Imaging ALMA data

Spectral line imaging & basic analysis

Dan Walker

UK ALMA Workshop 2025
The University of Manchester

EUROPEAN ARC
ALMA Regional Centre || UK

MANCHESTER
1824

Spectral line imaging

- The process of cleaning line data is broadly similar to cleaning the continuum, but there are some extra steps and parameters:
  - We now have a third dimension (frequency/velocity) made up of channels, the spacing of which is related to the spectral resolution of the data
  - The emission changes from channel-to-channel, which makes cleaning and masking more complex
  - More data → longer processing time
  - The dust continuum level must be subtracted to ensure that we are imaging only the line emission

Continuum subtraction

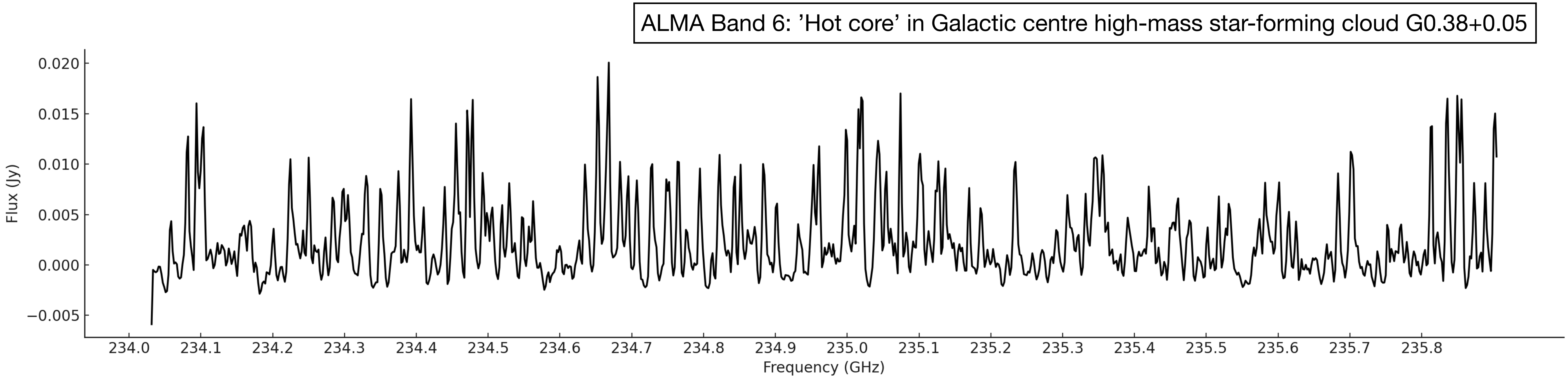
A note on continuum subtraction

- The ALMA pipeline can struggle in cases such as
  - Broad lines that fill the spectral window
  - Extremely line-rich spectra (e.g. ‘hot cores’)
- In these cases there is little true continuum in the spectrum, and a different approach is needed
- Alternative tools that handle difficult spectra
  - STATCONT
  - Lumberjack

Continuum subtraction

A note on continuum subtraction

- The ALMA pipeline can struggle in cases such as
    - Broad lines that fill the spectral window
    - Extremely line-rich spectra (e.g. ‘hot cores’)
- In these cases there is little true continuum in the spectrum, and a different approach is needed



Continuum subtraction

- In our example data, the line emission is quite simple, so we can manually identify the continuum channels
- Continuum subtraction can be done after imaging, but it's generally recommended to do this beforehand (if feasible)
- Use CASA task `uvcontsub` to subtract the continuum from the *uv* data
    - The contchans parameter is the same as what we used yesterday for imaging the continuum

```
uvcontsub(  vis             = filename + '.target',
            outputvis       = filename + '.target.contsub',
            fitspec         = contchans,
            fitorder        = 0)
```

Cube imaging

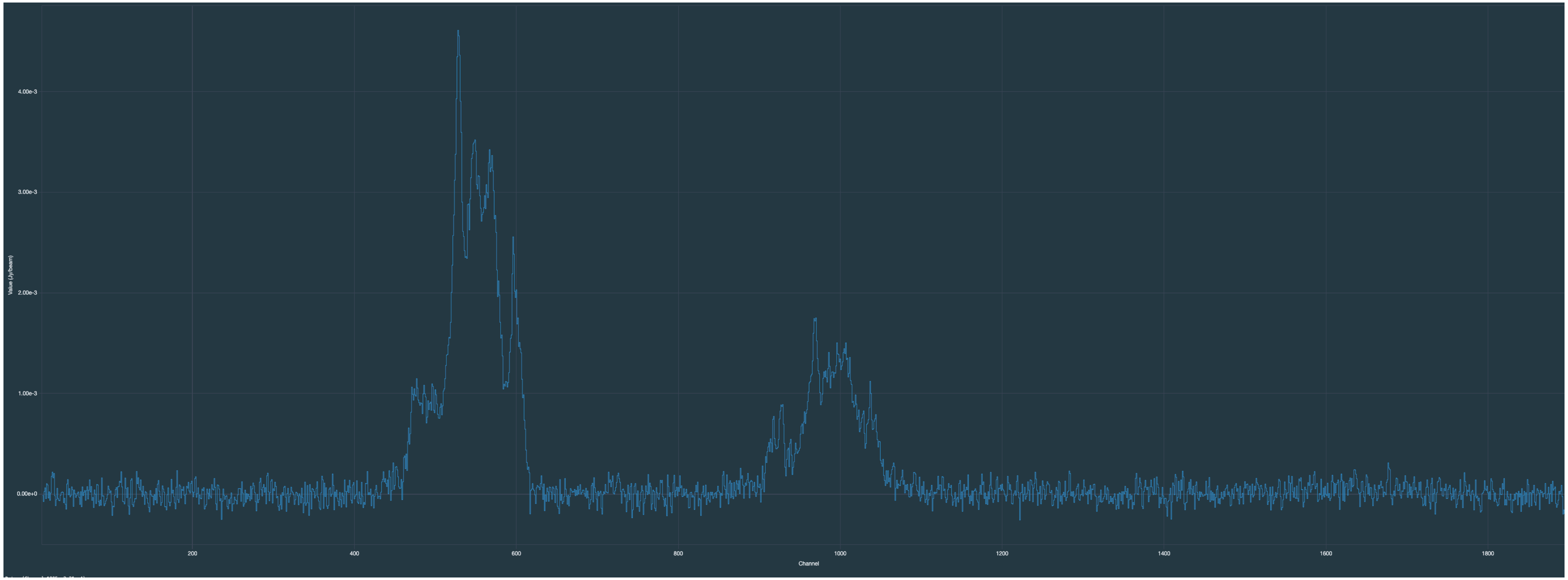
- Let's start by making a dirty image (0 clean iterations), just like we did for the continuum
- We will start by looking only at SPW 0

```
tclean(  vis            = filename + '.target.contsub',
         imagename      = 'PN_Hb_5.cube.dirty',
         spw            = '0',
         specmode       = 'cube',
         imsize         = [320, 320],
         cell           = '0.22arcsec',
         deconvolver    = 'hogbom',
         niter          = 0,
         weighting      = 'briggsbwtaper',
         robust         = 0.5,
         gridder        = 'mosaic',
         interactive    = False)
```

Cube imaging

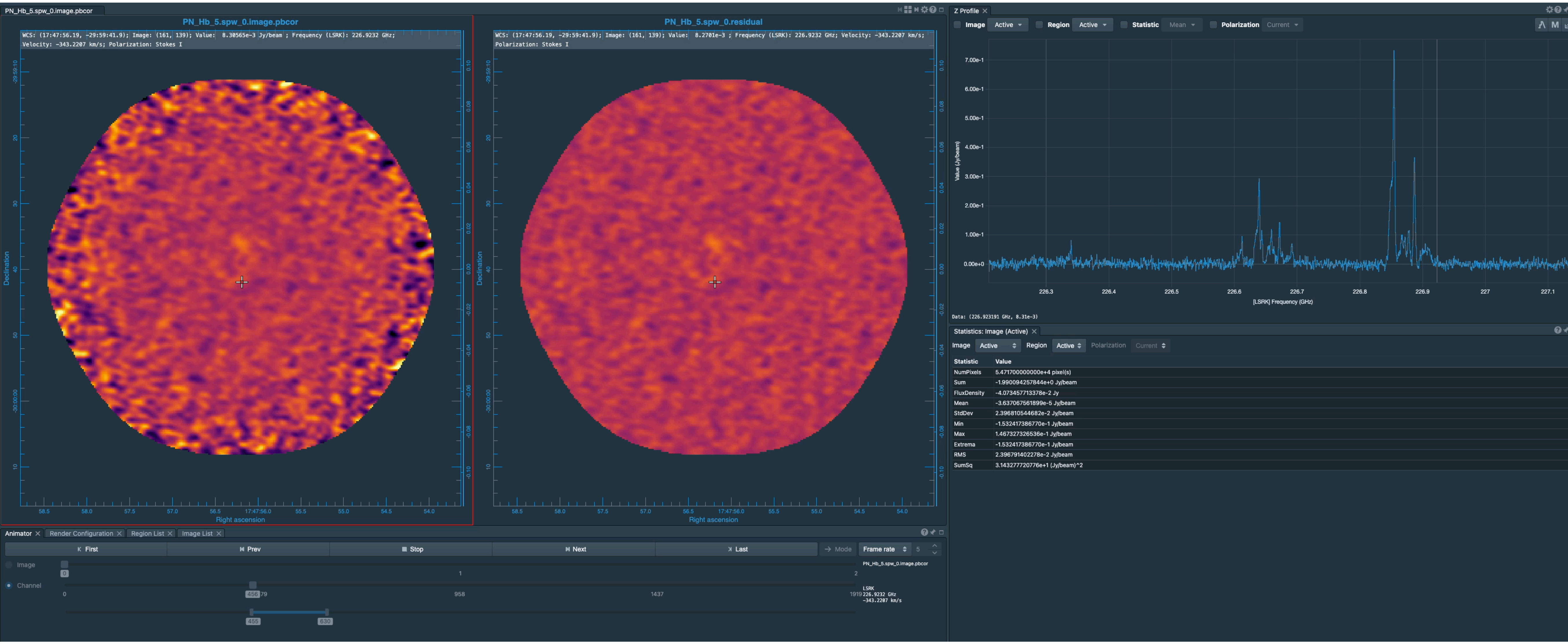
- There are two main lines and the rest is blank, so we can restrict the imaging to just the relevant channel/frequency ranges

- Once again, we want to increase the number of clean iterations, add a cleaning threshold, and add auto masking parameters …



Cube imaging

- There are two main lines and the rest is blank, so we can restrict the imaging to just the relevant channel/frequency ranges
  - Update `start`, `width`, and `nchan` parameters. E.g. if you want to image channels 300 - 400, use:
  - `start = 300`, `width = 1`, `nchan = 101`
- Once again, we also want to increase the number of clean iterations, add a cleaning threshold, and add auto masking parameters …
  - Update `imagename, threshold, niter, usemask`



PN_Hb_5.spw_0.image.pbcor
PN_Hb_5.spw_0.residual
Right ascension
Declination
Z Profile
Image
Region
Statistic
Polarization
Value (Jy/beam)
[LSRK] Frequency (GHz)
Statistics: Image (Active)
Statistic
Value
NumPixels
5.471700000000e+4 pixel(s)
Sum
-1.990094257844e+0 Jy/beam
FluxDensity
-4.073457713378e-2 Jy
Mean
-3.637067561899e-5 Jy/beam
StdDev
2.396810544682e-2 Jy/beam
Min
-1.532417386770e-1 Jy/beam
Max
1.467327326536e-1 Jy/beam
Extrema
-1.532417386770e-1 Jy/beam
RMS
2.396791402278e-2 Jy/beam
SumSq
3.143277720776e+1 (Jy/beam)^2
Animator
Render Configuration
Region List
Image List
First
Prev
Stop
Next
Last
Mode
Frame rate
Channel

Cube analysis

- Once you've imaged your data, it's time for analysis!
- Many tools available for image analysis. What you use will depend on goals & personal preference.
- We will introduce a few tools for basic image analysis in CASA, and some non-CASA Python packages to get
  - Image statistics
  - Moment maps (more on this later)
  - Extracting and fitting spectra
  - Position-velocity maps

followed by a hands-on session to try some of this yourself

Image statistics

CASA implementation

- Use task `imstat` to get statistics of an image, which are returned as a dictionary
  - Compute stats such as rms, peak, min/max, flux, etc.
  - Usage example

```
stats =  imstat(imagename = ‘image’,
                region = ‘region.crtf’,
                chans = ‘100~200’)
stats[‘rms’] # Will print measured RMS
```

Moment maps

See documentation for full definitions

```
immoments(imagename      = 'PN_Hb_5.spw_0.image',
          moments        = [0, 1, 8],
          region         = 'moment_region.crtf',
          chans          = '420~630',
          includepix     = [0.03, 100],
          outfile        = 'PN_Hb_5.spw_0.moment')
```

Integrated intensity

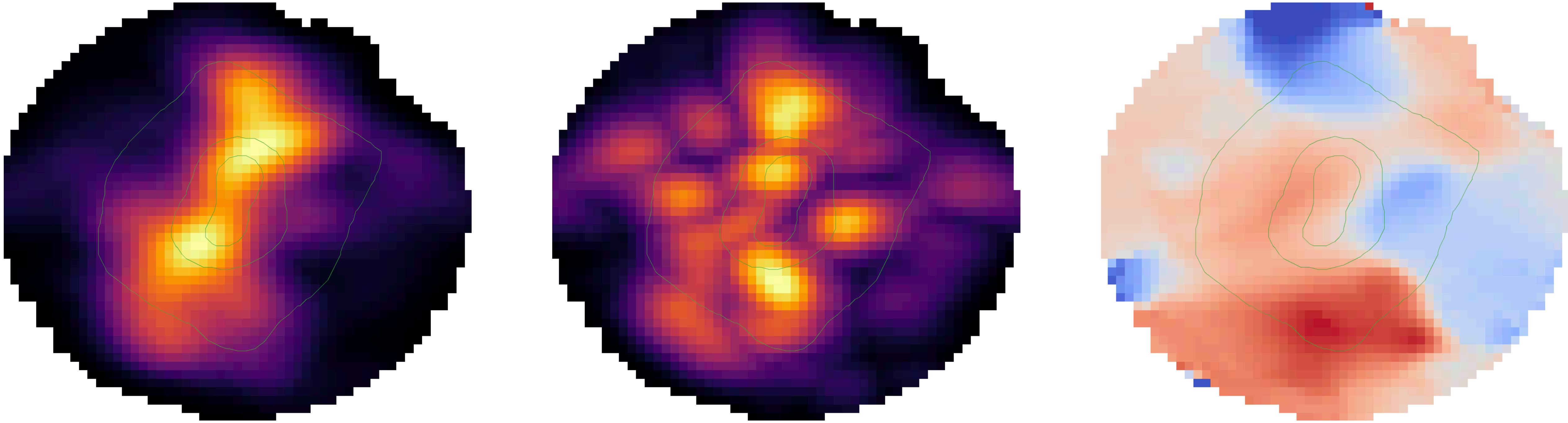
(moment 0)

Maximum intensity

(moment 8)

Velocity field

(moment 1)



Moment maps

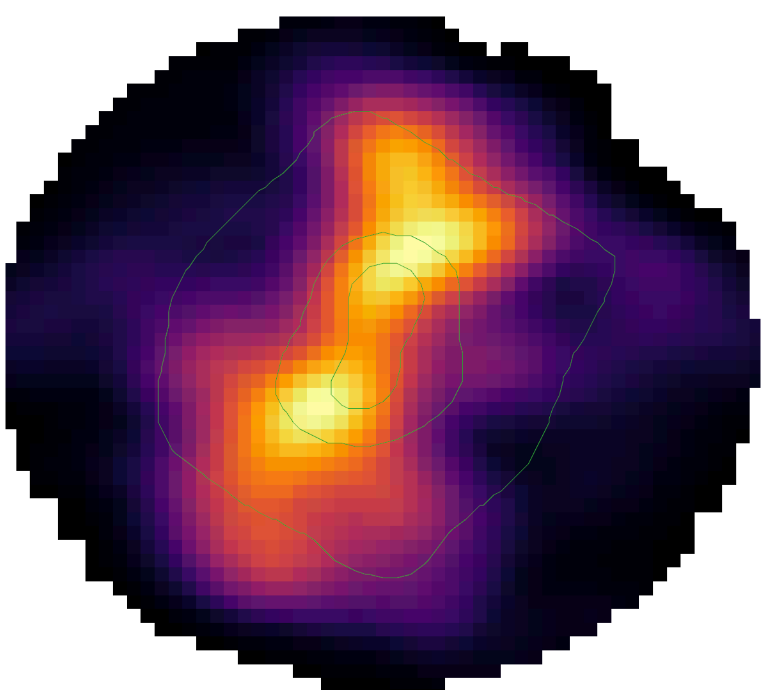
See documentation for full definitions

Try this yourself. You can use CASA, or do it interactively in CARTA

```
immoments(imagename      = 'PN_Hb_5.spw_0.image',
          moments        = [0, 1, 8],
          region         = 'moment_region.crtf',
          chans          = '420~630',
          includepix     = [0.03, 100],
          outfile        = 'PN_Hb_5.spw_0.moment')
```

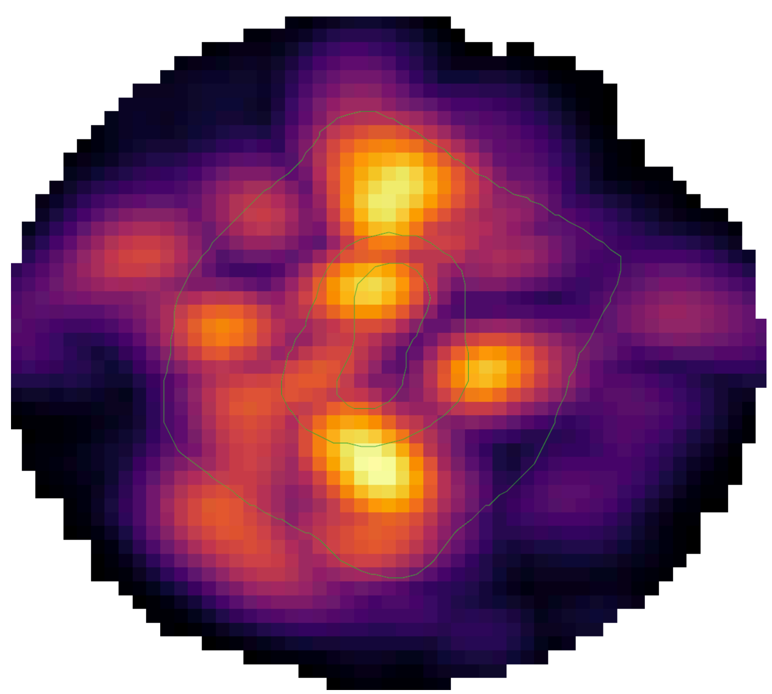
Integrated intensity

(moment 0)



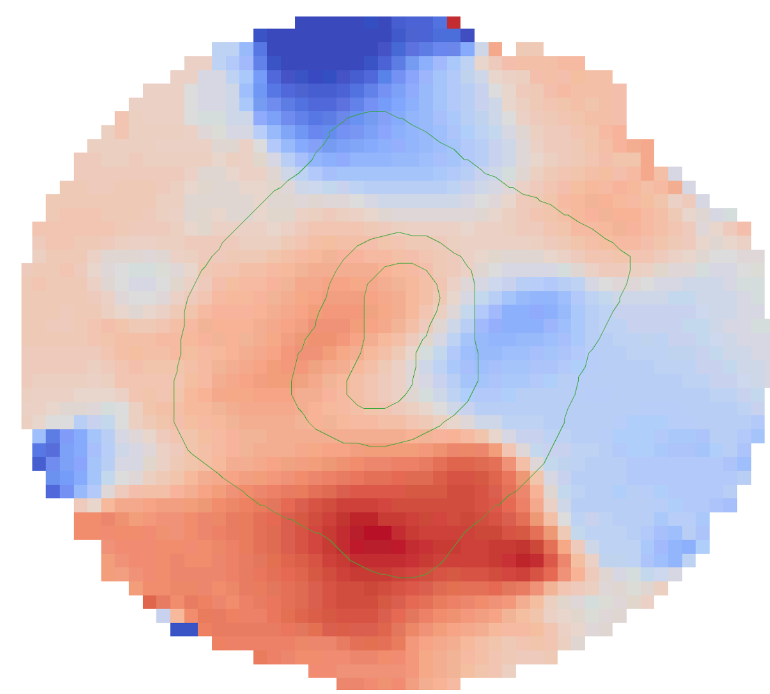
Maximum intensity

(moment 8)



Velocity field

(moment 1)



Moment analysis: caveat

- Moment analysis is widely used and is sufficient in many cases
- But if your source is kinematically complex (e.g. many velocity components), this complexity may be lost and result in poor constraints on velocity and velocity dispersion
- In such cases, full spectral decomposition — fitting spectra in every pixel with one or more components — may be desirable
    - Tools such as: SCOUSEpy and GaussPy+
    - More complicated and time-consuming than moment analysis

Alternative cube analysis tools

Spectral Cube (Python)

- Toolkit for reading, writing, manipulating, and analysing spectral cube data
- Create sub-cubes, moments, extract spectra etc.
- Designed to work with very large cubes that are too large to load into memory

Pyspeckit (Python)

- Analysis toolkit for analysing spectra
- Plotting, line fitting, line modelling, and more

Alternative cube analysis tools

Let's try some basic analyses with and without CASA tools

Please see the analysis script on the meeting webpage

Parallel processing (Linux only)

You can run `tclean` in parallel across multiple cores in order to distribute the processing and speed things up:

- In `tclean`, specify the parameter `parallel=True`
- Place your `tclean` command in a `.py` script
- Run your script via:

Number of cores

```
/path_to_casa/bin/mpicasa -n 8 /path_to_casa/bin/casa --
nologger -c ./imaging_script.py
```

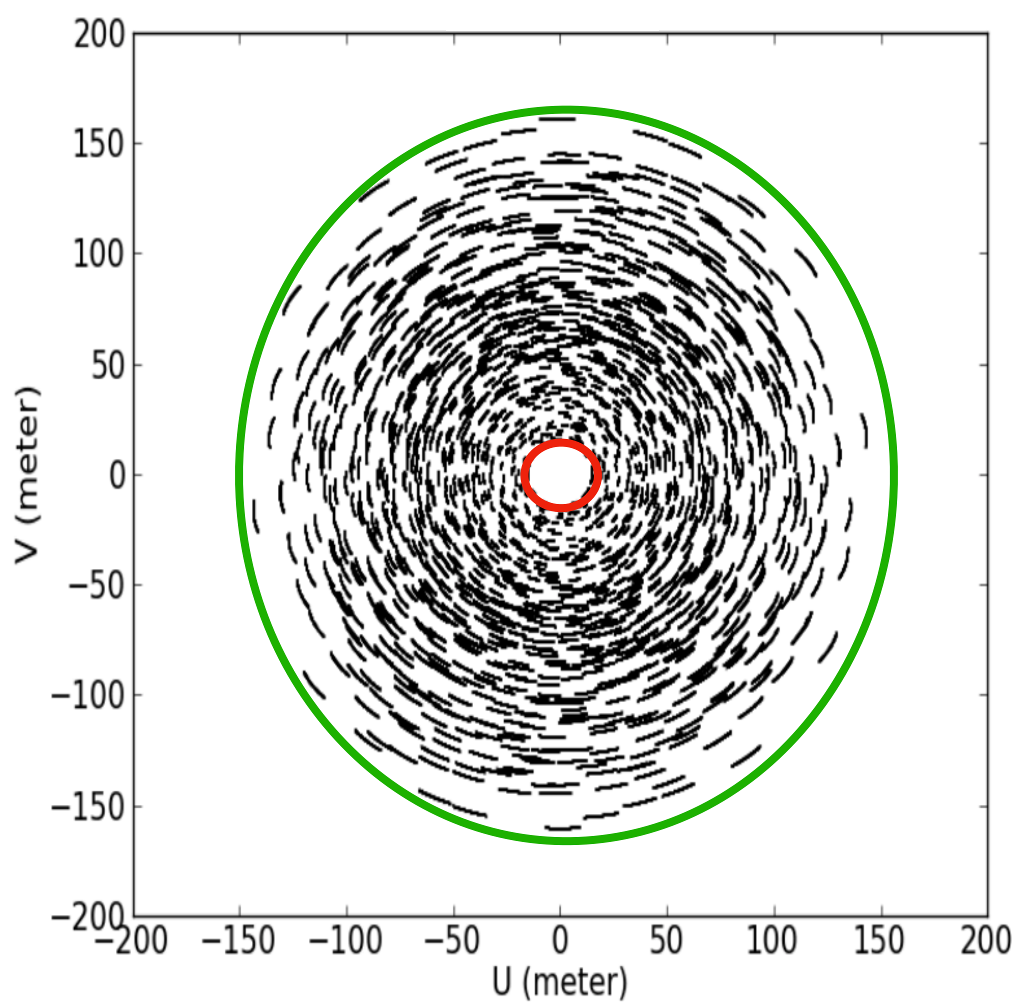
[You can also place the above command into a .sh script and execute it in the background]

A note on array combination

- Interferometer uv coverage is incomplete, which leads to spatial filtering
- If your emission is extended and resolved, you will lose information on certain scales
- ALMA offers main 12m array, 7m ‘compact’ array, and Total Power (single dish) antennas
    - Combining arrays minimises (but does not fully solve) these issues
    - Total Power dishes are for line only (not continuum)
    - Other non-ALMA data can be combined to fill in uv plane

Why combine the data?

- Interferometer *uv* coverage is incomplete, which leads to spatial filtering, flux loss, and image artefacts
- This problem is more pronounced with complex, large scale emission



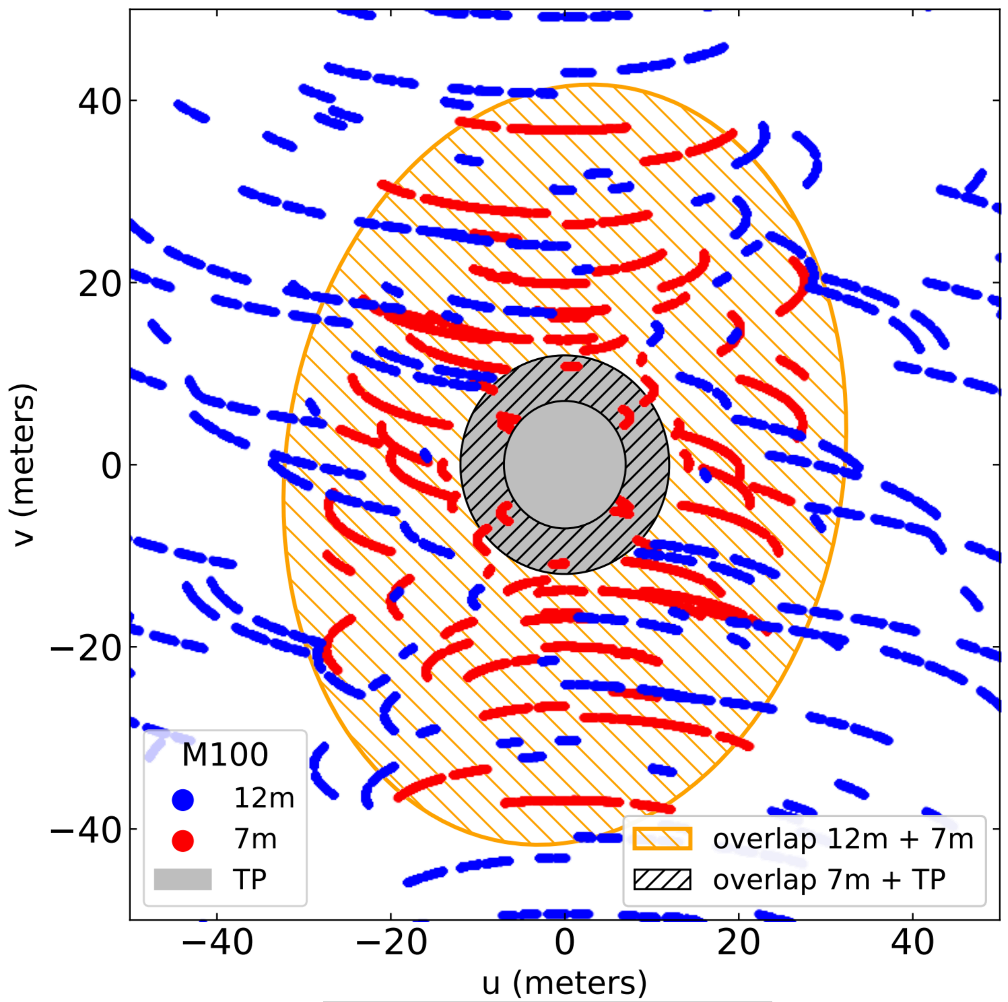
Source: ALMA Technical Handbook

- Angular resolution is related to the longest baselines
- Maximum recoverable scale is related to the shortest baselines
  - This is limited by how close you can physically place antennas
- Note the central hole, sparse coverage, and non-uniform sampling

Why combine the data?

- Interferometer *uv* coverage is incomplete, which leads to spatial filtering, flux loss, and image artefacts

- Arrays may be combined to minimise these issues, and to achieve high angular resolution & sensitivity to larger scale structure



Source: Plunkett+ 2023

- Example ALMA 12m, 7m, and TP overlap in *uv* space
  - Shorter baselines better sampled
  - Central hole now filled
  - Data can be combined to capture emission across a greater range of spatial scales

How to combine the data?

There are two main methods of data combination

- In the **visibility domain** e.g.:
  - 12m data from different array configurations
  - 12m + 7m data
- In the **image domain** e.g.:
  - (12m + 7m) + TP
  - (12m + 7m) + non-ALMA single dish

though some methods use a mix of these (more on this later)

Joint deconvolution

Combination in the visibility domain for interferometric data is relatively straightforward. If you have multiple 12m datasets or 12m + 7m* data, you can either

- Feed all measurement sets (MSs) directly into `tclean` as a list via the `vis` parameter e.g.

```
tclean(vis = ['12m_1.ms', '12m_2.ms', '7m.ms'], …)
```

- Concatenate MSs via the `concat` task, and use this as the input `vis`

In general the former is easier. However, if you have many MSs you may encounter issues due to having too many files open.

*Note that if you are cleaning 12m + 7m data, you must set `gridder='mosaic'` in `tclean`, even for a single pointing (this is related to the different antenna sizes in the arrays)

Single Dish combination

- As noted earlier, Single Dish (SD) data is crucial for filling in the central hole of the *uv* plane
  - This is particularly important when your source is resolved and contains extended emission. SD data recovers these large spatial features, along with the true flux distribution.
  - SD data is by definition non-interferometric — we have *images* not *visibilities*
  - There are several common methods to combine the SD and interferometric data, including:
    - Feathering
    - SDINT (Single Dish INTerferometric) imaging
    - Model-Assisted Clean & Feather (MACF)
    - TP2VIS (Total Power to VISibilities)

Feathering

Feathering is the simplest approach to SD combination, and is very widely used. It is implemented in CASA in the `feather` task, which does the following:

- Takes a high resolution (interferometric) and a low resolution (SD) image
- Takes a Fourier transform of both and combines them
- Transforms the data back into a combined image

The weighting and flux in the SD imaging can be scaled via the `effdishdiam` and `sdfactor` parameters

There are several preparation steps necessary to ensure `feather` will work as expected …

See also the Python package uvcombine for a non-CASA implementation of feathering

Feathering

Before running `feather` you should make sure that your SD image has:

- The same units as your interferometric data (likely Jy/beam)
- The same number and order of axes in the header. If the axis order is different, use task `imtrans` to re-order.
- A well-defined beam in the header (corresponding to the primary beam of the SD data)
- (If cube) The same rest frequency in the header, else use the `imreframe` task
- (If cube) The same spectral grid, else use the `imregrid` task*

*In principle feather does regridding, but this doesn't always work. In this case, regrid prior to feathering.

Feathering

What about feathering two interferometric datasets together?

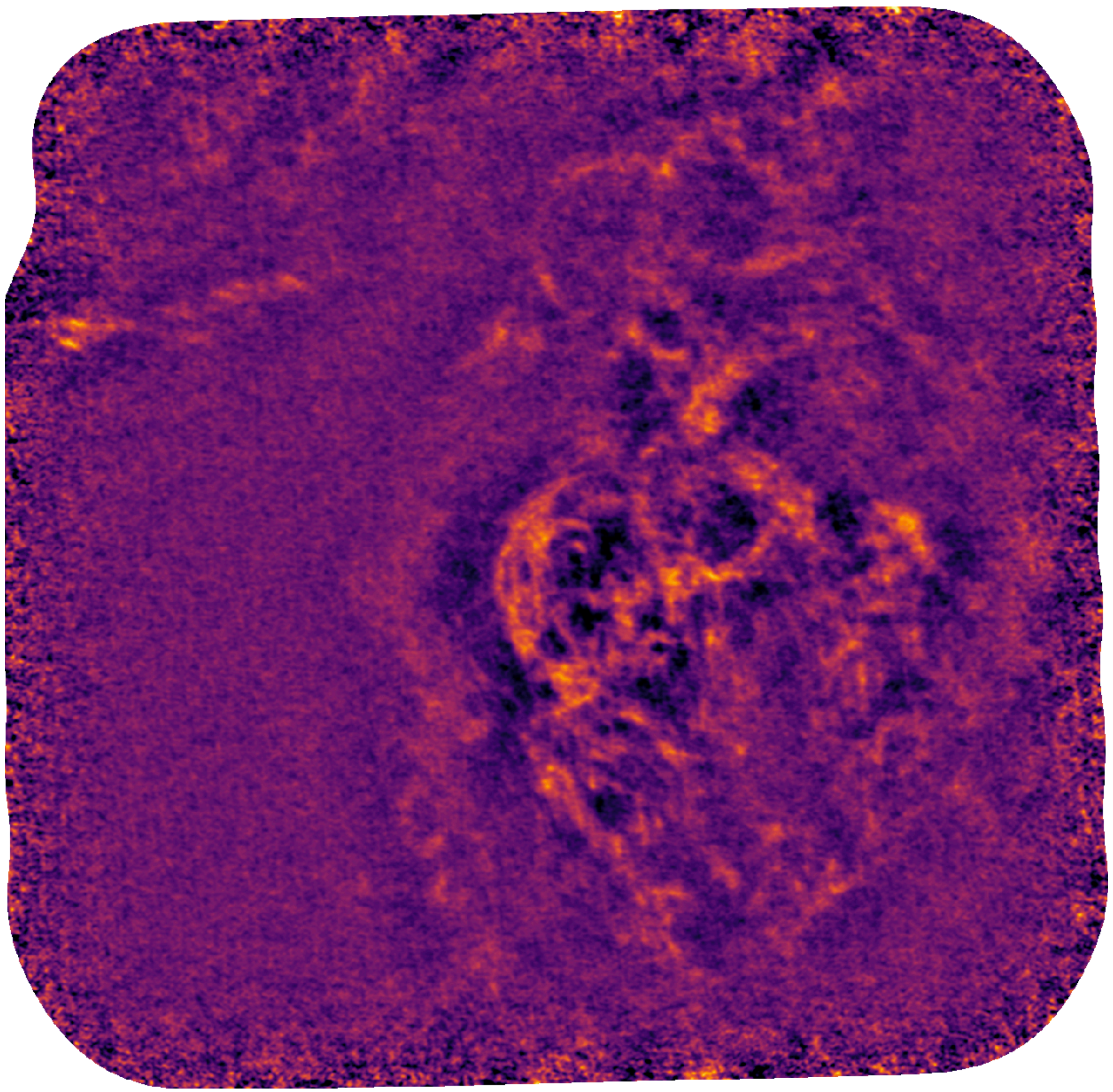
- In principle this does work. You can take e.g. your ALMA 12m and 7m images, and feed them into `feather` as the high and low res images, respectively

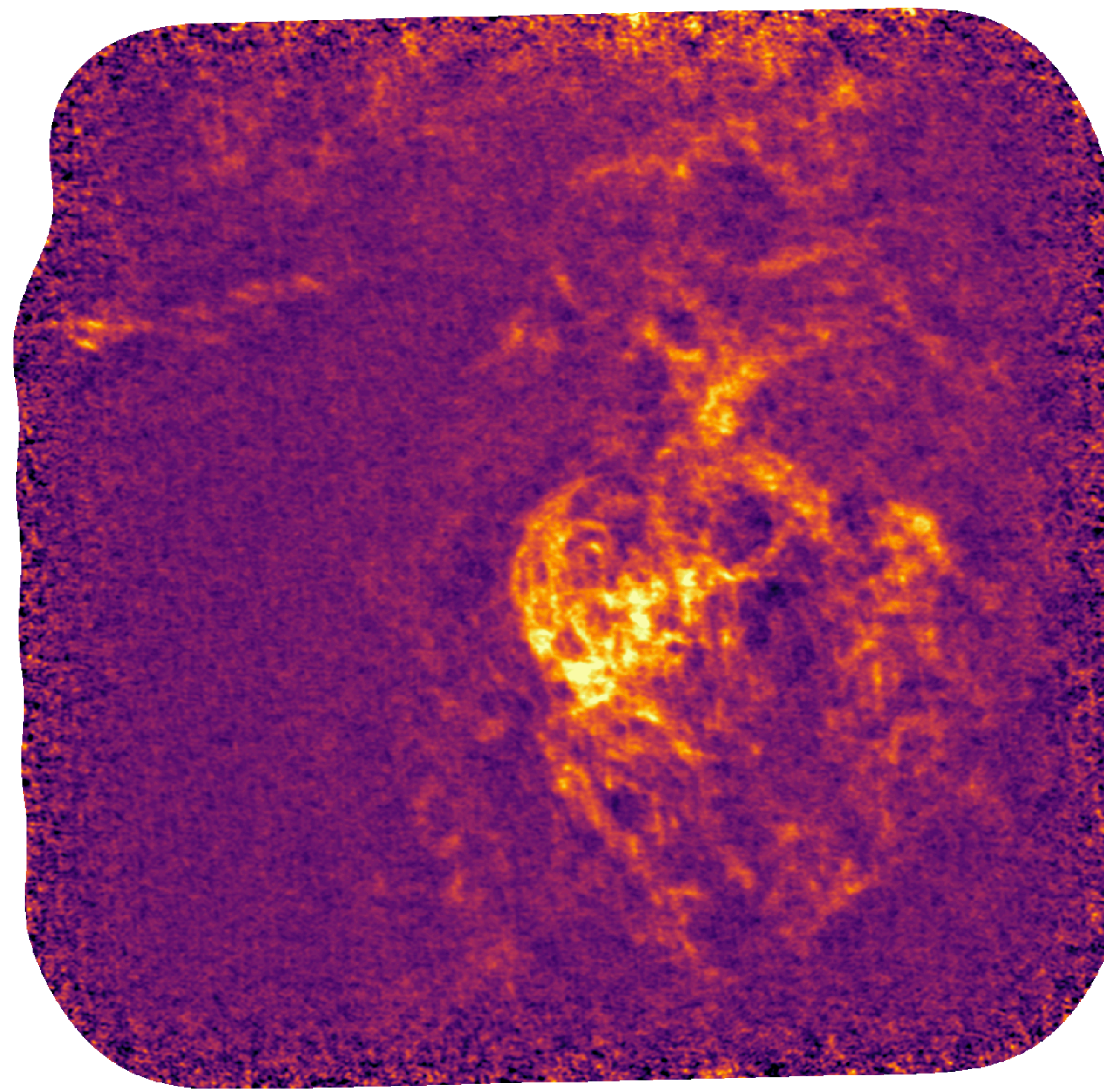
- In general this is not recommended over joint deconvolution as you are effectively losing information and therefore image fidelity

- This can still be a useful method to obtain a quick look at the combined interferometric data, just keep caveats in mind and plan to explore joint deconvolution

Galactic centre molecular cloud
HNCO 4-3, single channel

12m only | **12m + 7m + TP**

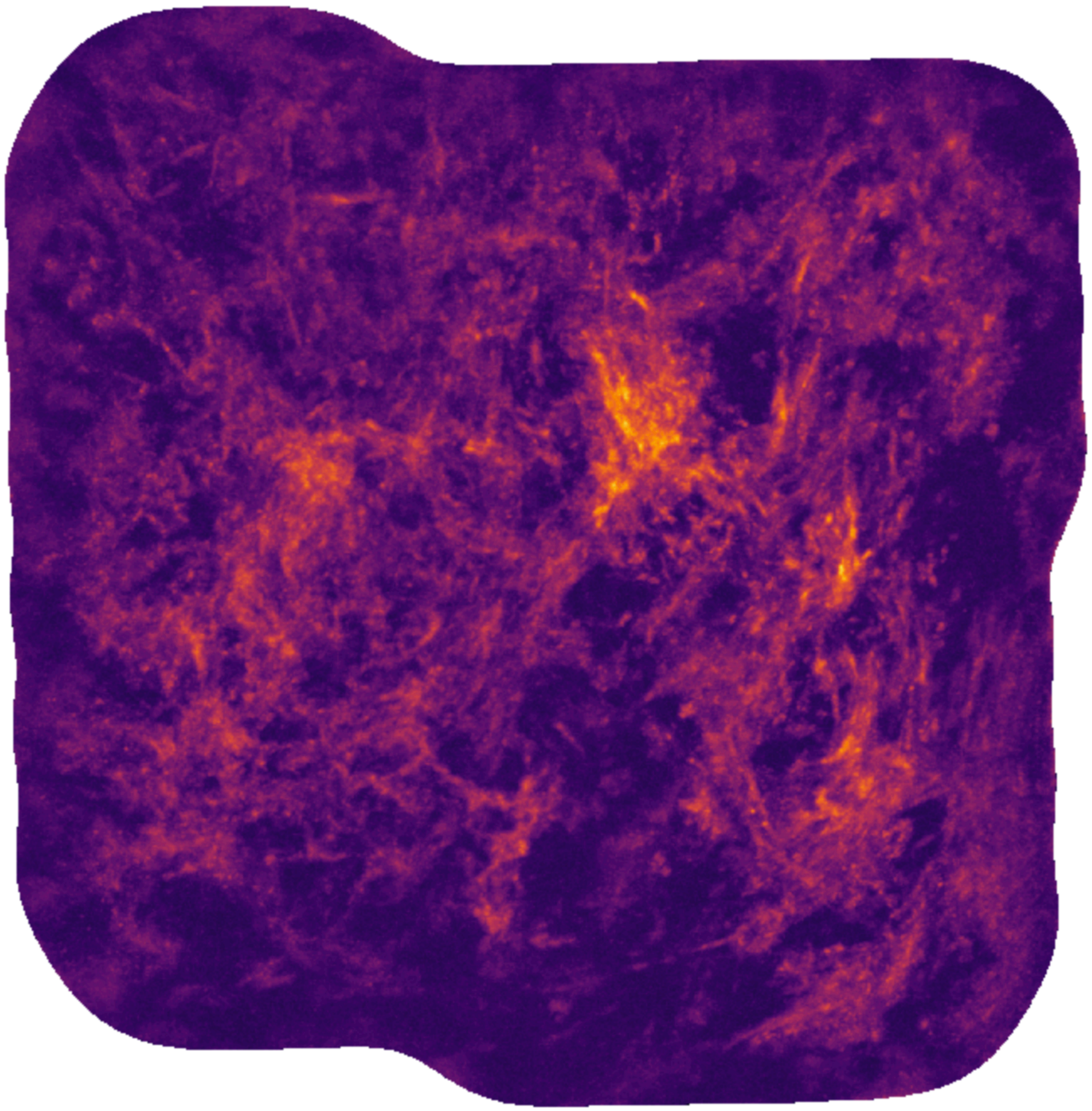




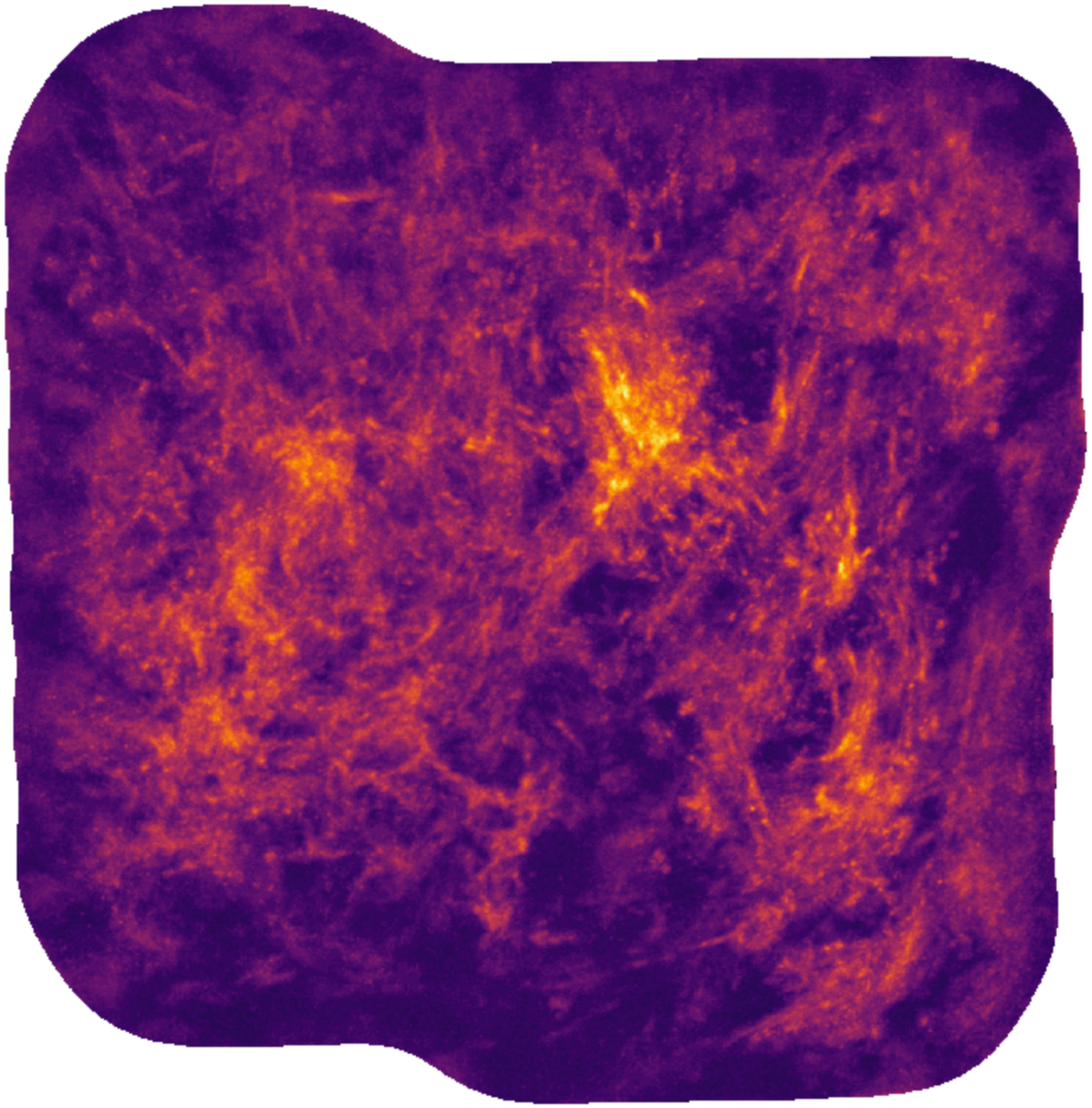
Source: ACES LP, ALMA ID 2021.1.00172.L (PI: S. Longmore)

Another Galactic centre molecular cloud
HNCO 4-3, peak intensity

12m + 7m



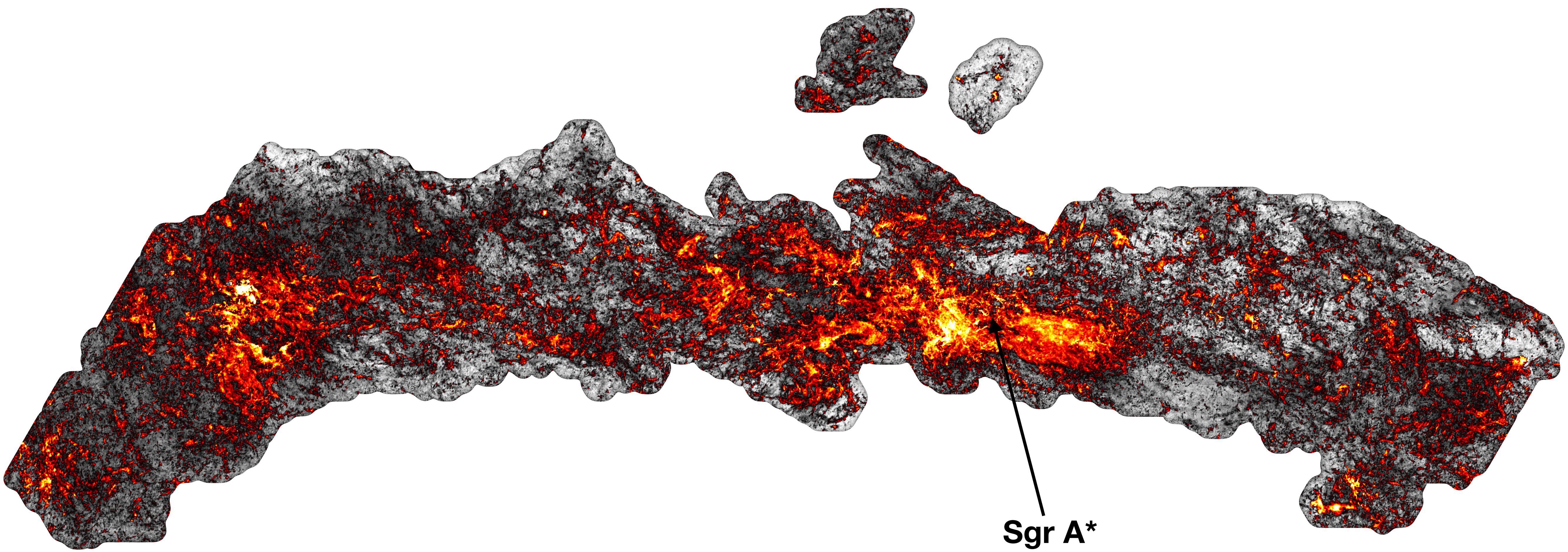
12m + 7m + TP



Source: ACES LP, ALMA ID 2021.1.00172.L (PI: S. Longmore)

Galactic centre inner ~ 200 pc
CS (2-1) peak intensity

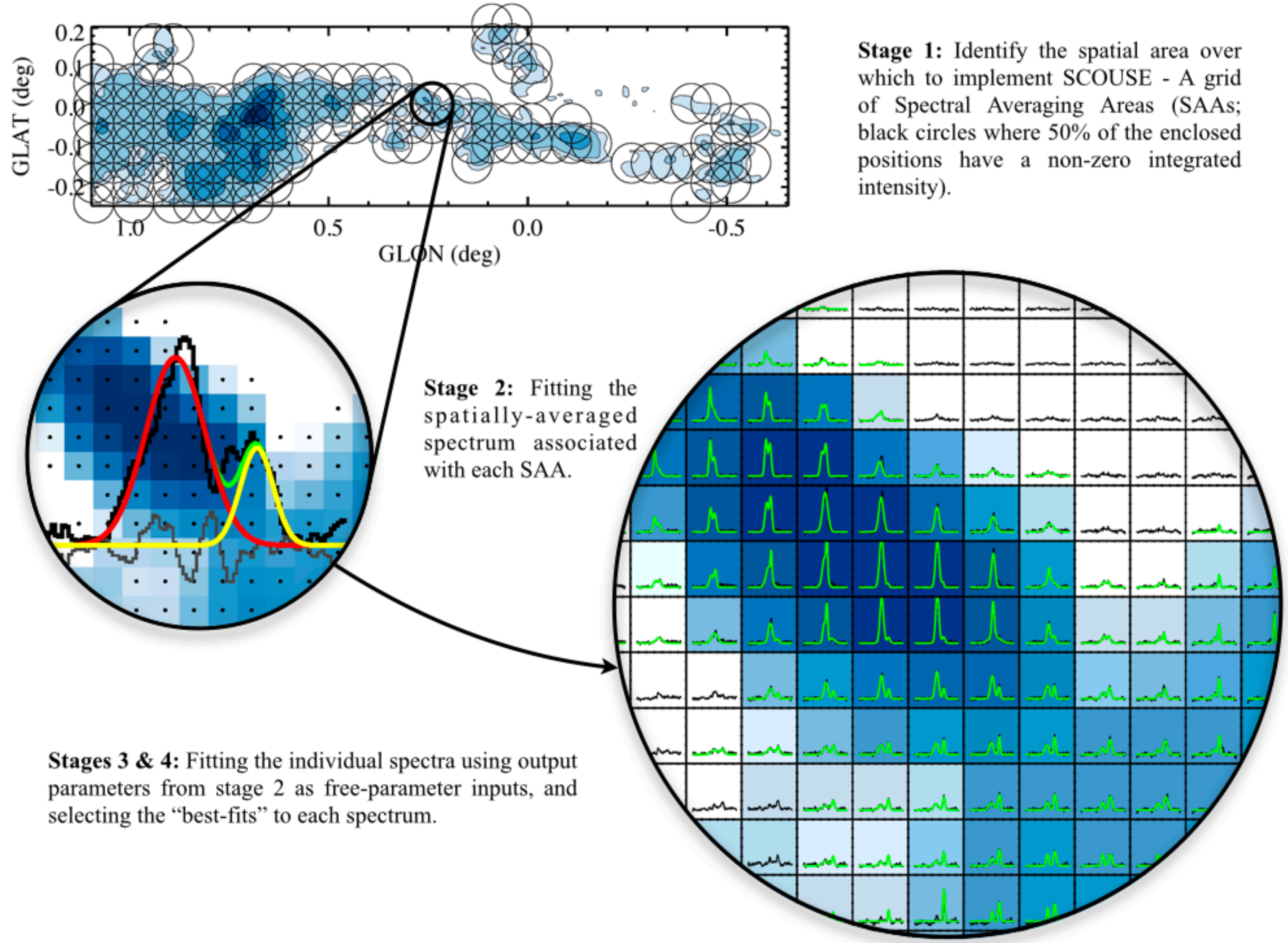
12m + 7m + TP



Source: ACES LP, ALMA ID 2021.1.00172.L (PI: S. Longmore)

SCOUSE(py): Semi-automated multi-COmponent Universal Spectral-line fitting Engine

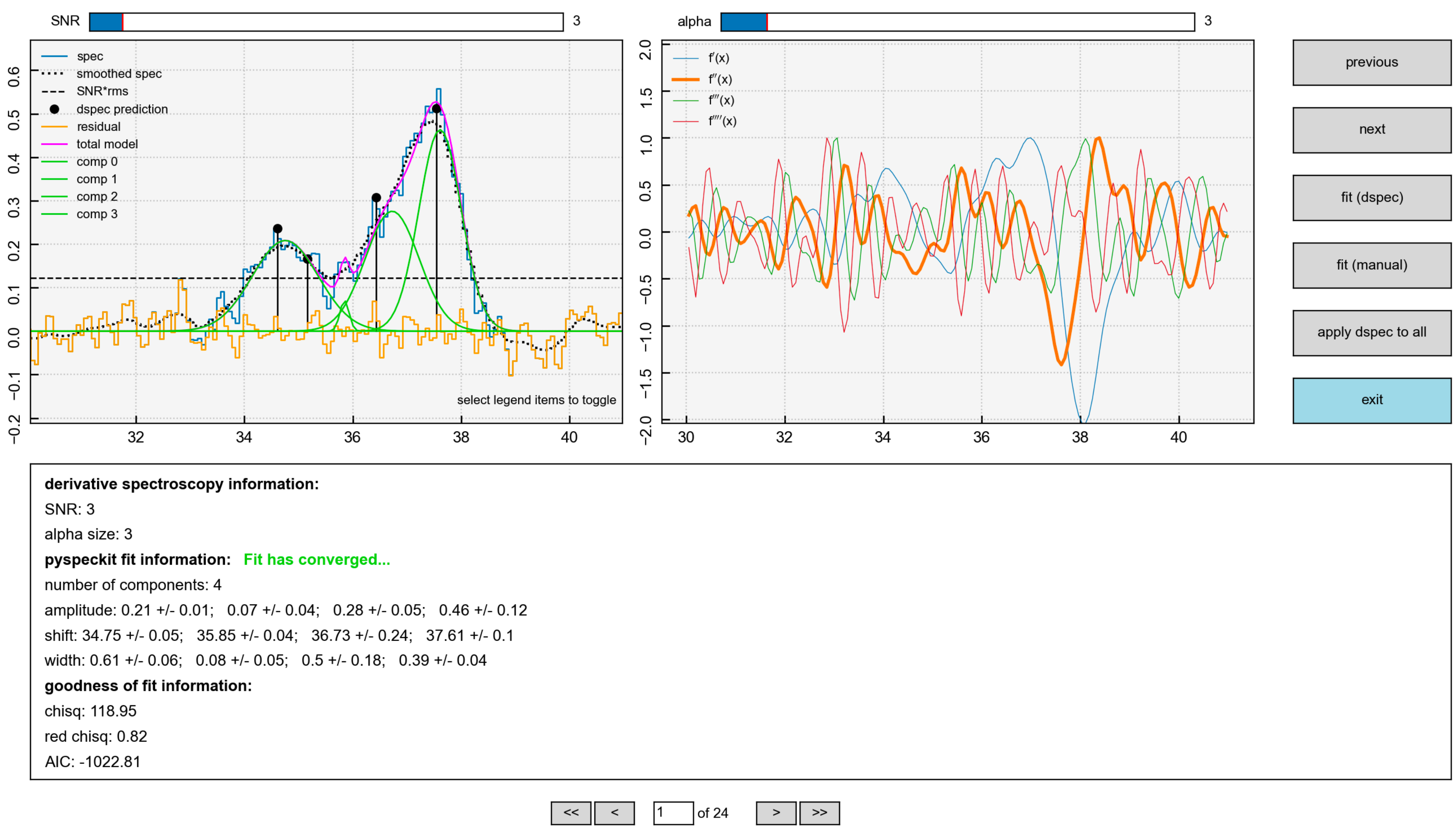
scouse



Available for download: **https://scousepy.readthedocs.io/en/latest/**

For each averaged spectrum, scousepy provides an initial fit, which the user cycles through an interactive GUI to either accept or update the fit.

scousepy will then enter the fully automated stage, where it will take these averaged fits, and pass the parameters to fit the spectrum at every single pixel within each averaging area.



Henshaw et al. 2019

Ubiquitous Velocity Fluctuations throughout the Molecular Interstellar Medium

G0.253+0.016 a.k.a. "The Brick"

Jonathan D. Henshaw et al.

Henshaw et al. 2020