a 5-sigma threshold
Cflag.stokesv_flagger.enable                 = true
Cflag.stokesv_flagger.threshold              = 5.0

# Enable selection based flagging with two rules
# Rule 1: Beams 0 and 1 on ant "ak01”, rule 2: Spectral Channels 0 to 16 on spectral window 0
Cflag.selection_flagger.rules                = [rule1, rule2]
Cflag.selection_flagger.rule1.antenna        = ak01
Cflag.selection_flagger.rule1.feed           = [0, 1]
Cflag.selection_flagger.rule2.spw            = 0:0~16
```

Cflag

- Visibility flagger. Limited in scope for HPC and eventual inclusion in *Ingest*
- Example 2

$ cflag -c config.in

```
Cflag.dataset                        = target.ms

# Elevation based flagging
Cflag.elevation_flagger.enable       = true
Cflag.elevation_flagger.low          = 12.0
Cflag.elevation_flagger.high         = 89.0

# Amplitude based flagging
Cflag.amplitude_flagger.enable       = true
Cflag.amplitude_flagger.high         = 10.25
Cflag.amplitude_flagger.low          = 1e-3
```

Cflag

- Visibility flagger. Limited in scope for HPC and eventual inclusion in *Ingest*
- Example 3

$ cflag -c config.in

```
Cflag.dataset                                          = target.ms

# Amplitude based flagging
Cflag.amplitude_flagger.enable                         = true

# Threshold using the median and IQR of each spectrum
Cflag.amplitude_flagger.dynamicBounds                  = true

# Threshold again after averaging spectra in time
Cflag.amplitude_flagger.integrateSpectra               = true
Cflag.amplitude_flagger.integrateSpectra.threshold     = 4.0
Cflag.amplitude_flagger.integrateTimes                 = true
```

Cbpcalibrator

- Bandpass calibration task.
- A good number of MPI ranks: 1 + nbeam x nchan / integer_factor

$ <MPI wrapper> cbpcalibrator -c config.in

```
Cbpcalibrator.dataset                        = calibration_data.ms
Cbpcalibrator.nAnt                           = 36
Cbpcalibrator.nChan                          = 16000
Cbpcalibrator.nBeam                          = 36
Cbpcalibrator.refantenna                     = 1
Cbpcalibrator.calibaccess                    = table
Cbpcalibrator.calibaccess.table              = calparameters.tab
Cbpcalibrator.calibaccess.table.maxant       = 36
Cbpcalibrator.calibaccess.table.maxchan      = 16000
Cbpcalibrator.calibaccess.table.maxbeam      = 36

Cbpcalibrator.sources.names                  = [src1]
Cbpcalibrator.sources.src1.components        = [cal]
Cbpcalibrator.sources.cal.calibrator         = 1934-638

Cbpcalibrator.ncycles                        = 25

Cbpcalibrator.gridder                        = SphFunc
```

Ccalibrator

- Calibration task.

$ <MPI wrapper> ccalibrator -c config.in

```
Ccalibrator.dataset                          = calibration_data.ms
Ccalibrator.refantenna                       = 0
calibrator.sources.names                     = [field1]
Ccalibrator.sources.field1.direction         = [12h30m00.000, -45.00.00.000, J2000]
Ccalibrator.sources.field1.components        = [src1]
Ccalibrator.sources.src1.flux.i              = 0.091
# "ra" and "dec" are actually l and m direction cosine offsets in radians relative to "direction"
Ccalibrator.sources.src1.direction.ra        = 0.00363277
Ccalibrator.sources.src1.direction.dec       = -0.00366022

Ccalibrator.ncycles                          = 25

# not building visibilities using a gridder, so specify a simple one here
Ccalibrator.gridder                          = SphFunc
```

Ccalibrator

- Calibration task.

$ <MPI wrapper> ccalibrator -c config.in

```
Ccalibrator.dataset                                   = calibration_data.ms
Ccalibrator.refantenna                                = 0
Ccalibrator.sources.names                             = [10uJy]
Ccalibrator.sources.10uJy.direction                   = [12h30m00.000, -45.00.00.000, J2000]
Ccalibrator.sources.10uJy.model                       = 10uJy.model.small

Ccalibrator.ncycles                                    = 25

# use wide-field gridder & A-projection to convolve w terms and the primary beam into the visibility model
Ccalibrator.gridder                                   = AProjectWStack
Ccalibrator.gridder.AProjectWStack.wmax               = 15000
Ccalibrator.gridder.AProjectWStack.nwplanes           = 100
Ccalibrator.gridder.AProjectWStack.oversample         = 4
Ccalibrator.gridder.AProjectWStack.diameter           = 12m
Ccalibrator.gridder.AProjectWStack.blockage           = 2m
Ccalibrator.gridder.AProjectWStack.maxfeeds           = 2
Ccalibrator.gridder.AProjectWStack.maxsupport         = 1024
Ccalibrator.gridder.AProjectWStack.frequencydependent  = false
```

Ccalapply

- Apply calibration solutions task.

ccalapply -c config.in

<or>

$ <MPI wrapper> ccalapply -c config.in

```
Ccalapply.dataset                  = science_data.ms

Ccalapply.calibaccess              = table
Ccalapply.calibaccess.table        = calparameters.tab
```

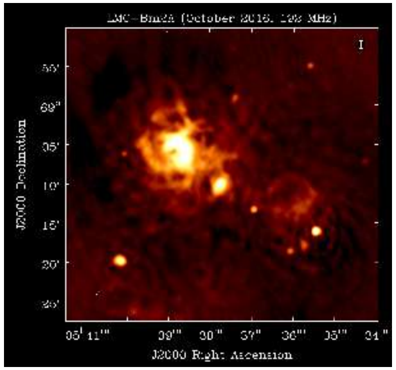
Cimager

- Continuum imaging task.
- Various parallelism options available, but *Imager* is more flexible.

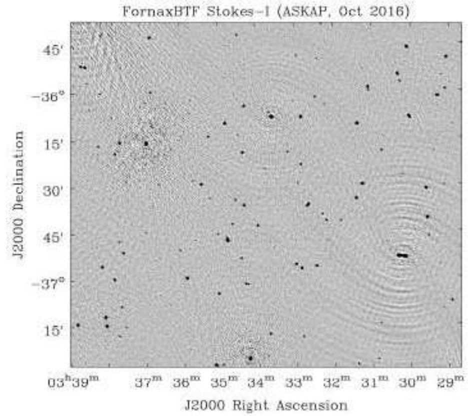
$ <MPI wrapper> cimager -c config.in

```
Cimager.dataset                              = cont_vis.ms
Cimager.Images.Names                         = [image.name] # must start with "image."
Cimager.Images.shape                         = [2048,2048]
Cimager.Images.cellsize                      = [6.0arcsec, 6.0arcsec]

Cimager.solver                               = Clean
Cimager.solver.Clean.algorithm               = BasisfunctionMFS # no Taylor terms here, so just the MS part
Cimager.solver.Clean.scales                  = [0,4,8,16]
Cimager.solver.Clean.niter                   = 1000
Cimager.solver.Clean.gain                    = 0.1
Cimager.threshold.minorcycle                 = [0.3mJy, 10%]
Cimager.threshold.majorcycle                 = 0.3mJy
Cimager.ncycles                              = 10

Cimager.restore                              = True
Cimager.restore.beam                         = fit

Cimager.preconditioner.Names                 = [Wiener]
Cimager.preconditioner.preservecf            = true
Cimager.preconditioner.Wiener.robustness     = -2
```

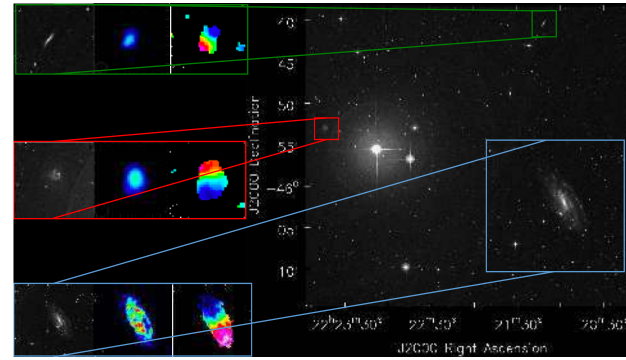
ASKAPsoft Examples



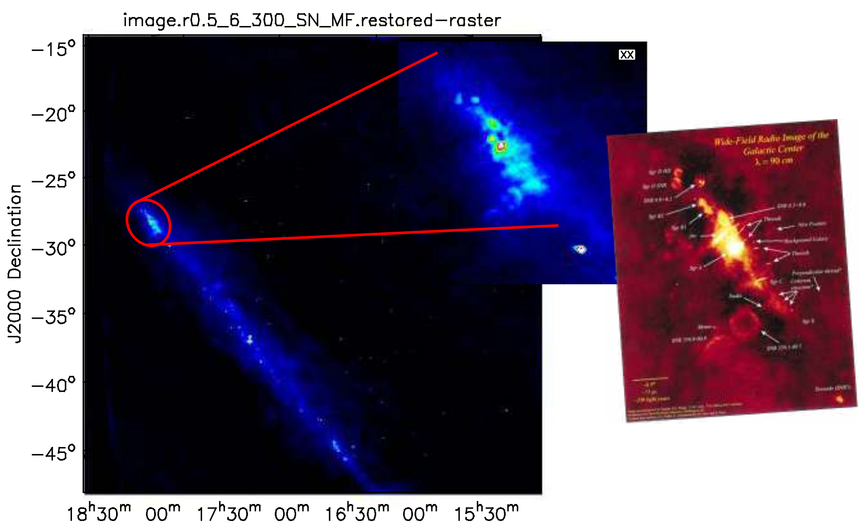
Continuum Imaging
Courtesy of Wasim Raja



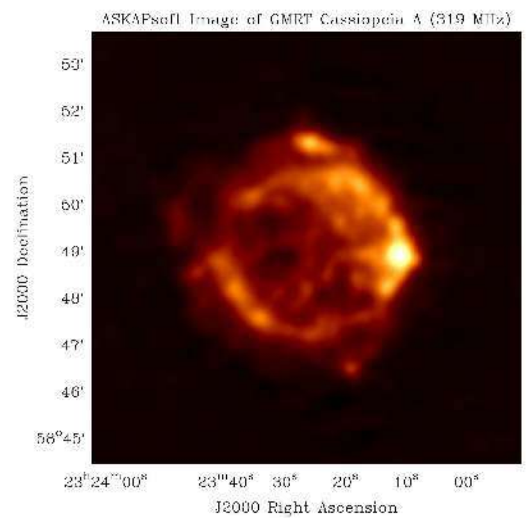
Polarised Imaging
Courtesy of Wasim Raja & Craig Anderson



Spectral Line Imaging
Courtesy of Dane Kleiner



MWA Imaging
Courtesy of Steve Ord & Daniel Mitchell



GMRT imaging
Courtesy of Wasim Raja

Linmos

- Linear Mosaicking task.
- Example 1

$ linmos -c config.in <or>

$ <MPI wrapper> linmos-mpi -c config.in

```
linmos.weighttype      = FromWeightImages
linmos.weightstate     = Inherent # expect for non-A gridders. Default “Corrected” is appropriate for A gridders.

linmos.names           = [image_feed00..35.restored]
linmos.weights         = [weights_feed00..35]

linmos.outname         = image_mosaic.restored
linmos.outweight       = weights_mosaic
```

Linmos

- Linear Mosaicking task.
- Example 2

$ linmos -c config.in <or>

$ <MPI wrapper> linmos-mpi -c config.in

```
linmos.weighttype      = FromPrimaryBeamModel

linmos.names           = [image_feed14..15.i.dirty.restored, image_feed20..21.i.dirty.restored]

linmos.outname         = image_mosaic.i.dirty.restored
linmos.outweight       = weights_mosaic.i.dirty

linmos.feeds.centre    = [12h30m00.00, -45.00.00.00]

# The following can be moved to a separate file
linmos.feeds.spacing   = 1deg
linmos.feeds.image_feed14.i.dirty.restored = [-0.5, -0.5]
linmos.feeds.image_feed15.i.dirty.restored = [-0.5,  0.5]
linmos.feeds.image_feed20.i.dirty.restored = [0.5, -0.5]
linmos.feeds.image_feed21.i.dirty.restored = [0.5,  0.5]
```

Selavy

- Source finding task.

See https://www.atnf.csiro.au/computing/software/askapsoft/sdp/docs/current/analysis/index.html

A walk through the tutorial

ASKAPsoft Imaging Tutorial

- Courtesy of (and many thanks to) Wasim Raja
- Wasim has prepared four scripts to:
  - Generate input slurm scripts, parsets and associated files
  - Launch jobs on Galaxy
- The scripts are:
  1. bandpass calibration: **do_cal_1934.sh**
  2. prepare science data: **do_pre_process_ras.sh**
  3. image/selfcal science data (continuum only): **do_selfcal_ras.sh**
  4. form linear mosaic: **do_linmos_ras.sh**
- Also:
  - a script to set up galaxy modules: **setup_modules_on_nodes.sh**
  - a file to configure various parameters: **process_ASKAPdata.config**

Tutorial — Getting Started

```
$ mkdir askap_tutorial
$ cd askap_tutorial
$ cp –r /group/askap/dmitchell/askap_tutorial/* .

$ . setup_modules_on_nodes.sh
```

setup_modules_on_nodes.sh

```
module use /group/askap/modulefiles
module unload askapsoft
module load askapsoft/0.22.1

module unload askapdata
module load askapdata

module unload askappipeline
module load askappipeline
#module load askapcli

export PMI_NO_PREINITIALIZE=1
export PMI_NO_FORK=1
export PMI_DEBUG=1

module unload askap-cray
module load askap-cray

module unload slurm
module load slurm
```

Tutorial — Getting Started

```
$ mkdir askap_tutorial
$ cd askap_tutorial
$ cp –r /group/askap/dmitchell/askap_tutorial/* .

$ . setup_modules_on_nodes.sh
$ . process_ASKAPdata.config
```

process_ASKAPdata.config

```
export TRIAL=0                                  # set to 1 to generate files but not run them

export SPLIT_CHAN=1                             # split out a subset of frequency channels
export BCHAN_SPLIT=8192
export ECHAN_SPLIT=8407 #9271

export MY_SBID_BPCAL=5181                       #  scheduling block for band-pass calibration (i.e. the id of the BP calibration
                                                # observation)
export MY_SBID_TARGET=5177                      #  scheduling block for science data (i.e. the id of the science observation)
export MY_FIELD_NAME=COSMOLOGY_T15-2            # name of the science field
export PATH_TO_SETUP_FILE=$PWD                  # change me if running from a different directory
export MY_OUTPATH=ras_data_processing_${this_user}/
mkdir -p ${MY_OUTPATH}msdata/${MY_SBID_TARGET} ${MY_OUTPATH}bpcal_solutions/${MY_SBID_BPCAL}

# Decide which beams you wish to process. Do bandpass calibration for all 36 beams, but restrict imaging and selfcal to 1 or a few
export BBEAM_BPCAL=0 # Must be 0 with the current structure of bptables
export EBEAM_BPCAL=35 # Can be less than maxBeams
export BBEAM=0 # image / selfcal beams 0 to 1
export EBEAM=1

# Some imaging parameters:
export ROBUST=-0.5
export BLOOP_SELFCAL=0
export ELOOP_SELFCAL=1
```

do_cal_1934.sh

$ ./do_cal_1934.sh

- mssplit — select a subset of channels (to limit the amount of processing)
- cflag — look for radio frequency interference and set flags
- cbpcalibrator — run the calibrator for each frequency channel

- Or just copy the solution table that I generated:

```
$ . process_ASKAPdata.config
$ mv cbpcal_1934_sb5181_bm0-bm35_refant-1_bp.tab \
  $MY_OUTPATH/bpcal_solutions/5181/
```

plot_bandpass.py

$ ssh –X username@galaxy.pawsey.org.au
- or:

$ ssh –Y username@galaxy.pawsey.org.au
- For the help menu:

$ plot_bandpass.py –h

$ optional arguments:

-t BP_TAB, --t BP_TAB Input Bandpass table (with path)

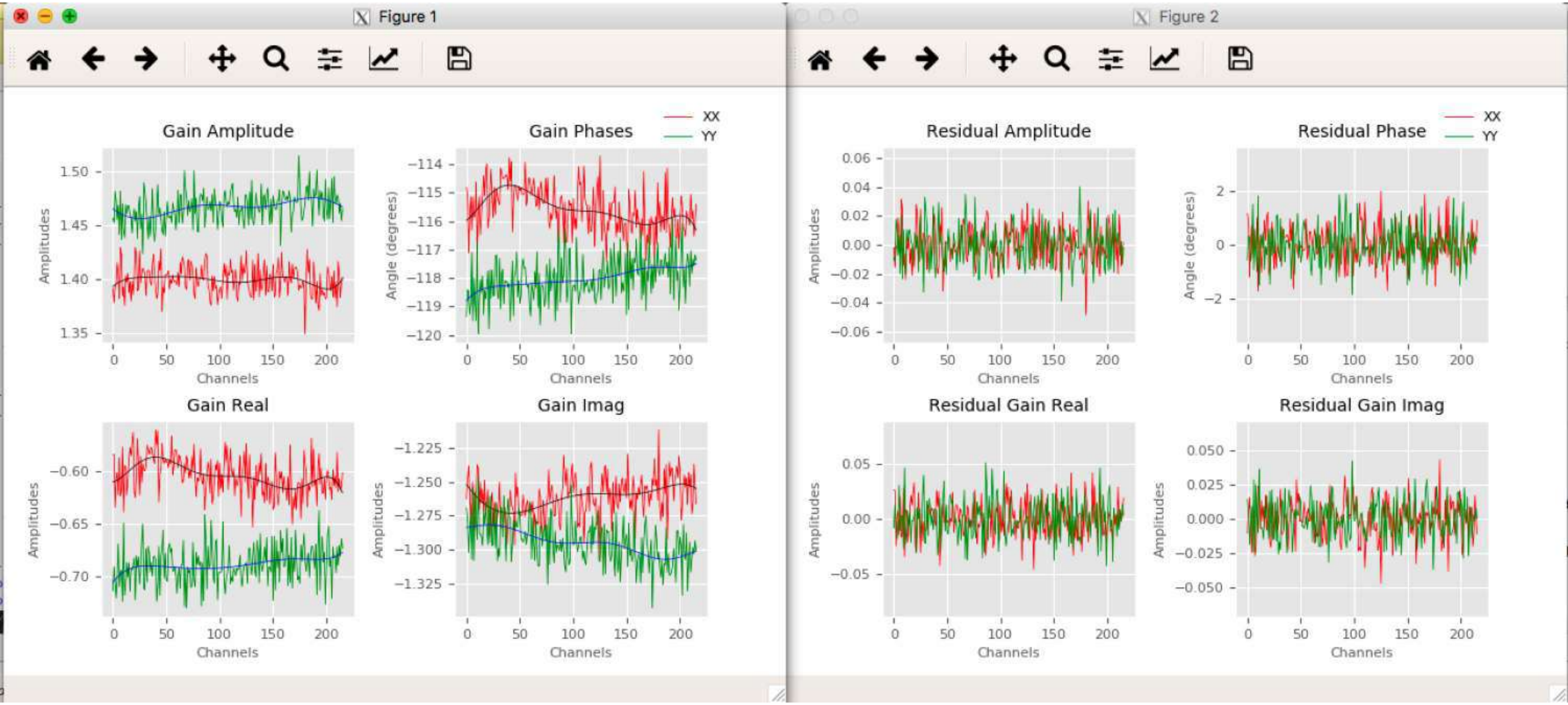
-ib BEAM_NUM, --ib BEAM_NUM The beam number you wish to process

-ia ANTE_NUM, --ia ANTE_NUM The antenna number you wish to process

$ plot_bandpass.py -t cbpcal_1934_sb5181_bm0-bm35_refant-1_bp.tab –ia 1

```
Successful readonly open of default-locked table cbpcal_1934_sb5181_bm0-bm35_refant-1_bp.tab: 3 columns, 1 rows
Plotting bandpass solutions for Input table: cbpcal_1934_sb5181_bm0-bm35_refant-1_bp.tab
For:
             Beam Num:  0
             Ante Num:  1
Smooth fits will be derived using:
           Poly Order:  2
           Harm Order:  3
   Fits will done in :  1  windows
 Residual plot bounds:  +/- 5.0 sigma
   Taper Width (npts):  15
 Niter (for F-interp):  100
    Reference Antenna:  1
```

plot_bandpass.py (-ia 0)

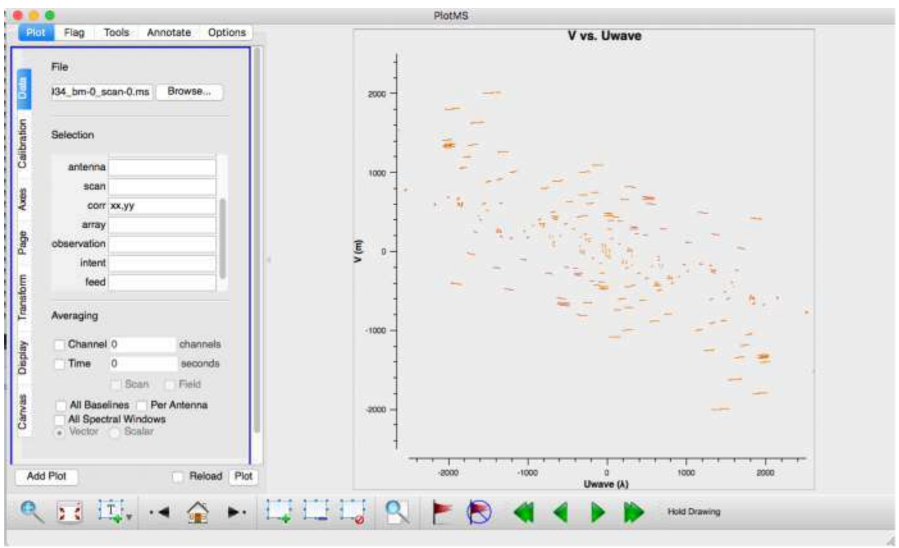


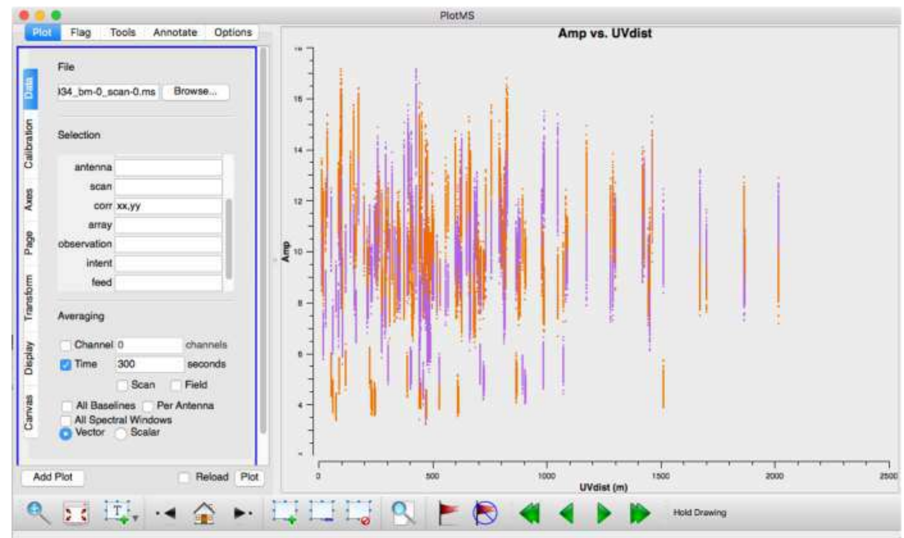
CASA tools

On local machine (replace $MY_OUTPATH with full directory path):

```
$  scp -r username@hpc-data.pawsey.org.au:$MY_OUTPATH/msdata/5181/FLAGGED_DYNAMIC/1934_bm-0_scan-0.ms .
```

```
$  casabrowser 1934_bm-0_scan-0.ms
$  casaplotms
```





do_pre_process_ras.sh

$ ./do_pre_process_ras.sh

- mssplit — select the same subset of channels from the science dataset
- ccalapply — apply calibration solutions to the science data
- cflag — look for radio frequency interference and set flags
- mssplit — average in frequency
- cflag — a final round of flagging

do_selfcal_ras.sh

$ ./do_selfcal_ras.sh

- ccalibrator — run calibration using a model of this field
- cimager — image and deconvolve the field with the new calibration solutions
- selavy — run relatively shallow source finder on the restored image
- cmodel — generate a model image from the selavy catalogue

- 1st run: set BLOOP_SELFCAL=0 & ELOOP_SELFCAL=0: imaging with no selfcal

$ squeue -u username

JOBID	USER	ACCOUNT	NAME	EXEC_HOST	ST	REASON	START_TIME	END_TIME	TIME_LEFT	NODES	PRIORITY
5055128	dmitchel	askaprt	IMG-5177-0A.I	nid00217	R	None	08:36:54	14:36:54	5:56:54	1	10001
5055129	dmitchel	askaprt	IMG-5177-1A.I	nid00299	R	None	08:36:54	14:36:54	5:56:54	1	10001

do_selfcal_ras.sh

```
$ ls -ld ${MY_OUTPATH}/image/5177/weight*
  image/5177/weights.I.COSMOLOGY_T15-2A_bm-0_iter-0
  image/5177/weights.I.COSMOLOGY_T15-2A_bm-1_iter-0
$ ls -ld ${MY_OUTPATH}/image/5177/image*restored
  image/5177/image.I.COSMOLOGY_T15-2A_bm-0_iter-0.restored
  image/5177/image.I.COSMOLOGY_T15-2A_bm-1_iter-0.restored
$ ls -ld ${MY_OUTPATH}/image/5177/image*restored.cmodel
  image/5177/image.I.COSMOLOGY_T15-2A_bm-0_iter-0.restored.cmodel
  image/5177/image.I.COSMOLOGY_T15-2A_bm-1_iter-0.restored.cmodel
$ ls -ld ${MY_OUTPATH}/image/5177/psf*
  image/5177/psf.I.COSMOLOGY_T15-2A_bm-0_iter-0
  image/5177/psf.I.COSMOLOGY_T15-2A_bm-1_iter-0
  image/5177/psf.image.I.COSMOLOGY_T15-2A_bm-0_iter-0
  image/5177/psf.image.I.COSMOLOGY_T15-2A_bm-1_iter-0
```

do_linmos_ras.sh

$ ./do_linmos_ras.sh

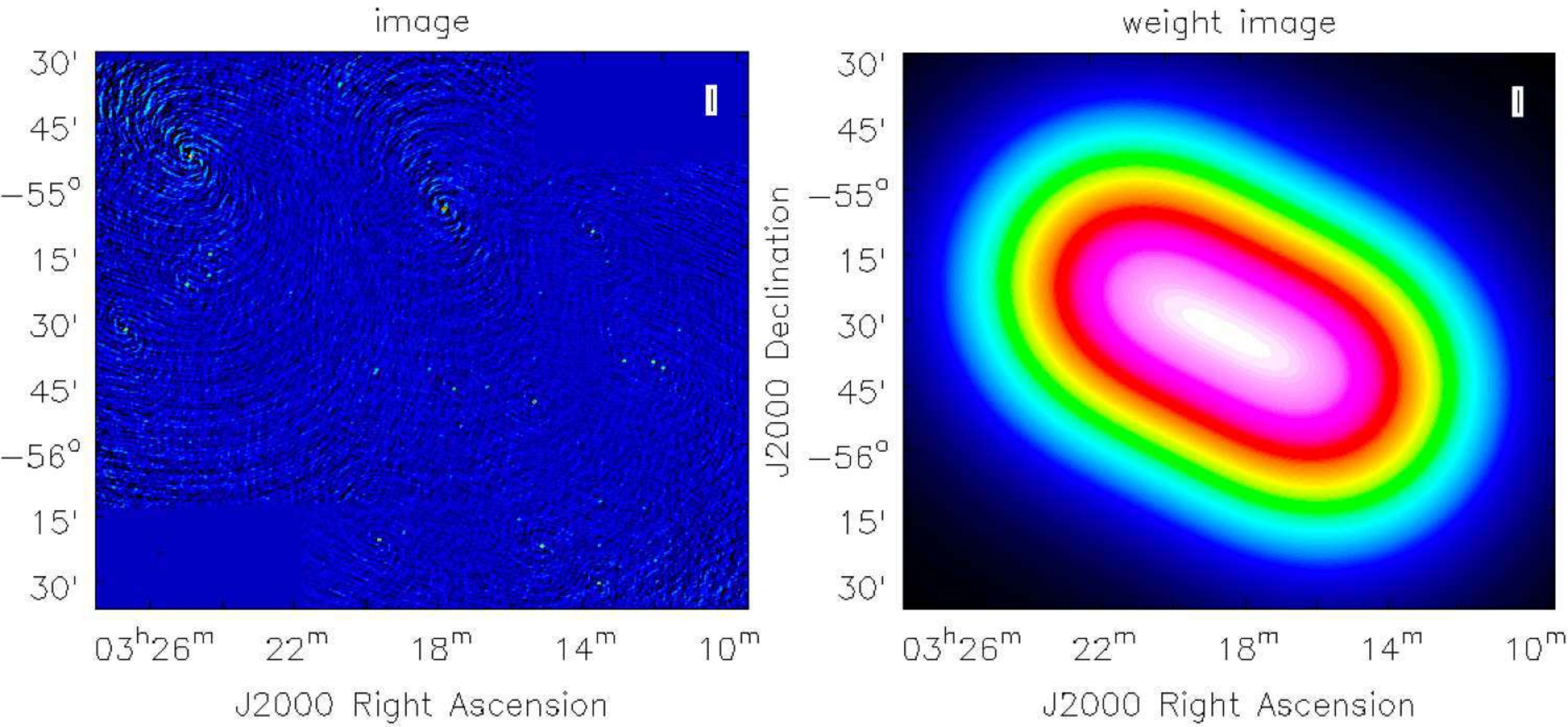
- linmos — form a linear mosaic of the final images

On local machine (replace $MY_OUTPATH with full directory path):

```
$ scp -r username@hpc-data.pawsey.org.au:$MY_OUTPATH/linmos/5177/\* .
```

```
$ casaviewer image.I.COSMOLOGY_T15-2iter-0.linmosRAS_5177
```

casaviewer image.I.COSMOLOGY_T15-2iter-0.linmosRAS_5177



One loop of self-cal

- 2nd run: set BLOOP_SELFCAL=1 & ELOOP_SELFCAL=1: imaging with a selfcal update

```
$ . process_ASKAPdata.config
$ ./do_selfcal_ras.sh
$ ./do_linmos_ras.sh

$ ls -l $MY_OUTPATH/linmos/5177/

$ scp -r username@hpc-data.pawsey.org.au:$MY_OUTPATH/linmos/5177/\*iter-1\* .

$ casaviewer image.I.COSMOLOGY_T15-2iter-1.linmosRAS_5177
```

Tutorial

- Run scripts, but also look at what they are doing

$ grep srun do_*

do_cal_1934.sh:

```
srun --export=ALL --ntasks=1 --ntasks-per-node=1 mssplit -c ${parset_split} > \${log}
srun --export=ALL --ntasks=1 --ntasks-per-node=1 cflag -c ${parset_flag} > \${log}
srun --export=ALL --ntasks=$nranks_bpcal --ntasks-per-node=20 $cbpcalibrator -c ${parset_bpcal} > \${log}
```

do_pre_process_ras.sh:

```
srun --export=ALL --ntasks=1 --ntasks-per-node=1 mssplit -c ${parset_split} > \${log}
srun --export=ALL --ntasks=19 --ntasks-per-node=19 ccalapply -c ${parset_ccalapply} > \${log}
srun --export=ALL --ntasks=1 --ntasks-per-node=1 cflag -c ${parset_flag} > \${log}
srun --export=ALL --ntasks=1 --ntasks-per-node=1 mssplit -c ${parset_average} > \${log}
srun --export=ALL --ntasks=1 --ntasks-per-node=1 cflag -c ${parset_flag2} > \${log}
```

do_selfcal_ras.sh:

```
srun --export=ALL --ntasks=$nranks_ccal --ntasks-per-node=$nppn_ccal $calib -c ${parset_ccal} > \${log}
srun --export=ALL --ntasks=$nranks_cimage --ntasks-per-node=$nppn_cimage $imager -c ${parset_cimage} > \${log}
srun --export=ALL --ntasks=$nranks_selavy --ntasks-per-node=$nppn_selavy $selavy -c ${parset_selavy} > \${log}
srun --export=ALL --ntasks=$nranks_cmodel --ntasks-per-node=$nppn_cmodel $cmodel -c ${parset_cmodel} > \${log}
```

do_linmos_ras.sh:

```
srun --export=ALL --ntasks=$nranks_linmos --ntasks-per-node=1 $linmos -c ${parset_linmos} > \${log}
```

Tutorial

```
$  ls -1 *.sbatch
IMG_COSMOLOGY_T15-2A_bm-0.I.sbatch
IMG_COSMOLOGY_T15-2A_bm-1.I.sbatch
cbpcal_1934_bm0-bm35.sbatch
linmos_COSMOLOGY_T15-2iter-0.sbatch
linmos_COSMOLOGY_T15-2iter-1.sbatch
preimag_1934_bm-0_scan-0.sbatch
preimag_1934_bm-10_scan-10.sbatch
preimag_1934_bm-11_scan-11.sbatch
preimag_1934_bm-12_scan-12.sbatch
…
prep_COSMOLOGY_T15-2A_bm-0.sbatch
prep_COSMOLOGY_T15-2A_bm-1.sbatch
```

- e.g.:

```
$  sbatch linmos_COSMOLOGY_T15-2iter-0.sbatch
```