The ALMA (and JVLA) Pipeline

Andy Biggs and Liz Humphreys

ALMA Regional Centre,

European Southern Observatory

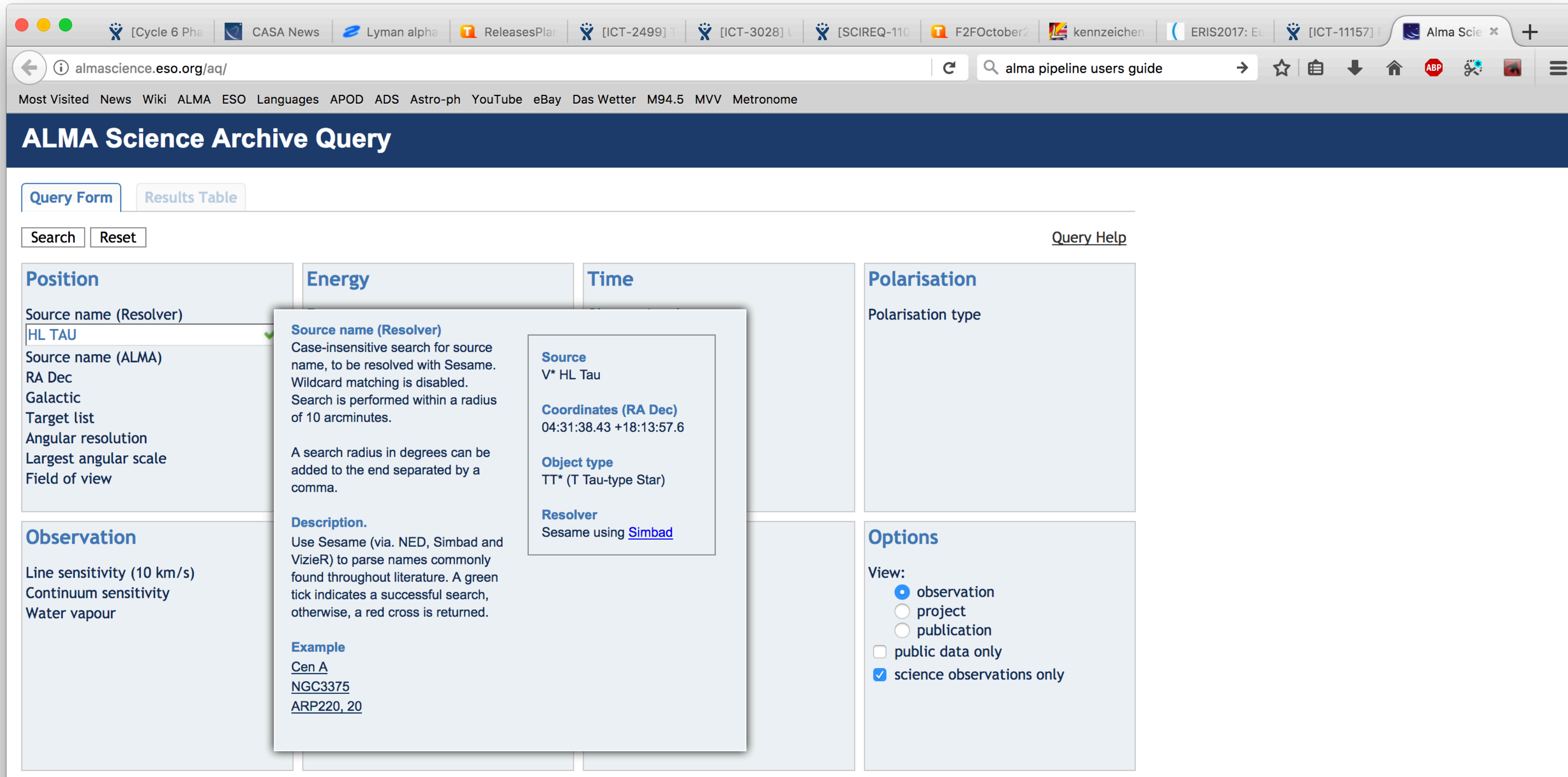
EUROPEAN ARC
ALMA Regional Centre

The need for pipelines

- ALMA datasets can be HUGE
  - Up to 64 antennas (>2000 baselines!)
  - Up to 8 x 8192 channels per integration
  - Targets observed for tens of hours
  - Data sets with TB of data are not unusual
- Processing times are similarly now very long
  - Several days can be required in some cases
- Flagging, calibration and imaging by a human is often simply not possible
- Automatic re-processing of data becomes possible
  - Data products in archive can be updated as Pipeline improves

The ALMA archive

Aims to provide science-ready data products



https://almascience.eso.org/aq

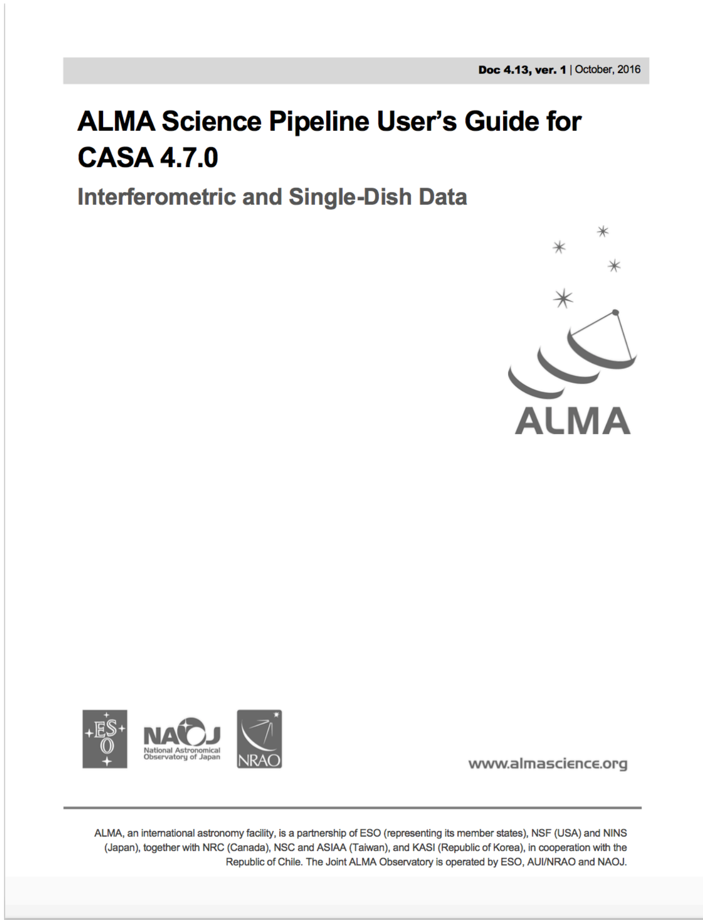
CASA pipeline

- Used by both ALMA and the JVLA
- Dedicated pipeline tasks within **CASA** are used
  - E.g. hif_flagdata, hif_setjy, hif_bandpass, hif_applycal, etc.
- Prefixes are actually meaningful
  - hif: interferometry (ALMA and VLA)
  - hifa: interferometry (ALMA-only)
  - hifv: interferometry (VLA-only)
  - hfs: single dish
- Pipeline is data-driven
  - In principle, no user input is required
  - Heuristics have been developed to help Pipeline decision making

The ALMA Pipeline

- All ALMA data is calibrated and imaged by the Observatory
  - Data not released to PI until it has undergone ‘Quality Assurance’
- Huge progress in developing the Pipeline has been made
  - The calibration pipeline was first used in Cycle 2
    - 75% of data is now calibrated without human intervention
  - First imaging deliveries in Cycle 4
  - Imaging in Cycle 5
    - Expect that only 30% of projects will require human intervention
    - 50% of data will be calibrated and imaged completely automatically
- Not all observing modes can be pipelined
  - e.g. polarization, bandwidth-switching (for narrow bandwidths)
  - “Non-standard modes” – restricted to 20% of ALMA observing time

Pipeline User's Guide


Doc 4.13, ver. 1 | October, 2016

ALMA Science Pipeline User's Guide for CASA 4.7.0

Interferometric and Single-Dish Data

ALMA

ESO NAOJ National Astronomical Observatory of Japan NRAO

www.almascience.org

ALMA, an international astronomy facility, is a partnership of ESO (representing its member states), NSF (USA) and NINS (Japan), together with NRC (Canada), NSC and ASIAA (Taiwan), and KASI (Republic of Korea), in cooperation with the Republic of Chile. The Joint ALMA Observatory is operated by ESO, AUI/NRAO and NAOJ.

Available from the ALMA Science Portal:

`https://almascience.eso.org`

A CASA Guide also exists:

`https://casaguides.nrao.edu/index.php/ALMA_Imaging_Pipeline_Reprocessing`

Running the Pipeline

- Must use a version of CASA which includes the Pipeline
  - Not every CASA release includes the Pipeline
  - The last release with a Pipeline was 4.7.2
  - The Cycle-5 Pipeline will appear in CASA 5.1
    - **Current release does not include the Pipeline**
    - This will be updated at some point
- Must start CASA in a special mode
  - `>casa --pipeline` (must be done from the command line)
- Data download will contain various Python scripts e.g.
  - casa_pipescript.py (imaging and/or calibration)
  - scriptForImaging.py (if imaging was done manually)
  - Run in usual way e.g. `>execfile('casa_pipescript.py')`

Why re-run the Pipeline?

- Remove datasets from the calibration/imaging
  - ALMA observes in approx. 1-hour sessions called Execution Blocks
  - Each is delivered to the user as an ‘ASDM’ (ALMA Science Data Model)
- Introducing additional flagging
  - Edit the flagtemplate.txt file
- Setting more accurate calibrator flux densities
  - More reliable values may now be available
- Modifying the imaging parameters
  - Weighting
  - Tapering
  - Redoing continuum subtraction
  - And many, many more …

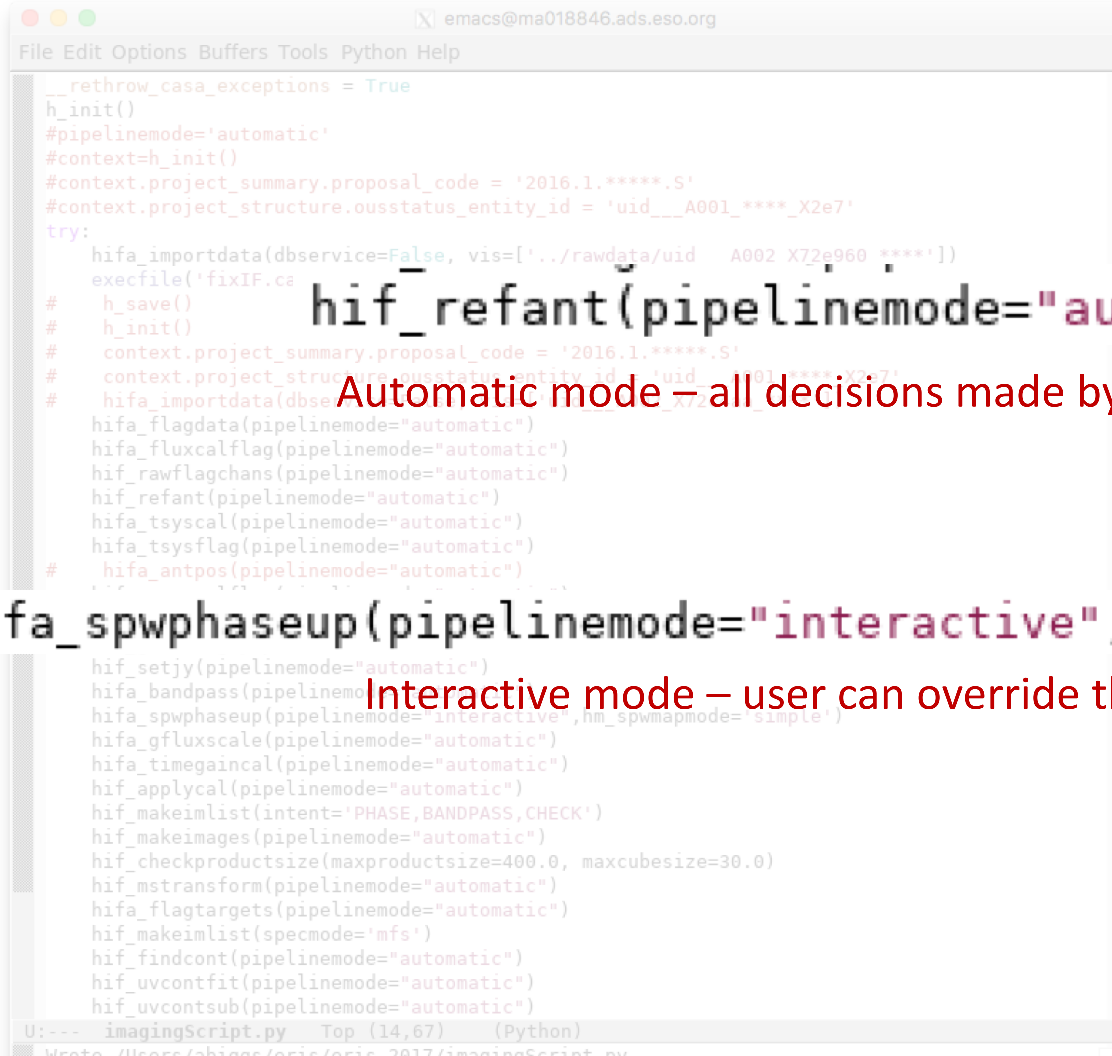
Example script

emacs@ma018846.ads.eso.org

File Edit Options Buffers Tools Python Help

```
__rethrow_casa_exceptions = True
h_init()
#pipelinemode='automatic'
#context=h_init()
#context.project_summary.proposal_code = '2016.1.*****.S'
#context.project_structure.ousstatus_entity_id = 'uid__A001_****_X2e7'
try:
    hifa_importdata(dbservice=False, vis=['../rawdata/uid__A002_X72e960_****'])
    execfile('fixIF.casa.py')
#     h_save()
#     h_init()
#     context.project_summary.proposal_code = '2016.1.*****.S'
#     context.project_structure.ousstatus_entity_id = 'uid__A001_****_X2e7'
#     hifa_importdata(dbservice=False, vis=['uid__A002_X72c4aa_****'])
    hifa_flagdata(pipelinemode="automatic")
    hifa_fluxcalflag(pipelinemode="automatic")
    hif_rawflagchans(pipelinemode="automatic")
    hif_refant(pipelinemode="automatic")
    hifa_tsyscal(pipelinemode="automatic")
    hifa_tsysflag(pipelinemode="automatic")
#     hifa_antpos(pipelinemode="automatic")
    hifa_wvrgcalflag(pipelinemode="automatic")
    hif_lowgainflag(pipelinemode="automatic")
    hif_gainflag(pipelinemode="automatic")
    hif_setjy(pipelinemode="automatic")
    hifa_bandpass(pipelinemode="automatic")
    hifa_spwphaseup(pipelinemode="interactive",hm_spwmapmode='simple')
    hifa_gfluxscale(pipelinemode="automatic")
    hifa_timegaincal(pipelinemode="automatic")
    hif_applycal(pipelinemode="automatic")
    hif_makeimlist(intent='PHASE,BANDPASS,CHECK')
    hif_makeimages(pipelinemode="automatic")
    hif_checkproductsize(maxproductsize=400.0, maxcubesize=30.0)
    hif_mstransform(pipelinemode="automatic")
    hifa_flagtargets(pipelinemode="automatic")
    hif_makeimlist(specmode='mfs')
    hif_findcont(pipelinemode="automatic")
    hif_uvcontfit(pipelinemode="automatic")
    hif_uvcontsub(pipelinemode="automatic")
U:---  imagingScript.py   Top (14,67)    (Python)
Wrote /Users/abiggs/eris/eris_2017/imagingScript.py
```

Example script




```
__rethrow_casa_exceptions = True
h_init()
#pipelinemode='automatic'
#context=h_init()
#context.project_summary.proposal_code = '2016.1.*****.S'
#context.project_structure.ousstatus_entity_id = 'uid___A001_****_X2e7'
try:
    hifa_importdata(dbservice=False, vis=['../rawdata/uid___A002_X72e960_****'])
    execfile('fixIF.ca
#    h_save()
#    h_init()
#    context.project_summary.proposal_code = '2016.1.*****.S'
#    context.project_struc
#    hifa_importdata(dbse
    hifa_flagdata(pipelinemode="automatic")
    hifa_fluxcalflag(pipelinemode="automatic")
    hif_rawflagchans(pipelinemode="automatic")
    hif_refant(pipelinemode="automatic")
    hifa_tsyscal(pipelinemode="automatic")
    hifa_tsysflag(pipelinemode="automatic")
#    hifa_antpos(pipelinemode="automatic")
    hif_setjy(pipelinemode="automatic")
    hifa_bandpass(pipelinemo
    hifa_spwphaseup(pipelinemode="interactive",hm_spwmapmode='simple')
    hifa_gfluxscale(pipelinemode="automatic")
    hifa_timegaincal(pipelinemode="automatic")
    hif_applycal(pipelinemode="automatic")
    hif_makeimlist(intent='PHASE,BANDPASS,CHECK')
    hif_makeimages(pipelinemode="automatic")
    hif_checkproductsize(maxproductsize=400.0, maxcubesize=30.0)
    hif_mstransform(pipelinemode="automatic")
    hifa_flagtargets(pipelinemode="automatic")
    hif_makeimlist(specmode='mfs')
    hif_findcont(pipelinemode="automatic")
    hif_uvcontfit(pipelinemode="automatic")
    hif_uvcontsub(pipelinemode="automatic")
```

`hif_refant(pipelinemode="automatic")`

Automatic mode – all decisions made by the Pipeline

`hifa_spwphaseup(pipelinemode="interactive",hm_spwmapmode='simple')`

Interactive mode – user can override the defaults

The Weblog

- Results of the Pipeline are reported in the Weblog
  - View ‘index.html’ file in your browser
- Shows clickable menu of all tasks that were run
- Clicking on a task will show results of that task
  - Tasks will report warnings, useful messages
  - **Many** plots are included

file:///Users/ehumphre/pipelineC5_r40783weblog/t2-4m.html?sidebar=sidebar_stage1

Mail - Elizabeth Humphreys - Outlook | 1273635062492792877.png 757×573 pixels | 12. hifa_bandpassflag

Home By Topic By Task 2016.1.00496.S

Tasks in execution order

1. hifa_importdata
2. hifa_flagdata
3. hifa_fluxcalflag
4. hif_rawflagchans
5. hif_refant
6. h_tsyscal
7. hifa_tsysflag
8. hifa_antpos
9. hifa_wvrgcalflag
10. hif_lowgainflag
11. hif_setmodels
12. hifa_bandpassflag
13. hifa_spwphaseup
14. hifa_gfluxscaleflag
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hifa_imageprecheck
19. hif_makeimlist
20. hif_makeimages
21. hif_checkproductsize

12. Bandpass Calibration and Flagging

BACK

Task notifications

Warning! Evaluation of uid___A002_Xb9cc97_X3ce7.ms raised 1 flagging command(s)

This task performs a preliminary bandpass solution and applies it, then computes the flagging heuristics by calling hif_correctedampflag which looks for outlier visibility points by statistically examining the scalar difference of the corrected amplitude minus model amplitudes, flags those outliers, then derives a final bandpass solution (if any flags were generated). The philosophy is that only outlier data points that have remained outliers after calibration will be flagged. Note that the phase of the data is not assessed.

In further detail, the workflow is as follows: an a priori calibration is applied using pre-existing caltables in the calibration state, a preliminary bandpass solution and amplitude gaincal solution is solved and applied, the flagging heuristics are run and any outliers are flagged, a final bandpass solution is solved (if necessary) and the name "final" is appended to this caltable. Plots are generated at three points in this workflow: after a priori calibration, after bandpass calibration but before flagging heuristics are run, and after flagging heuristics have been run and applied. If no points were flagged, the "after" plots are not generated or displayed. The score for this stage is a simple combination (multiplication) of the standard data flagging score (depending on the fraction of data flagged) and the score for the bandpass solution.

Flagging

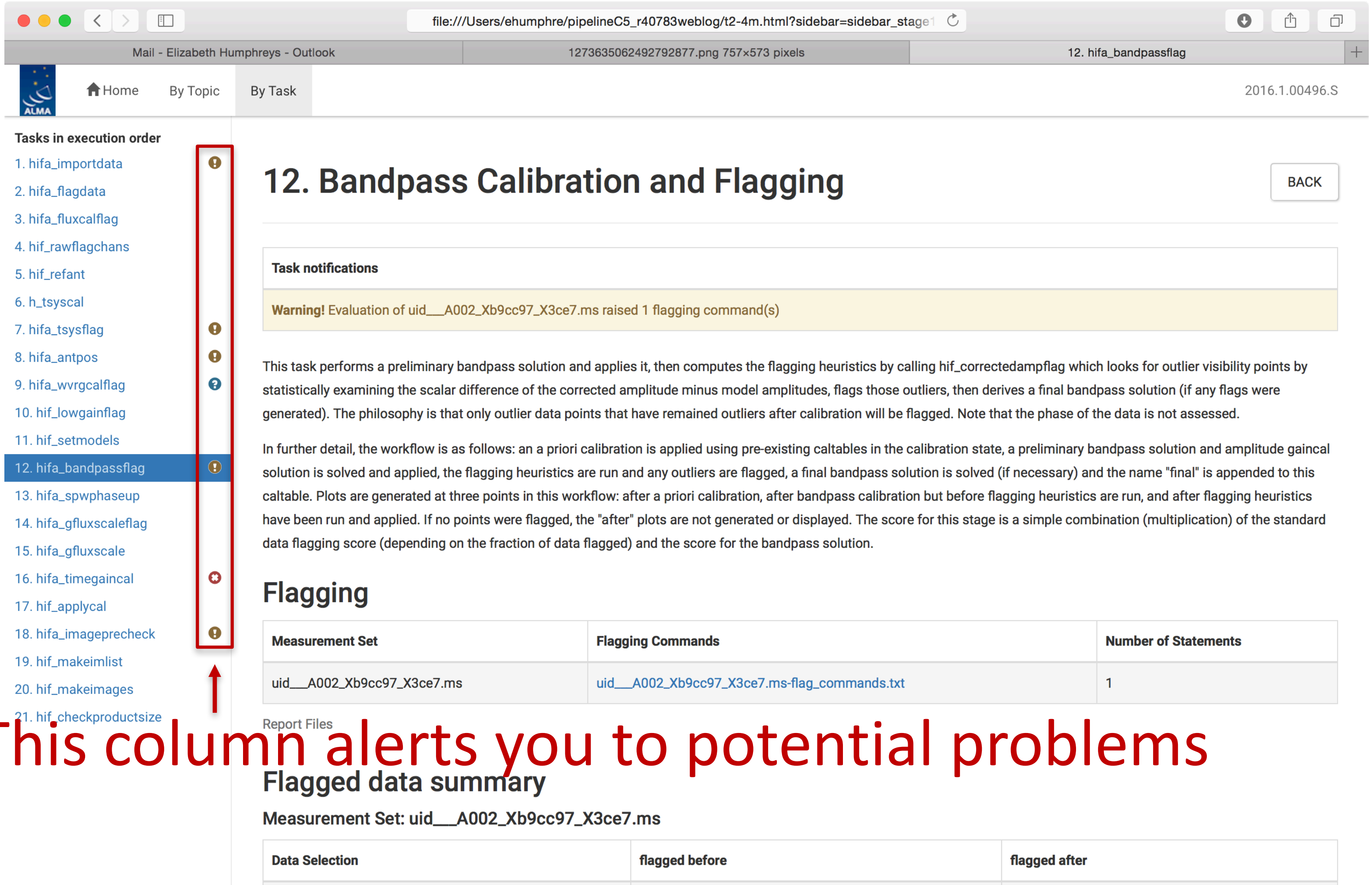
Measurement Set	Flagging Commands	Number of Statements
uid___A002_Xb9cc97_X3ce7.ms	uid___A002_Xb9cc97_X3ce7.ms-flag_commands.txt	1

Report Files

Flagged data summary

Measurement Set: uid___A002_Xb9cc97_X3ce7.ms

Data Selection	flagged before	flagged after


file:///Users/ehumphre/pipelineC5_r40783weblog/t2-4m.html?sidebar=sidebar_stage

Mail - Elizabeth Humphreys - Outlook | 1273635062492792877.png 757×573 pixels | 12. hifa_bandpassflag

Home | By Topic | By Task | 2016.1.00496.S

Tasks in execution order

1. hifa_importdata
2. hifa_flagdata
3. hifa_fluxcalflag
4. hif_rawflagchans
5. hif_refant
6. h_tsyscal
7. hifa_tsysflag
8. hifa_antpos
9. hifa_wvrgcalflag
10. hif_lowgainflag
11. hif_setmodels
12. hifa_bandpassflag
13. hifa_spwphaseup
14. hifa_gfluxscaleflag
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hifa_imageprecheck
19. hif_makeimlist
20. hif_makeimages
21. hif_checkproductsize

12. Bandpass Calibration and Flagging

BACK

Task notifications

Warning! Evaluation of uid___A002_Xb9cc97_X3ce7.ms raised 1 flagging command(s)

This task performs a preliminary bandpass solution and applies it, then computes the flagging heuristics by calling hif_correctedampflag which looks for outlier visibility points by statistically examining the scalar difference of the corrected amplitude minus model amplitudes, flags those outliers, then derives a final bandpass solution (if any flags were generated). The philosophy is that only outlier data points that have remained outliers after calibration will be flagged. Note that the phase of the data is not assessed.

In further detail, the workflow is as follows: an a priori calibration is applied using pre-existing caltables in the calibration state, a preliminary bandpass solution and amplitude gaincal solution is solved and applied, the flagging heuristics are run and any outliers are flagged, a final bandpass solution is solved (if necessary) and the name "final" is appended to this caltable. Plots are generated at three points in this workflow: after a priori calibration, after bandpass calibration but before flagging heuristics are run, and after flagging heuristics have been run and applied. If no points were flagged, the "after" plots are not generated or displayed. The score for this stage is a simple combination (multiplication) of the standard data flagging score (depending on the fraction of data flagged) and the score for the bandpass solution.

Flagging

Measurement Set	Flagging Commands	Number of Statements
uid___A002_Xb9cc97_X3ce7.ms	uid___A002_Xb9cc97_X3ce7.ms-flag_commands.txt	1

Report Files

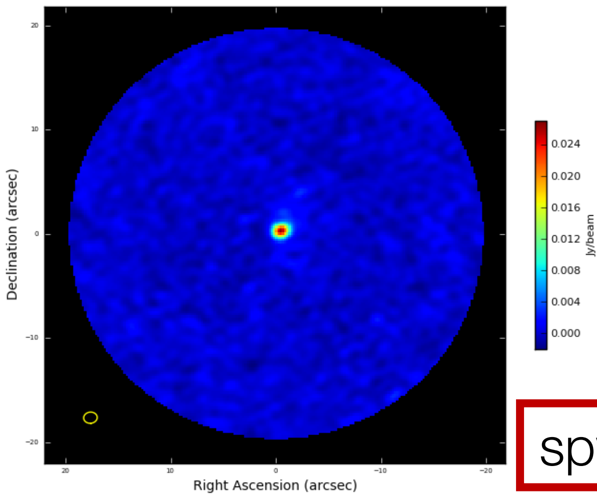
Flagged data summary

Measurement Set: uid___A002_Xb9cc97_X3ce7.ms

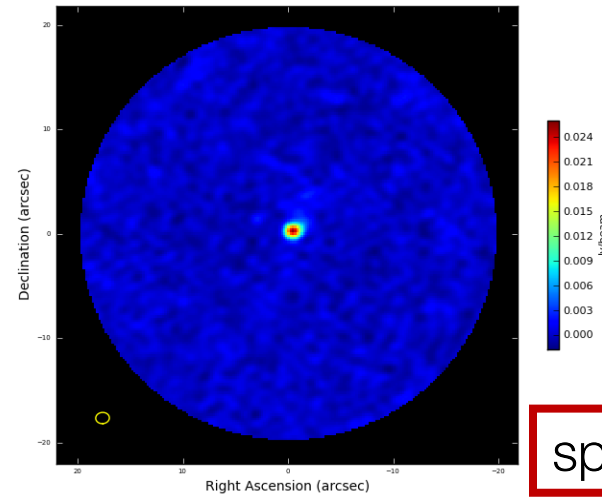
Data Selection	flagged before	flagged after

This column alerts you to potential problems

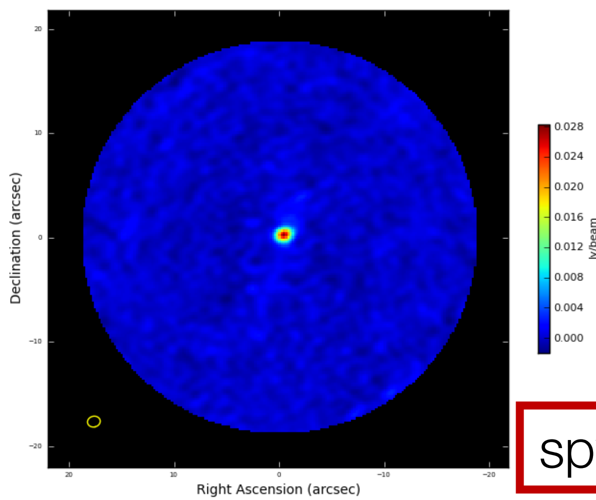
Continuum Imaging



spw 25

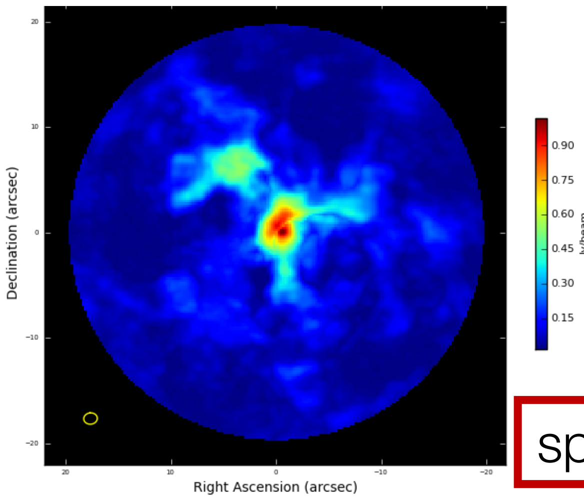


spw 27

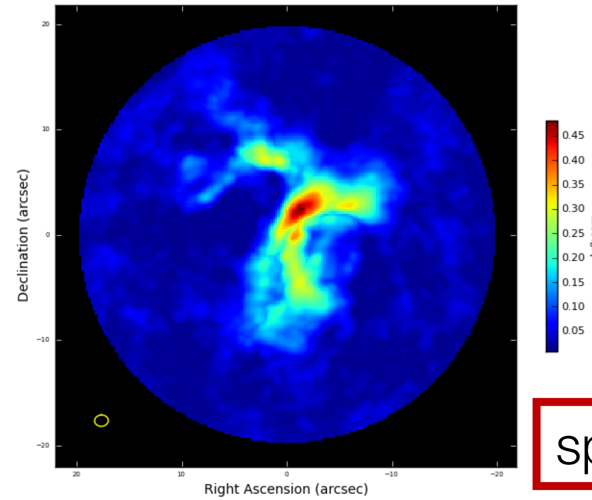


spw 39

Moment-8 maps



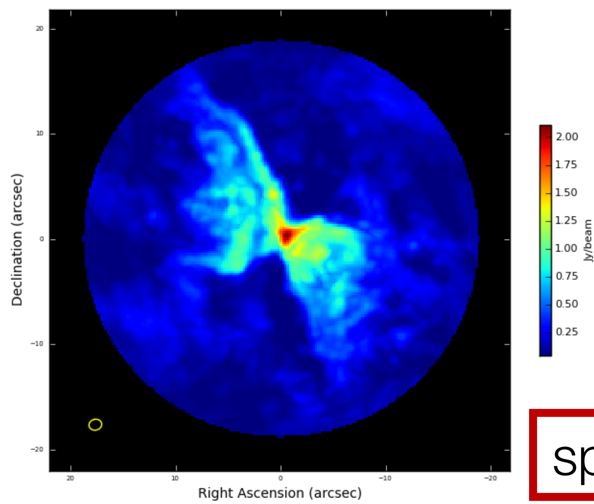
spw 25



spw 27

Continuum-subtracted **cubes**

Moment-8 maps show brightest channel at each spatial position

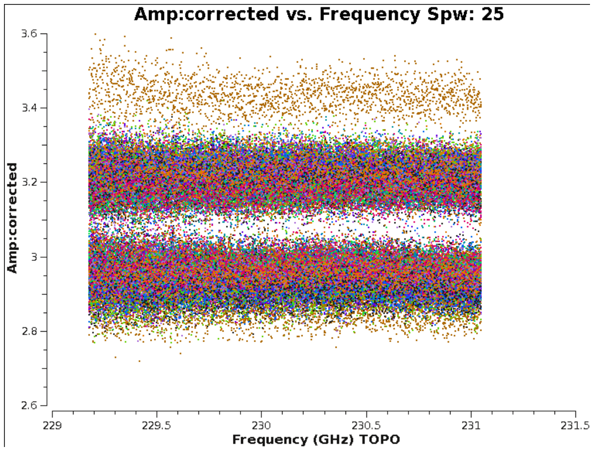


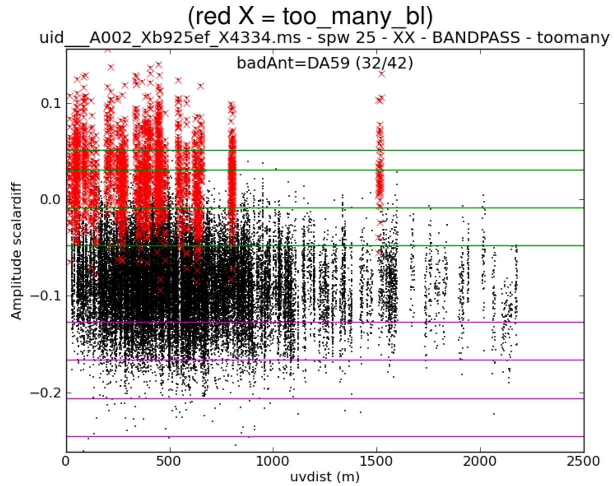
spw 39

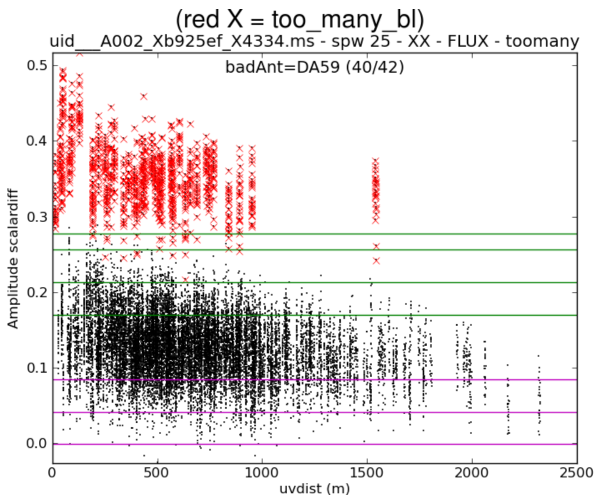
Recent improvements

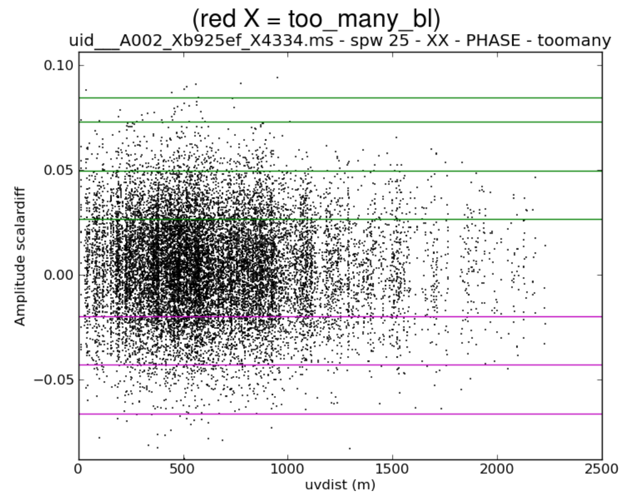
- For Cycle 5 there have been a number of improvements
  - Attempt to reduce the runtime if data products are too large
    - i.e. any cube > 30 GB or total cubes > 400 GB
    - Will channel average, reduce image size, number of sources imaged, etc.
  - Automatic flagging of calibrated calibrator data
    - Reduces the amount of manual flagging required
    - Note: Calibrated target data is not yet flagged
  - Auto-masking (auto-boxing) during cleaning
    - Better imaging and allows a deeper clean
  - Plus many others!

Auto-flagging of calibrators (1)

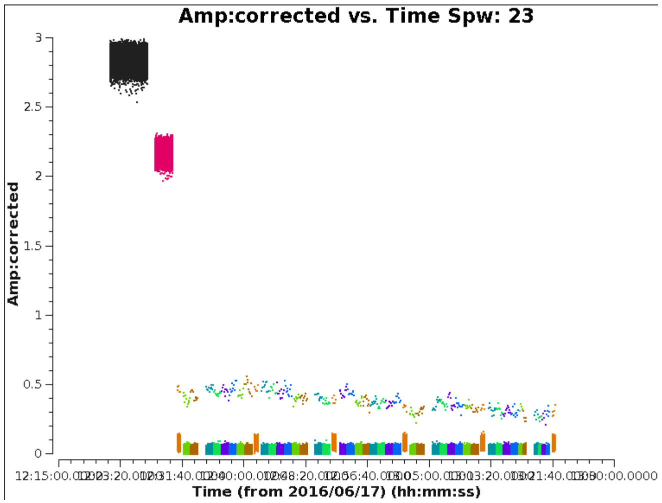


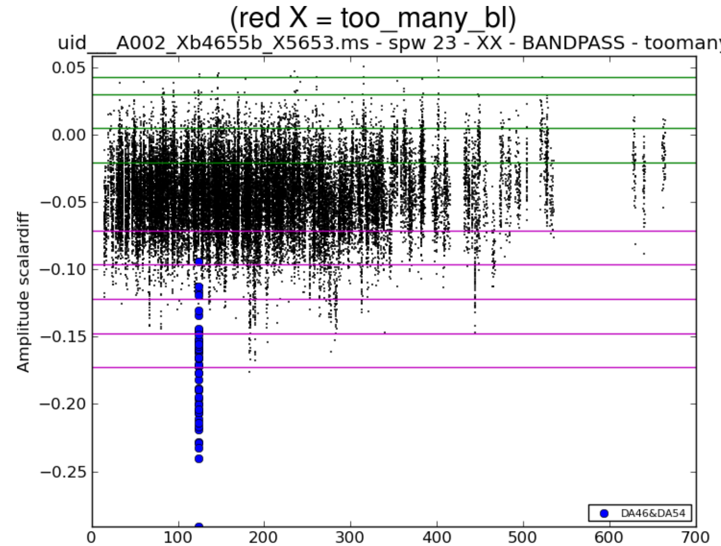


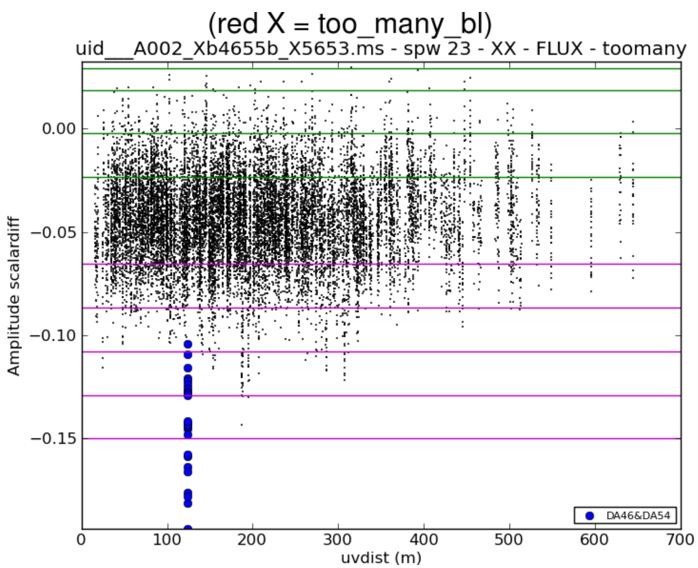


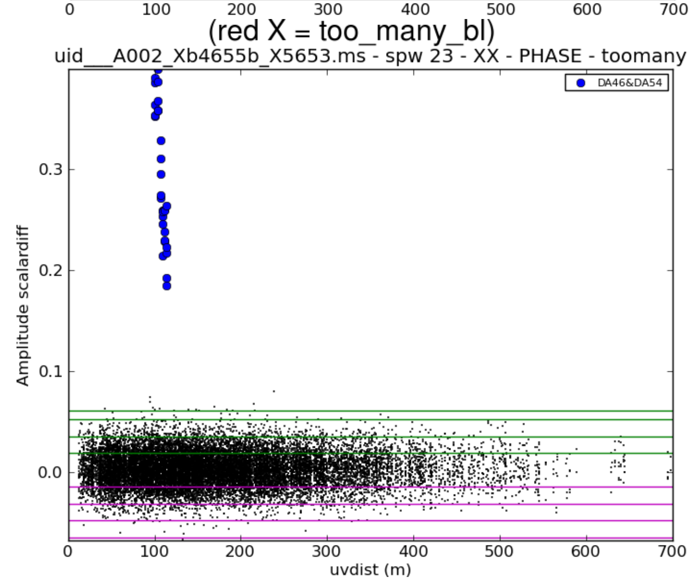


Auto-flagging of calibrators (2)







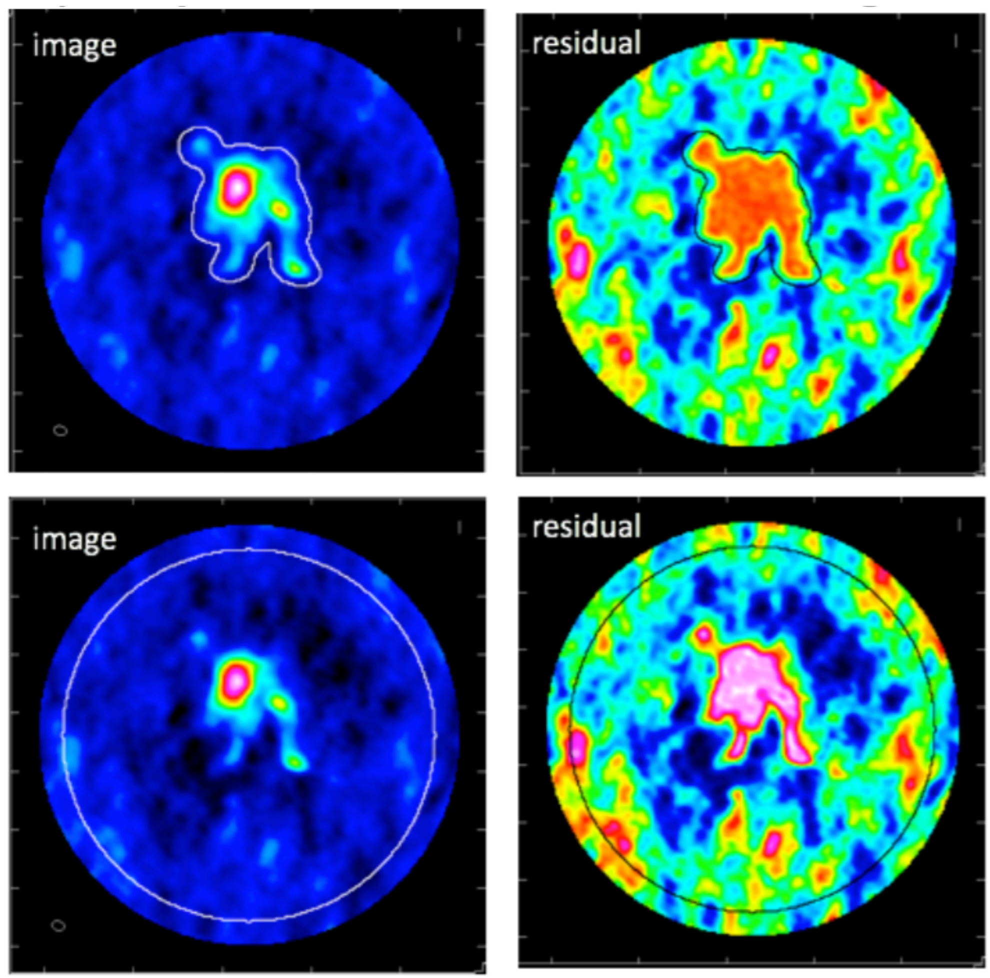


Auto-masking

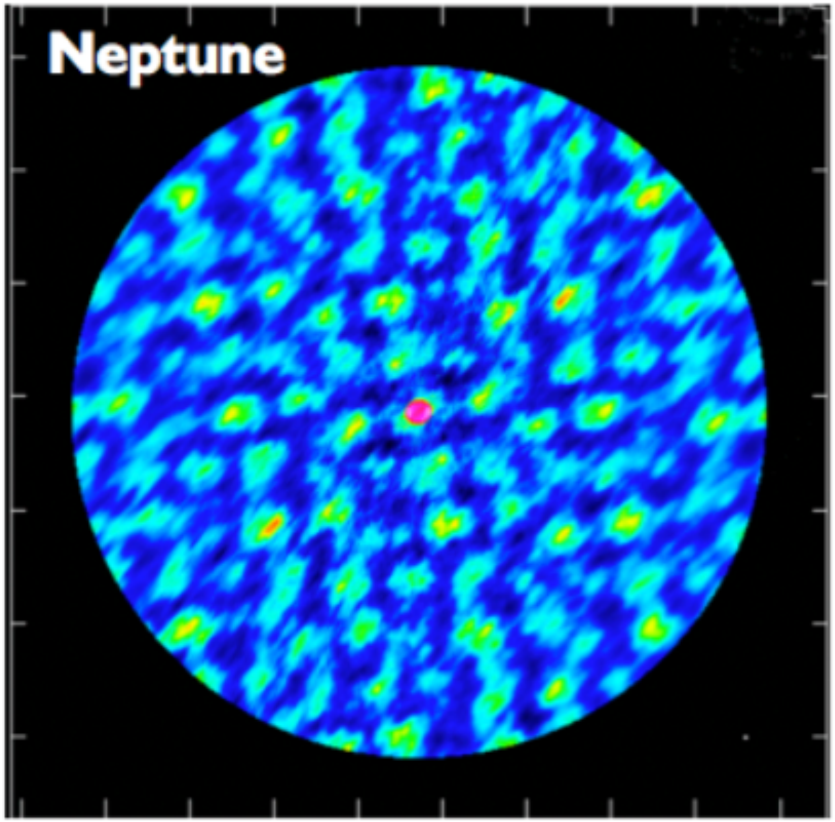
Will allow deeper cleaning

Cycle 5 pipeline with automasking

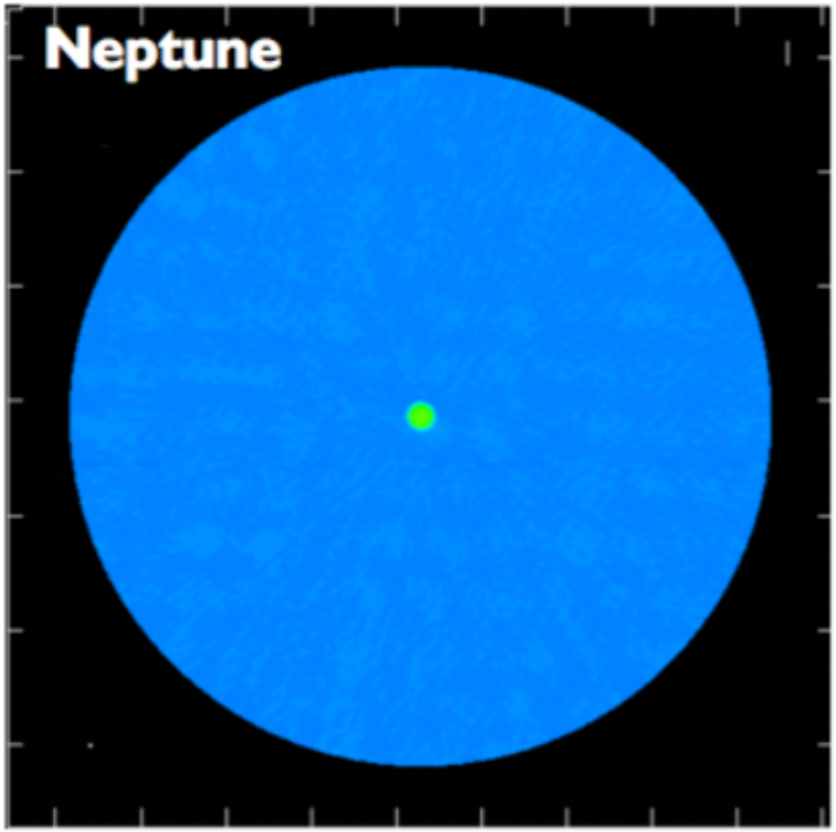
Cycle 4 pipeline with primary beam mask



Auto-masking



Cycle 4 pipeline with primary beam mask



Cycle 5 pipeline with automasking

Summary

- The CASA pipeline is used to calibrate ALMA and JVLA data
  - Including ALMA Total Power array (single dish)
- Fraction of pipelined ALMA data continually increasing
  - Expect about 70% of projects to be pipeline-imaged in Cycle 5
- Cycle-5 Pipeline will be released in an update of CASA 5.1
- Parallelization of `tclean`* will appear in CASA 5.2

*Refactor of `clean` – will become the standard version of ‘clean’

RadioNet has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 730562