Introduction to CASA, Calibration & Basic Imaging

Seventeenth Synthesis Imaging Workshop

29 June – July 17 2020

NRAO ALMA AUI NSF New Mexico Tech SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY The University of New Mexico New Mexico CONSORTIUM

Introduction to CASA, Calibration & Basic Imaging

ALMA Data Reduction Tutorial

Synthesis Imaging Summer School

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Robert C. Byrd Green Bank Telescope
Very Long Baseline Array

NRAO

Outline

- **Short introduction to CASA and the Python interface**
  - How to use tasks
  - What is a measurement set?
- The Flow of Calibration
- Overview of your Directory
  - Data preparation and set up
  - Getting oriented with your data
- Data Calibration
- Data Inspection and Flagging
- Basic Imaging

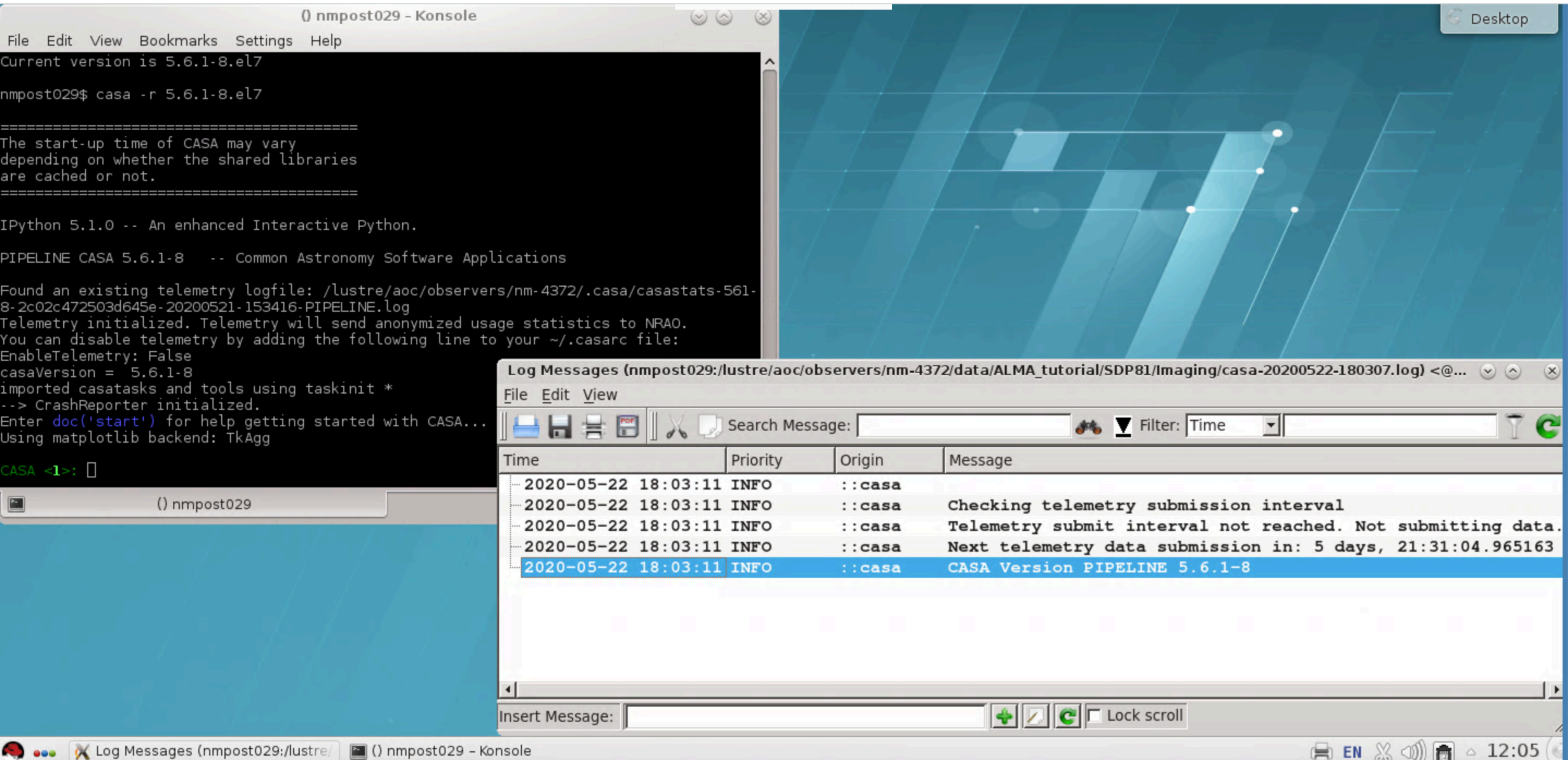
CASA (Common Astronomy Software Applications)

- CASA is the offline data reduction package for ALMA and the VLA (data from other telescopes usually work, too, but not primary goal of CASA)
- Code is C++ (fast) bound to Python (easy access and scripting) (plus some Qt or other apps)
- Import/export data, inspect, edit, calibrate, image, view, analyze
- Also supports single dish data reduction
- CASA has many tasks and a LOT of tool methods
- Easy to write scripts and tasks
- We have a lot of documentation, reduction tutorials, helpdesk, user forum
- CASA has some of the most sophisticated algorithms implemented (multi-scale clean, Taylor term expansion for wide bandwidths, W-term projection, OTF mosaicing, etc.)
- We have a active Algorithm Research Group, so expect more features in future versions…

NRAO

CASA Startup

$ **casa (or casa -r version, e.g. casa -r 5.6.1-8.el7 if you have multiple casa versions installed)**



CASA Interactive Interface

- CASA runs within pythons scripts or through the interactive *IPython* (ipython.org) interface
- IPython Features:
  - shell access
  - auto-parenthesis (autocall)
  - Tab auto-completion
  - command history (arrow up and "hist [-n]")
  - session logging
    - casapyTIME.log – casa logger messages
  - numbered input/output
  - history/searching

NRAO

Basic Python tips

- CASA uses python 2.7.x, not python 3
  - transition to python 3 happening soon
- to run a python ".py" script:

  `execfile('<scriptname>')`

  example: `execfile('ngc5921_demo.py')`

Some python specialties:

- python counts from 0 to n-1!
- variables are global when using task interface
- tasknames are objects (not variables)

Basic Python tips

Cutting and pasting in CASA:

- indentation matters!
  - indentation in python is for loops, conditions etc.
  - be careful when doing cut-and-paste to python
  - cut a few (4-6) lines at a time
- for longer commands and loops:
  - use **%cpaste** and **--**

```
CASA <1>: %cpaste


Long list of CASA commands


--
```

Tasks and tools in CASA

- Tasks - high-level functionality
  - function call or parameter handling interface
  - these are what you should use in tutorials
- Tools - complete functionality
  - `tool.method()` calls, they are internally used by tasks or can be used on their own
  - sometimes shown in tutorial scripts and CASAGuides
- Applications – some tasks/tools invoke standalone apps
  - e.g. `casaviewer, casaplotms, casabrowser, asdm2MS`
- Shell commands can be run with a leading exclamation mark `!du -ls` or inside `os.system("shell command")`

(some key shell commands like "ls" work without the exclamation mark and we will use `os.system()` exclusively within this tutorial.)

Find the right Task

To see list of tasks organized by type:

tasklist

IPython: rfriesen/SIS18

File Edit View Search Terminal Help

```
CASA <4>: tasklist
--------> tasklist()
Available tasks, organized by category (experimental tasks in parenthesis ()
  deprecated tasks in curly brackets {}).

Import/export      Information        Editing            Manipulation
------------------ ------------------ ------------------ ------------------
exportasdm         imhead             fixplanets         concat
exportfits         imreframe          fixvis             conjugatevis
exportuvfits       imstat             flagcmd            cvel
importasdm         imval              flagdata           fixvis
importatca         listcal            flagmanager        hanningsmooth
importfits         listfits           msview             imhead
importfitsidi      listhistory        plotms             mstransform
importmiriad       listobs                               oldhanningsmooth
importuvfits       listpartition                         oldsplit
importvla          listvis                               partition
(importevla)       plotms                                plotms
(importgmrt)       plotuv                                split
                   vishead                               testconcat
                   visstat                               uvcontsub
                   visstat2                              virtualconcat
                   visstatold                            vishead
                   (asdmsummary)                         (cvel2)
                   (listsdm)                             (statwt)
                   (makemask)                            (uvcontsub3)

Calibration        Modeling           Imaging            Analysis
------------------ ------------------ ------------------ ------------------
accum              predictcomp        clean              imcollapse
applycal           setjy              deconvolve         imcontsub
bandpass           uvcontsub          feather            imdev
blcal              uvmodelfit         ft                 imfit
calstat            uvsub              imcontsub          imhead
clearcal           (uvcontsub3)       (boxit)            imhistory
delmod                                (csvclean)         immath
fixplanets                            (tclean)           immoments
fluxscale                             (tclean2)          impbcor
ft                                    (widebandpbcor)    impv
gaincal                               {mosaic}           imrebin
gencal                                {widefield}        imreframe
initweights                                              imregrid
listcal                                                  imsmooth
plotants                                                 imstat
plotbandpass                                             imsubimage
plotcal                                                  imtrans
polcal                                                   imval
predictcomp                                              listvis
rerefant                                                 rmfit
setjy                                                    slsearch
smoothcal                                                specflux
uvmodelfit                                               specsmooth
```

Find the right Task

To see list of tasks with short help:

taskhelp

```
IPython: rfriesen/SIS18
File  Edit  View  Search  Terminal  Help

CASA <3>: taskhelp
--------> taskhelp()
Available tasks:

accum             : Accumulate incremental calibration solutions into a calibration table
applycal          : Apply calibrations solutions(s) to data
asdmsummary       : Summarized description of an ASDM dataset.
autoclean         : CLEAN an image with automatically-chosen clean regions.
bandpass          : Calculates a bandpass calibration solution
blcal             : Calculate a baseline-based calibration solution (gain or bandpass)
boxit             : Box regions in image above given threshold value.
browsetable       : Browse a table (MS, calibration table, image)
calstat           : Displays statistical information on a calibration table
caltabconvert     : Convert old-style caltables into new-style caltables.
clean             : Invert and deconvolve images with selected algorithm
clearcal          : Re-initializes the calibration for a visibility data set
clearplot         : Clear the matplotlib plotter and all layers
clearstat         : Clear all autolock locks
concat            : Concatenate several visibility data sets.
conjugatevis      : Change the sign of the phases in all visibility columns.
csvclean          : This task does an invert of the visibilities and deconvolve in the image
 plane.
cvel              : regrid an MS to a new spectral window / channel structure or frame
cvel2             : Regrid an MS or MMS to a new spectral window, channel structure or frame
deconvolve        : Image based deconvolver
delmod            : Deletes model representations in the MS
exportasdm        : Convert a CASA visibility file (MS) into an ALMA or EVLA Science Data Mo
del
exportfits        : Convert a CASA image to a FITS file
exportuvfits      : Convert a CASA visibility data set to a UVFITS file:
feather           : Combine two images using their Fourier transforms
find              : Find string in tasks, task names, parameter names:
fixplanets        : Changes FIELD and SOURCE table entries based on user-provided direction
or POINTING table, optionally fixes the UVW coordinates
fixvis            : Recalculates (u, v, w) and/or changes Phase Center
flagcmd           : Flagging task based on batches of flag-commands
flagdata          : All-purpose flagging task based on data-selections and flagging modes/al
gorithms.
flagmanager       : Enable list, save, restore, delete and rename flag version files.
```

NRAO

Task Interface

examine task parameters with **inp** :

IPython: Calibration/test

File Edit View Search Terminal Help

```
CASA <6>: inp
--------> inp()
#  tclean :: Radio Interferometric Image Reconstruction
vis                 =         ''        #  Name of input visibility file(s)
selectdata          =       True        #  Enable data selection parameters
     field          =         ''        #  field(s) to select
     spw            =         ''        #  spw(s)/channels to select
     timerange      =         ''        #  Range of time to select from data
     uvrange        =         ''        #  Select data within uvrange
     antenna        =         ''        #  Select data based on antenna/baseline
     scan           =         ''        #  Scan number range
     observation    =         ''        #  Observation ID range
     intent         =         ''        #  Scan Intent(s)

datacolumn          = 'corrected'       #  Data column to image(data,corrected)
imagename           =         ''        #  Pre-name of output images
imsize              =      [100]        #  Number of pixels
cell                = ['1arcsec']       #  Cell size
phasecenter         =         ''        #  Phase center of the image
stokes              =        'I'        #  Stokes Planes to make
projection          =      'SIN'        #  Coordinate projection (SIN, HPX)
startmodel          =         ''        #  Name of starting model image
specmode            =      'mfs'        #  Spectral definition mode
                                        #   (mfs,cube,cubedata)
     reffreq        =         ''        #  Reference frequency

gridder             = 'standard'        #  Gridding options (standard, wproject,
                                        #   widefield, mosaic, awproject)
     vptable        =         ''        #  Name of Voltage Pattern table
     pblimit        =        0.2        #  >PB gain level at which to cut off
                                        #   normalizations

deconvolver         =   'hogbom'        #  Minor cycle algorithm (hogbom,clark,m
                                        #   ultiscale,mtmfs,mem,clarkstokes)
restoration         =       True        #  Do restoration steps (or not)
     restoringbeam  =         []        #  Restoring beam shape to use. Default
                                        #   is the PSF main lobe
     pbcor          =      False        #  Apply PB correction on the output
                                        #   restored image

outlierfile         =         ''        #  Name of outlier-field image
                                        #   definitions
weighting           =  'natural'        #  Weighting scheme
                                        #   (natural,uniform,briggs)
     uvtaper        =         []        #  uv-taper on outer baselines in uv-
                                        #   plane

niter               =          0        #  Maximum number of iterations
usemask             =     'user'        #  Type of mask(s) for deconvolution
                                        #   (user, pb, auto-thresh, auto-
                                        #   thresh2, or auto-multithresh)
```

Task Interface

- standard tasking interface, similar to AIPS, MIRIAD, etc.
- parameter manipulation commands
  - `inp, default, saveinputs, tget, tput`
- use parameters set as global Python variables

  `<param> = <value>`

  (e.g. `vis = 'ngc5921.demo.ms'`)
- execute

  `<taskname>` or `go` (e.g. `clean()`)

- return values (except when using "go")
  - some tasks return Python dictionaries, assign a variable name to get them, e.g. `myval=imval()`
  - Very useful for scripting based on task outputs

Expandable Parameters

- Boldface parameters have *subparameters* that unfold when main parameter is set

```
IPython: Calibration/test
File  Edit  View  Search  Terminal  Help
CASA <21>: inp
---------> inp()
#  tclean :: Radio Interferometric Image Reconstruction
vis                 = 'SDP81_B4_uncalibrated.ms.split' #  Name of input
                                         #   visibility file(s)
selectdata          =       True         #  Enable data selection parameters
     field          =         ''         #  field(s) to select
     spw            =         ''         #  spw(s)/channels to select
     timerange      =         ''         #  Range of time to select from data
     uvrange        =         ''         #  Select data within uvrange
     antenna        =         ''         #  Select data based on antenna/baseline
     scan           =         ''         #  Scan number range
     observation    =         ''         #  Observation ID range
     intent         =         ''         #  Scan Intent(s)

datacolumn          = 'corrected'        #  Data column to image(data,corrected)
imagename           = 'SDP81_B4_uncalibrated' #  Pre-name of output images
imsize              =      [100]         #  Number of pixels
cell                = ['1arcsec']        #  Cell size
phasecenter         =         ''         #  Phase center of the image
stokes              =        'I'         #  Stokes Planes to make
projection          =      'SIN'         #  Coordinate projection (SIN, HPX)
startmodel          =         ''         #  Name of starting model image
specmode            =      'mfs'         #  Spectral definition mode
                                         #   (mfs,cube,cubedata)
     reffreq        =         ''         #  Reference frequency
```

Parameter Checking

sanity checks of parameters in inp :

```
IPython: SDP81/Calibration
File  Edit  View  Search  Terminal  Help
CASA <20>: inp
---------> inp()
#  tclean :: Radio Interferometric Image Reconstruction
vis                 =         ''        #  Name of input visibility file(s)
selectdata          =       True        #  Enable data selection parameters
     field          =         ''        #  field(s) to select
     spw            =         ''        #  spw(s)/channels to select
     timerange      =         ''        #  Range of time to select from data
     uvrange        =         ''        #  Select data within uvrange
     antenna        =         ''        #  Select data based on antenna/baseline
     scan           =         ''        #  Scan number range
     observation    =         ''        #  Observation ID range
     intent         =         ''        #  Scan Intent(s)

datacolumn          =         ''        #  Data column to image(data,corrected)
imagename           =         ''        #  Pre-name of output images
imsize              = 'MakeItReallyBig' #  Number of pixels
cell                = ['1arcsec']       #  Cell size
phasecenter         =         ''        #  Phase center of the image
stokes              =        'I'        #  Stokes Planes to make
projection          =      'SIN'        #  Coordinate projection (SIN, HPX)
startmodel          =         ''        #  Name of starting model image
specmode            =      'mfs'        #  Spectral definition mode (mfs,cube,cubedata)
     reffreq        =         ''        #  Reference frequency

gridder             = 'standard'        #  Gridding options (standard, wproject, widefield,
                                        #   mosaic, awproject)
     vptable        =         ''        #  Name of Voltage Pattern table
```

erroneous values in red

Help on Tasks

CASAdocs: https://casa.nrao.edu/casadocs/

CASA

Log in | Search Site | Search

only in current section

Home | Latest | Previous Versions

CASA Documentation Homepage

CASA Docs is the official documentation for the CASA software. A new version of CASA Docs is released with every version of CASA...

CASA, the *Common Astronomy Software Applications* package, is the primary data processing software for the Atacama Large Millimeter/submillimeter Array (ALMA) and the Karl G. Jansky Very Large Array (VLA), and has a versatility that also benefits the reduction and analysis of data from other radio telescopes. The CASA package can be used for processing both interferometric and single dish data, and it supports the ALMA, VLA and VLA Sky Survey (VLASS) data-processing pipelines.

The CASA infrastructure is built on top of CASAcore, and consists of a suite of tools that are implemented in C++ and bundled together under an Interactive Python interface as data reduction 'tasks'. This structure provides flexibility to process the data via task interface or as a python script. In addition to the data reduction tasks, many post-processing tools are available for even more flexibility and special purpose reduction needs.

CASA is developed by an international consortium of scientists based at the National Radio Astronomical Observatory (NRAO), the European Southern Observatory (ESO), the National Astronomical Observatory of Japan (NAOJ), the Academia Sinica Institute of Astronomy and Astrophysics (ASIAA), CSIRO Astronomy and Space Science (CSIRO/CASS), and the Netherlands Institute for Radio Astronomy (ASTRON), under the guidance of NRAO.

NRAO

Help on Tasks

CASAdocs: https://casa.nrao.edu/casadocs/latest

CASA

Log in | Search Site | Search
only in current section

Home | Latest | Previous Versions

- Release Information
- Global Task List
- Global Tool List
- CASA Fundamentals
- Using CASA
- Calibration & Visibilities
- Imaging & Analysis
- Pipeline
- Simulations
- Parallel Processing
- Memo Series
- CASA Development

CASA 5.6

Warning to all VLA users, 11 Feb 2020: A bug was found in the version of the pipeline that was packaged with CASA 5.6.2-2 (the initial CASA 5.6.2 version for VLA). This bug impacts the restoration of calibrated MSs, and affects all VLA projects that restored pipeline calibration after October 21, 2019. If an existing calibration was restored using the new interface to the NRAO archive or following the instructions on the pipeline web-pages, you will obtain incorrect results. The output from a full calibration run of the pipeline is not affected by this bug, nor is data restored prior to October 21 2019, or data restored with a different version of CASA. As per Feb 19, a new CASA 5.6.2-3 version is available in which the problem has been fixed. The pipeline team will update affected users.

CASA, the *Common Astronomy Software Applications*, is the primary data processing software for the Atacama Large Millimeter/submillimeter Array (ALMA) and Karl G. Jansky Very Large Array (VLA), and is often used also for other radio telescopes.

The CASA 5.6 series includes three official CASA releases to date, CASA 5.6.0, 5.6.1, and 5.6.2, which can be downloaded here. All three 5.6 versions are suited for manual data processing. CASA 5.6.1 has also been scientifically validated for ALMA and includes the ALMA Cycle 7 pipeline, while CASA 5.6.2 includes the latest version of the pipeline that has been scientifically validated for the VLA.

The CASA 5.6 releases build on CASA 5.5, but have the following main new features:

New Features

- Task **bandpass** now supports relative frequency-dependent interpolation when applying bandpass tables.
- For **fringefit**, a new keyword *niter* determines a maximum number of iterations for the global least squares solver.
- In **tclean**, a new parameter *'smallscalebias'* for *deconvolver = 'mtmfs'* more efficiently cleans signal on different spatial scales.
- **plotms** can show atmospheric and sky temperature curves, enhanced pointing plots and more accurate axis labels.
- In **tclean**, the auto-multithresh now functions with polarization data.

NRAO

Help on Tasks

Documentation inside CASA:

`doc "tclean"`

Home | Latest | Previous Versions

- Release Information
- **Global Task List**
- Global Tool List
- CASA Fundamentals
- Using CASA
- Calibration & Visibilities
- Imaging & Analysis
- Pipeline
- Simulations
- Parallel Processing
- Memo Series
- CASA Development

tclean | Description | Parameters | Changelog | Examples | Developer | Planning

Description

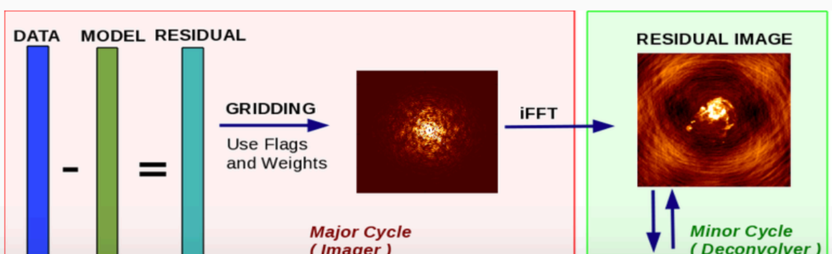
task tclean description

Overview

The **tclean** task forms images from visibilities and reconstructs a sky model.

tclean handles continuum images and spectral line cubes, supports outlier fields, contains point-source CLEAN based algorithms as well as options for multi-scale and wideband image reconstruction, widefield imaging correcting for the w-term, full primary-beam imaging and joint mosaic imaging (with heterogeneous array support for ALMA). Parallelization of the major cycle is also available.

Image reconstruction in CASA typically comprises an outer loop of *major cycles* and an inner loop of *minor cycles*. The major cycle implements transforms between the data and image domains and the minor cycle operates purely in the image domain. Together, they implement an iterative weighted χ^2 minimization that solves the measurement equation. Minor cycle algorithms can have their own (different) optimization schemes and the imaging framework and task interface allow for considerable freedom in choosing options separately for each step of the process.



Task Execution

- In addition to typing in all variables in the task interface and executing with **go** one can write the full parameter set in a line:

```
taskname( arg1=val1, arg2=val2, ... )
```

  e.g.

```
tclean(vis='input.ms',imagename='galaxy',
robust=0.5, imsize=[200,200])
```

  - unspecified parameters will be set to their *default* values (globals not used; i.e. not to previously set variables)
  - Useful in scripts, but also in 'pseudo-scripts':
    - To keep a record it is frequently a good idea to write down the full line as above in an editor, then cut and paste into CASA.
    - When changes are needed, change in editor and cut and paste again. That is good practice to keep a record of the exact input.
    - But note that the logger is also repeating the full task command

NRAO

What is a Measurement Set?

- CASA stores u-v data in directories called “Measurement Sets”
  TO DELETE THEM USE `rmtables(“measurement_set.ms”)` or `os.system(”rm –rf measurement_set.ms”)`

- These data sets store two copies of the data (called “columns”):

“Data” Column	“Corrected” Column
Contains the raw, unprocessed measurements.	Usually created by applying one or more calibration terms to the data.

- Additionally a “model” may be stored separately.
  THIS IS USED TO CALCULATE WHAT THE TELESCOPE SHOULD HAVE OBSERVED.

- Each data point may also be “flagged,” i.e., marked bad.
  IN THIS CASE IT IS IGNORED (TREATED AS MISSING) BY CASA OPERATIONS.

Outline

- Short introduction to CASA and the Python interface
  - How to use tasks
  - What is a measurement set?
- **The Flow of Calibration**
- Overview of your Directory
  - Data preparation and set up
  - Getting oriented with your data
- Data Calibration
- Data Inspection and Flagging
- Basic Imaging

Steps to a Calibrated Dataset

Correct for System Temperature, WVR (Water Vapor), Antenna Positions
IMPROVES SHORT TERM VARIABILITY OF PHASE, DATA WEIGHTS AND FLUX SCALE

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna
ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE TIME VARIABILITY

Calibrate the Amplitude and Phase vs. Time of Each Antenna
ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE FREQ. VARIABILITY

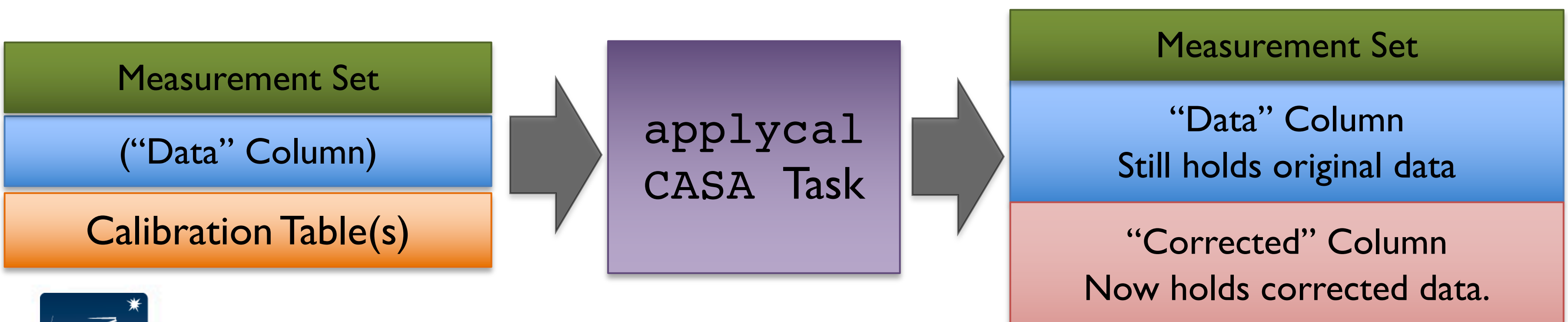
Set the Absolute Amplitude Scale With Reference to a Known Source
PLANET (MODELED), MONITORED QUASAR, ETC.

Apply all corrections to produce calibrated data

Applying Calibration in Practice: Calibration Tables

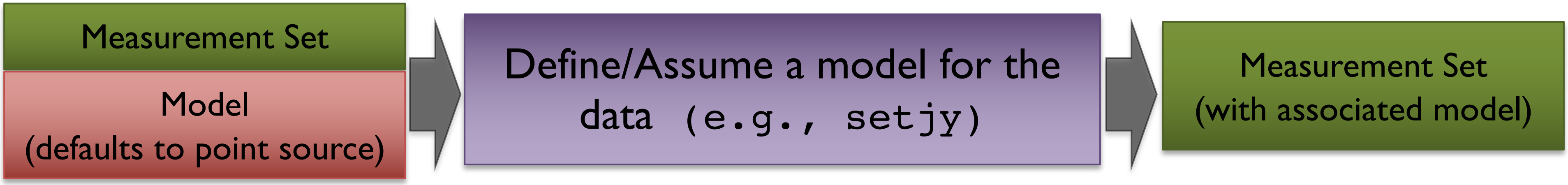
- Calibration yields estimates of phase and amplitude corrections.
  E.G., AS A FUNCTION OF TELESCOPE, TIME, FREQUENCY, POLARIZATION.
- CASA stores these corrections in directories called "calibration tables."
  TO DELETE THEM USE `rmtables("my_table.gcal")`

  OR `os.system("rm -rf my_table.gcal")`
- These are created by calibration tasks:
  E.G., `gaincal, bandpass, gencal`
- Applied via "`applycal`" to the data column and saved as corrected.



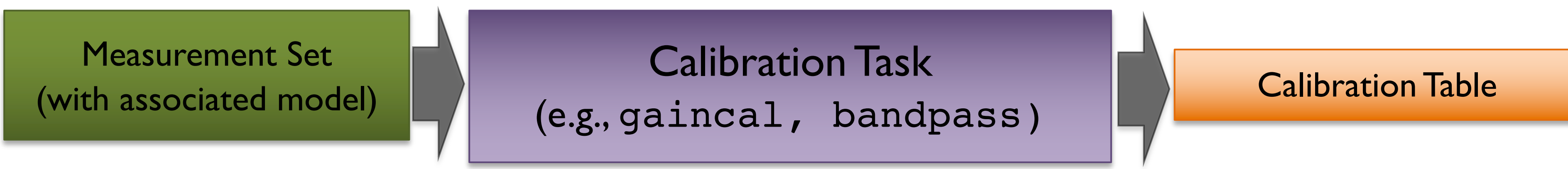
Basic Flow to Create/Apply a Calibration Table

Define what the telescope SHOULD have seen.



Basic Flow to Create/Apply a Calibration Table

Derive the corrections needed to make the data match the model.



Basic Flow to Create/Apply a Calibration Table

Apply these corrections to derive the corrected (calibrated) data.

Basic Flow to Create/Apply a Calibration Table

CASA
Common Astronomy Software Applications

Define what the telescope SHOULD have seen.

Measurement Set
Model
(defaults to point source)
→ Define/assume a model for the data (e.g., `setjy`)
→ Measurement Set
(with associated model)

Derive the corrections needed to make the data match the model.

Measurement Set
(with associated model)
→ Calibration Task
(e.g., `gaincal`, `bandpass`)
→ Calibration Table

Apply these corrections to derive the corrected (calibrated) data.

Measurement Set
Data Column
Calibration Table
→ Apply Calibration
`applycal`
→ Measurement Set
Corrected column now holds calibrated data.

NRAO

Steps to a Calibrated Data set

Correct for System Temperature, WVR (Water Vapor), Antenna Positions
IMPROVES SHORT TERM VARIABILITY OF PHASE, DATA WEIGHTS AND FLUX SCALE

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna
ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE TIME VARIABILITY

Calibrate the Amplitude and Phase vs. Time of Each Antenna
ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE FREQ. VARIABILITY

Set the Absolute Amplitude Scale With Reference to a Known Source
PLANET (MODELED), MONITORED QUASAR, ETC.

Apply all corrections to produce calibrated data

Steps to a Calibrated Data set

Correct for System Temperature, WVR (Water Vapor), Antenna Positions
`gencal, wvrgcal`

Tsys, WVR, Antenna Correction Tables

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna
`bandpass`

Bandpass Calibration Table

Calibrate the Amplitude and Phase vs. Time of Each Antenna
`gaincal`

Phase Calibration Table
Amplitude Calibration Table

Set the Absolute Amplitude Scale With Reference to a Known Source
`fluxscale`

Flux Calibration Table

Apply all corrections to produce calibrated data
`applycal`

Measurement Set

Corrected column now holds calibrated data.

Our Goal Today: Calibrate and Image the data for the Gravitationally Lensed Galaxy SDP.81

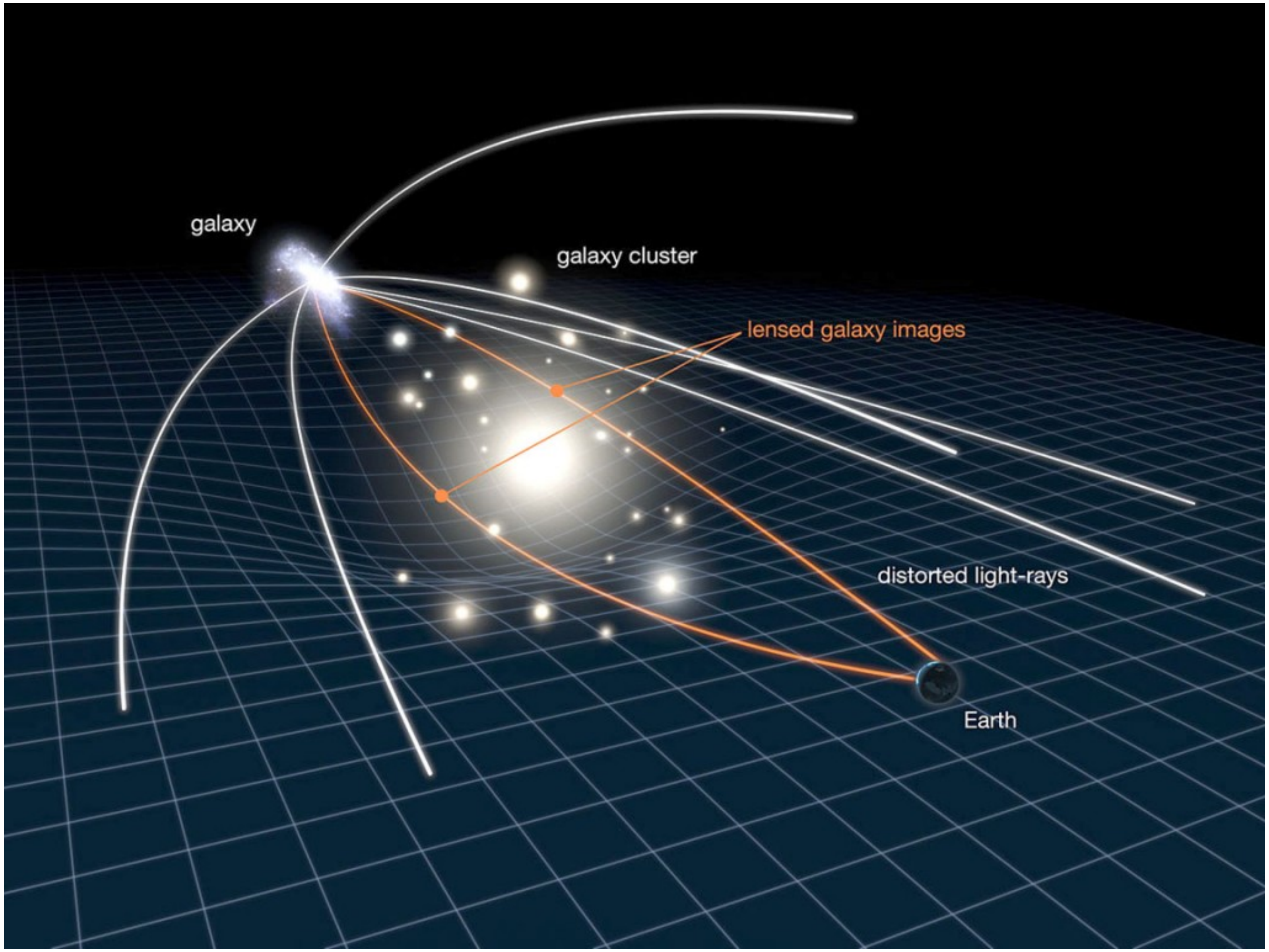
ALMA Long Baseline Campaign

- Successful test of ALMA's longest baselines (i.e. highest resolutions) run from September through December 2014
- Baselines out to 15km (resolution up to 0.023")

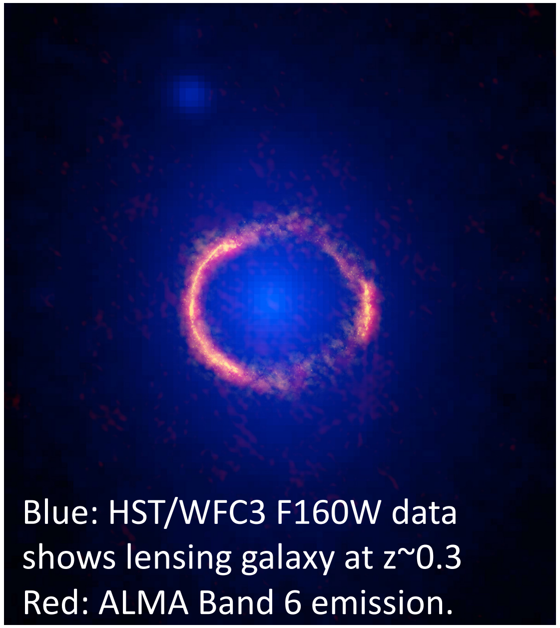
The Gravitationally Lensed Galaxy SDP.81

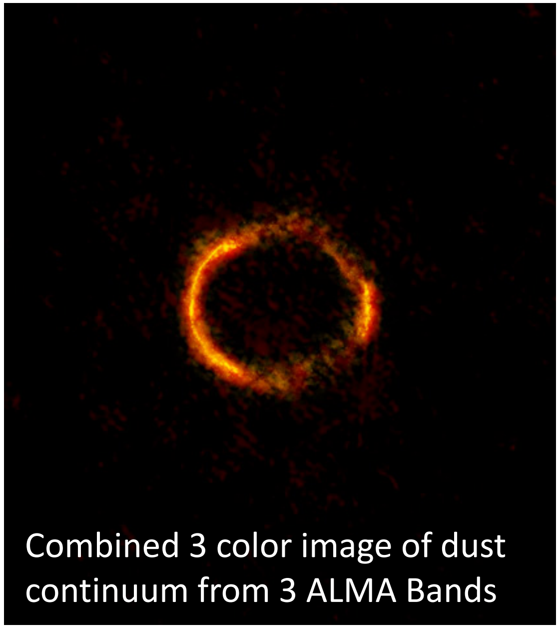
- At z = 3.04, the star-forming galaxy SDP.81 sits behind a massive foreground elliptical galaxy (z = 0.299) which acts as a gravitational lens.
- During the Long Baseline Campaign, the dust continuum at 151, 236, and 290 GHz was mapped as well as emission lines from CO and water.
- These images allow for the determination of the physical and chemical properties of the lensed galaxy down to 180 pc size scales (similar to giant molecular clouds in the Milky Way … but at a redshift of 3!)

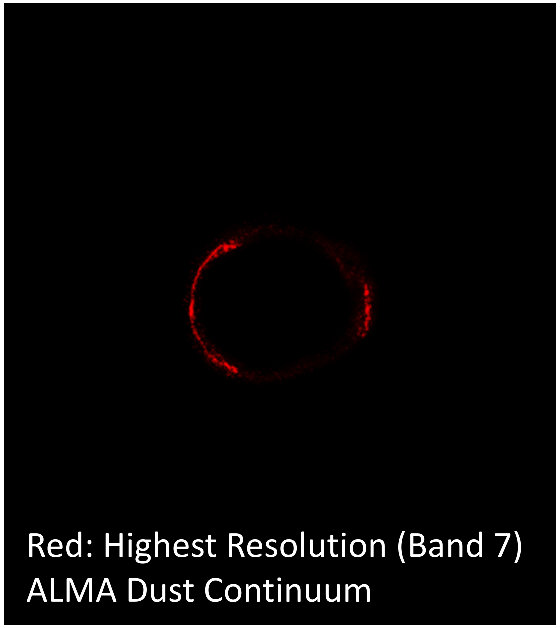
Our Goal Today: Calibrate and Image the data for the Gravitationally Lensed Galaxy SDP.81



Our Goal Today: Calibrate and Image the data for the Gravitationally Lensed Galaxy SDP.81







We will image the dust continuum emission and CO line emission observed at Band 4.

Link to paper: http://arxiv.org/abs/1503.02652

Image Credits: ALMA (NRAO/ESO/NAOJ); B. Saxton NRAO/AUI/NSF; NASA/ESA Hubble, T. Hunter (NRAO)

NRAO

Outline

- Short introduction to CASA and the Python interface
  - How to use tasks
  - What is a measurement set?
- The Flow of Calibration
- **Overview of your Directory**
  - Data preparation and set up
  - Getting oriented with your data
- Data Calibration
- Data Inspection and Flagging
- Basic Imaging

An Overview of your Directory

In your home directory there should be two sub-directories labeled /Calibration and /Imaging.

In **/Calibration** you should have:

- **SDP81_B4_uncalibrated.ms.split** (the data file containing uncalibrated data with minor initial processing applied)
- **data_prep.py** (script detailing the initial processing that has already been applied)
- **calibration.py** (the script we will work through together to calibrate the data)

In **/Imaging** you have:

- **SDP.81_Band4_continuum.ms** (fully calibrated continuum measurement set ready for imaging)
- **SDP.81_Band4.ms** (fully calibrated measurement set containing both continuum and line emission ready for imaging)
- **SDP.81_Band4_COline.ms.contsub** (fully calibrated line-only measurement set)
- **imaging.py** (the script we will work through together to image the data)
- **combination.py** (a script detailing the steps taken to create the measurement sets ready for imaging: this is just for reference **we won't be using it!)**

An Overview of your Directory

To begin, if you haven't already done so … start casa:

```
casa
```

Note that you can run system commands from within casa via:

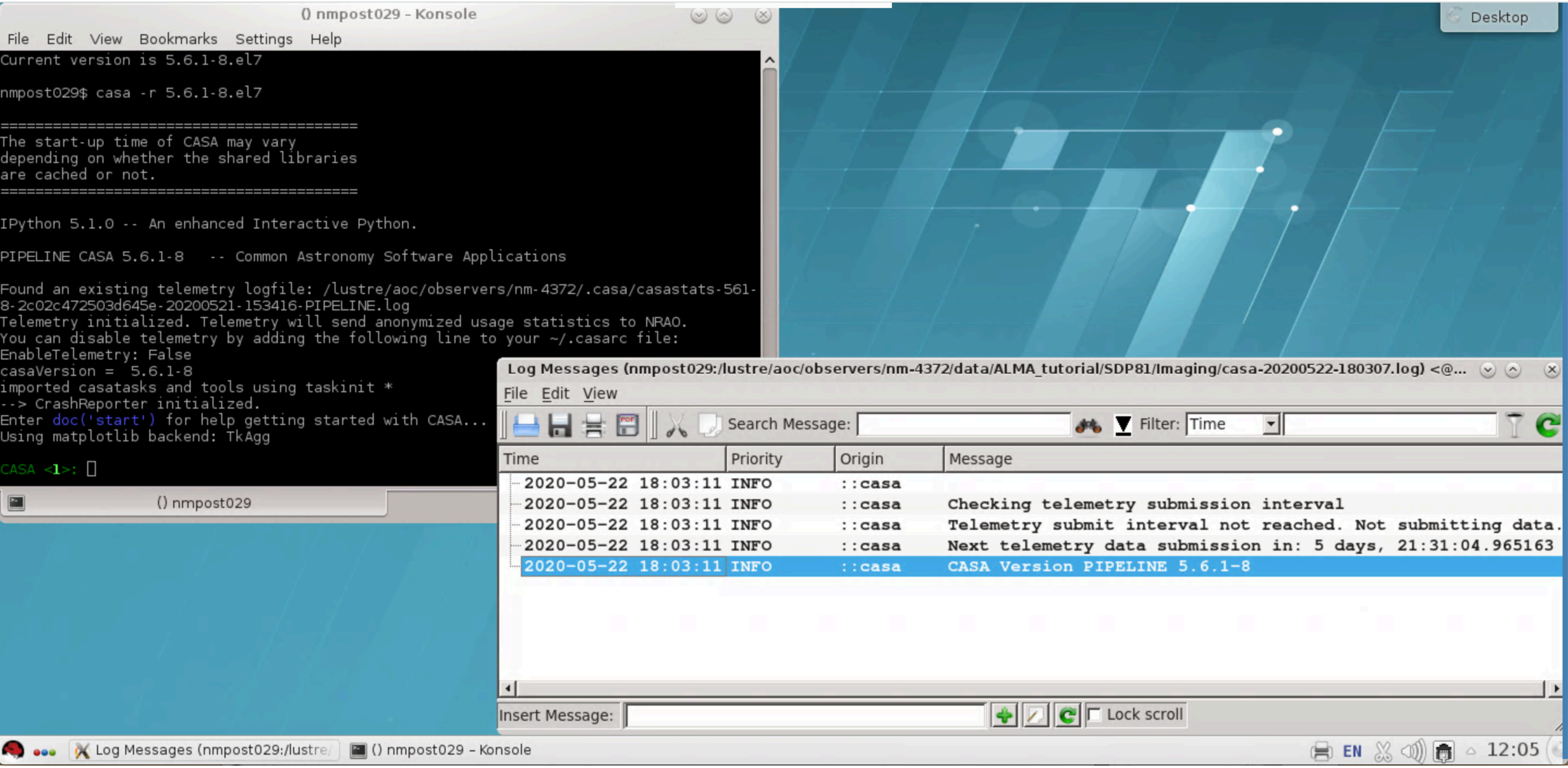
```
os.system("ls")

!ls
```

The dataset we will be working with is large, so there is likely not enough memory to save the data at various steps throughout the reduction process. Should your dataset get corrupted, you can grab a new copy from the main repository to start fresh either at the start of calibration.py or at the start of the imaging.py:

```
os.system("rm -rf SDP81_B4_uncalibrated.ms.split")

os.system("tar xvf /lustre/aoc/siw/nrao/ALMA/SDP81/
SDP81_B4_uncalibrated.ms.split.tgz")
```

Be sure you have run all of the commands in *Startup*

When you start casa …



NRAO

Initial Data Preparation

Downloading data from the ALMA archive will return raw data along with the scripts necessary for calibrating the data. In the interest of time, we have already applied some initial corrections to the raw data for you. All of these steps are detailed in

data_prep.py

Here we will briefly explain the steps taken in data_prep.py

- Import the raw data into a casa measurement set.
- Occasionally a dataset will require a fix to some of the metadata (i.e. the header). In this case, some coordinates in the metadata are adjusted.
- Data that is known to be irrelevant to calibration or to be problematic (even without inspection of the data) is flagged. Examples: data taken when the telescope was not yet on source yet, when the system temperature load was too close to the beam, when the receivers were not yet tuned)
- Create 3 correction tables (WVR, Tsys, antenna positions) and apply them.
- The output of data_prep.py is SDP81_B4_uncalibrated.ms.split

  (we will start calibration with this data file)

ALMA Online Corrections

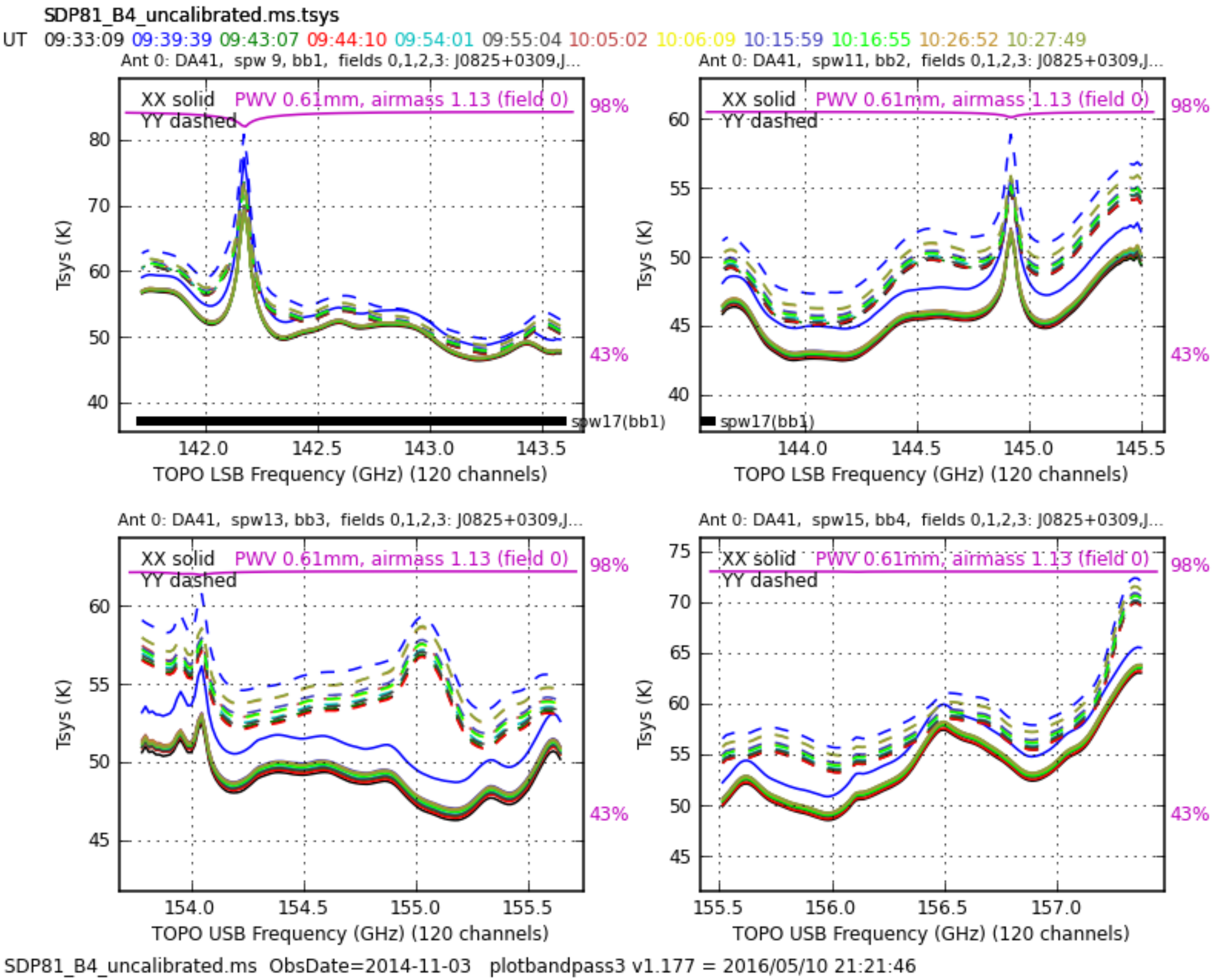
- Water Vapor Radiometer (WVR) – phase delay due to atmosphere
  - Key to correct short-timescale phase variations
  - Phase calibration, variable with time
- System Temperature (Tsys) – atmospheric emission/opacity
  - Key to gain transfer across elevation
  - Amplitude calibration, variable with frequency (observed in "TDM")
  - System temperatures of order ~100 K at Band 3 to ~1000 K at Band 9
- Antenna Positions – updates in accuracy of antenna positions

These corrections are provided by the observatory for each dataset.

The datasets associated with this tutorial already have these corrections applied and the steps are detailed in data_prep.py only for reference.

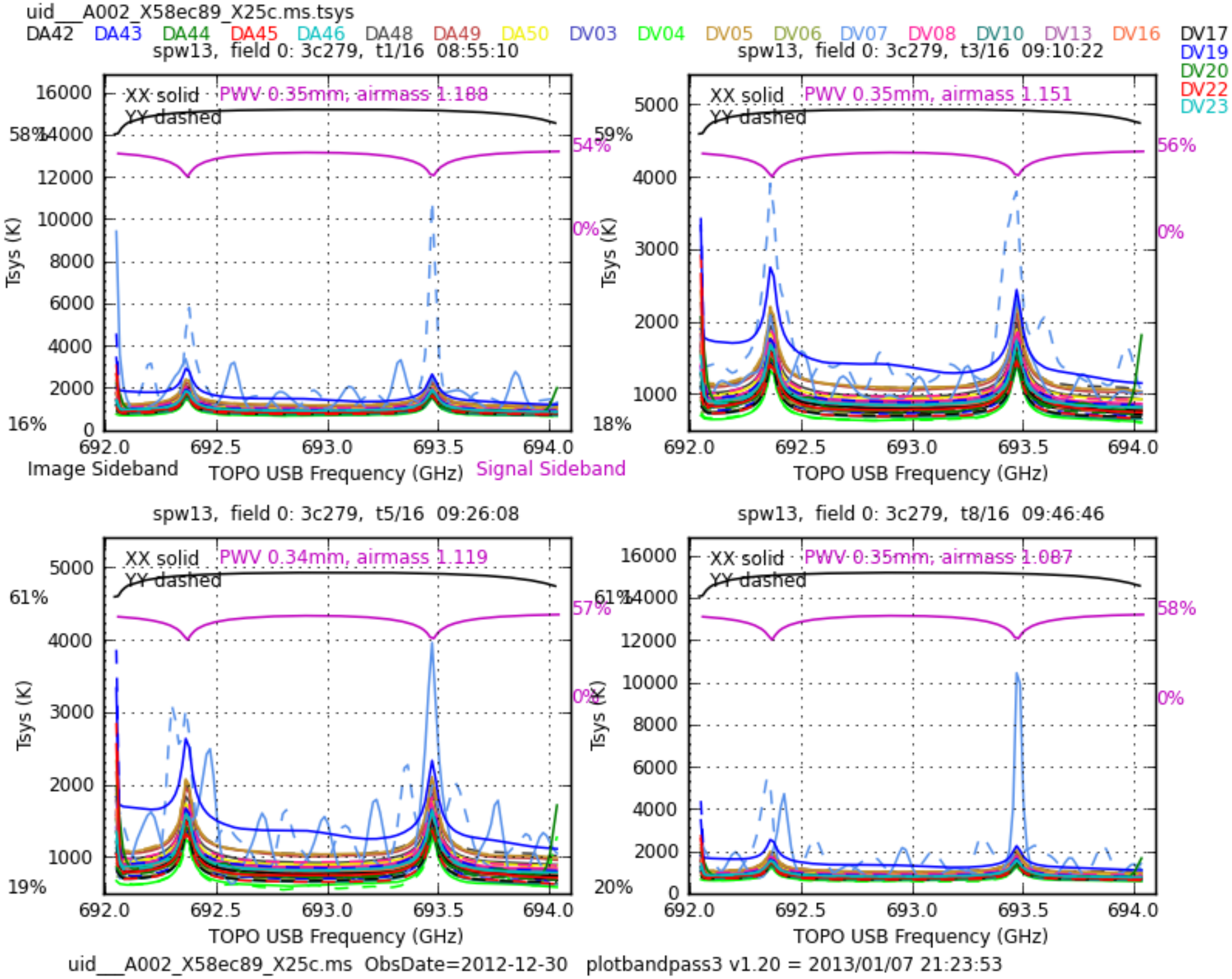
ALMA Online Corrections: Tsys

SDP.81



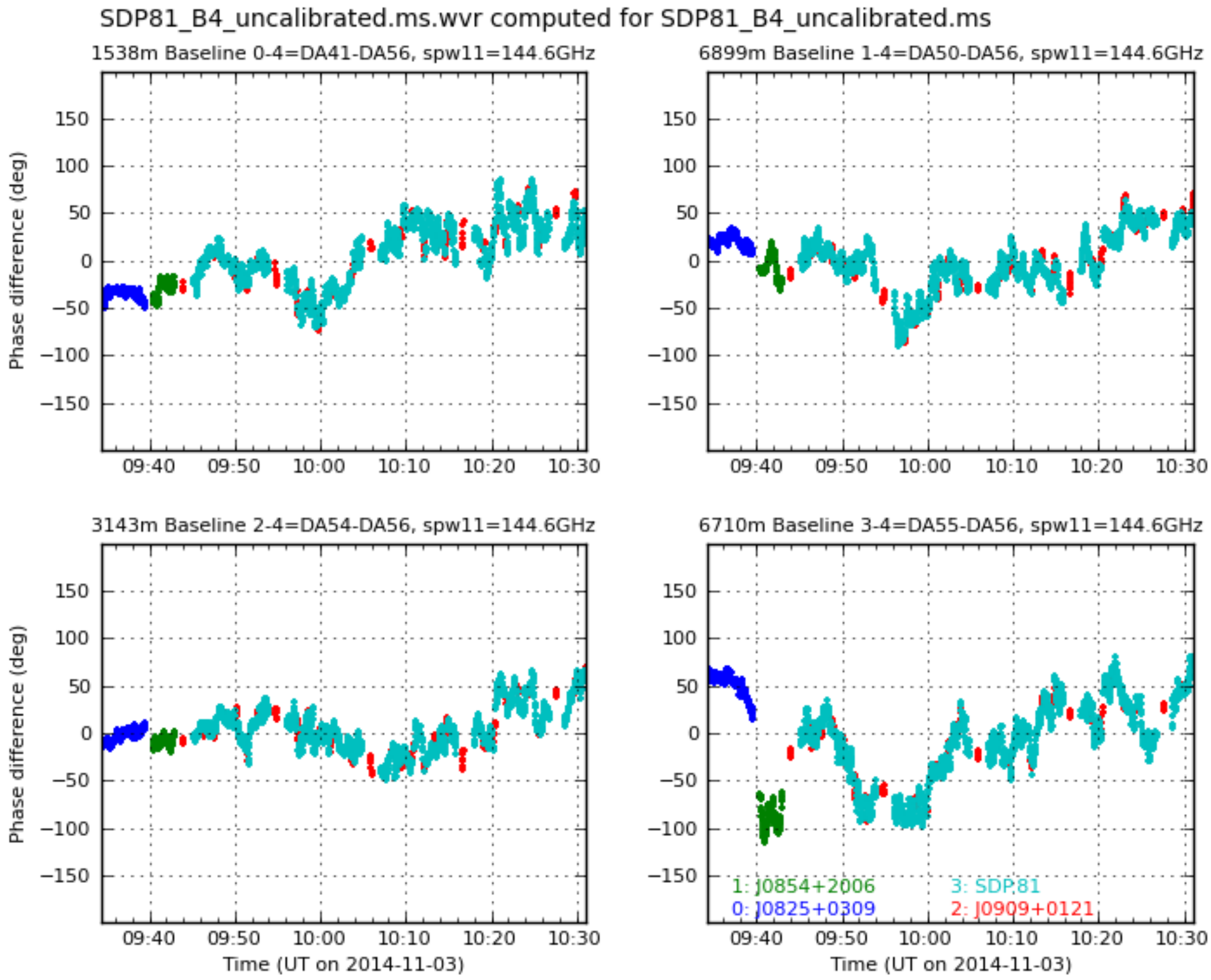
ALMA Online Corrections: Tsys

High Frequency Example: TW Hydra
(note much higher system temperatures)



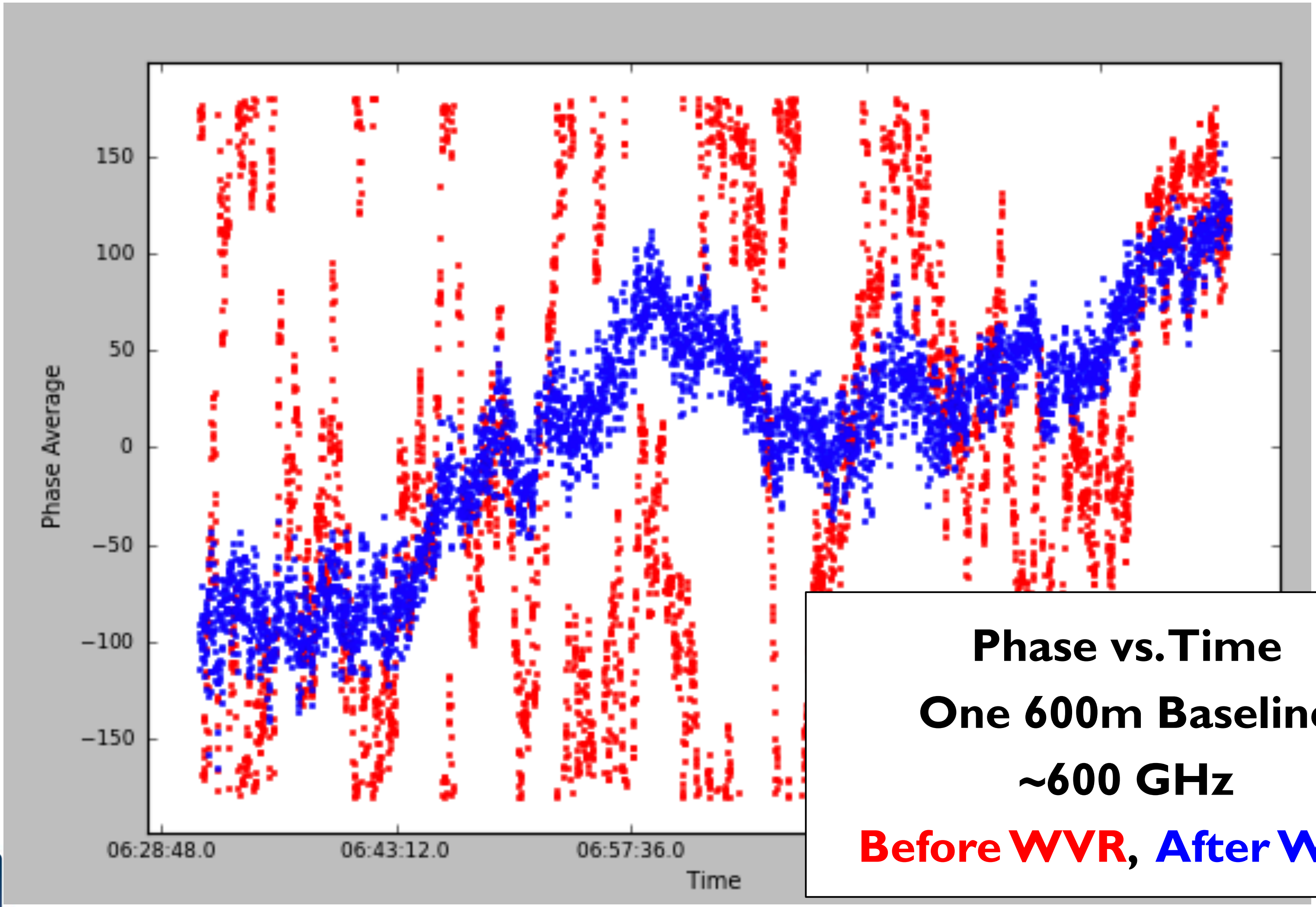
ALMA Online Corrections: WVR

SDP.81



ALMA Online Corrections: WVR

High Frequency Example: TW Hydra



Phase vs. Time

One 600m Baseline

~600 GHz

Before WVR, After WVR

ALMA Online Corrections: Antenna Positions

SDP.81: These are the offsets determined for our dataset.

```
# antenna x_offset y_offset z_offset total_offset baseline_date
 # DV14    -4.61575e-04   7.57190e-04   1.74002e-03  1.95296e-03    2014-10-31 11:27:40
 # DA50     4.24031e-05  -4.98282e-04   1.51997e-03  1.60012e-03    2014-10-31 11:27:40
 # DV22    -9.64679e-04   1.07473e-03   3.88599e-04  1.49554e-03    2014-10-31 11:27:40
 # DV08     5.53798e-04  -1.32566e-03   2.52869e-04  1.45877e-03    2014-10-31 11:27:40
 # DA64    -2.80747e-04   2.60536e-04   1.39146e-03  1.44321e-03    2014-10-31 11:27:40
 # DA54     7.92693e-04  -1.16213e-03  -4.01242e-05  1.40731e-03    2014-10-31 11:27:40
 # DA62     1.95323e-04  -4.82360e-06   1.32798e-03  1.34227e-03    2014-10-31 11:27:40
 # DV17     1.09515e-04  -3.07546e-04   1.20603e-03  1.24944e-03    2014-10-31 11:27:40
 # DV04     3.70800e-04  -4.36427e-04   4.07359e-04  7.02782e-04    2014-10-31 11:27:40
 # DA41     5.09151e-04  -3.88547e-04   1.20386e-04  6.51687e-04    2014-10-31 11:27:40
```

Note: these offsets are in units of meters!!

Getting Oriented

Run the listobs task (output sent to casalogger)

```
listobs("SDP81_B4_uncalibrated.ms.split")
```

==

MeasurementSet Name: SDP81_B4_uncalibrated.ms.split MS Version

==

Timerange (UTC)	Scan	FldId	FieldName	nRows	SpwIds	Average Interval(s)	ScanIntent
09:33:43.0 - 09:33:58.5	2	0	J0825+0309	23400	[0,1,2]	[0.48, 0.48, 0.48]	[CALIBRATE_ATMOSPHERE,CALIBRATE_WVR]
09:34:19.2 - 09:39:35.9	3	0	J0825+0309	195000	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_BANDPASS,CALIBRATE_WVR]
09:39:53.7 - 09:40:09.3	4	1	J0854+2006	23400	[0,1,2]	[0.48, 0.48, 0.48]	[CALIBRATE_ATMOSPHERE, CALIBRATE_WVR]
09:40:24.8 - 09:43:02.6	5	1	J0854+2006	97500	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_AMP,CALIBRATE_FLUX,CALIBRATE_WVR]
09:43:20.9 - 09:43:36.5	6	2	J0909+0121	23400	[0,1,2]	[0.48, 0.48, 0.48]	[CALIBRATE_ATMOSPHERE,CALIBRATE_WVR]
09:43:54.3 - 09:44:04.4	7	2	J0909+0121	6500	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_PHASE,CALIBRATE_WVR]
09:44:20.0 - 09:44:35.5	8	3	SDP.81	23400	[0,1,2]	[0.48, 0.48, 0.48]	[CALIBRATE_ATMOSPHERE,CALIBRATE_WVR]
09:45:08.1 - 09:46:12.1	9	3	SDP.81	39000	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[OBSERVE_TARGET#ON_SOURCE]
09:46:14.1 - 09:46:24.2	10	2	J0909+0121	6500	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_PHASE,CALIBRATE_WVR]
09:46:25.7 - 09:47:29.8	11	3	SDP.81	39000	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[OBSERVE_TARGET#ON_SOURCE]
09:47:31.8 - 09:47:41.9	12	2	J0909+0121	6500	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_PHASE,CALIBRATE_WVR]
09:47:43.4 - 09:48:47.4	13	3	SDP.81	39000	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[OBSERVE_TARGET#ON_SOURCE]
09:48:49.4 - 09:48:59.5	14	2	J0909+0121	6500	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_PHASE,CALIBRATE_WVR]
09:49:01.1 - 09:50:05.1	15	3	SDP.81	39000	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[OBSERVE_TARGET#ON_SOURCE]
09:50:07.1 - 09:50:17.2	16	2	J0909+0121	6500	[0,1,2,3]	[2.02, 2.02, 2.02, 2.02]	[CALIBRATE_PHASE,CALIBRATE_WVR]

Getting Oriented

Run the listobs task

```
listobs("SDP81_B4_uncalibrated.ms.split")
```

==

MeasurementSet Name: SDP81_B4_uncalibrated.ms.split   MS Version

==

Fields: 4

ID	Code	Name	RA	Decl	Epoch	SrcId	nRows
0	none	J0825+0309	08:25:50.338355	+03.09.24.52006	J2000	0	218400
1	none	J0854+2006	08:54:48.874929	+20.06.30.64088	J2000	1	120900
2	none	J0909+0121	09:09:10.091592	+01.21.35.61768	J2000	2	318500
3	none	SDP.81	09:03:11.610000	+00.39.06.70000	J2000	3	1287000

Spectral Windows: (4 unique spectral windows and 1 unique polarization setups)

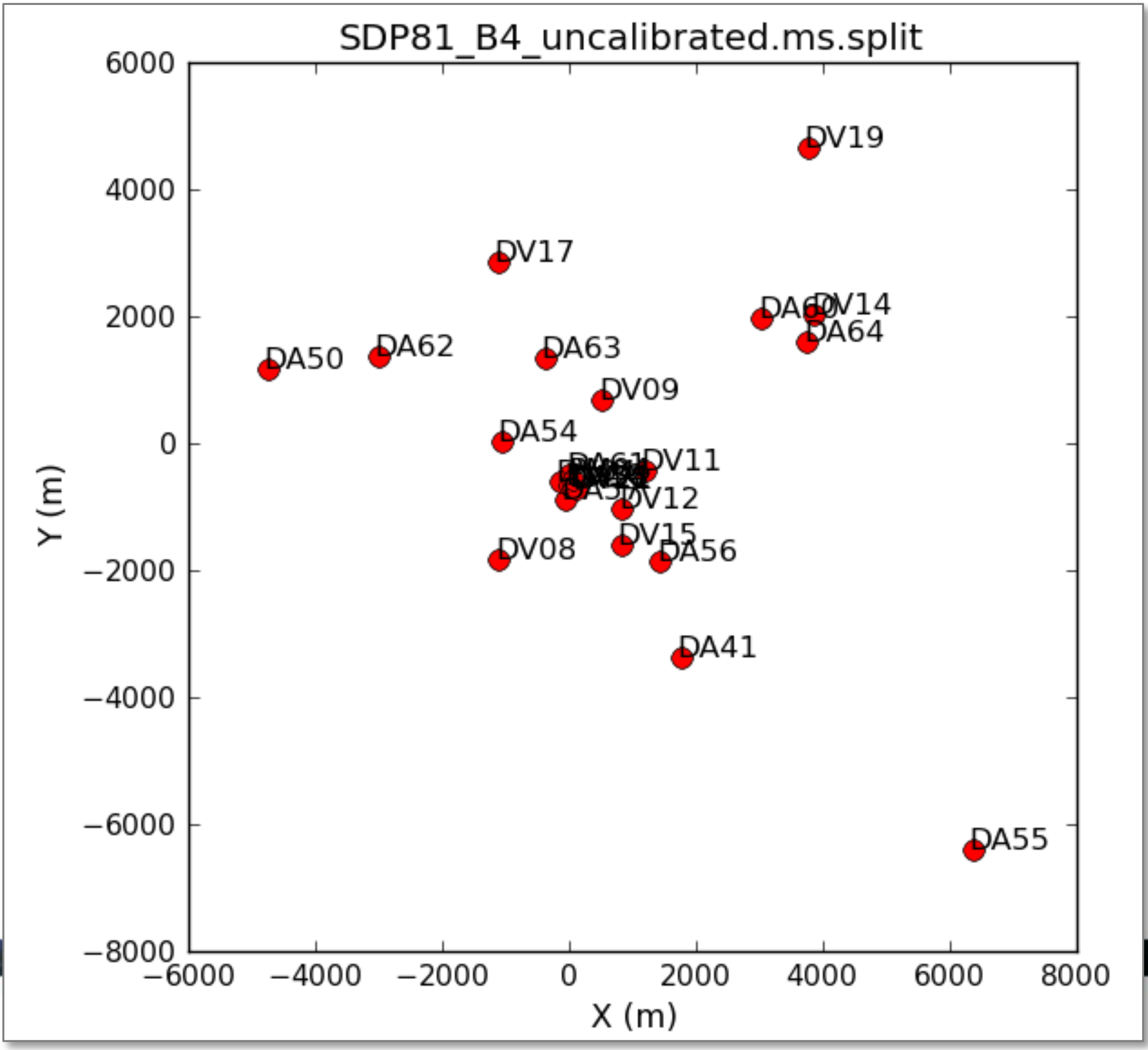
SpwID	Name	#Chans	Frame	Ch0(MHz)	ChanWid(kHz)	TotBW(kHz)	CtrFreq(MHz)	Num	Corrs
0	ALMA_RB_04#BB_2	64	TOPO	145550.922	-31250.000	2000000.0	144566.5468	2	XX YY
1	ALMA_RB_04#BB_3	64	TOPO	153727.218	31250.000	2000000.0	154711.5928	3	XX YY
2	ALMA_RB_04#BB_4	64	TOPO	155459.988	31250.000	2000000.0	156444.3626	4	XX YY
3	ALMA_RB_04#BB_1	1920	TOPO	143586.559	-976.562	1875000.0	142649.5468	1	XX YY

NRAO

Getting Oriented

Run the plotants task

```
plotants("SDP81_B4_uncalibrated.ms.split",
          figfile="plotants.png")
```



plotms

A general-purpose graphical interface for plotting and flagging UV data

Can be started in the usual *casapy* interface:

```
inp plotms
```

Can be fully specified in the CASA command line (e.g.):

```
plotms(vis=“SDP81_B4_uncalibrated.ms.split",
       xaxis="time", yaxis="amp", ydatacolumn="corrected",
       field="0,1,2", averagedata=T, avgchannel="1e3",
       avgtime="1e3", coloraxis="field")
```

Also can be started directly from the unix prompt:

```
% casaplotms
```

Getting Oriented

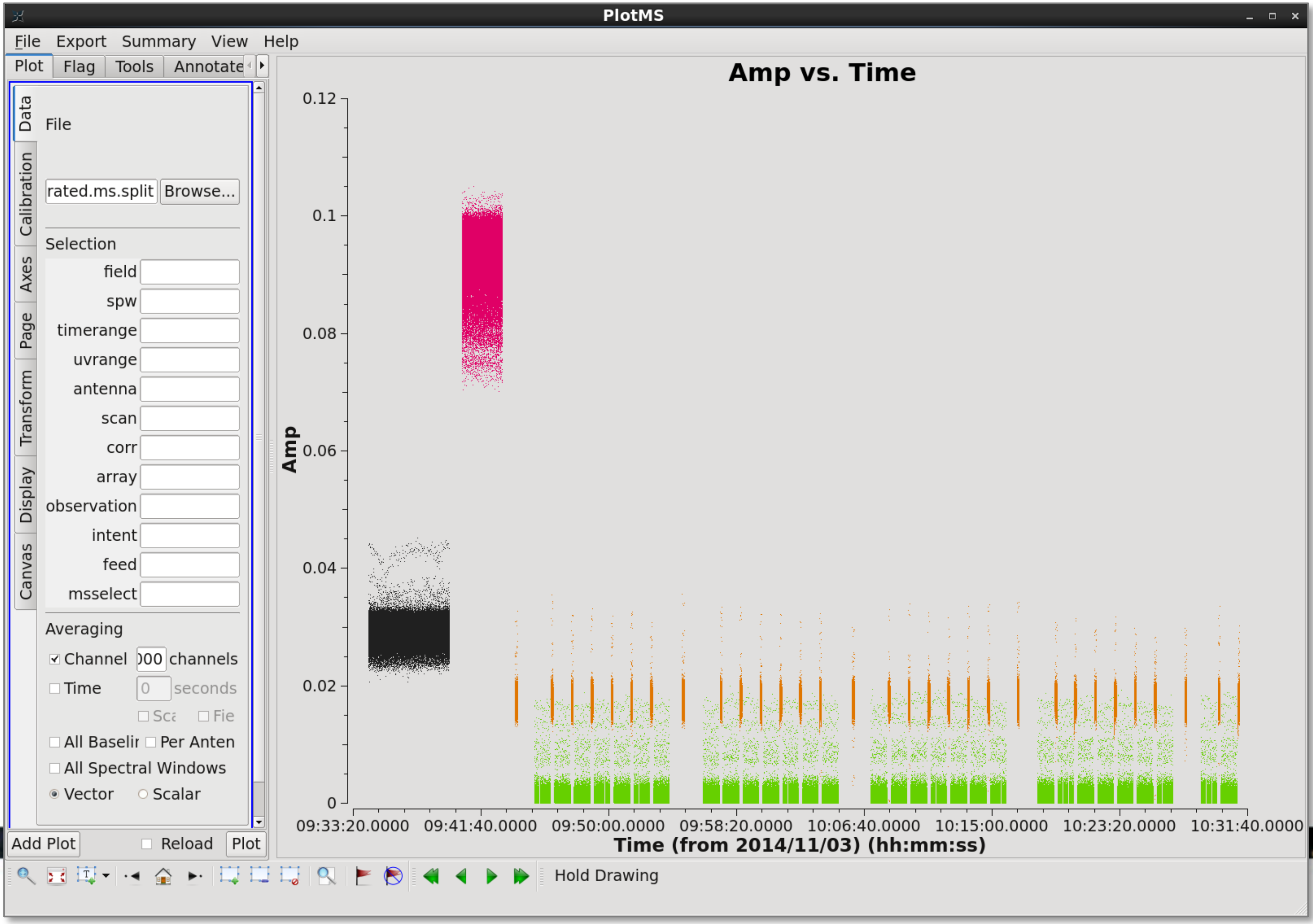
inp plotms

```
CASA <35>: inp plotms
---------> inp(plotms)
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 = 'SDP81_B4_uncalibrated.ms.split' #  Input MS (or CalTable) (blank for
                                                       #   none)
gridrows            =          1        #  Number of subplot rows
gridcols            =          1        #  Number of subplot columns
rowindex            =          0        #  Row location of the plot (0-based)
colindex            =          0        #  Column location of the plot (0-based)
plotindex           =          0        #  Index to address a subplot (0-based)
xaxis               =         ''        #  Plot x-axis (blank for default/current)
yaxis               =         ''        #  Plot y-axis (blank for default/current)
selectdata          =       True        #  Data selection parameters
     field          =         ''        #  Field names or field index numbers (blank for all)
     spw            =         ''        #  Spectral windows:channels (blank for all)
     timerange      =         ''        #  Time range (blank for all)
     uvrange        =         ''        #  UV range (blank for all)
     antenna        =         ''        #  Antenna/baselines (blank for all)
     scan           =         ''        #  Scan numbers (blank for all)
     correlation    =         ''        #  Correlations (blank for all)
     array          =         ''        #  (Sub)array numbers (blank for all)
     observation    =         ''        #  Observation IDs (blank for all)
     intent         =         ''        #  Observing intent (blank for all)
     feed           =         ''        #  Feed numbers (blank for all)
     msselect       =         ''        #  MS selection (blank for all)

averagedata         =       True        #  Data averaging parameters
     avgchannel     =         ''        #  Average over channel (blank = False, otherwise
                                        #   value in channels)
     avgtime        =         ''        #  Average over time (blank = False, otherwise value
                                        #   in seconds)
     avgscan        =      False        #  Average over scans. Only valid with time averaging
     avgfield       =      False        #  Average over fields. Only valid with time
                                        #   averaging
     avgbaseline    =      False        #  Average over all baselines (mutually exclusive
                                        #   with avgantenna)
     avgantenna     =      False        #  Average per antenna (mutually exclusive with
                                        #   avgbaseline)
     avgspw         =      False        #  Average over all spectral windows
     scalar         =      False        #  Scalar averaging (False=vector averaging)

transform           =      False        #  Transform data in various ways
extendflag          =      False        #  Extend flagging to other data points
iteraxis            =         ''        #  The axis over which to iterate
```

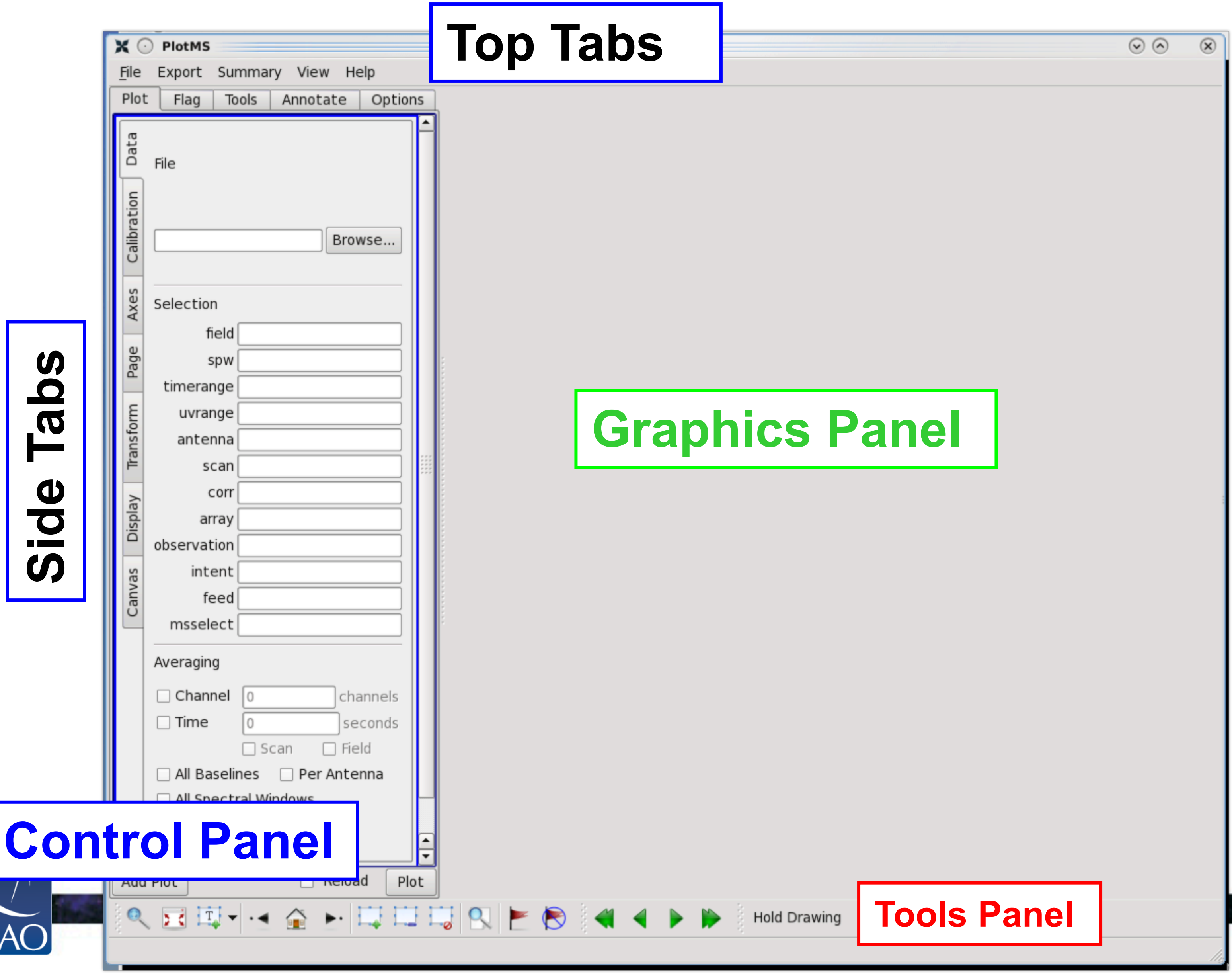
NRAO

Getting Oriented

```
plotms(vis="SDP81_B4_uncalibrated.ms",
        xaxis="time", yaxis="amp", averagedata=True,
        avgchannel="1e3", coloraxis="field")
```



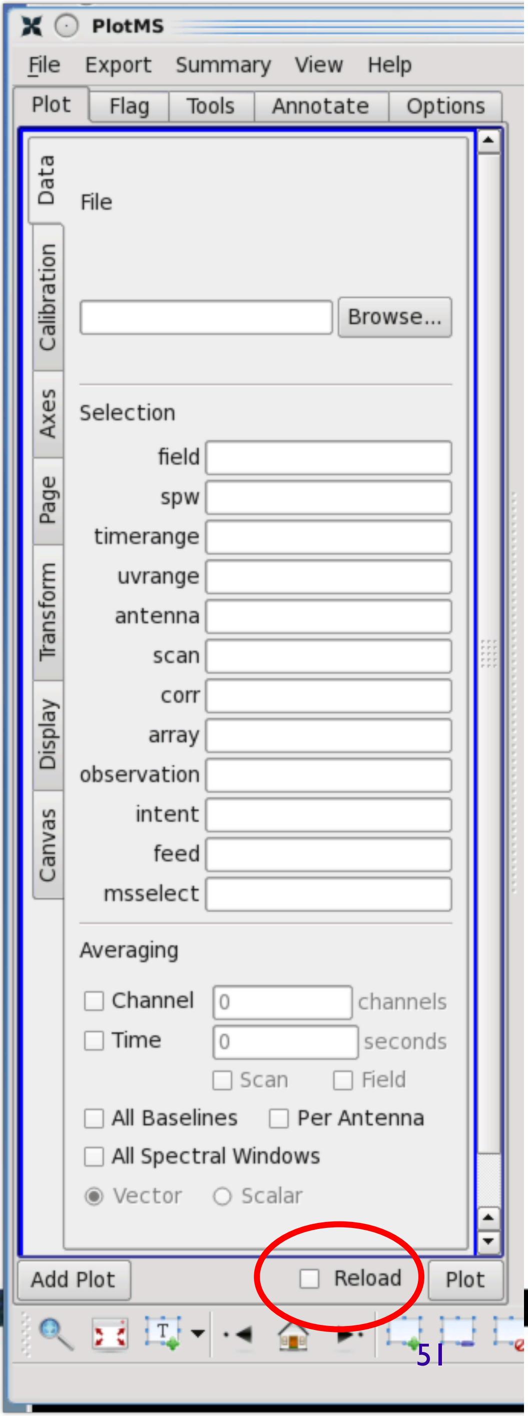
Data Review: *plotms*



Data Review: *plotms*

Control panel: Data

The modification of certain parameters may not be applied if ‘Plot’ is clicked and ‘force reload’ is unchecked.

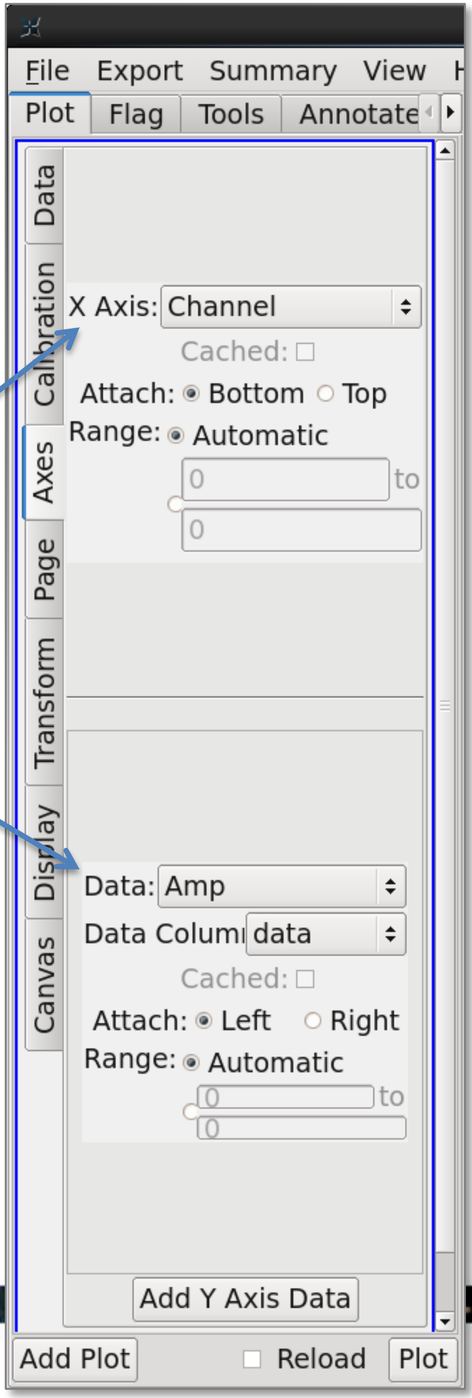


Data Review: *plotms*

Control panel: Axes

Drop down menus to select x and y axes:

time, channel, frequency, velocity, amplitude, phase, uvdist, elevation, etc.

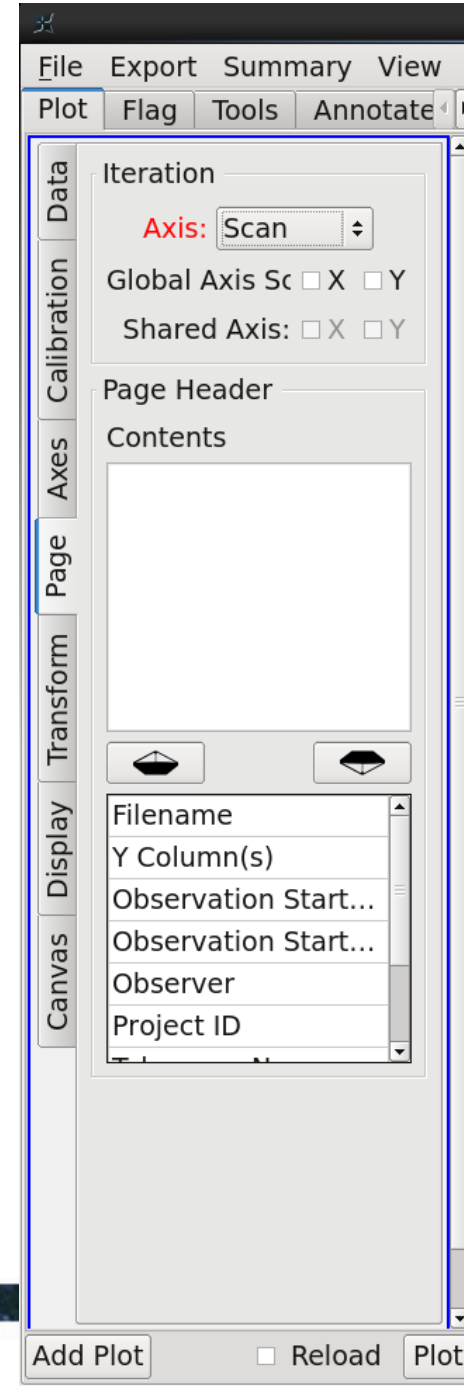


Data Review: *plotms*

Iteration

- Scan
- Field
- Spw
- Baseline
- Antenna

Tool panel

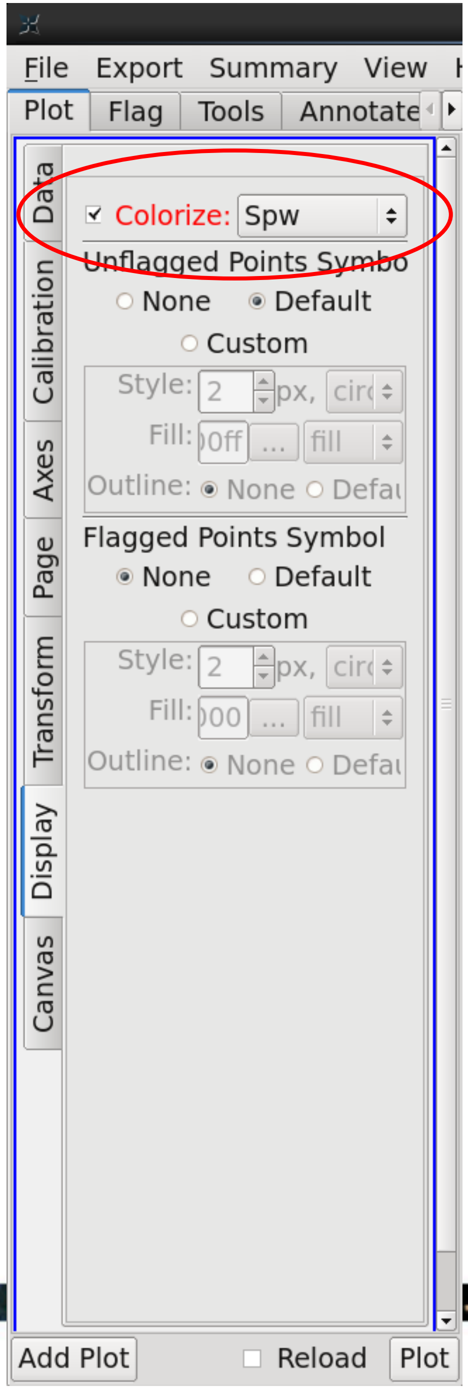


Data Review: *plotms*

Display

Colorize by:

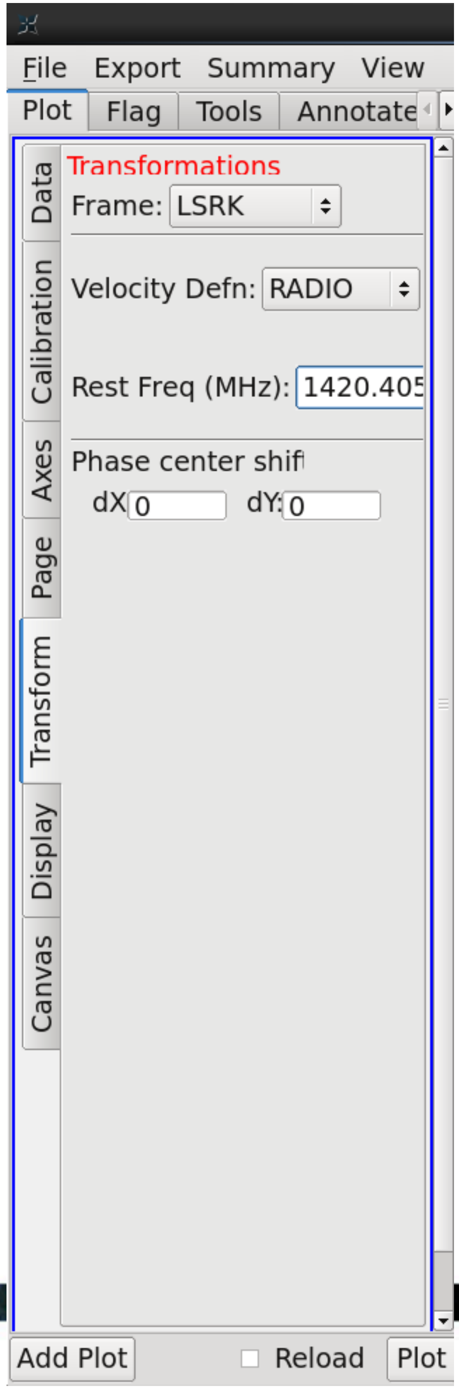
- Scan
- Field
- Spw
- Antenna1
- Antenna2
- Baseline
- Channel
- Correlation



Data Review: *plotms*

Transformations

Frame: TOPO, GEO, BARY, LSRK, LSRD, etc..

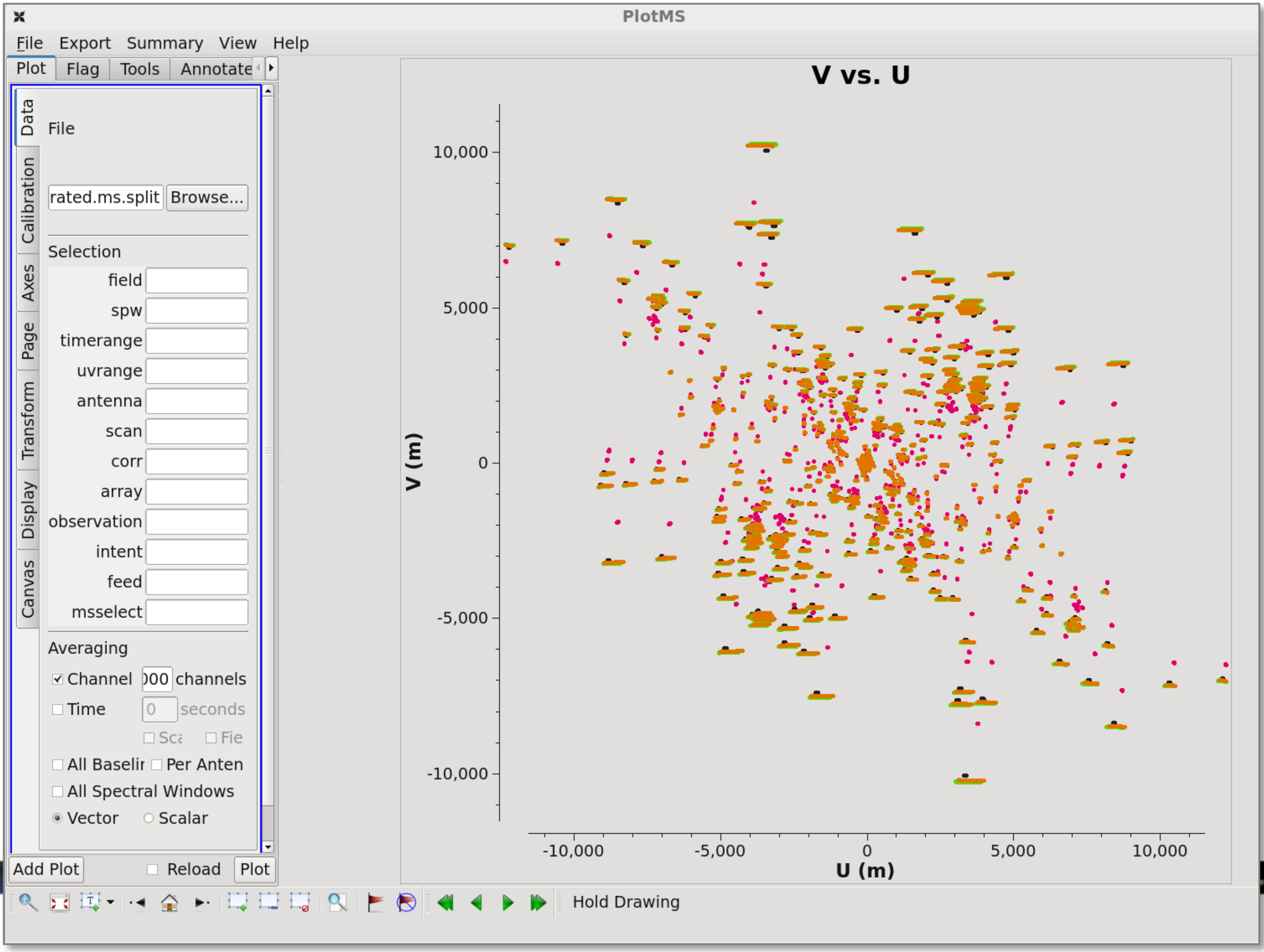


Getting Oriented

```
plotms(vis="SDP81_B4_uncalibrated.ms.split",
        xaxis="u", yaxis="v", averagedata=True,
        avgchannel="1e3", coloraxis="field")
```

'u' and 'v' in meters

Plot 'uwave' Vs. 'vwave' for units of wavelength



Initial Flagging

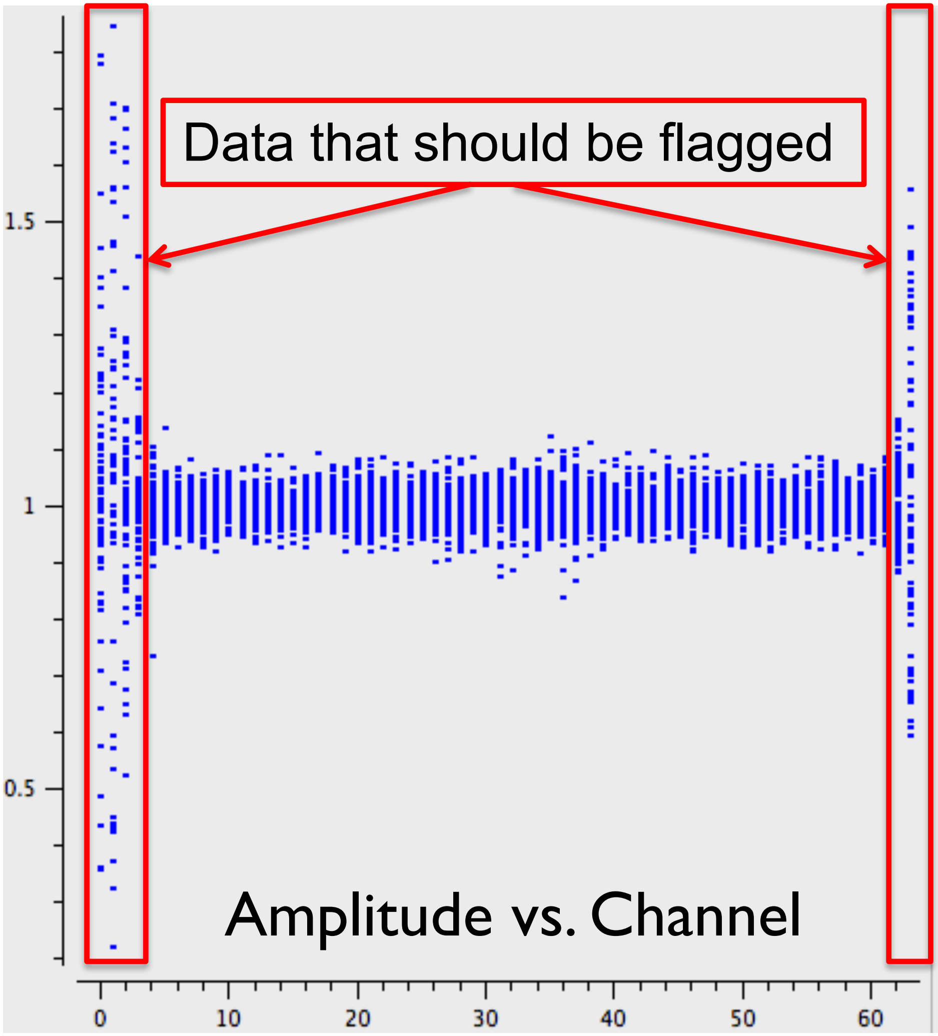
Initial Flagging includes data we know to be problematic even without visual inspection:

- Shadowing
  - Issue at low elevations
  - Issue for compact arrays
  - In CASA: `flagdata(vis='my_data.ms', mode='shadow')`
- Observing Log
  - Many observatories will note weather or hardware problems that affect the data.
- Other obvious errors

Be sure you have run all of the commands in *Getting Oriented and Initial Flagging*

NRAO

An Example of Initial Flagging: Edge Channels



Outline

- Short introduction to CASA and the Python interface
  - How to use tasks
  - What is a measurement set?
- The Flow of Calibration
- Overview of your Directory
  - Data preparation and set up
  - Getting oriented with your data
- **Data Calibration**
- Data Inspection and Flagging
- Basic Imaging

Bandpass, Phase and Amplitude Calibration

ALMA Data Reduction Tutorial

Synthesis Imaging Summer School

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Robert C. Byrd Green Bank Telescope
Very Long Baseline Array

Key Tasks for Calibration

Derive Calibration Tables

- `setjy`: set "model" (correct) visibilities using known model for a calibrator
- `bandpass`: calculate bandpass calibration table (amp/phase vs frequency)
- `gaincal`: calculate temporal gain calibration table (amp/phase vs time)
- `fluxscale`: apply absolute flux scaling to calibration table from known source

Manipulate Your Measurement Set

- `flagdata/flagcmd/flagmanager`: flag (remove) bad data
- `applycal`: apply calibration table(s) from previous steps
- `split`: split off calibrated data from your ms

Inspect Your Data and Results

- `plotms`: inspect your data interactively
- `plotcal`: examine a calibration table

What is Bandpass Calibration?

As we have seen all week, the goal of calibration is to find the relationship between the observed visibilities, V_{obs}, and the true visibilities, V :

$$V_{ij}(t,\nu)_{obs} = V_{ij}(t,\nu) G_{ij}(t) B_{ij}(t,\nu)$$

where t is time, ν is frequency, i and j refer to a pair of antennas (i,j) (i.e., one baseline), G is the complex "continuum" gain, and B is the complex frequency-dependent gain (the "bandpass").

Bandpass calibration is the process of measuring and correcting the *frequency-dependent* part of the gains, $B_{ij}(t,\nu)$.

B_{ij} may be constant over the length of an observation, or it may have a slow time dependence.

NRAO

Why is BP Calibration important?

Good bandpass calibration is a key to detection and accurate measurement of spectral features, especially weak, broad features.

Bandpass calibration can also be the limiting factor in dynamic range of continuum observations.

- Bandpass amplitude errors may mimic changes in line structure with ν
- ν-dependent phase errors may lead to spurious positional offsets of spectral features as a function of frequency, mimicking doppler motions
- ν-dependent amplitude errors limit ability to detect/measure weak line emission superposed on a continuum source. Consider trying to measure a weak line on a strong continuum with ~ 10% gain variation across the band.

Bandpass Calibration

- Determine the variations of phase and amplitude with frequency
- Account for slow time-dependency of the bandpass response
- We will arrive at antenna-based solutions against a reference antenna
  - In principle, could use autocorrelation data to measure antenna-based amplitude variations, but not phase
  - Most bandpass corruption is antenna-based, yet we are measuring N(N-1)/2 baseline-based solutions
  - Amounts to channel-by-channel self-cal

Bandpass Calibration: What makes good calibrators?

- Best targets are bright, flat-spectrum sources with featureless spectra
  - Although point-source not absolutely required, beware frequency dependence of resolved sources
  - If necessary, can specify a spectral index using *setjy*

- Don't necessarily need to be near science target on the sky

CASA Tasks for Bandpass Calibration

- We will use *gaincal* to measure time variation of phase

- Then use *bandpass* task
  - We will calibrate channel-to-channel variation (preferred method)
  - Alternatively, could fit a smooth function
  - Pay close attention to solutions; e.g. bright calibrators are rare, esp. at Band 9

- Use *applycal* to apply the bandpass solution to other sources

Create a phase solution for the bandpass calibrator

Run a listobs and note which source is the bandpass calibrator. This is J0825+0309 (identified as field 0).

```
listobs(”SDP81_B4_uncalibrated.ms.split")
```

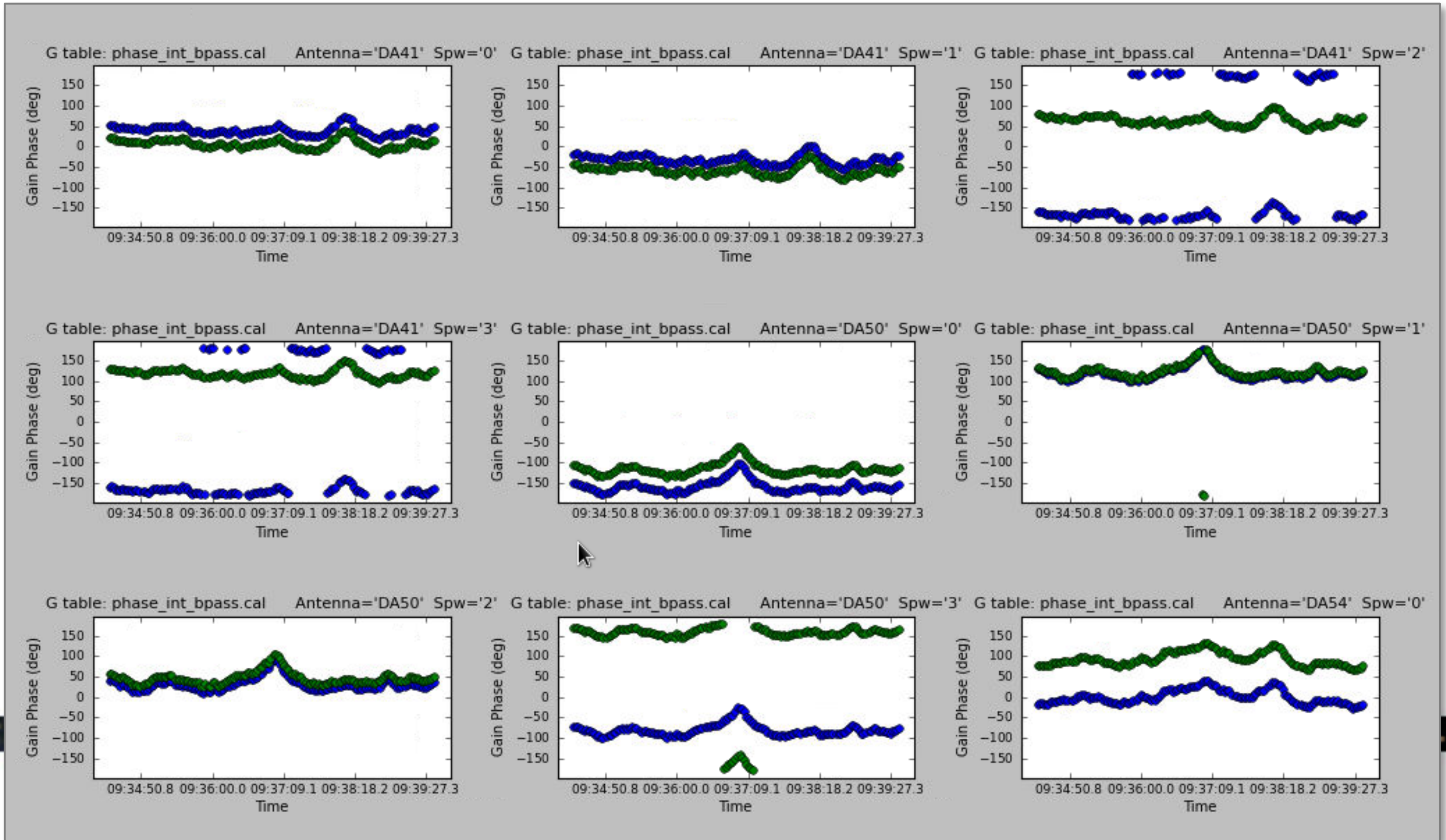
Gaincal is the general purpose task to solve for time-dependent amplitude and phase variations for each antenna. Here we carry out a short-timescale phase solution ("int") on the bandpass calibrator. This is saved as a calibration table "phase_int_bpass.cal".

```
os.system("rm -rf phase_int_bpass.cal")
gaincal(vis=“SDP81_B4_uncalibrated.ms.split",
      caltable="phase_int_bpass.cal",
      field="0",
      spw=“0:22~42,1:22~42,2:22~42,3:800~1200”,
      scan=“3”,solint="int",refant=”DA56”, calmode="p”)
```

Plot phase solutions (phase vs. time)

Plot the calibration table, showing phase vs. time with a separate plot for each antenna. The two colors are the two correlations (i.e., polarizations).

```
plotcal(caltable="phase_int_bpass.cal",
  xaxis="time",yaxis="phase", subplot=331,
  iteration="antenna,spw", plotrange=[0,0,-180,180])
```



NRAO

Create the bandpass solution

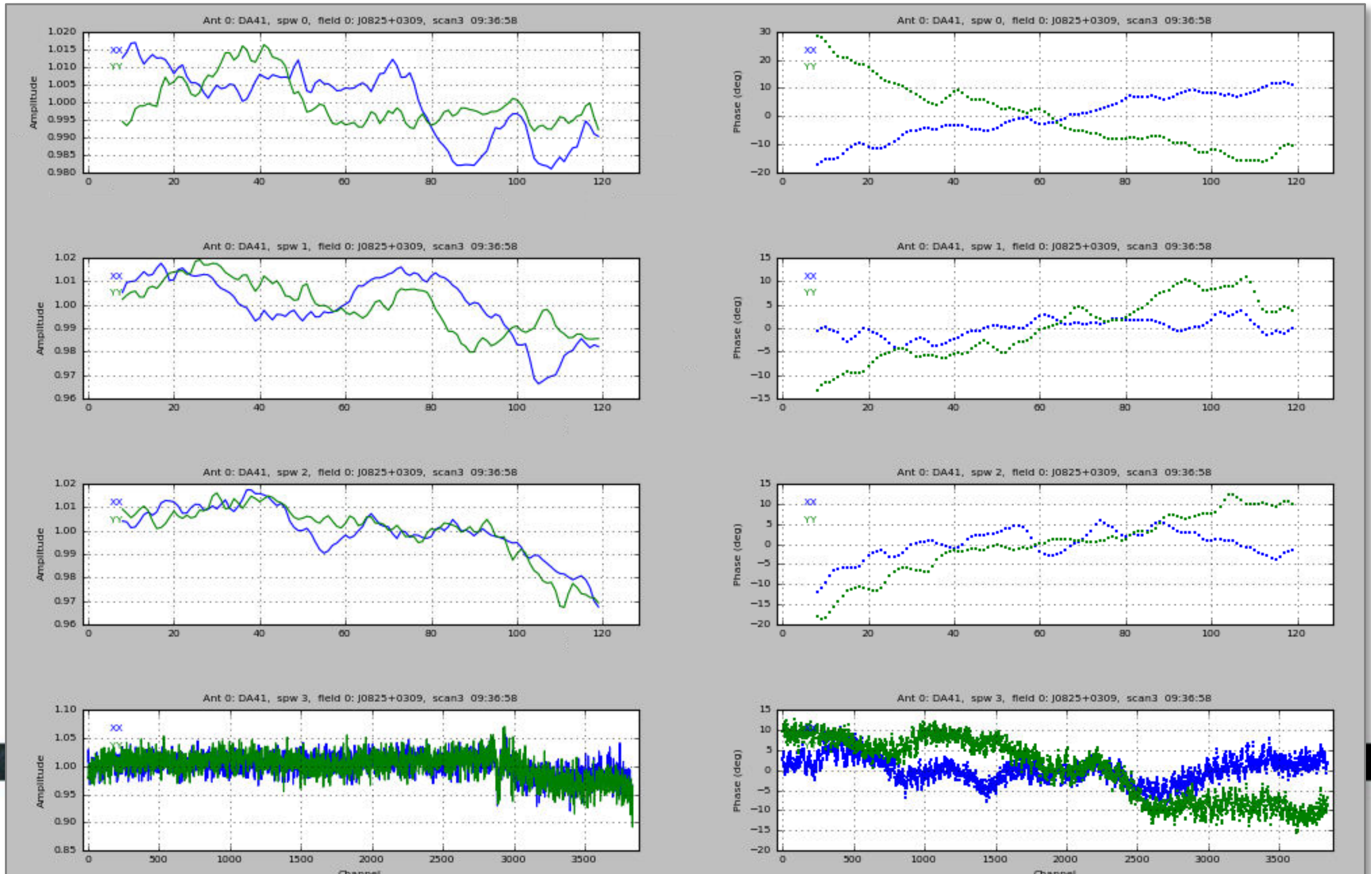
Now carry out a bandpass solution. This will solve for the amplitude and phase corrections needed for each channel for antenna. We use gaintable to feed the short-timescale phase solution to the task. This means that this table will be applied before the bandpass solution is carried out. We will deal with the overall normalization of the data later, for now we tell the task to solve for normalized (average=1) solutions via solnorm=True.

```
os.system("rm -rf bandpass.cal")
bandpass(vis="SDP81_B4_uncalibrated.ms",
         caltable="bandpass.cal",
         field="0",
         solint="inf",
         scan="3",combine="scan",refant="DA56",
         solnorm=True,bandtype="B",
         gaintable="phase_int_bpass.cal")
```

Plot the result with plotbandpass

We inspect the phase and amplitude behavior of the calibration plotting the corrections for each antenna using plotbandpass. We tell it to plot both phase and amplitude for four spectral windows at a time. Cycle through the plots.

```
plotbandpass(caltable="bandpass.cal",
             xaxis="chan", yaxis="both", subplot=42)
```



Create a smoother bandpass for spw 3

Notice how noisy the solutions are on one of the spectral windows (spw 3). We can also calibrate the bandpass by averaging several channels at once, which is good if you think that signal-to-noise may be an issue and the solutions can be described as smoothly varying functions. We do this for the noisy spectral window by setting a solution interval of 5 channels.

For spws 0,1,2:

```
os.system("rm -rf bandpass_smooth.cal")
bandpass(vis="SDP81_B4_uncalibrated.ms",
         caltable="bandpass.cal",
         field="0",
         spw="0,1,2",
         scan="3",
         solint="inf",
         combine="scan",
         refant="DA56",
         solnorm=True,
         bandtype="B",
         gaintable="phase_int_bpass.cal")
```

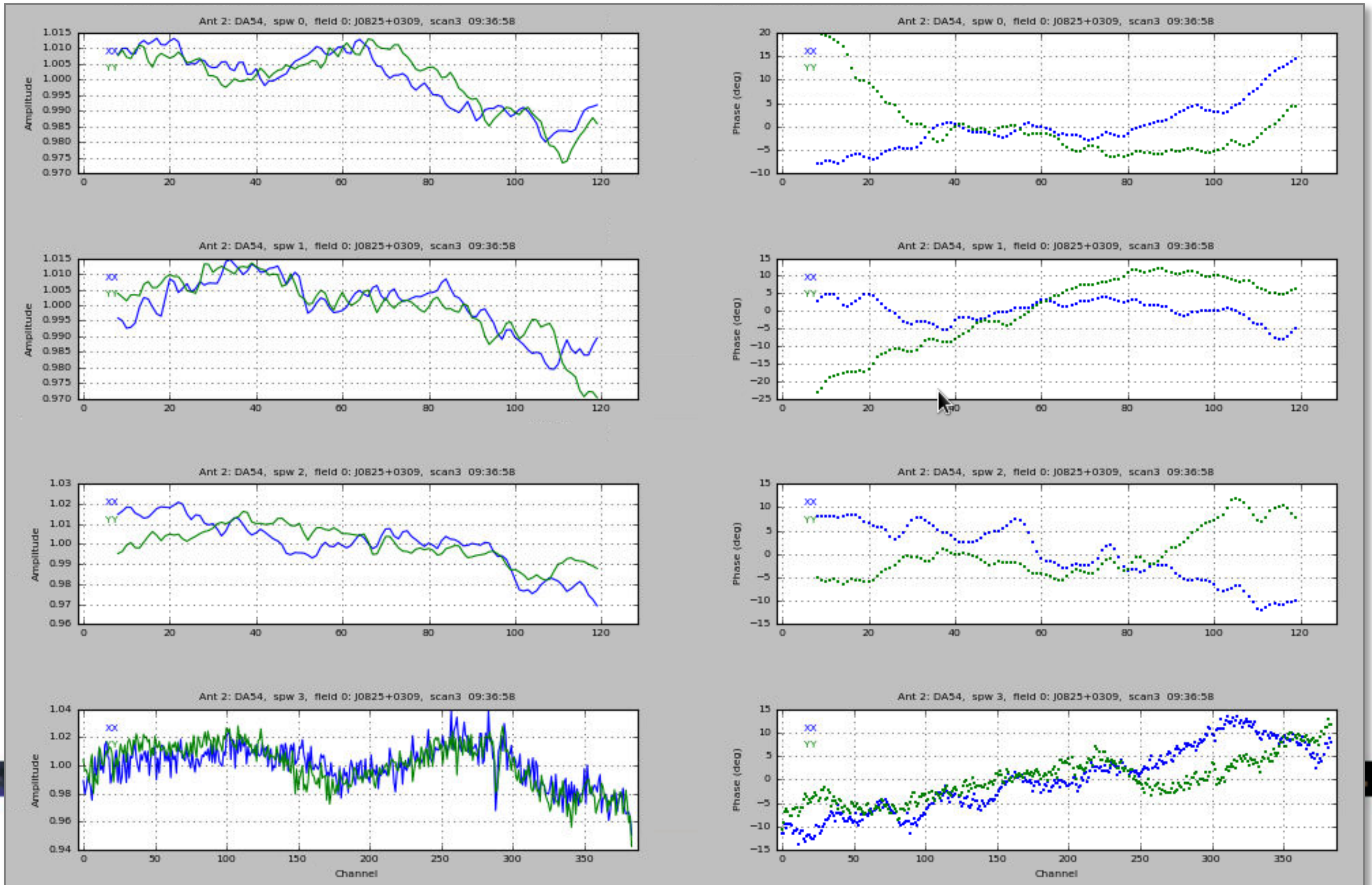
For spw 3:

```
os.system("rm -rf bandpass_smooth.cal")
bandpass(vis="SDP81_B4_uncalibrated.ms",
         caltable="bandpass.cal",
         field="0",
         spw="3",
         scan="3",
         solint="inf,5ch",
         combine="scan",
         refant="DA56",
         solnorm=True,
         bandtype="B",
         append=True,
         gaintable="phase_int_bpass.cal")
```

Plot the new (smoother) bandpass solutions

Now plot the new (smoother) bandpass solutions. There are less points and they are less noisy in absolute scale. We will use these in our calibration.

```
plotbandpass(caltable="bandpass_smooth.cal",
             xaxis="chan", yaxis="both", subplot=42)
```



NRAO

Apply the bandpass solutions

Apply the solutions - both in time and frequency - to the data using applycal. This creates a new corrected data column.

```
applycal(vis="SDP81_B4_uncalibrated.ms.split",
         field="",
         gaintable=["bandpass_smooth.cal","phase_int_bpass.cal"],
         interp=["linear","linear"],
         gainfield=["0","0"],
         applymode='calonly')
```

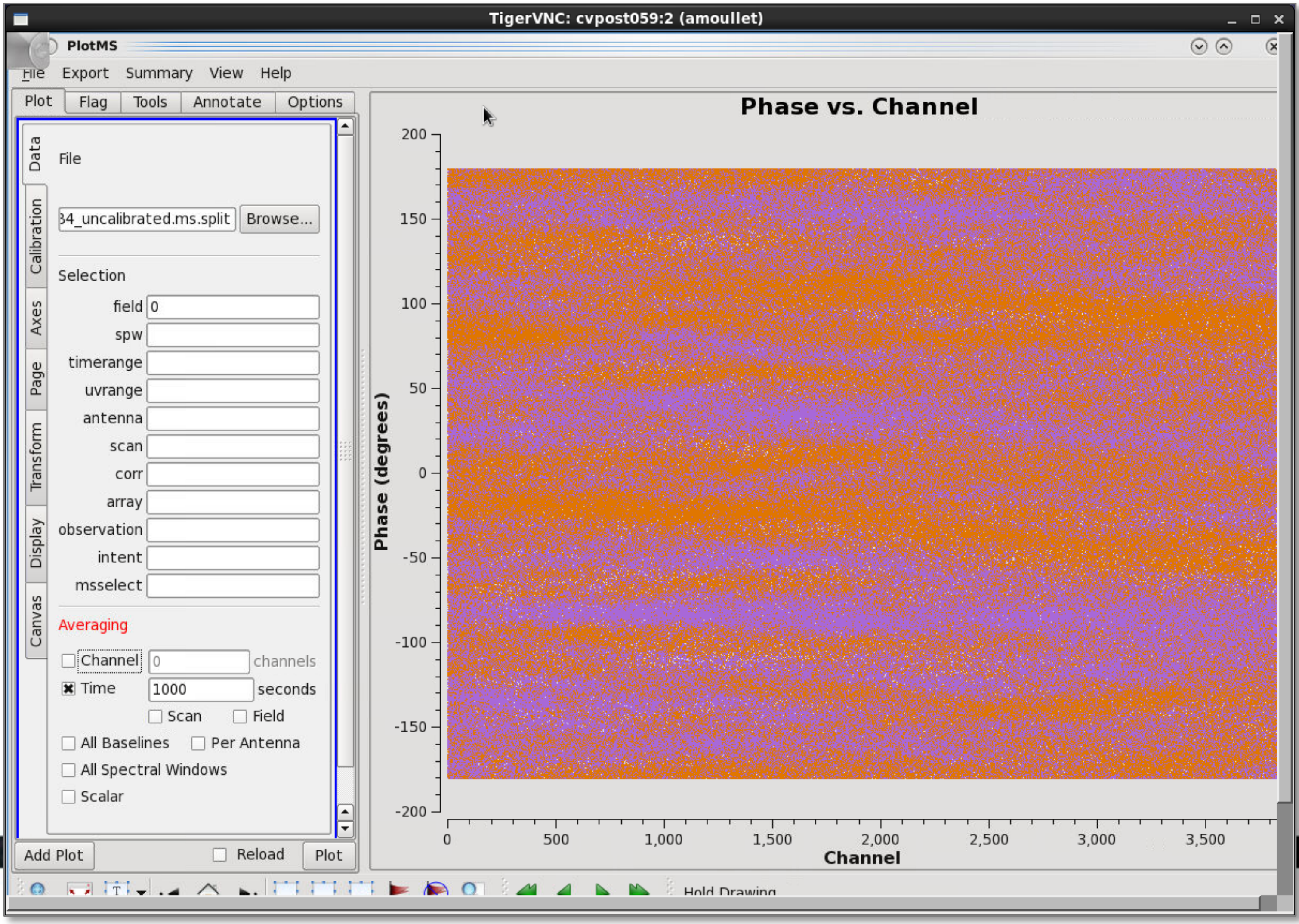
Plot the results of the calibration by comparing the dependence of phase and amplitude on channel before and after calibration.

At this point, we are going to look at how the solutions have fixed the phase and amplitude variations vs. frequency. You can try the non-channel averaged data to see if there are any differences.

NRAO

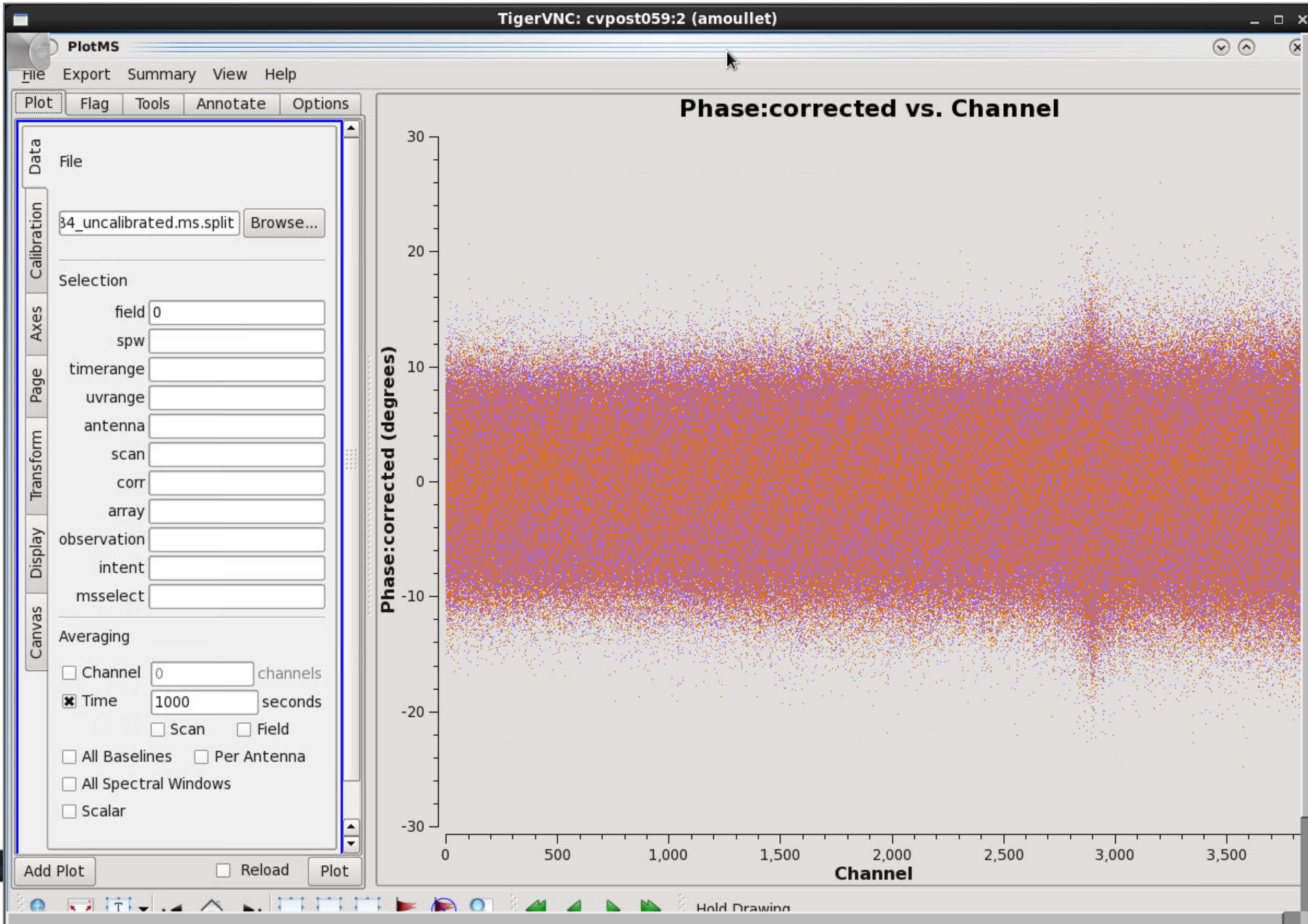
Phase vs Channel before

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="chan",
   yaxis="phase",  ydatacolumn="data", field="0",
   averagedata=True, avgtime="1e3", coloraxis="corr")
```



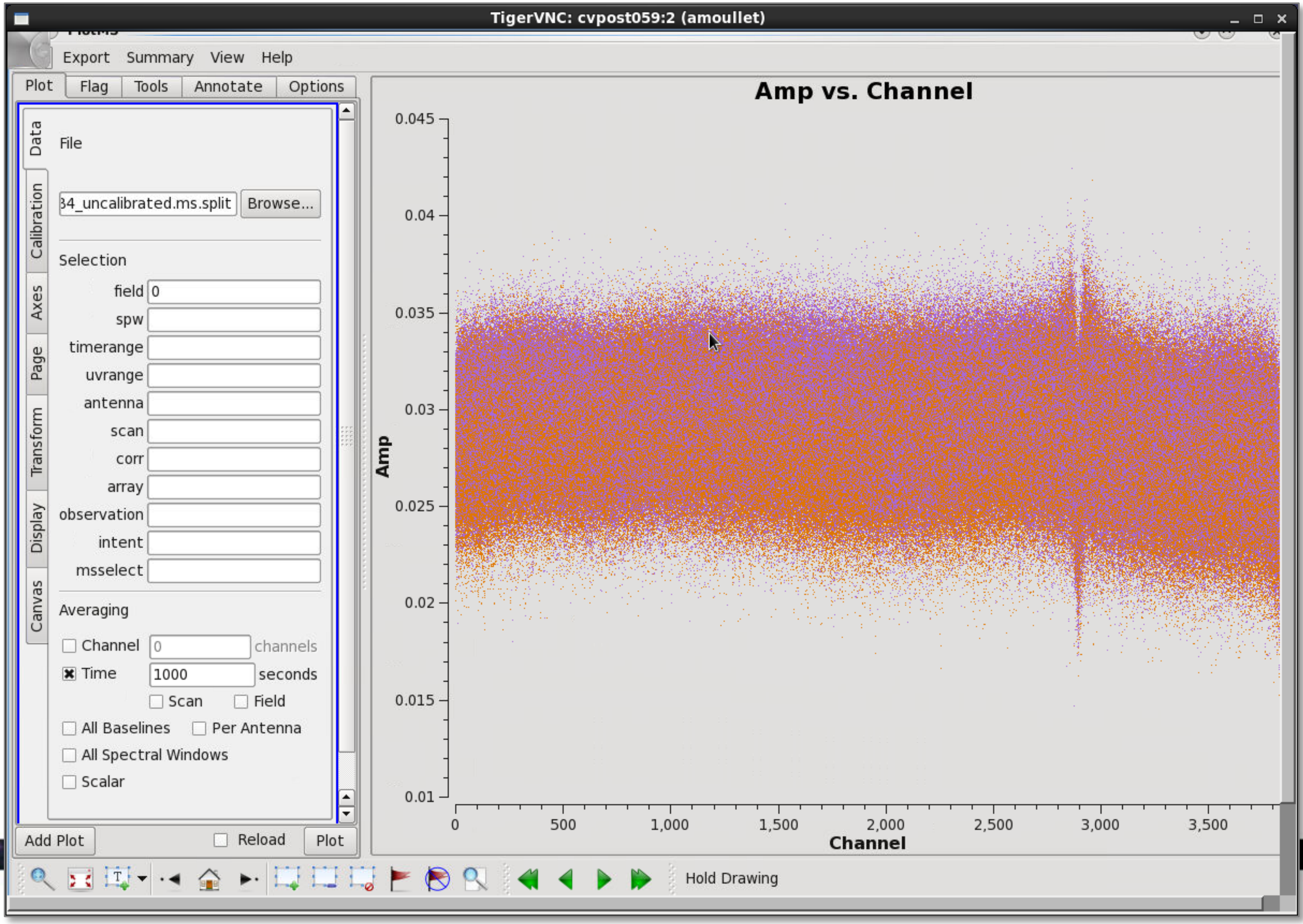
Phase vs Channel after

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="chan",
   yaxis="phase",  ydatacolumn="corrected", field="0",
   averagedata=True, avgtime="1e3", coloraxis="corr")
```



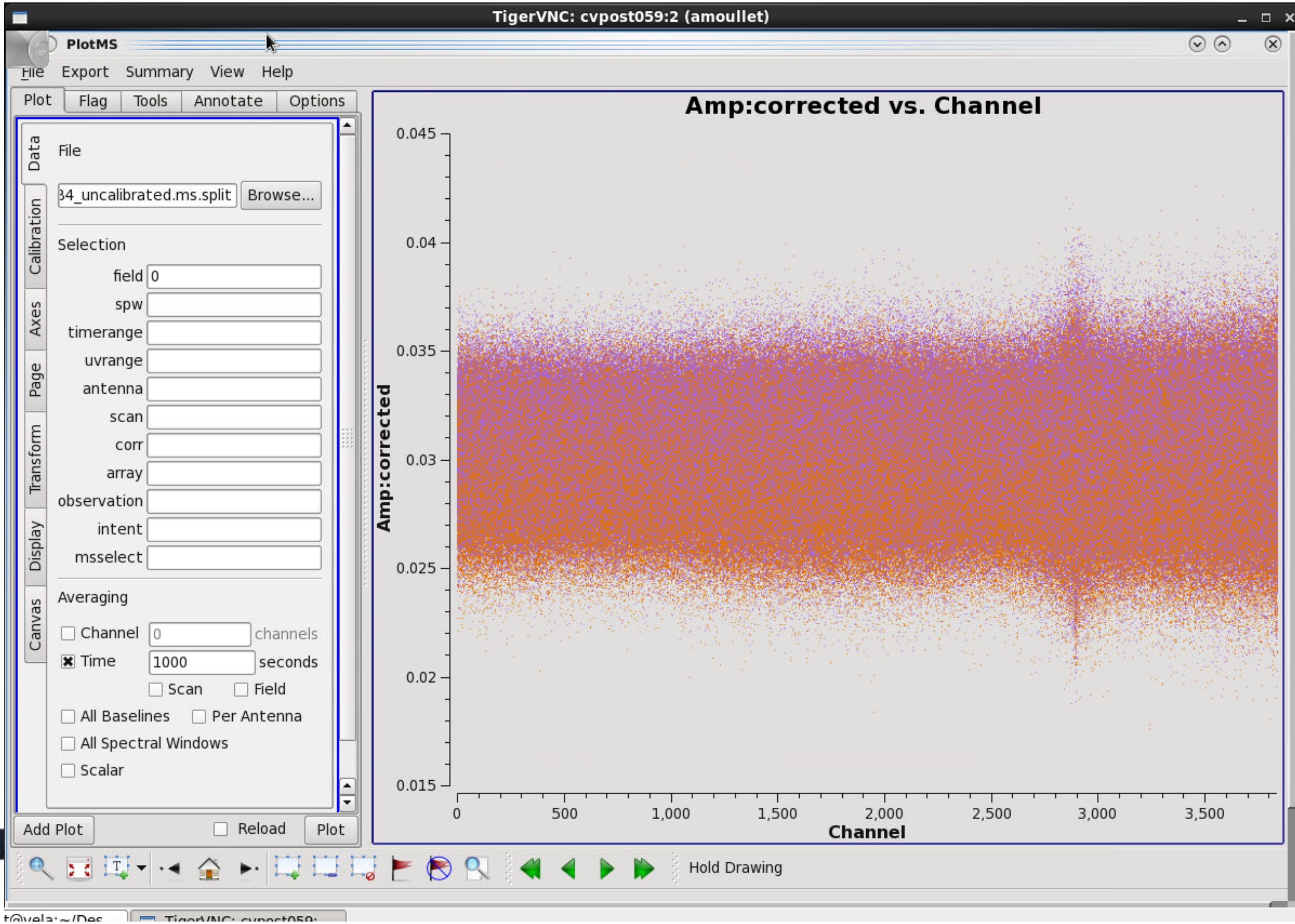
Amp vs. Chan before

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="chan",
yaxis="amp", ydatacolumn="data",field="0",
averagedata=True, avgtime="1e3",coloraxis="corr")
```



Amp vs. Chan after

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="chan",
yaxis="amp", ydatacolumn="corrected",field="0",
averagedata=True, avgtime="1e3",coloraxis="corr")
```



Our first attempt at bandpass calibration is now complete.

Be sure you have run all of the commands in *Bandpass Calibration*

NRAO

Steps to a Calibrated Data set

DONE

Correct for System Temperature, WVR (Water Vapor), Antenna Positions
`gencal, wvrgcal`

DONE

Tsys, WVR, Antenna Correction Tables

DONE

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna
`bandpass`

DONE

Bandpass Calibration Table

Calibrate the Amplitude and Phase vs. Time of Each Antenna
`gaincal`

Phase Calibration Table
Amplitude Calibration Table

Set the Absolute Amplitude Scale With Reference to a Known Source
`fluxscale`

Flux Calibration Table

Apply all corrections to produce calibrated data
`applycal`

Measurement Set

Corrected column now holds calibrated data.

gaincal

Measurement Set

Data column holds observations.

↓

gaincal

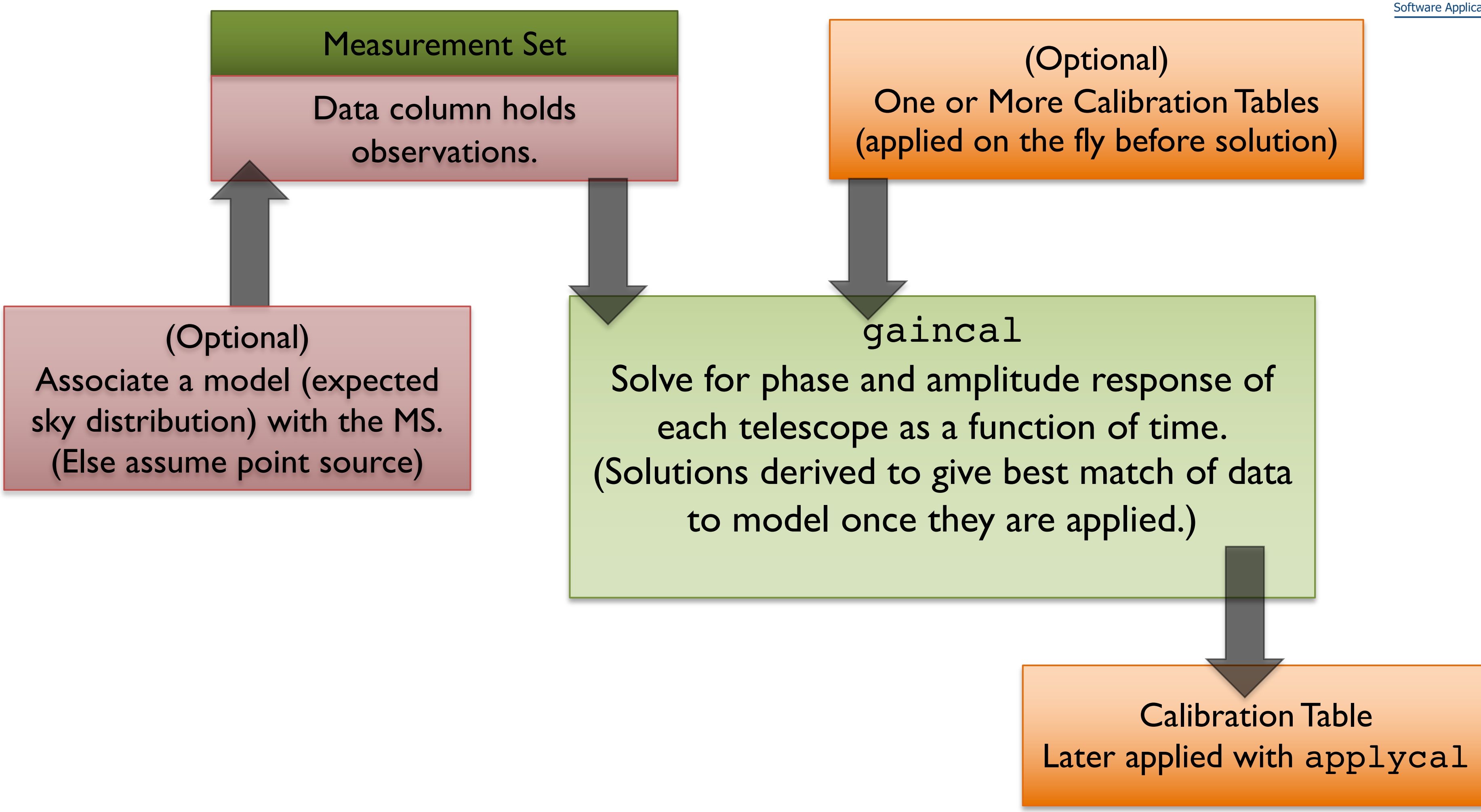
Solve for phase and amplitude response of each telescope as a function of time. (Solutions derived to give best match of data to model once they are applied.)

↓

Calibration Table

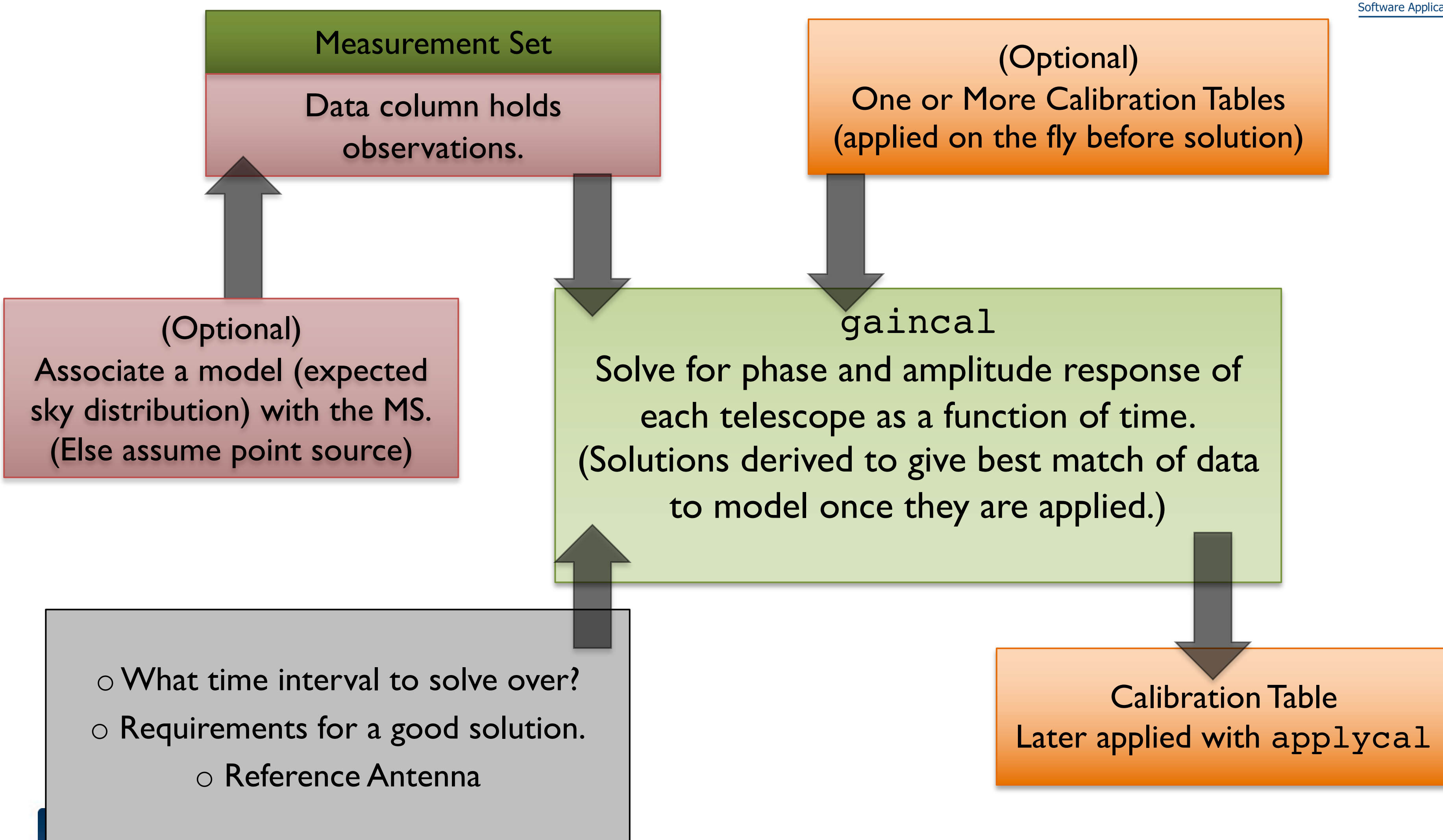
Later applied with `applycal`

gaincal



Measurement Set
Data column holds observations.
(Optional)
One or More Calibration Tables
(applied on the fly before solution)
(Optional)
Associate a model (expected sky distribution) with the MS.
(Else assume point source)
gaincal
Solve for phase and amplitude response of each telescope as a function of time.
(Solutions derived to give best match of data to model once they are applied.)
Calibration Table
Later applied with applycal

gaincal


Measurement Set
Data column holds observations.
(Optional)
One or More Calibration Tables
(applied on the fly before solution)
(Optional)
Associate a model (expected sky distribution) with the MS.
(Else assume point source)
gaincal
Solve for phase and amplitude response of each telescope as a function of time.
(Solutions derived to give best match of data to model once they are applied.)
○ What time interval to solve over?
○ Requirements for a good solution.
○ Reference Antenna
Calibration Table
Later applied with applycal

Set Model for the Quasar

First things first - we need to make sure that we have valid models in place for our data. Our flux reference source is a quasar J0854+2006 (field 1). We will first query the calibrator catalog and then use those outputs in the task "setjy” to apply the model to our data. In other words, we use a routine to parse the ALMA calibrator database, interpolate the expected flux for the calibrator reference, and put in the 'model' column of the data using setjy.

```
aU.getALMAFluxForMS(“SDP81_B4_uncalibrated.ms.split”)
setjy(vis=”SDP81_B4_uncalibrated.ms.split”,
      standard=“manual”,
      field=1,
      fluxdensity = [3.986837, 0, 0, 0],
      spix = -0.456158813,
      reffreq = “149.593012274GHz’)
```

NRAO

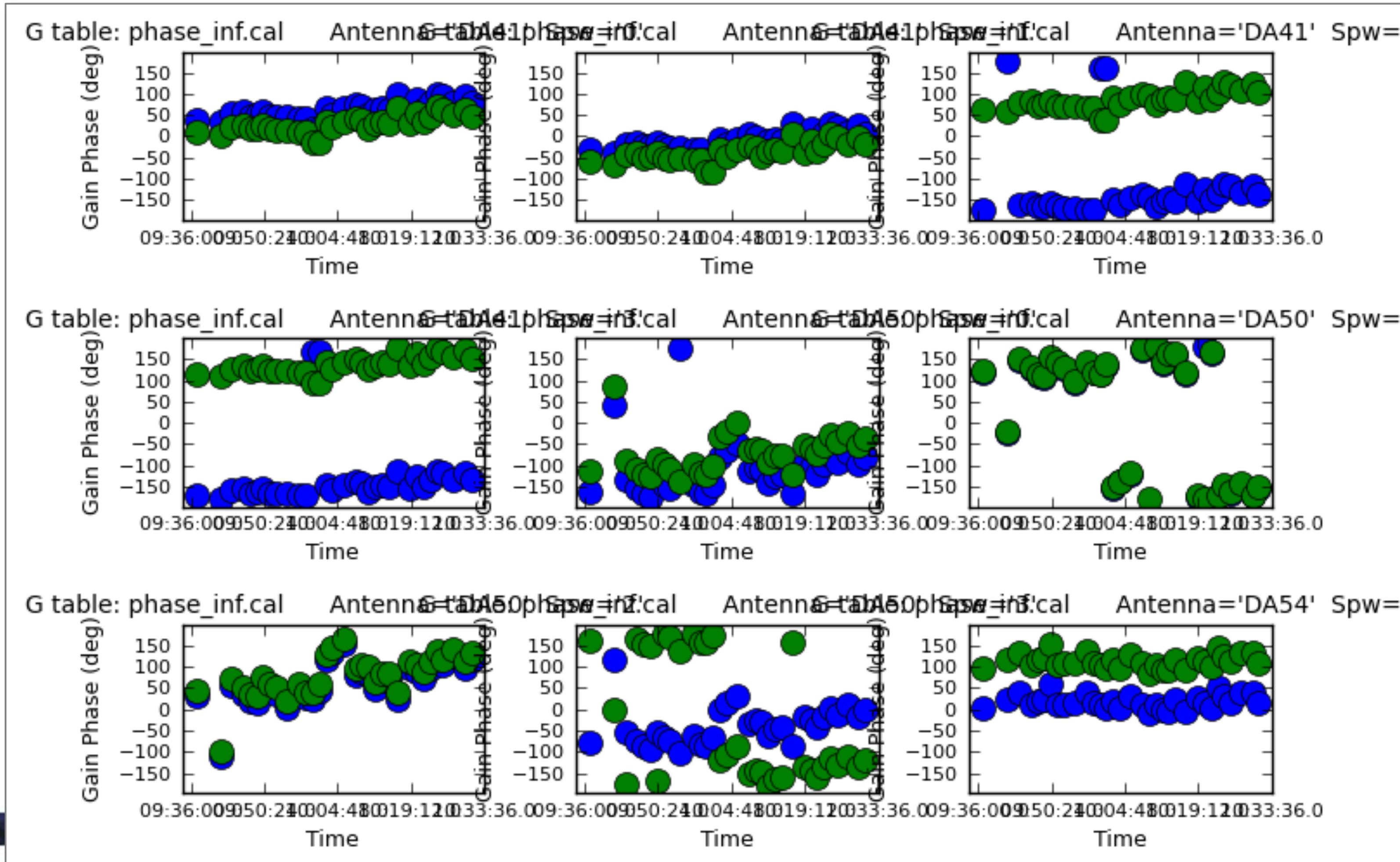
Gain Calibration: Long-term phase solutions

First, we calibrate the phase for each antenna for each scan. This is the right cadence to transfer to the science target, which is visited only on a ~ every-other-scan timescale.

```
os.system("rm -rf phase_inf.cal")
gaincal(vis=”SDP81_B4_uncalibrated.ms.split",
        caltable="phase_inf.cal",
        field=”0~2",
        solint="inf",
        refant=”DA56”,
        gaintype=“G”,
        gaintable=“bandpass_smooth.cal”)
```

Plot the resulting phase calibration

```
plotcal(caltable="phase_inf.cal",xaxis="time",yaxis="phase,
   subplot=331,iteration="antenna,spw”, plotrange=[0,0,-
   180,180],markersize=10, figfile=”ss20_phase_scan.png")
```



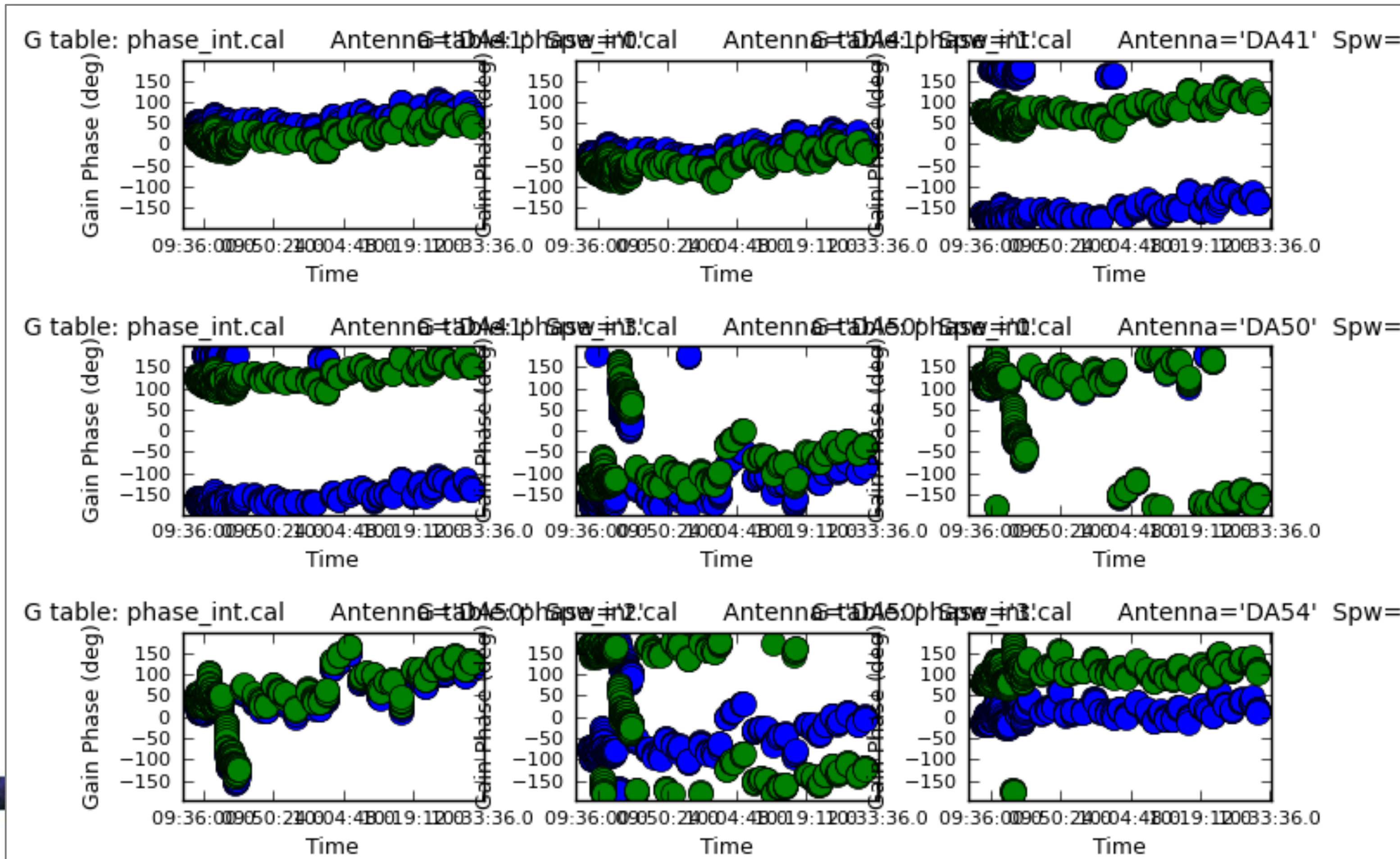
Gain Calibration: Short-term Phase Solutions

Now we want to remove any short timescale phase variation from the sources involved in the bandpass and flux calibration. We do so using gaincal.

```
os.system("rm -rf phase_int.cal")
gaincal(vis="SDP81_B4_uncalibrated.ms.split",
        caltable="phase_int.cal",
        field="0~2",
        solint="int",
        refant="DA56",
        gaintype="G",
        calmode="p",
        gaintable="bandpass_smooth.cal")
```

Plot the resulting short timescale phase calibration

```
plotcal(caltable="phase_int.cal",xaxis="time",yaxis="phase",
   subplot=331, iteration="antenna,spw",plotrange=[0,0,-
   180,180],markersize=10,figfile="ss20_phase_int.png")
```



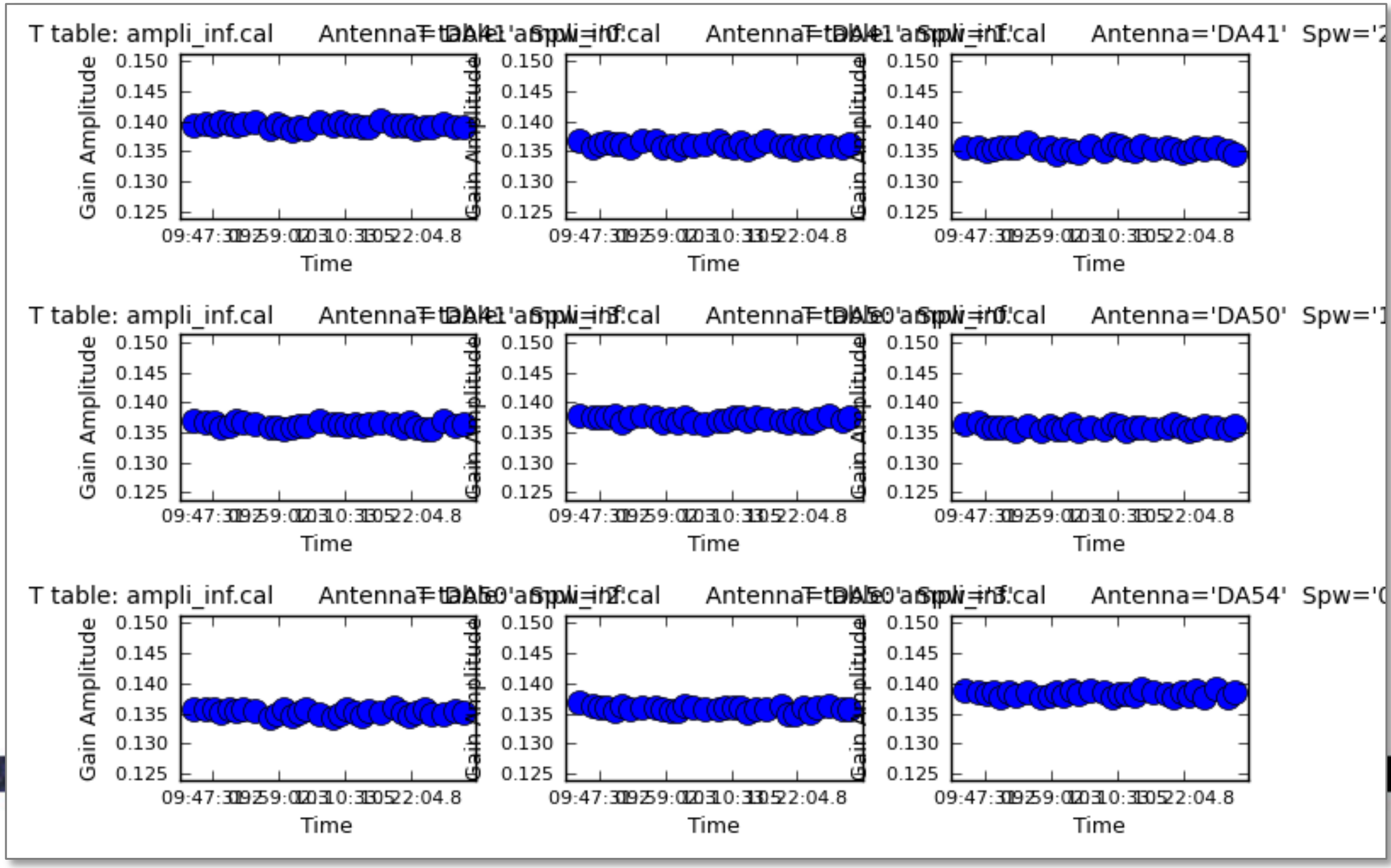
Gain Calibration: Long-Term Amplitude Solutions

Now let's derive an amplitude solution, first applying the short-timescale phase solution.

```
os.system("rm -rf ampli_inf.cal")
gaincal(vis=”SDP81_B4_uncalibrated.ms.split",
        caltable="ampli_inf.cal",
        field=”0~2",
        solint="inf",
        refant=”DA56”,
        gaintype=“T”,
        calmode="a",
        gaintable=[”bandpass_smooth.cal”,”phase_int.cal”])
```

Plot the solution as amplitude vs. time for each antenna and spectral window

- `plotcal(caltable="ampli_inf.cal", xaxis="time", yaxis="amp", subplot=331, iteration="antenna,spw”, plotrange=[0,0,0.125,0.15], markersize=10, field=“2”, figfile=“ss20_ampli_scan.png)`



Our first attempt at gain calibration is now complete.

Be sure you have run all of the commands in *Gain Calibration*

Set flux scale of calibrators

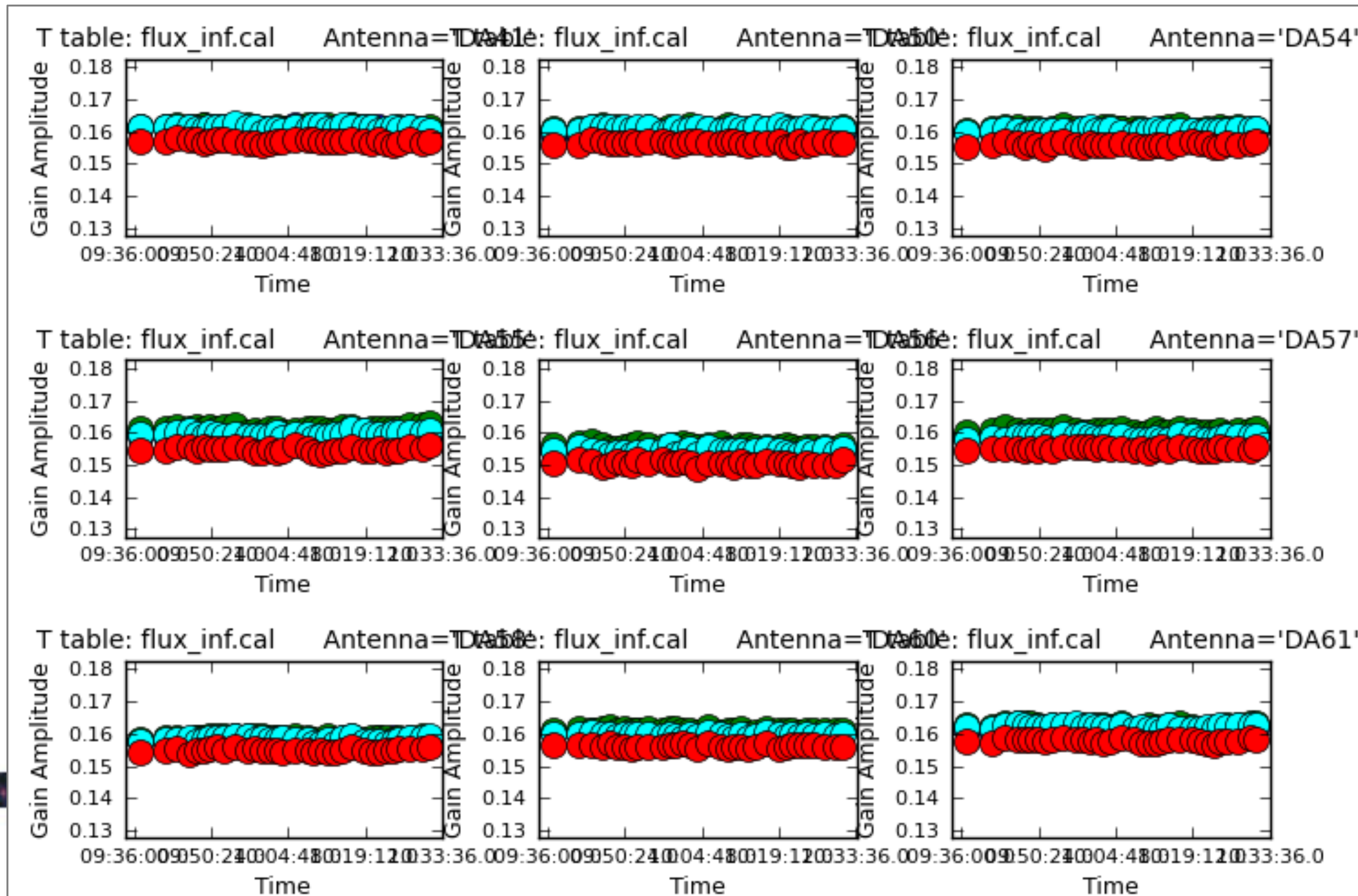
The gaincal solved for the amplitude scaling to make the data match the current model. For the quasar J0854+2006, we have taken care to set the correct model using setjy. For the other two calibrators, however, we don't a priori know the flux. Those have been calibrated using the default model, which is a point source of amplitude 1 Jy at the middle of the field. We now use fluxscale to bootstrap from the (correct) flux of the quasar through the amplitude calibration table to estimates of the true flux of the other two calibrators. This will output both a new table and the flux estimates themselves.

```
os.system("rm -rf flux_inf.cal")
fluxscaleDict = fluxscale(vis="SDP81_B4_uncalibrated.ms.split",
          caltable="ampli_inf.cal",
          fluxtable="flux_inf.cal",
          reference="1")
```

Plot the rescaled flux solutions

Plot the rescaled flux table, which now should contain the correct flux calibrations.

```
plotcal(caltable="flux_inf.cal", xaxis="time",yaxis="amp",
    subplot=331,iteration="antenna”,plotrange=[0,0,0.13,0.18],
    markersize=10,figfile=“ss20_flux_scan.png”)
```



We have now bootstrapped the known flux of the flux reference quasar to the fluxes of our other calibrators.

Be sure you have run all of the commands in
Setting the Flux Scale

Steps to a Calibrated Data set

CASA
Common Astronomy Software Applications

Correct for System Temperature, WVR (Water Vapor), Antenna Positions
`gencal, wvrgcal`
→ Tsys, WVR, Antenna Correction Tables

DONE DONE

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna
`bandpass`
→ Bandpass Calibration Table

DONE DONE

Calibrate the Amplitude and Phase vs. Time of Each Antenna
`gaincal`
→ Phase Calibration Table
Amplitude Calibration Table

DONE DONE

Set the Absolute Amplitude Scale With Reference to a Known Source
`fluxscale`
→ Flux Calibration Table

DONE DONE

Apply all corrections to produce calibrated data
`applycal`
→ Measurement Set
Corrected column now holds calibrated data.

NRAO

Apply Bandpass, Phase, & Flux Calibration Tables

For our bandpass and flux calibrators (fields 0 & 1), we apply our bandpass calibration and our gain calibration
(short term phase + flux).

For our science target and phase calibrator (fields 2 & 3), we apply our bandpass calibration and our gain calibration
(long term phase + flux).

For field 0:

```
applycal(vis=”SDP81_B4_uncalibrated.ms.split",
   field=”0",
   gaintable=[”bandpass_smooth.cal”,”phase_int.cal","flux_inf.cal"],
   gainfield=[“”,”0”,”0”],
   interp="linear,linear",
   calwt=True,
   flagbackup=False)
```

Apply Bandpass, Phase, & Flux Calibration Tables

For field 1:

```
applycal(vis=″SDP81_B4_uncalibrated.ms.split",
   field=″1",
   gaintable=[″bandpass_smooth.cal″,″phase_int.cal", "flux_inf.cal"],
   gainfield=[‶″,″1″,″1″], interp="linear,linear",
   calwt=True, flagbackup=False)
```

For fields 2 & 3:

```
applycal(vis=″SDP81_B4_uncalibrated.ms.split",
   field=″2,3",
   gaintable=[″bandpass_smooth.cal″,″phase_inf.cal", "flux_inf.cal"],
   gainfield=[‶″,″2″,″2″],
   interp="linear,linear",
   calwt=True, flagbackup=False)
```

Be sure you have run all of the commands in *Applying Calibrations*

Outline

- Short introduction to CASA and the Python interface
  - How to use tasks
  - What is a measurement set?
- The Flow of Calibration
- Overview of your Directory
  - Data preparation and set up
  - Getting oriented with your data
- Data Calibration
- **Data Inspection and Flagging**
- Basic Imaging

Data Inspection, Flagging and End to End processing

ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Robert C. Byrd Green Bank Telescope
Very Long Baseline Array

Key Tasks for Data Inspection/Editing

CASA
Common Astronomy Software Applications

Initial Inspection Tools

- `listobs`: list contents of a MS
- `plotant`: plot antenna positions

Inspect Your Data and Results

- `plotms`: inspect/flag your data interactively
- `plotcal`: examine a calibration table
- `listcal`: list calibration table data

Flagging

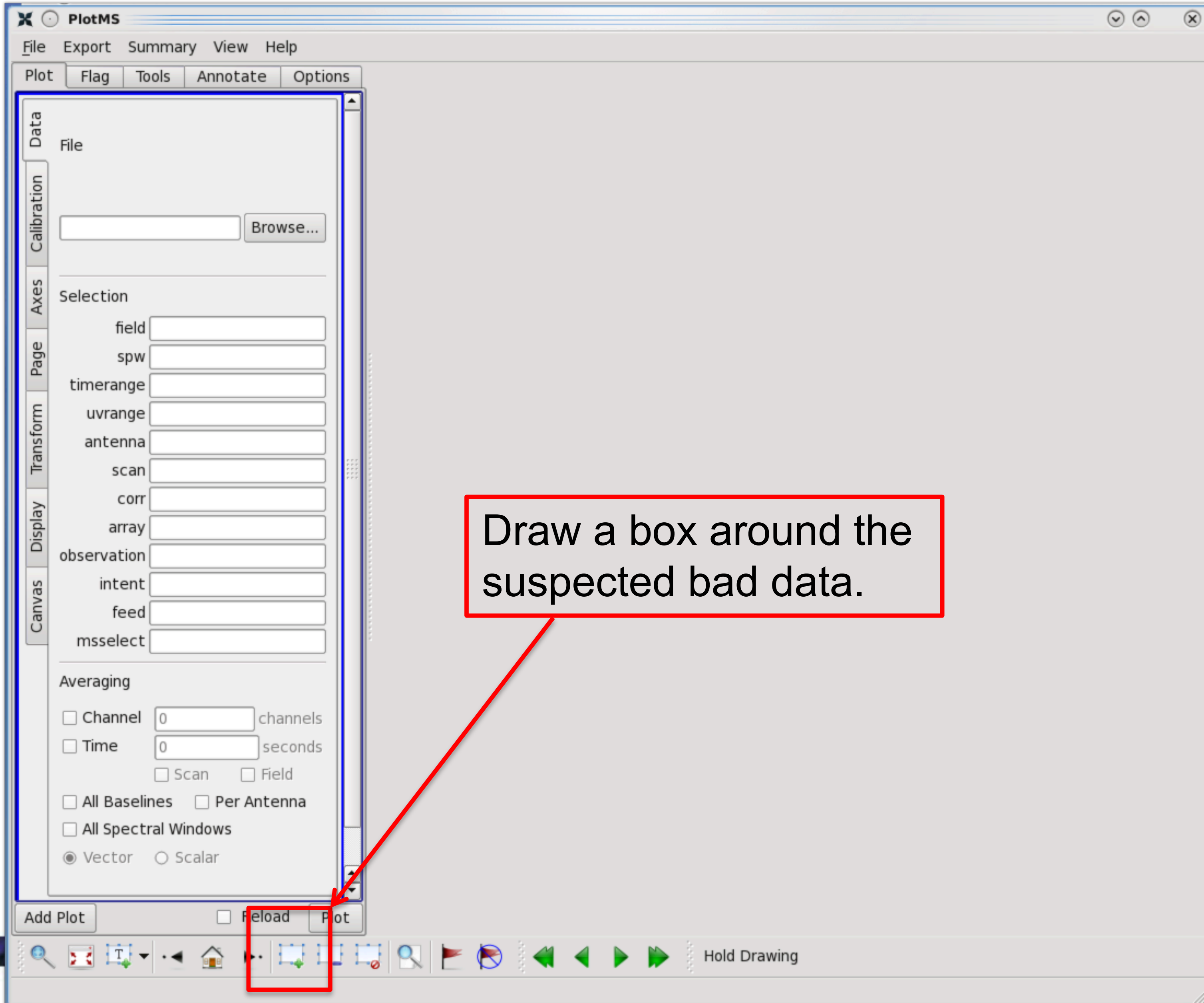
- `flagdata`: flag (remove) bad data
- `flagcmd`: batch flagging using lists/tables
- `flagmanager`: storage/retrieval of flagging state

NRAO

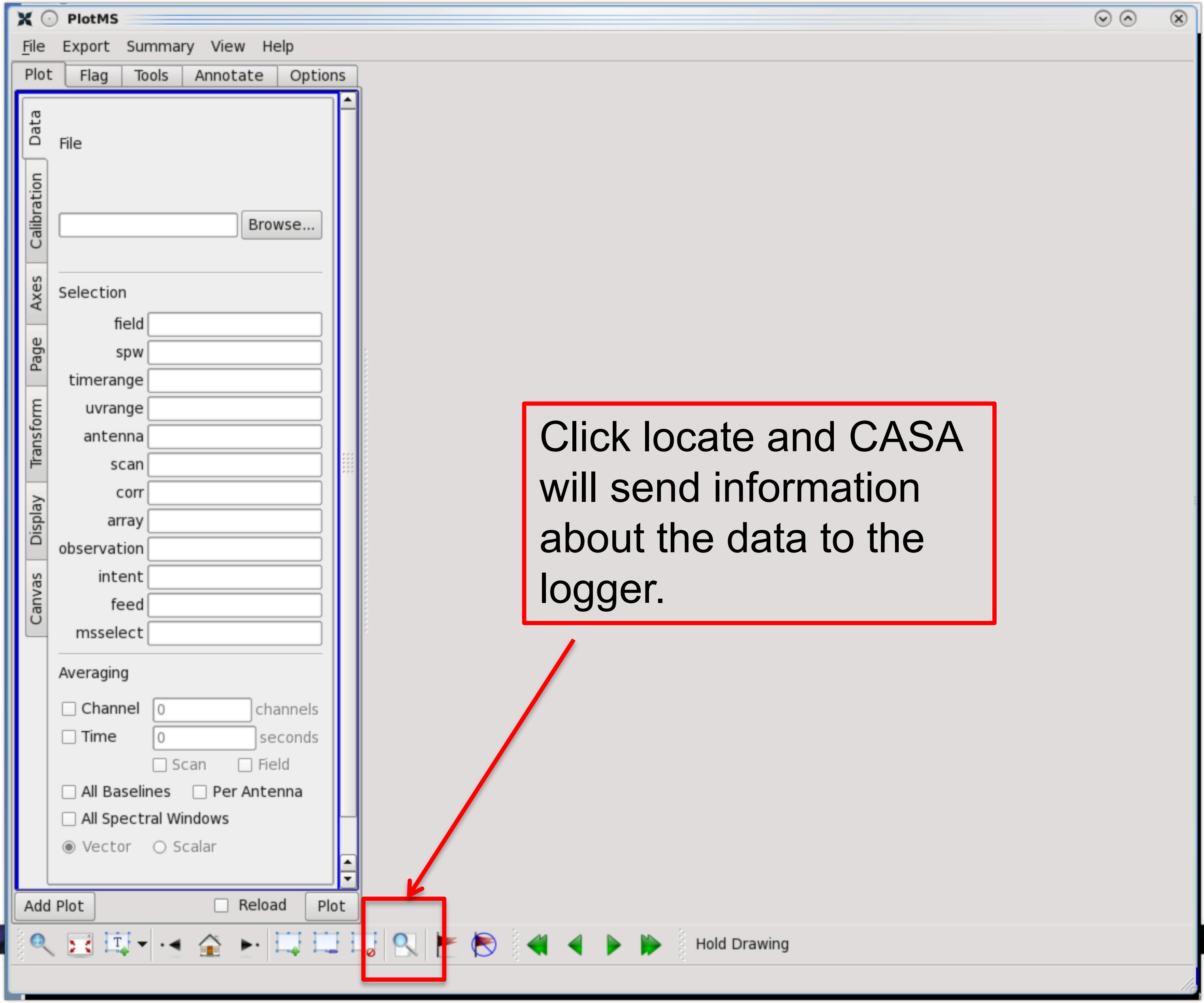
Data Inspection and Flagging

- This next step goes through the basics of data inspection and flagging.
- Throughout the calibration process you will want to create a series of diagnostic plots and use these to identify and remove problematic data. This lesson steps through common steps in identifying and flagging problematic data.
- In the next lesson, we will see how this interplays with calibration in a typical iterative workflow.
- We will now use plotms to make a series of diagnostic plots. These plots have been picked because we have a good expectation of what the calibrators (fields 0, 1, and 2 here) should look like in each space. Before that however, let's walk through the plotms GUI to familiarize ourselves with the interface.

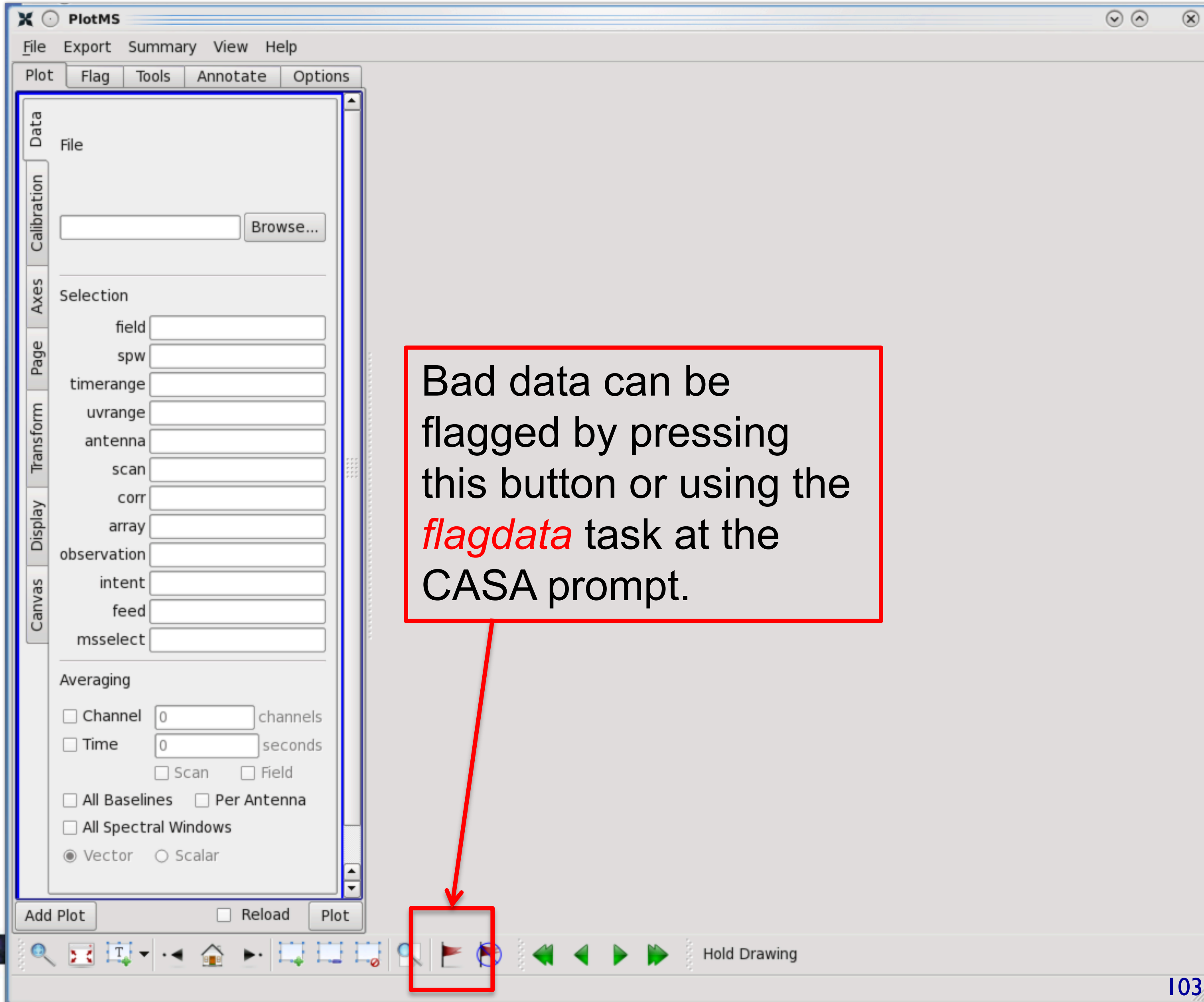
Flagging: Locating Bad Data - *plotms*



Flagging: Locating Bad Data - *plotms*

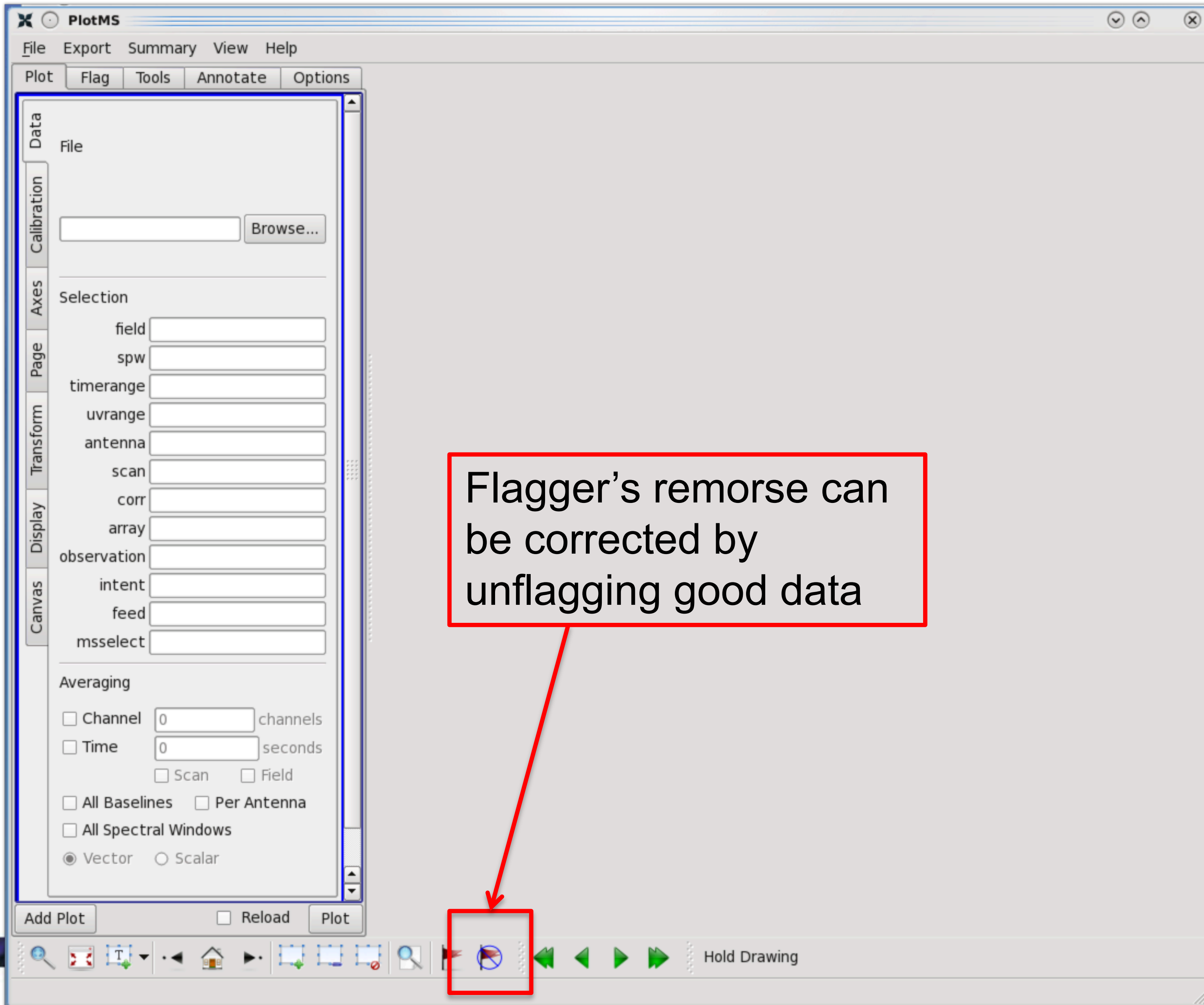


Flagging: Locating Bad Data - *plotms*



Bad data can be flagged by pressing this button or using the *flagdata* task at the CASA prompt.

Flagging: Locating Bad Data - *plotms*

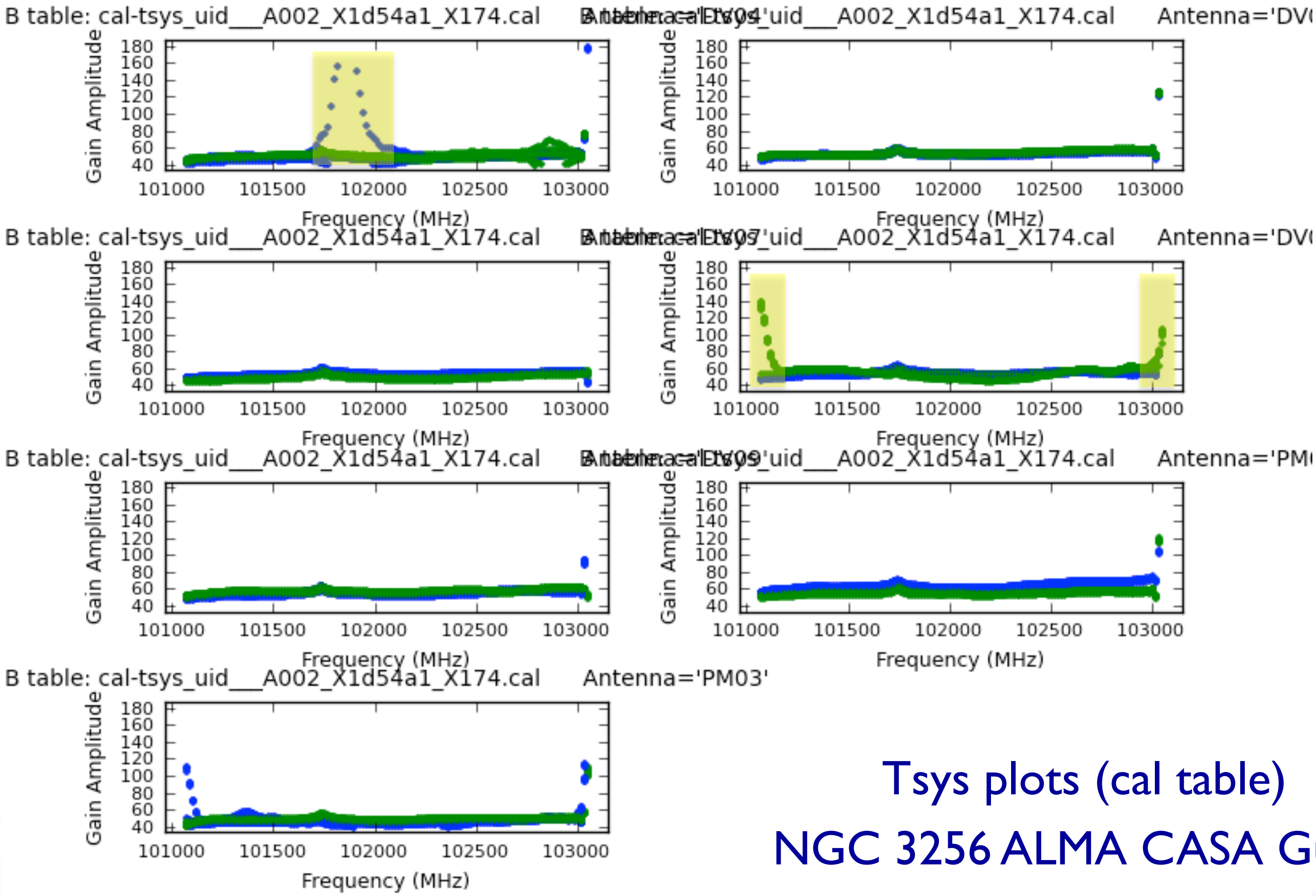


Flagging: What to Look For

- Plots of amplitude and phase vs. time and frequency (gain solutions, visibilities)
- Iterate over
  – Antenna
  – Spectral window
  – Source
- Make plots of calibrators first
  – Easier to find problems in observations of bright point source
  – Harder to find problems in observations of a faint and extended source

Flagging:
Example of an Obvious Issue

Flag the target data for the affected periods (yellow)



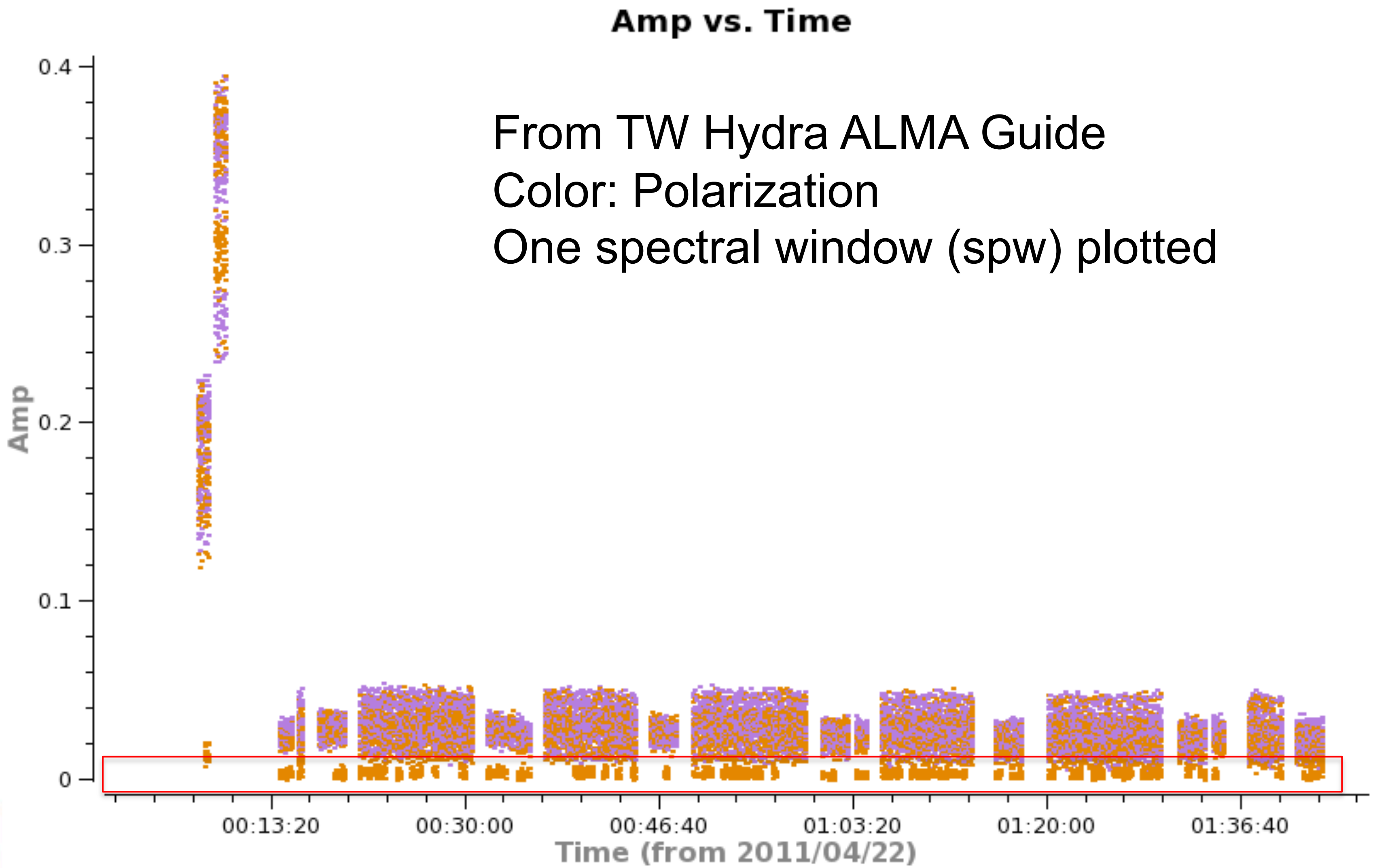
Tsys plots (cal table)

NGC 3256 ALMA CASA Guide

Flagging: What to Look For

- Smoothly varying phases and amplitudes can be calibrated
- Discontinuities can not be calibrated
- Features in the calibrators that may not be in the target data can cause problems

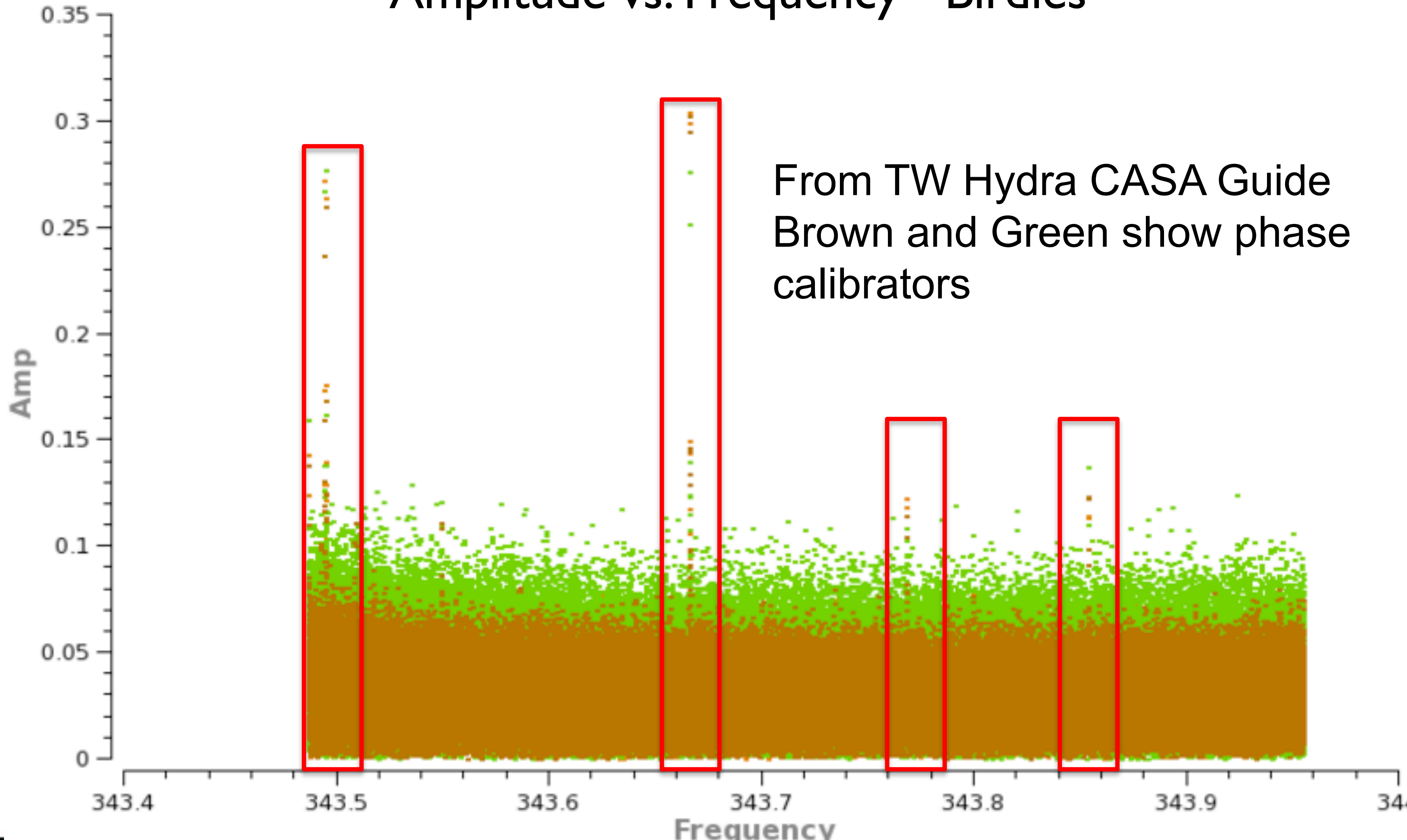
Flagging: What to Look For



From TW Hydra ALMA Guide
Color: Polarization
One spectral window (spw) plotted

Flagging: What to Look For

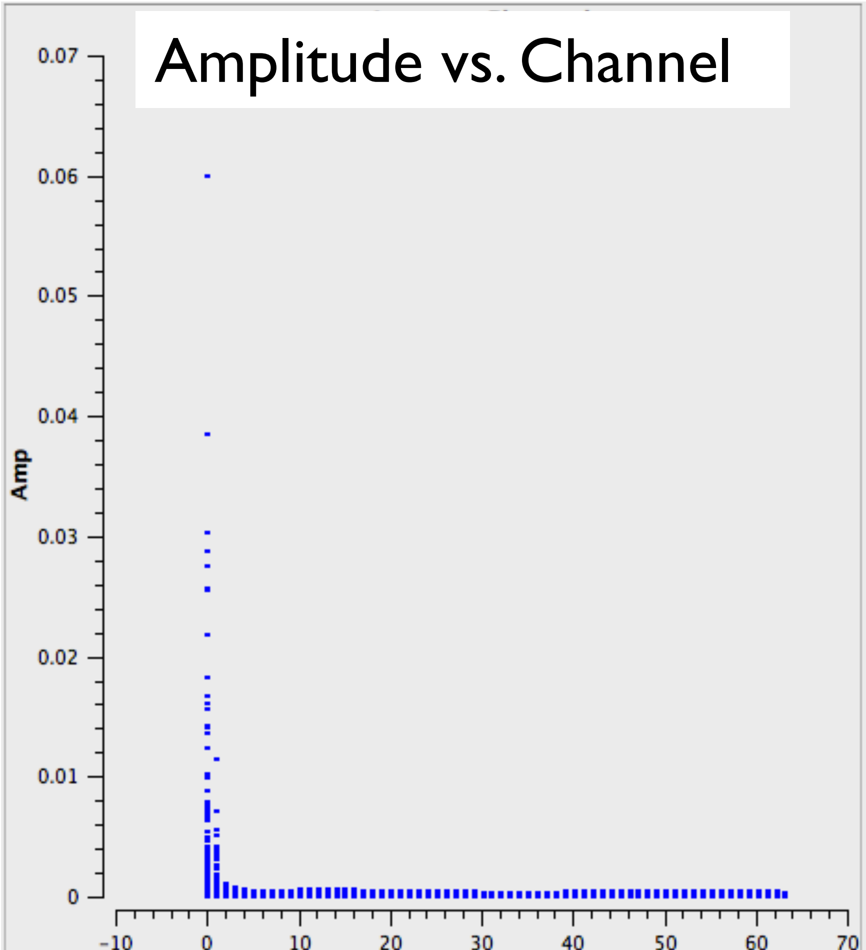
Amplitude vs. Frequency - Birdies

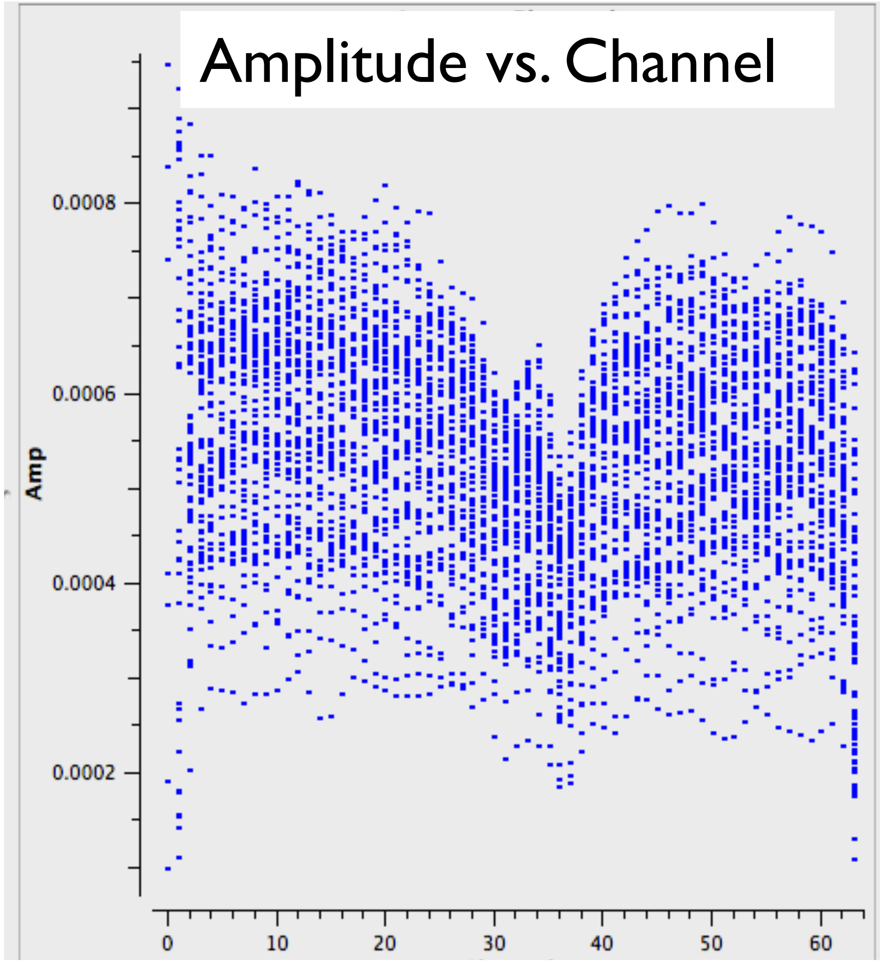


From TW Hydra CASA Guide
Brown and Green show phase calibrators

Flagging: What to Look For

Edge Channels

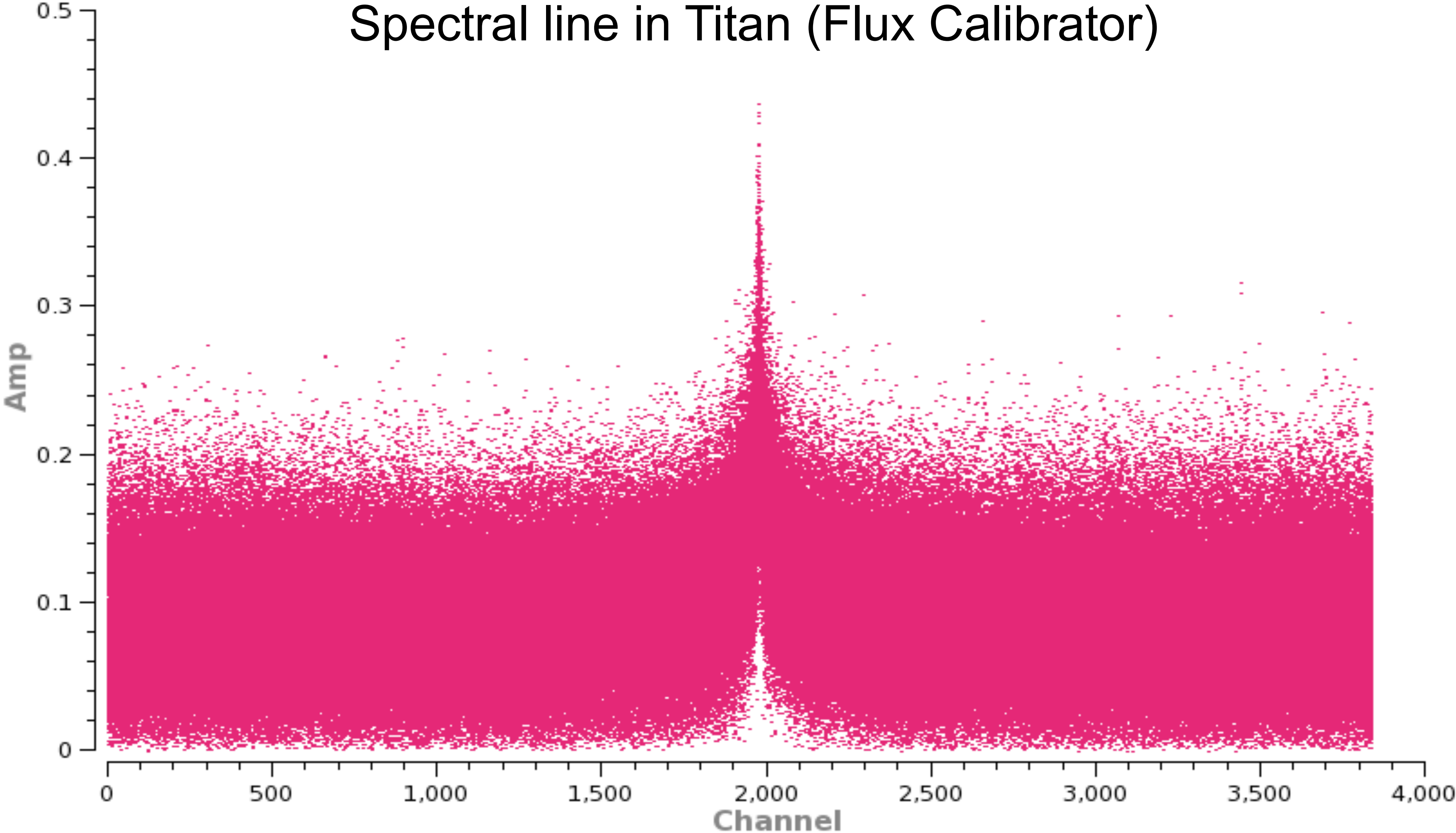




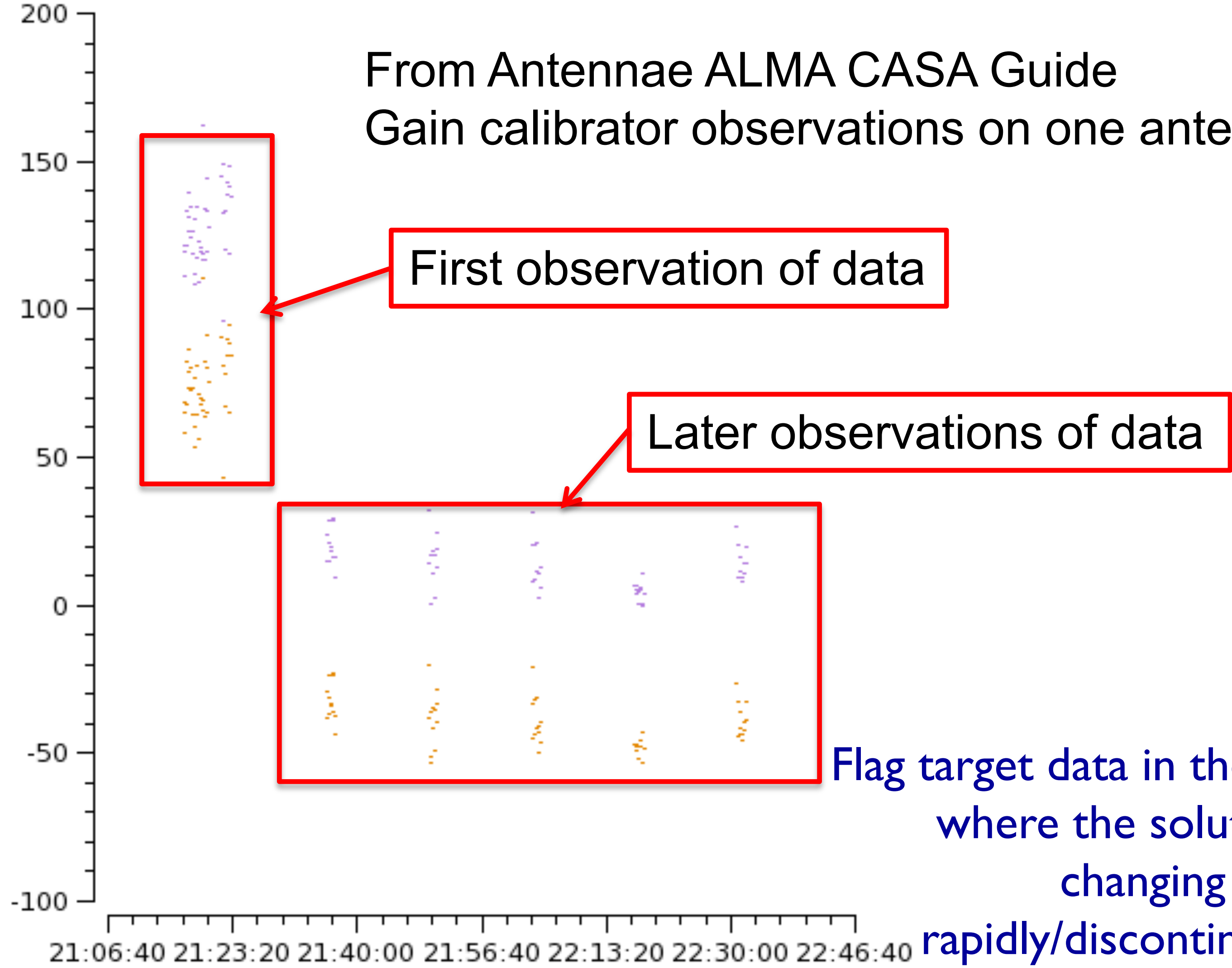
Flagging:
What to Look For

From TW Hydra Band 7 Guide

Spectral line in Titan (Flux Calibrator)



Flagging: What to Look For
Phase vs. Time on Gain Calibrator



From Antennae ALMA CASA Guide
Gain calibrator observations on one antenna

Flag target data in the region where the solution is changing rapidly/discontinuously

Sage Advice

From Rick Perley:
"When in doubt, throw it out."

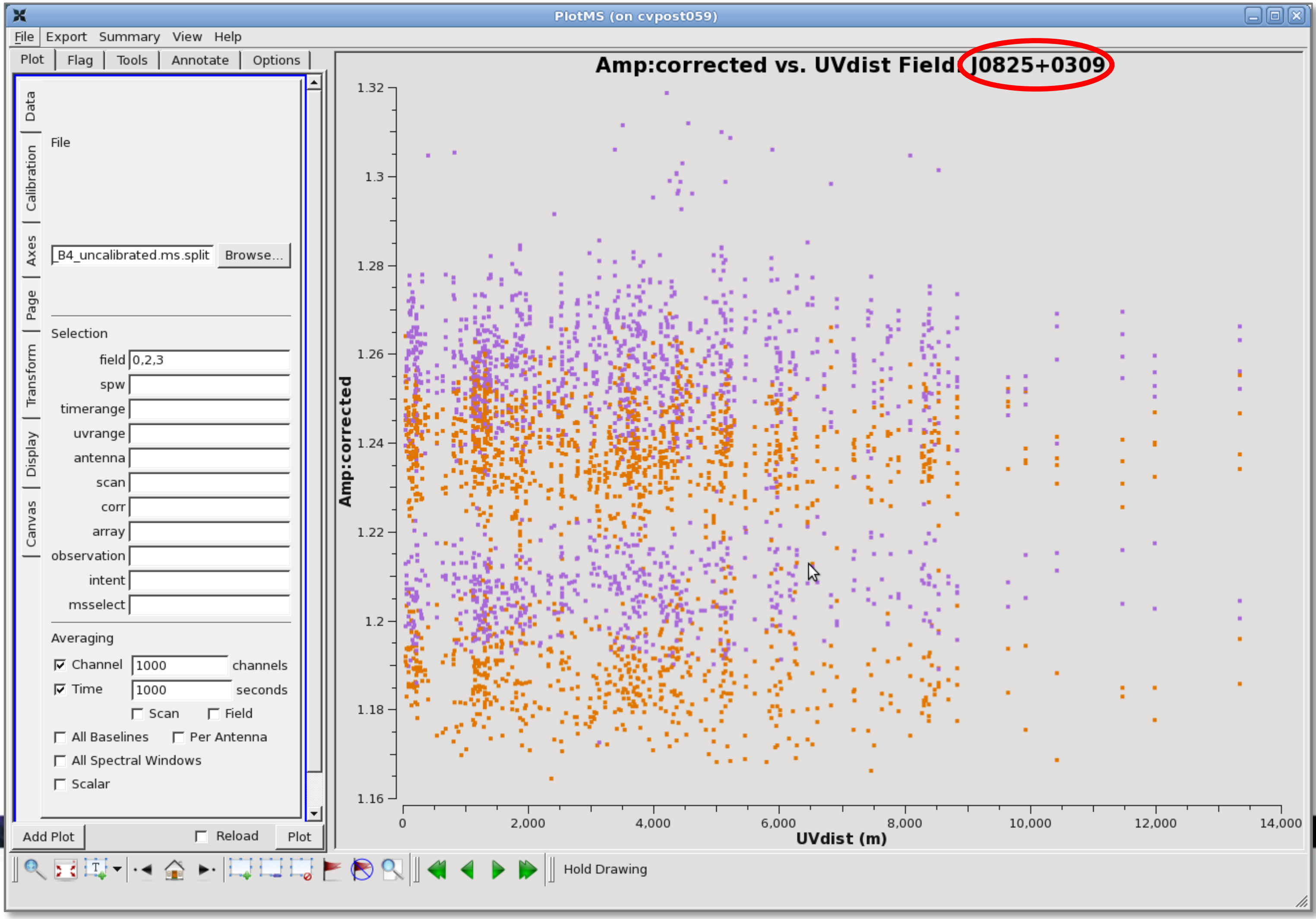
Inspect your Data

In general, we will look through these plots one at a time and look for data that appears as outliers. Use the "locate" function, manipulate the plotted axes, and change the data selection and averaging to try to identify the minimum way to specify the problem data (antenna, scan, channel, etc.). Keep in mind that issues like bad antennas are usually identified using calibrators but are flagged for both calibrators and for the science target.

We will walk you through a few suggested ways of viewing your data for inspection and then give you time to explore on your own. Start with plots of amplitude and phase vs. uv distance. For point sources we expect flat amplitude and zero phases for these plots.

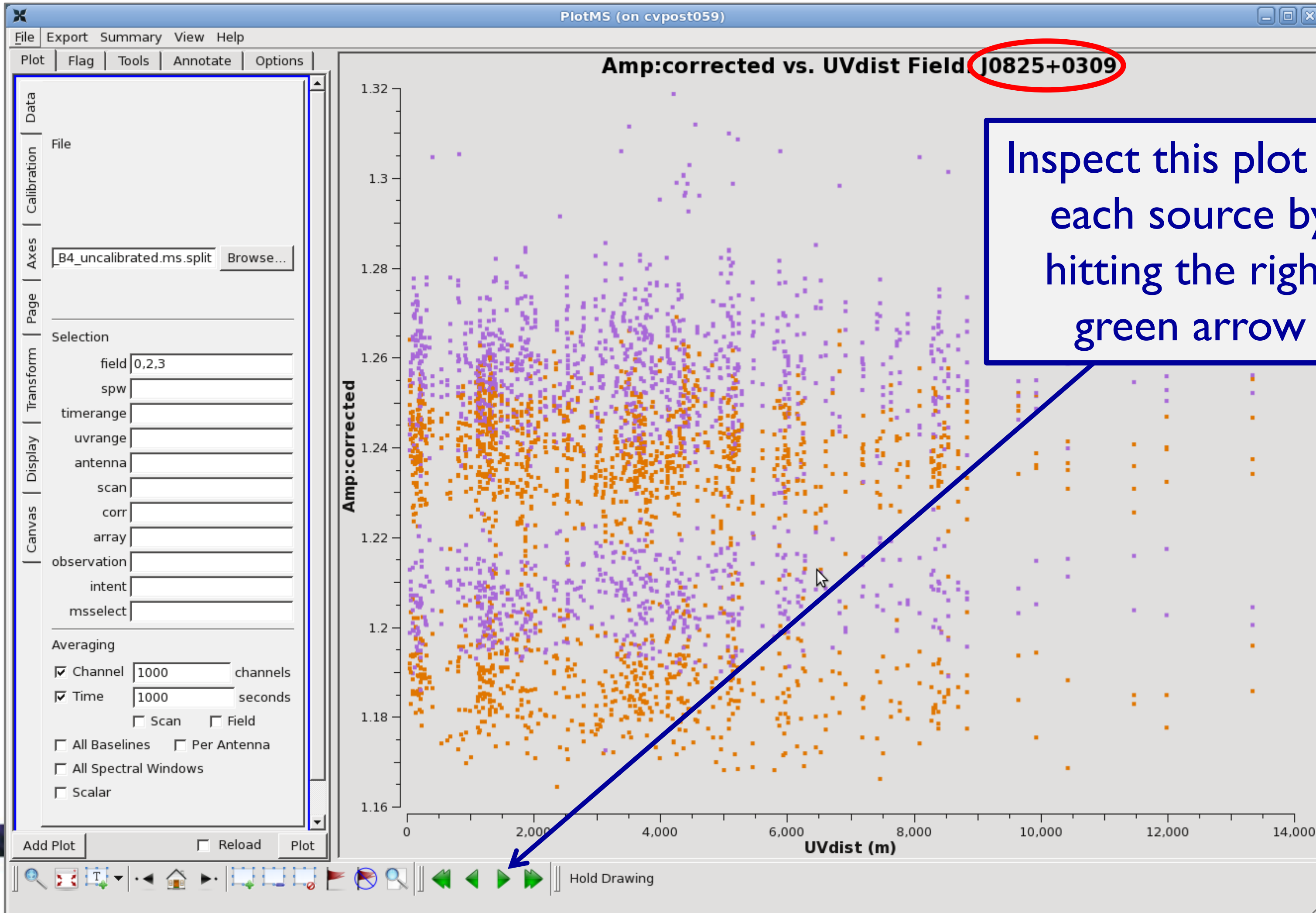
Inspection: Amplitude vs. UVdistance

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="uvdist",
   yaxis="amp", ydatacolumn="corrected", field="0,2,3",
   averagedata=True, avgchannel="1e3", avgtime="1e3",
   iteraxis="field", coloraxis="corr")
```

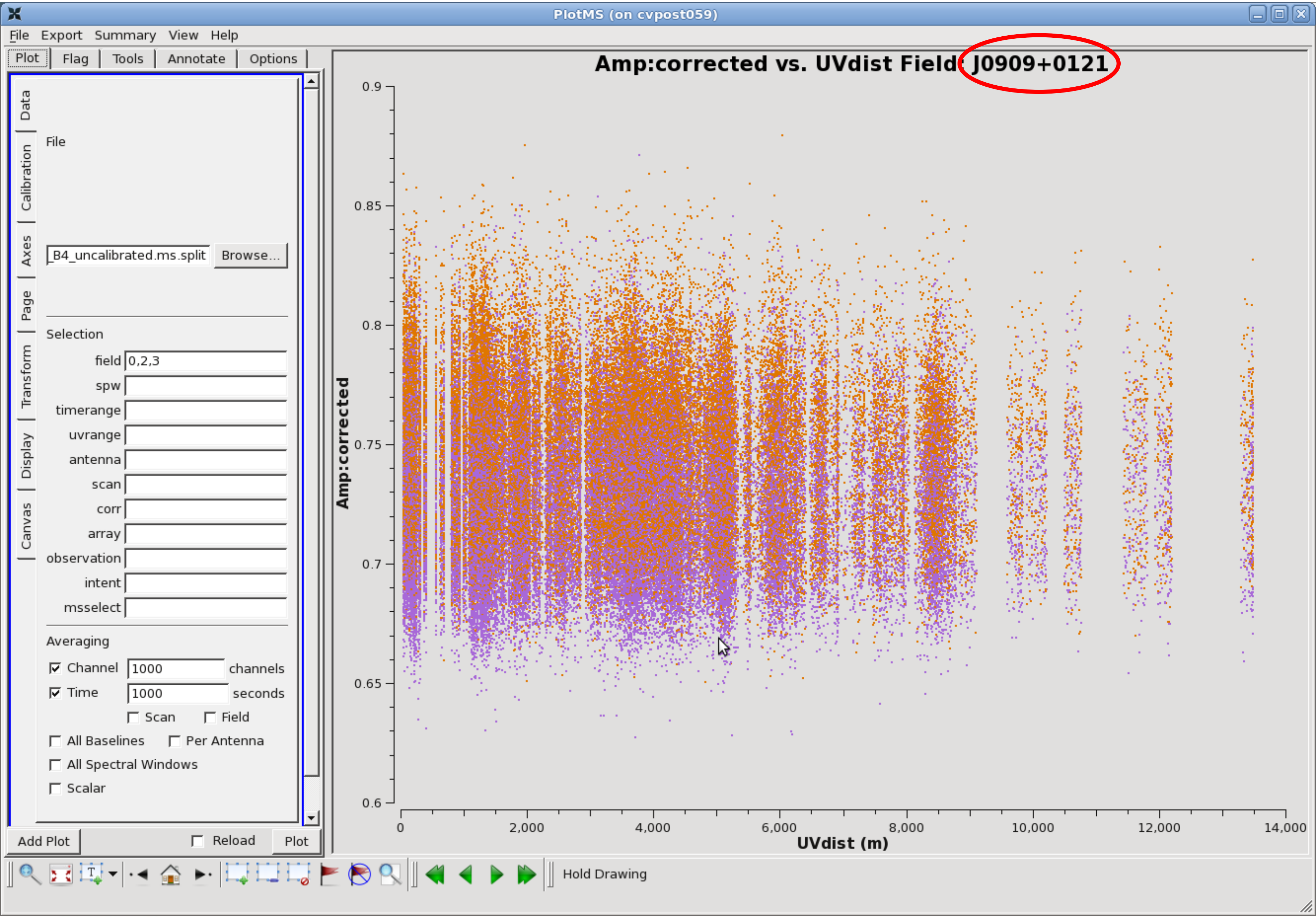


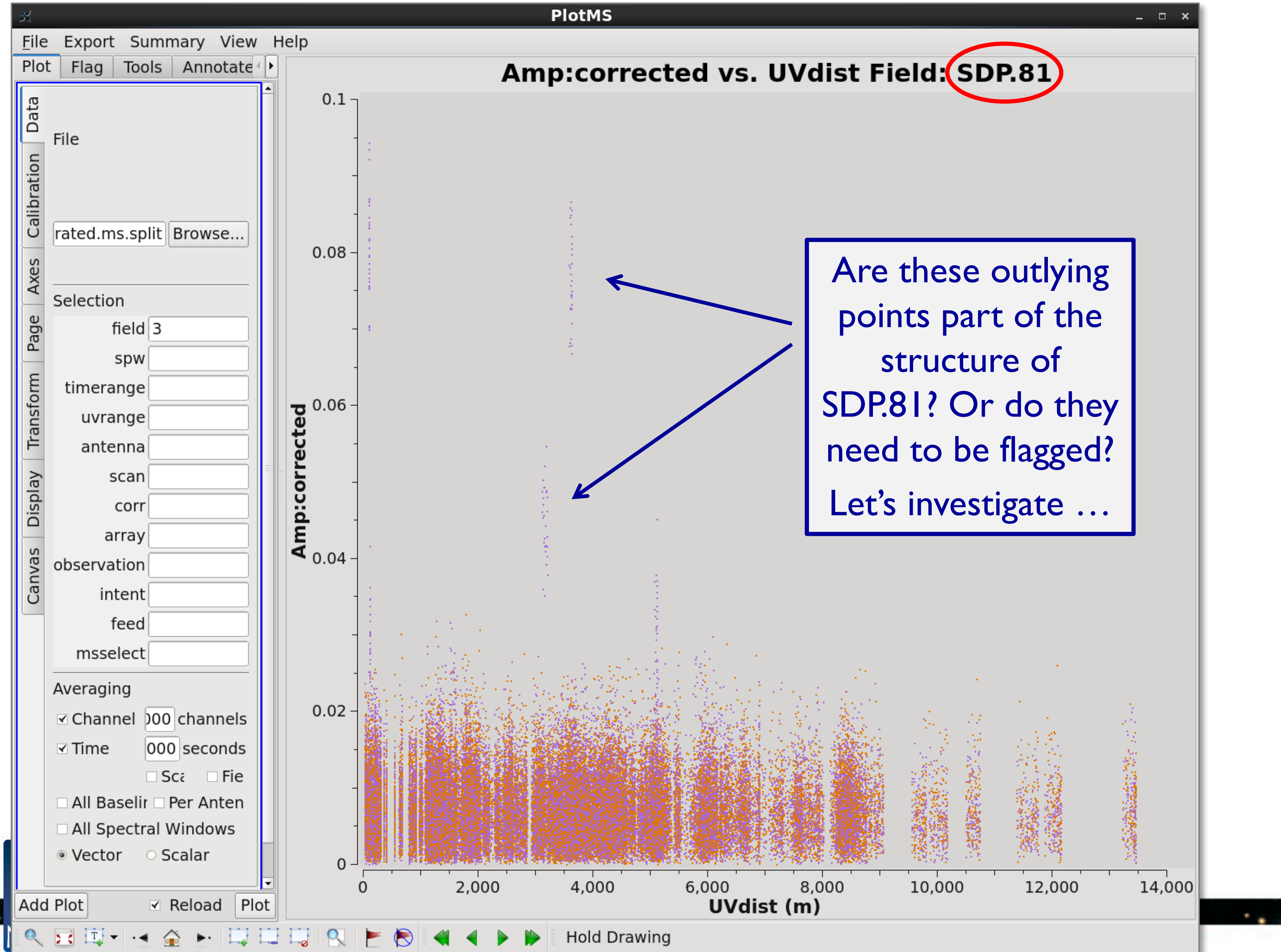
Inspection: Amplitude vs. UVdistance

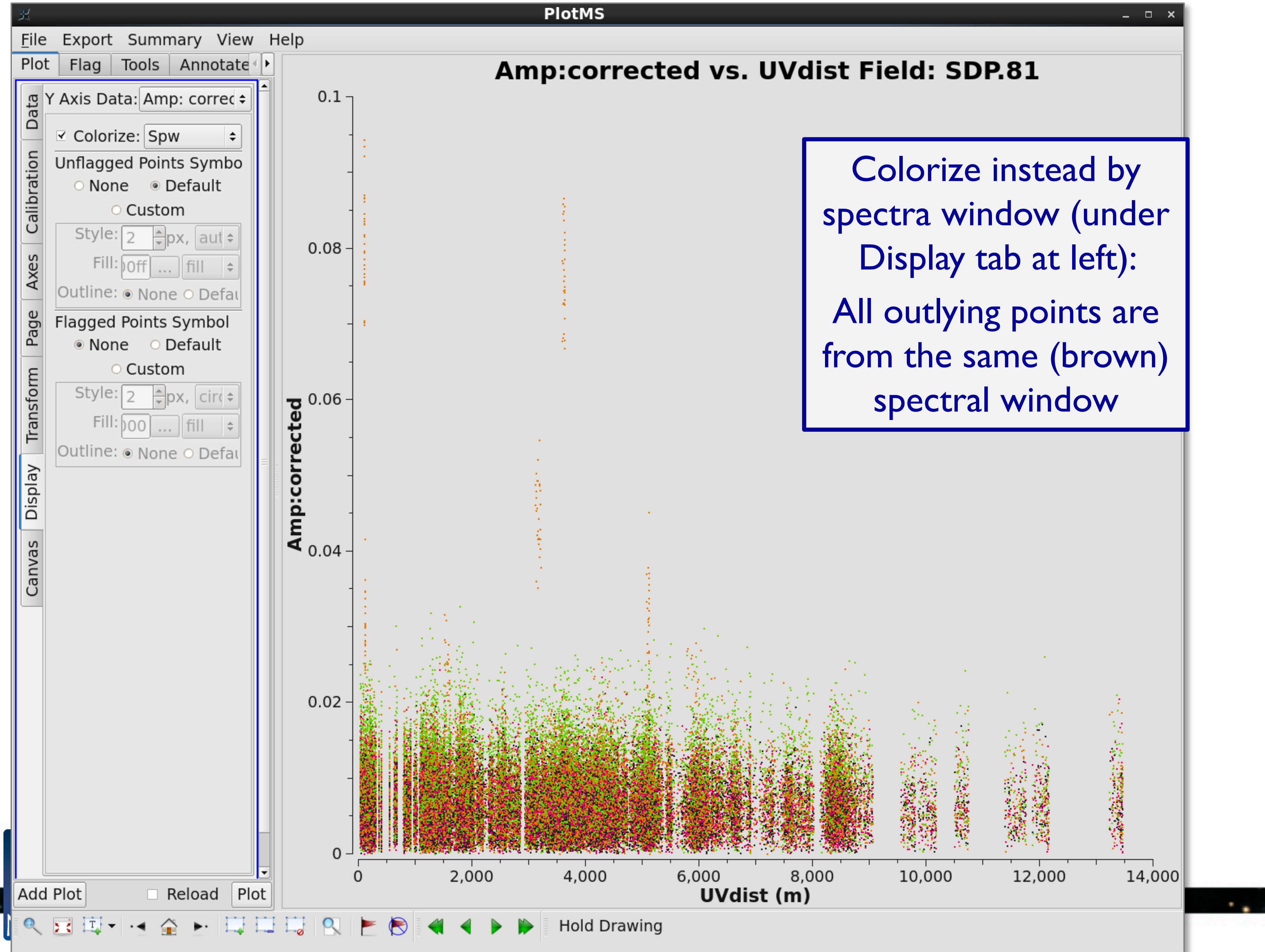
```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="uvdist",
   yaxis="amp", ydatacolumn="corrected", field="0,2,3",
   averagedata=True, avgchannel="1e3", avgtime="1e3",
   iteraxis="field", coloraxis="corr")
```

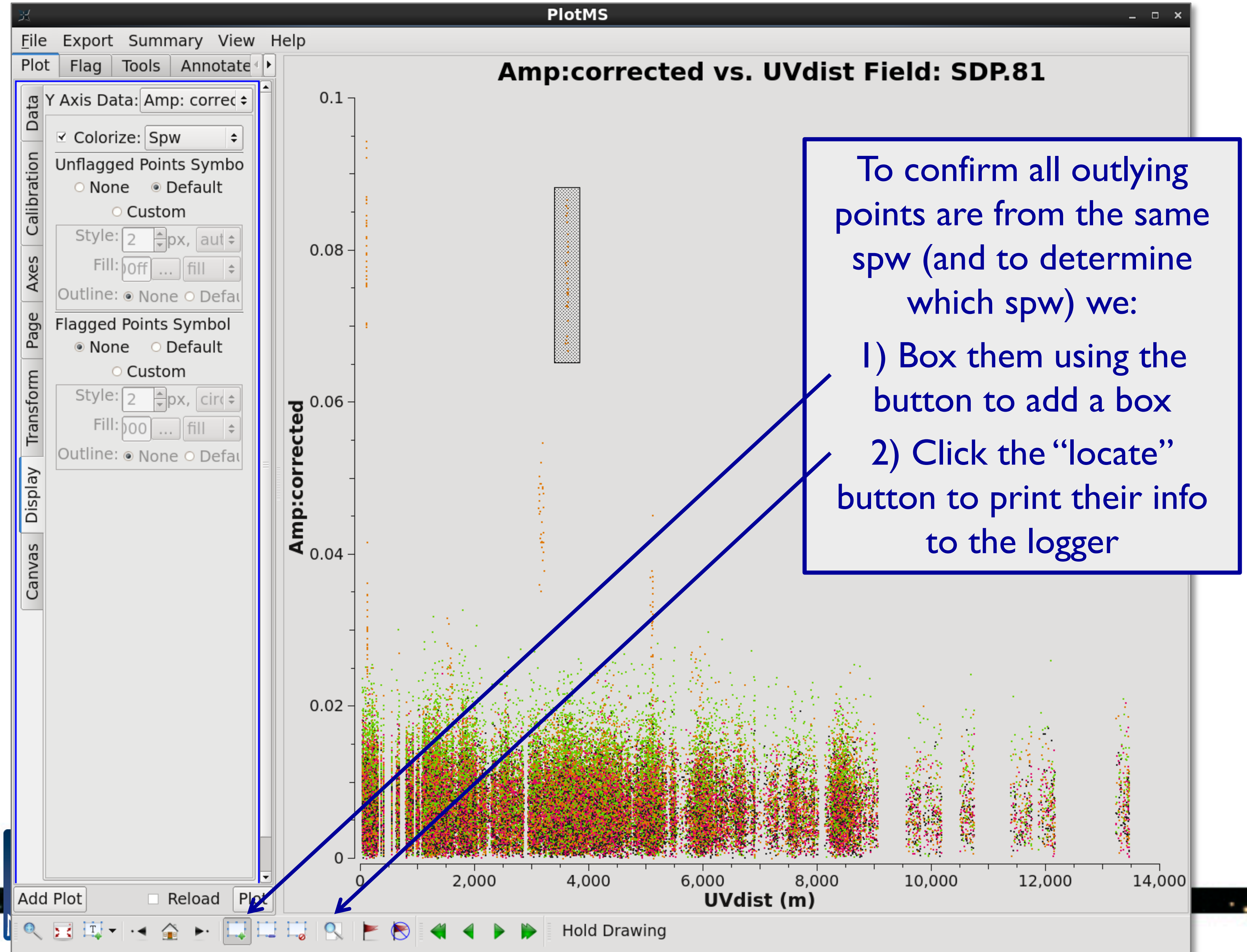


Inspect this plot for each source by hitting the right green arrow


PlotMS (on cvpost059)
File Export Summary View Help
Plot Flag Tools Annotate Options
Data
Calibration
Axes
Page
Transform
Display
Canvas
File
B4_uncalibrated.ms.split
Browse...
Selection
field 0,2,3
spw
timerange
uvrange
antenna
scan
corr
array
observation
intent
msselect
Averaging
Channel 1000 channels
Time 1000 seconds
Scan
Field
All Baselines
Per Antenna
All Spectral Windows
Scalar
Add Plot
Reload
Plot
Amp:corrected vs. UVdist Field: J0909+0121
Amp:corrected
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
2,000
4,000
6,000
8,000
10,000
12,000
14,000
UVdist (m)
Hold Drawing

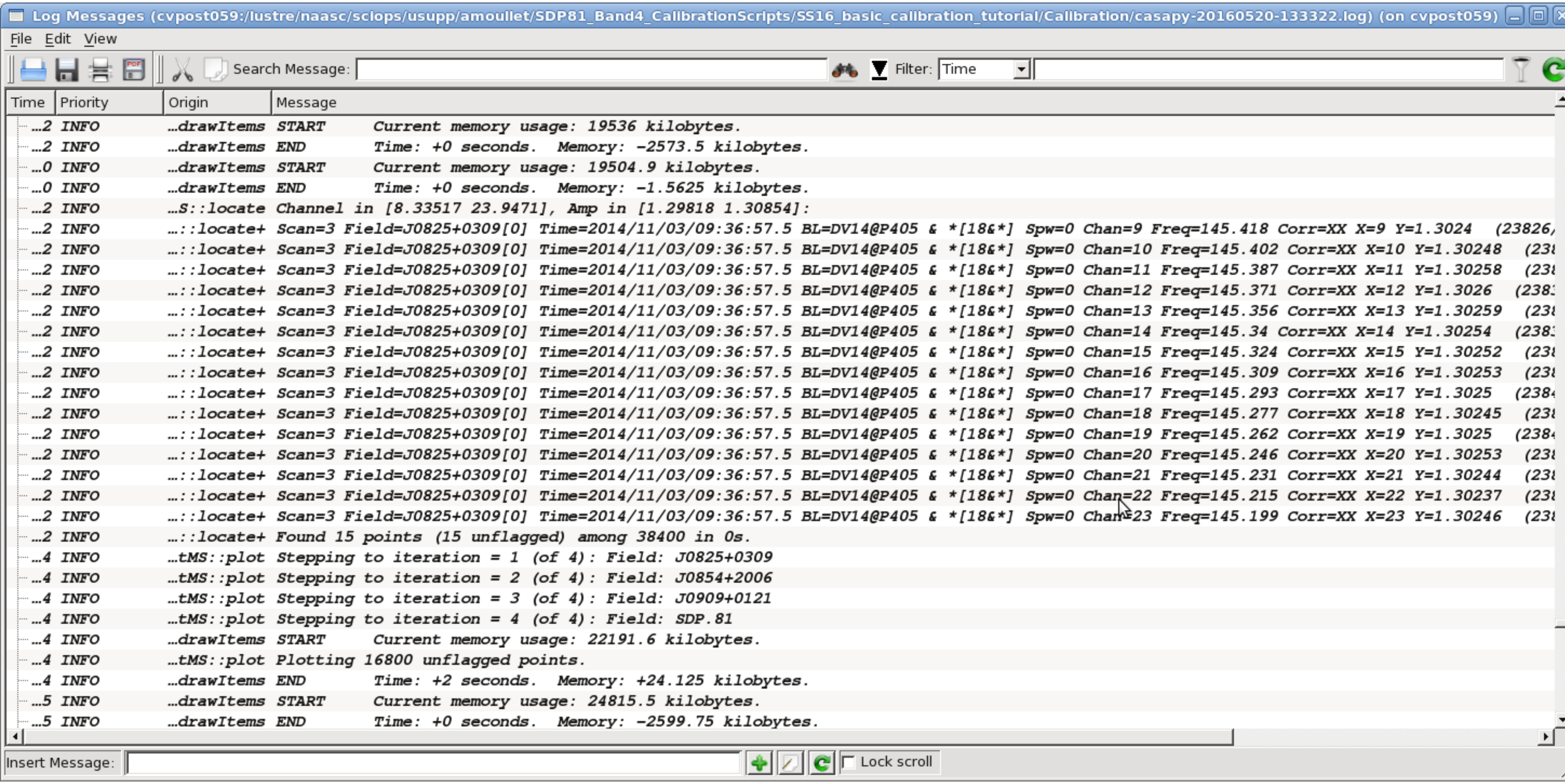

PlotMS
File Export Summary View Help
Plot Flag Tools Annotate
Data Calibration Axes Page Transform Display Canvas
File
rated.ms.split Browse...
Selection
field 3
spw
timerange
uvrange
antenna
scan
corr
array
observation
intent
feed
msselect
Averaging
Channel)00 channels
Time 000 seconds
Sca Fie
All Baselir Per Anten
All Spectral Windows
Vector Scalar
Add Plot Reload Plot
Hold Drawing
Amp:corrected vs. UVdist Field: SDP.81
Amp:corrected
UVdist (m)
Are these outlying points part of the structure of SDP.81? Or do they need to be flagged?
Let's investigate …

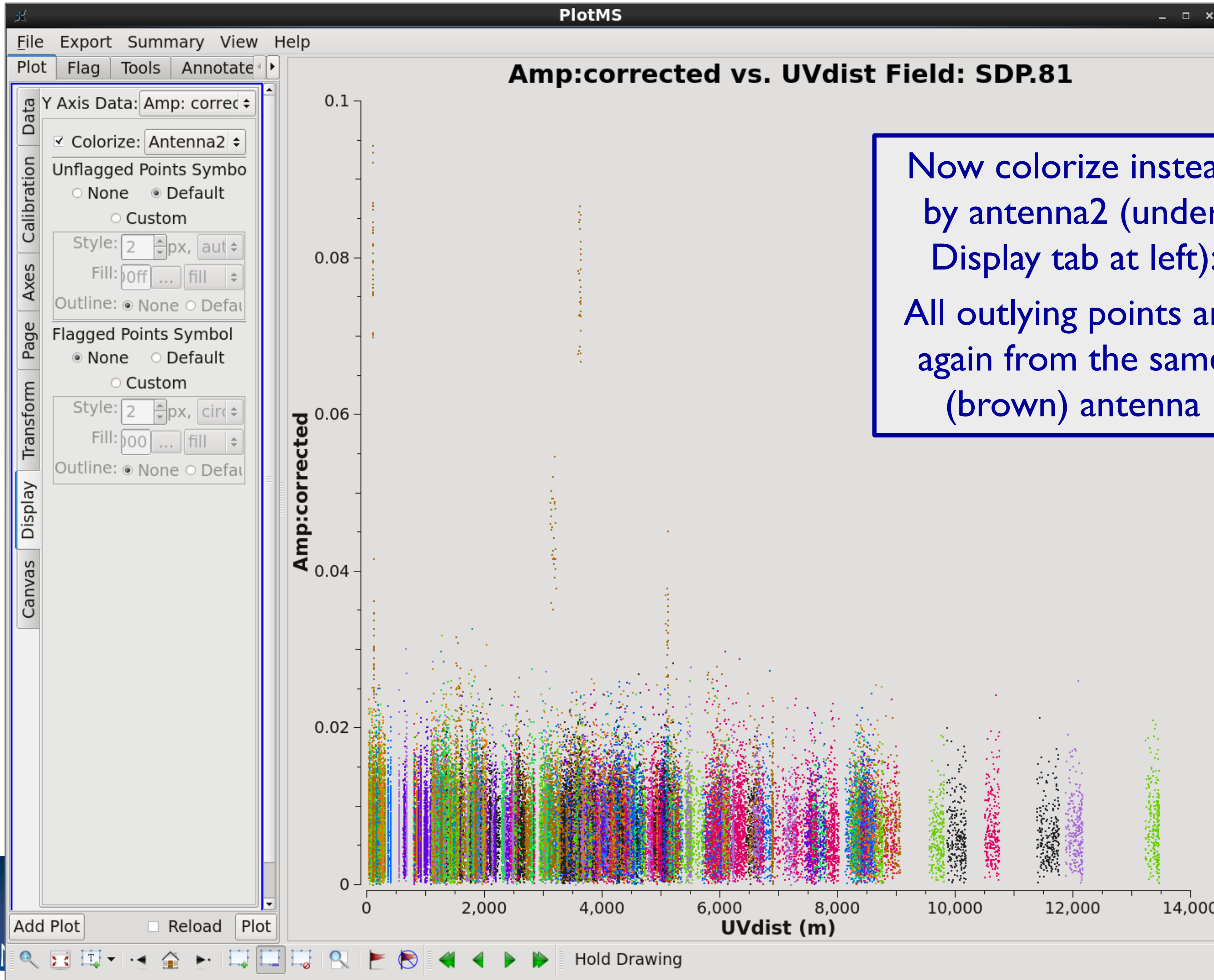

PlotMS
File Export Summary View Help
Plot Flag Tools Annotate
Data Calibration Axes Page Transform Display Canvas
Y Axis Data: Amp: correc
Colorize: Spw
Unflagged Points Symbo
None Default
Custom
Style: 2 px, aut
Fill: 0ff ... fill
Outline: None Defau
Flagged Points Symbol
None Default
Custom
Style: 2 px, circ
Fill: 00 ... fill
Outline: None Defau
Add Plot Reload Plot
Hold Drawing
Amp:corrected vs. UVdist Field: SDP.81
Amp:corrected
UVdist (m)
0 0.02 0.04 0.06 0.08 0.1
0 2,000 4,000 6,000 8,000 10,000 12,000 14,000
Colorize instead by spectra window (under Display tab at left):
All outlying points are from the same (brown) spectral window



PlotMS
File Export Summary View Help
Plot Flag Tools Annotate
Y Axis Data: Amp: correc
Colorize: Spw
Unflagged Points Symbo
None Default
Custom
Style: 2 px, aut
Fill: 0ff ... fill
Outline: None Defau
Flagged Points Symbol
None Default
Custom
Style: 2 px, circ
Fill: 000 ... fill
Outline: None Defau
Data
Calibration
Axes
Page
Transform
Display
Canvas
Add Plot
Reload
Plot
Amp:corrected vs. UVdist Field: SDP.81
Amp:corrected
UVdist (m)
Hold Drawing
To confirm all outlying points are from the same spw (and to determine which spw) we:
1) Box them using the button to add a box
2) Click the "locate" button to print their info to the logger

Inspection: Example output from locate tool in plotms





Now colorize instead by antenna2 (under Display tab at left):

All outlying points are again from the same (brown) antenna

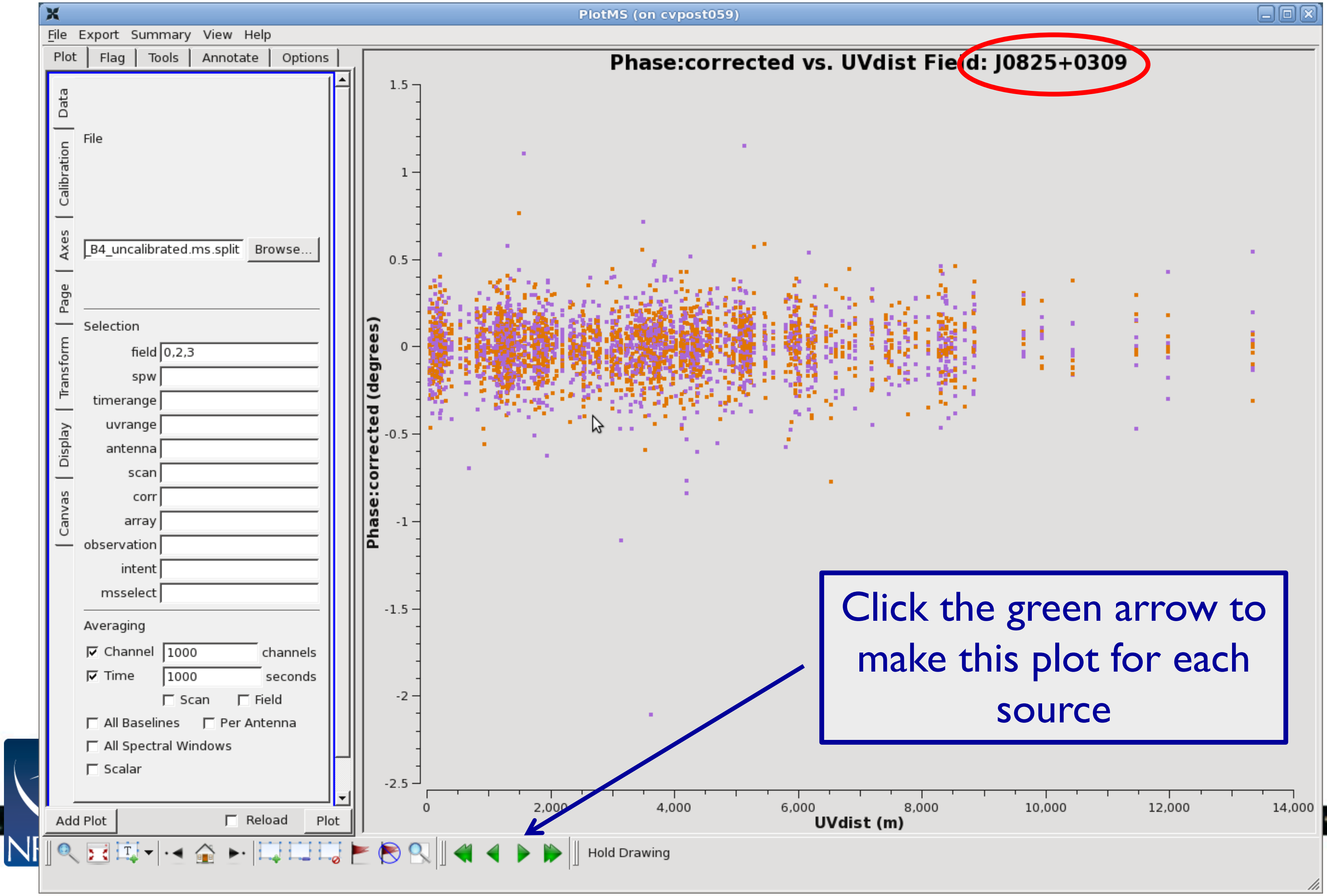
Inspection: Determining what data to flag

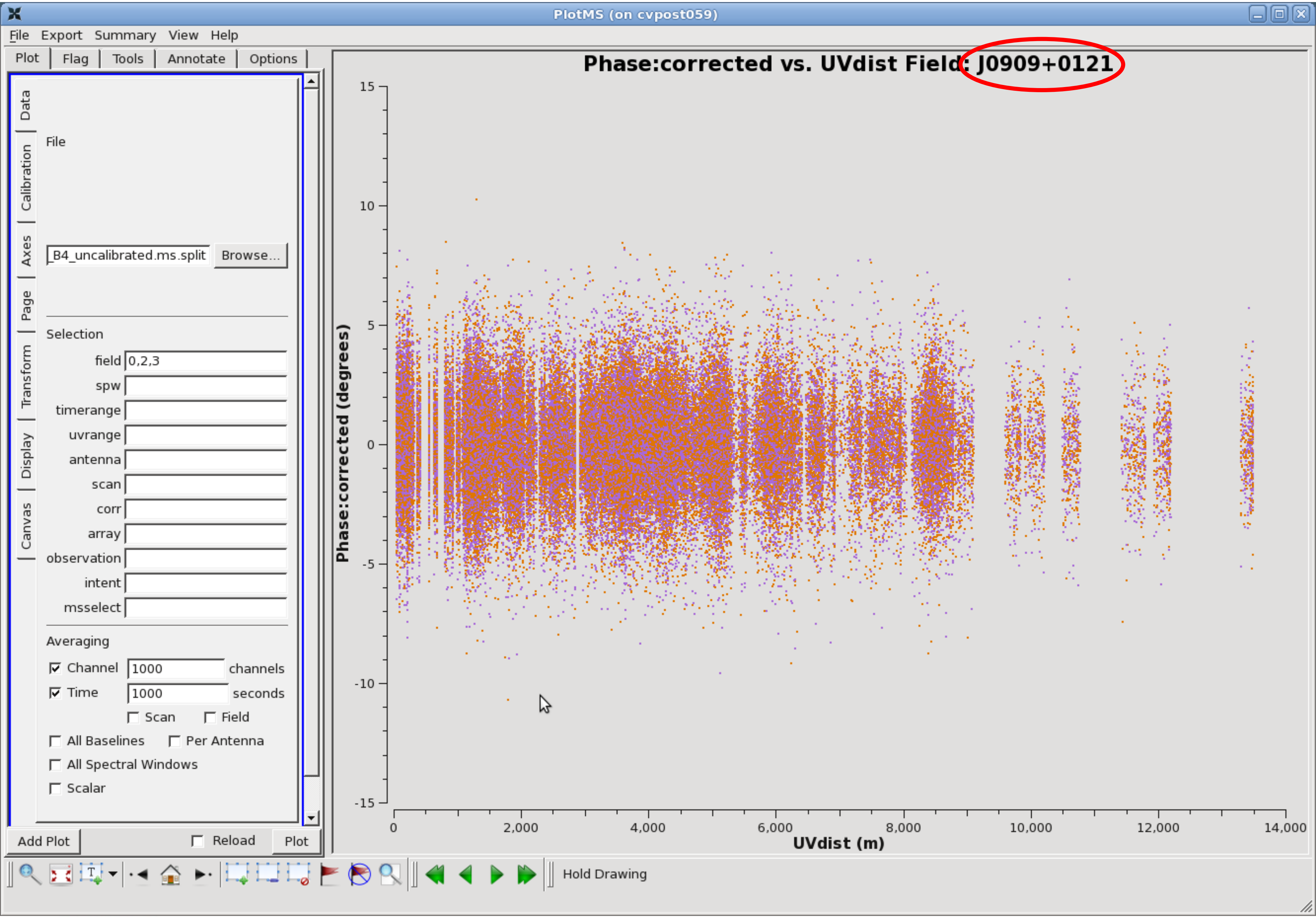
Given the often weaker flux of a science target, it is often difficult to discern features that could be representative of real source structure from problematic data that needs flagging.

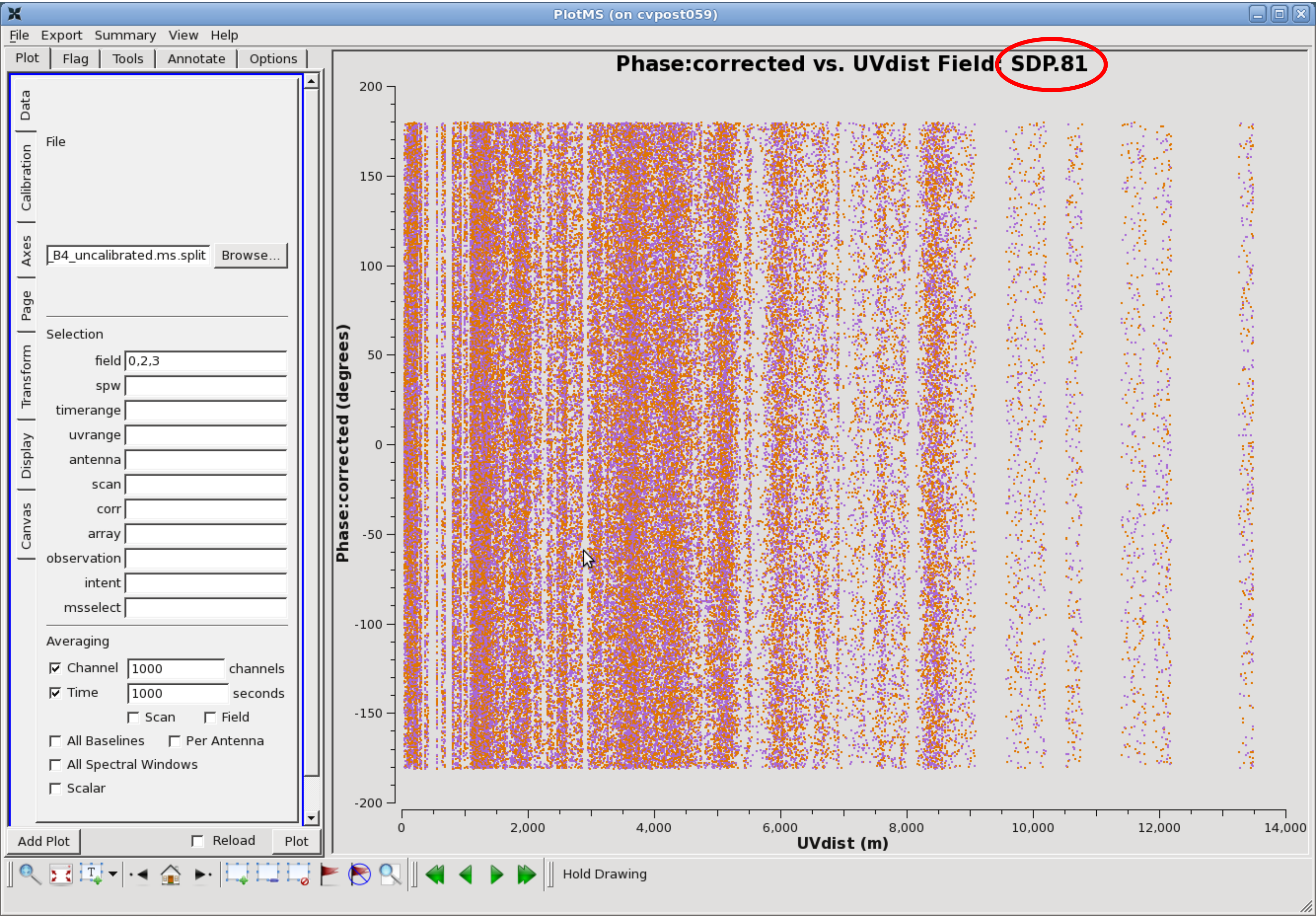
In the case of the outlying points in the plots we have inspected for SDP.81, they are all from the same antenna and the same spectral window. This is highly unlikely to be source structure and so can should flagged.

Inspection: Phase vs. UVdistance

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="uvdist", yaxis="phase",
    ydatacolumn="corrected",   field="0,2,3", avgdata=True,
    avgchannel="1e3", avgtime="1e3", iteraxis="field", coloraxis="corr")
```

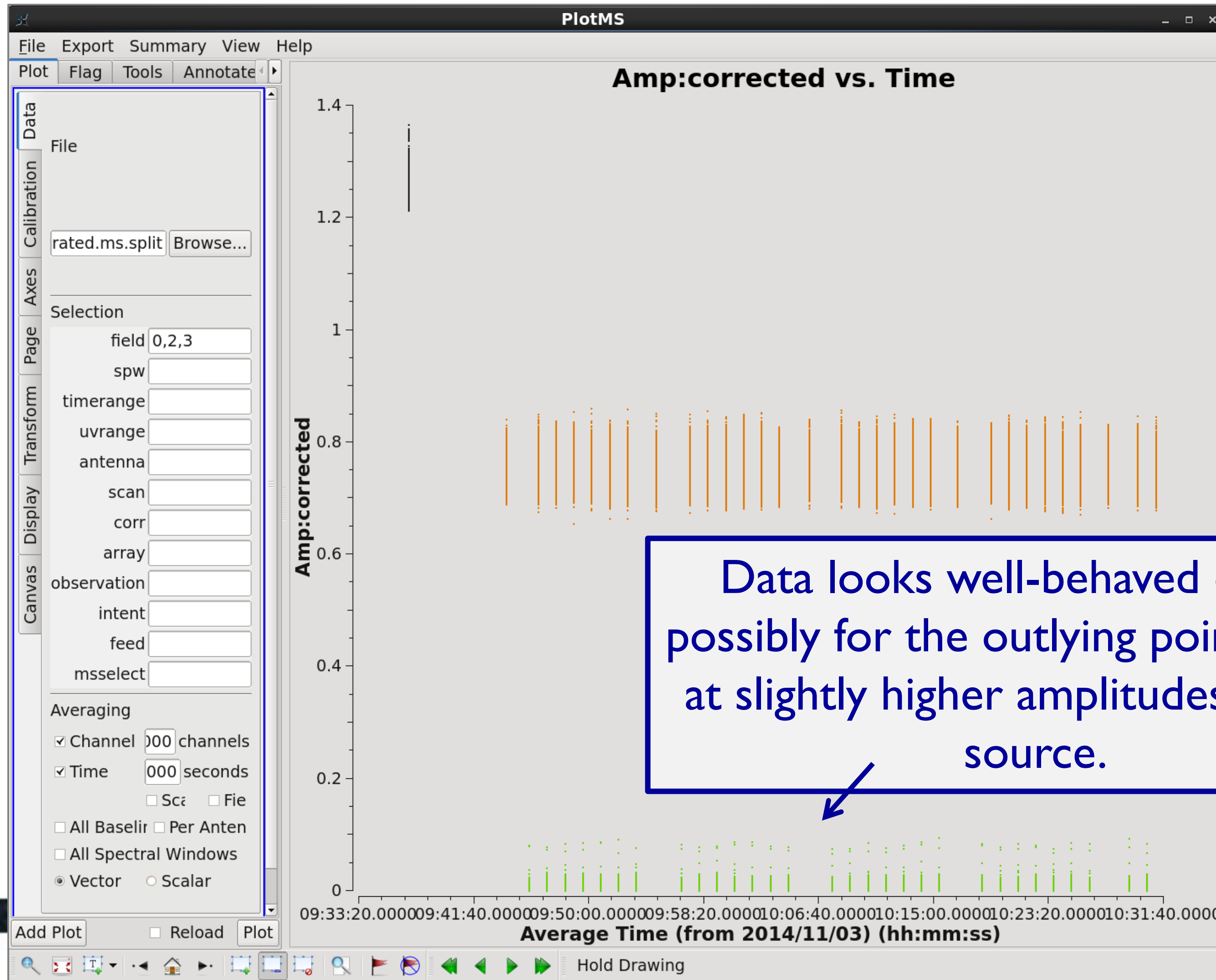



PlotMS (on cvpost059)
File Export Summary View Help
Plot Flag Tools Annotate Options
Data
Calibration
Axes
Page
Transform
Display
Canvas
File
B4_uncalibrated.ms.split
Browse...
Selection
field 0,2,3
spw
timerange
uvrange
antenna
scan
corr
array
observation
intent
msselect
Averaging
Channel 1000 channels
Time 1000 seconds
Scan
Field
All Baselines
Per Antenna
All Spectral Windows
Scalar
Add Plot
Reload
Plot
Phase:corrected vs. UVdist Field: J0909+0121
Phase:corrected (degrees)
UVdist (m)
Hold Drawing


PlotMS (on cvpost059)
File Export Summary View Help
Plot Flag Tools Annotate Options
Data
Calibration
Axes
Page
Transform
Display
Canvas
File
B4_uncalibrated.ms.split
Browse...
Selection
field 0,2,3
spw
timerange
uvrange
antenna
scan
corr
array
observation
intent
msselect
Averaging
Channel 1000 channels
Time 1000 seconds
Scan
Field
All Baselines
Per Antenna
All Spectral Windows
Scalar
Add Plot
Reload
Plot
Phase:corrected vs. UVdist Field: SDP.81
Phase:corrected (degrees)
UVdist (m)
Hold Drawing

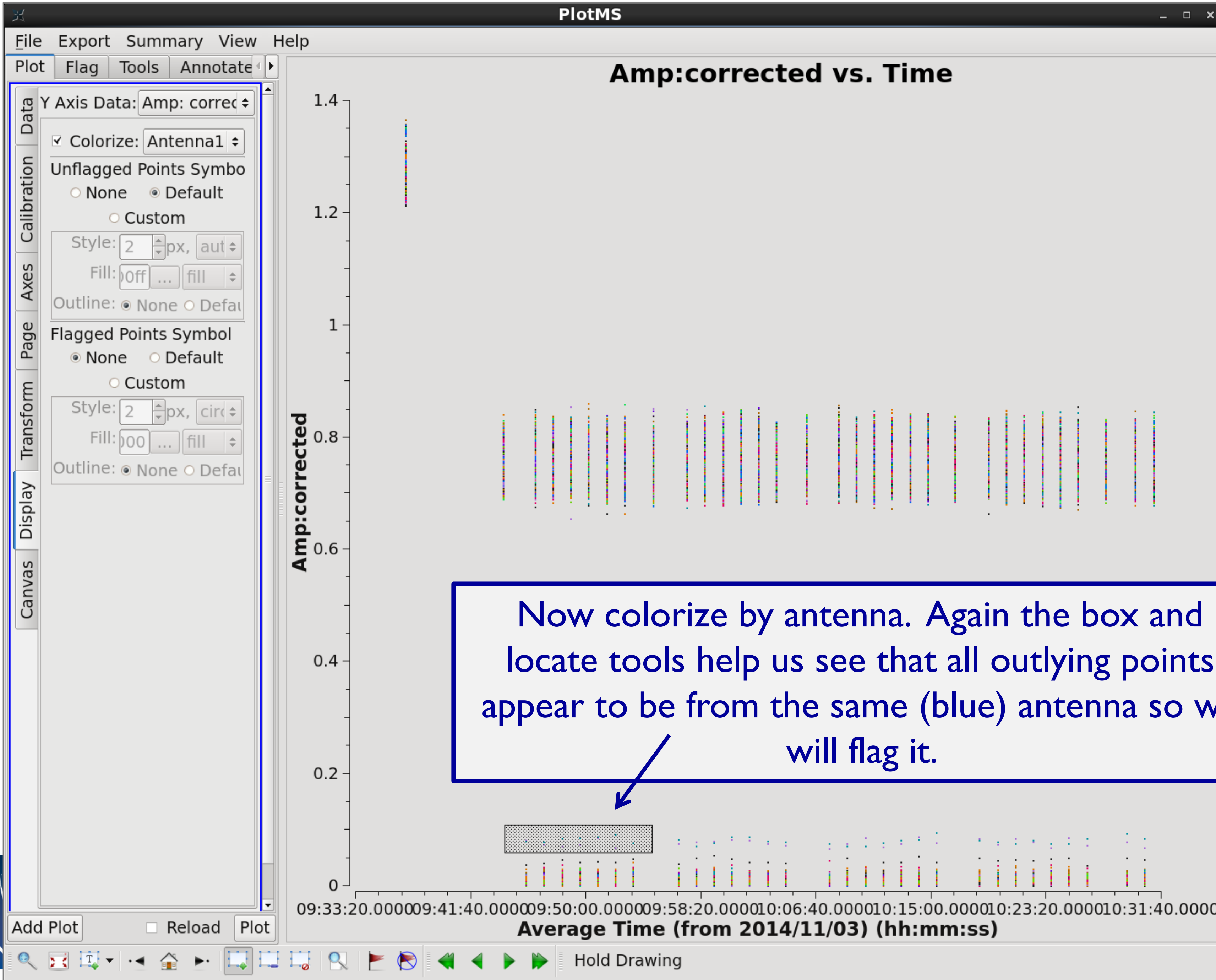
Inspection: Scan-to-Scan Variations in Amplitude

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="time",
   yaxis="amp", ydatacolumn="corrected",   field="0,2,3",
   avgdata=True, avgchannel="1e3", avgtime="1e3", coloraxis="field")
```



Data looks well-behaved except possibly for the outlying points sitting at slightly higher amplitudes for our source.

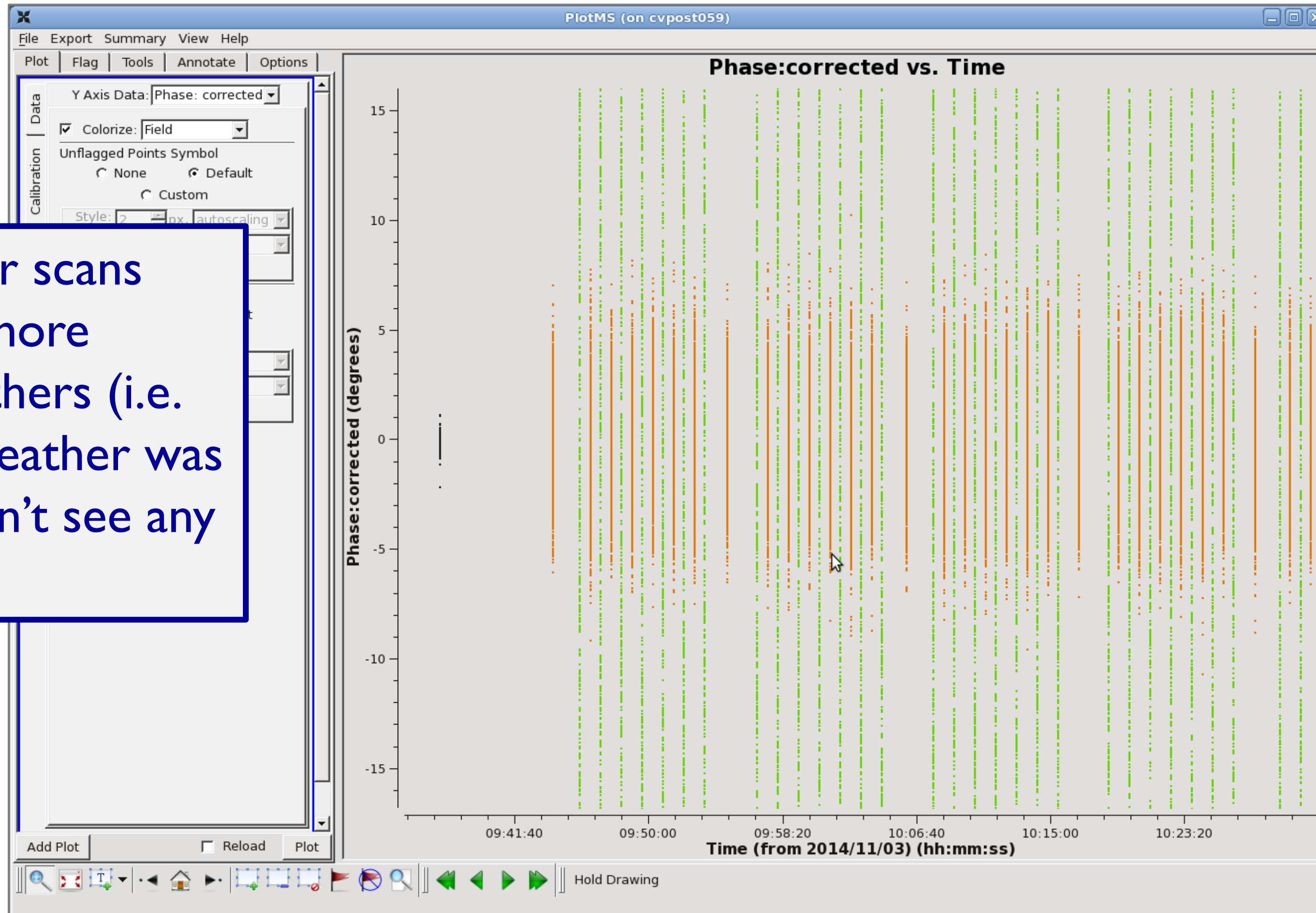
NRAO



Now colorize by antenna. Again the box and locate tools help us see that all outlying points appear to be from the same (blue) antenna so we will flag it.

Inspection: Scan-to-Scan Variations in Phase

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="time",
   yaxis="phase", ydatacolumn="corrected", field="0,2,3",
   avgdata=True, avgchannel="1e3", avgtime="1e3", coloraxis="field")
```



We are looking for scans with significantly more dispersion than others (i.e. times when the weather was worse) but we don't see any issues.

Lines or Spikes

Finally, we don't expect strong lines in the calibrators and sharp unexpected spikes anywhere are likely to be spurious. We will likely want to flag any lines or spikes. Plot the amplitude and phase as function of channel for the calibrators and the source.

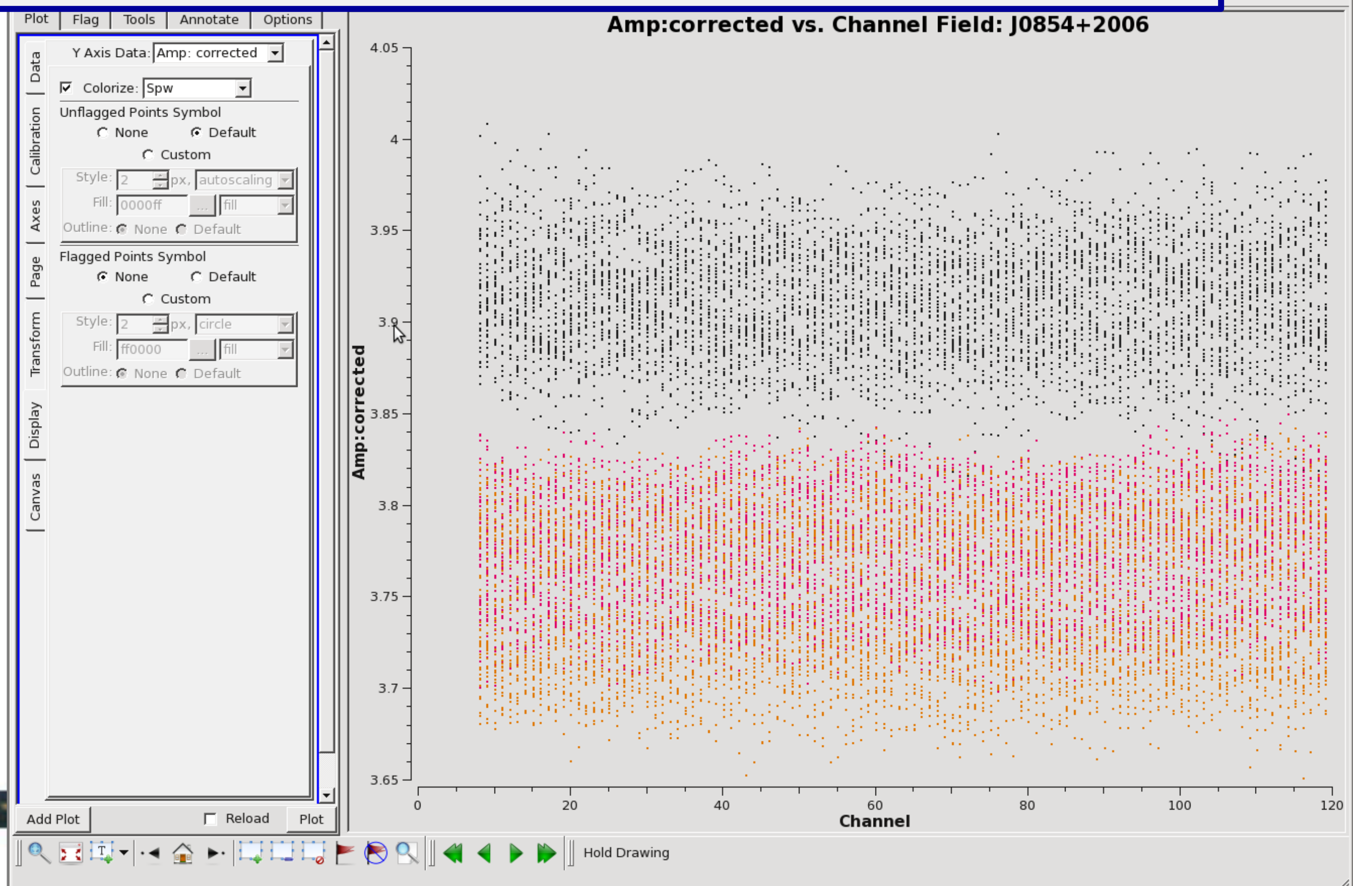
First we will plot our three spectral windows with wide channels (128 channels with 15625 kHz each; i.e. those set for continuum – see listobs output).

Then we will plot our final spectra window set with narrower channels (3840 channels with 488 kHz each).

Inspection: Spectral Windows with Wide Channels

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="channel",
   yaxis="amp", ydatacolumn="corrected", field="0,1,2,3",
   avgdata=True, avgchannel=" ", avgtime="1e6", coloraxis="spw",
   iteraxis="field", spw="0,1,2", avgantenna=True)
```

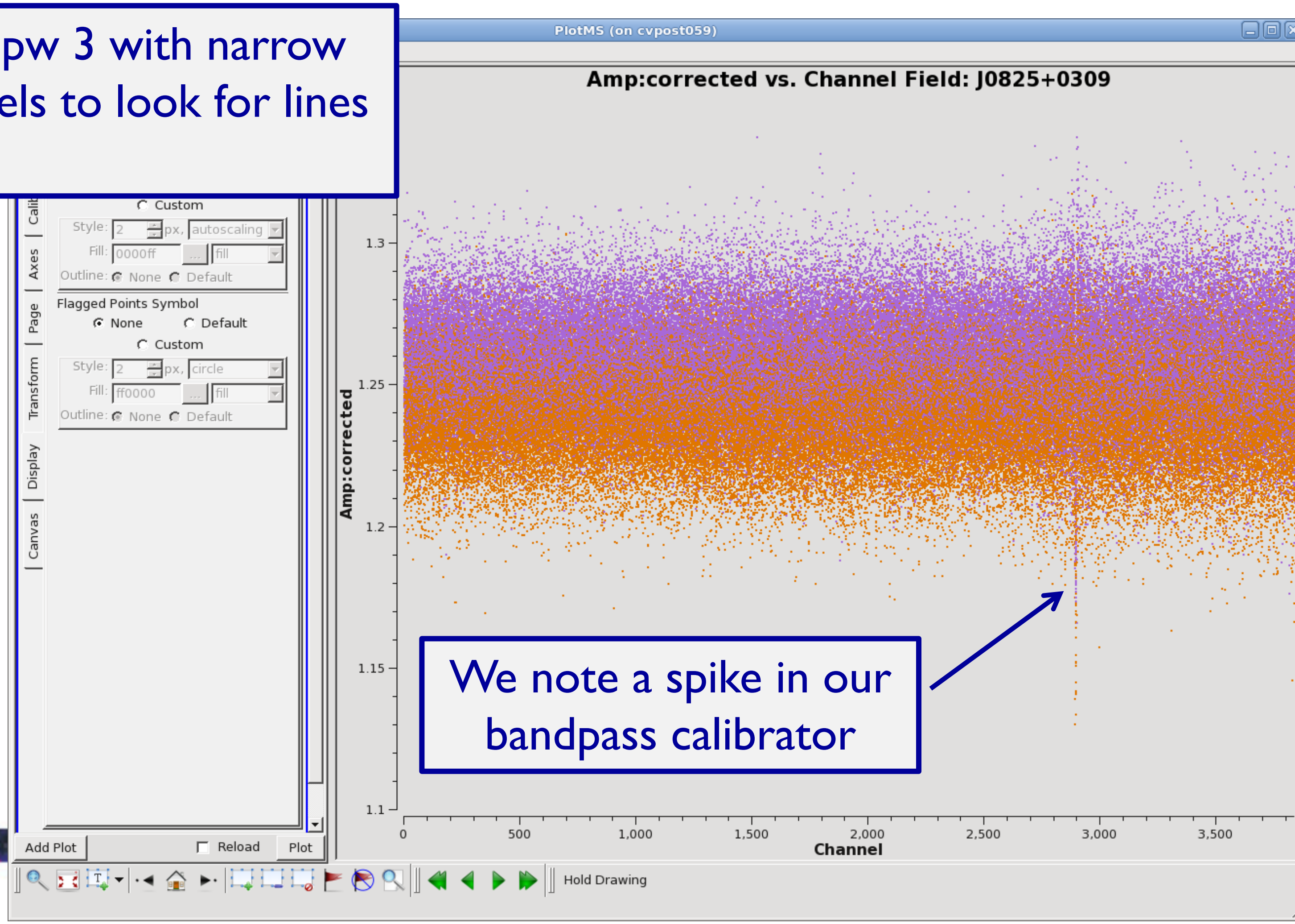
We inspect the three spws (0,1, & 2) with wide (15625 kHz) channels to look for lines or spikes. We don't note any issues (i.e. spikes).



Inspection: Spectral Windows with Narrow Channels

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="channel",
   yaxis="amp", ydatacolumn="corrected", field="0,1,2,3",
   avgdata=True, avgchannel=" ", avgtime="1e6", coloraxis="corr",
   iteraxis="field", spw="3", avgantenna=True)
```

We inspect the spw 3 with narrow (488 kHz) channels to look for lines or spikes.

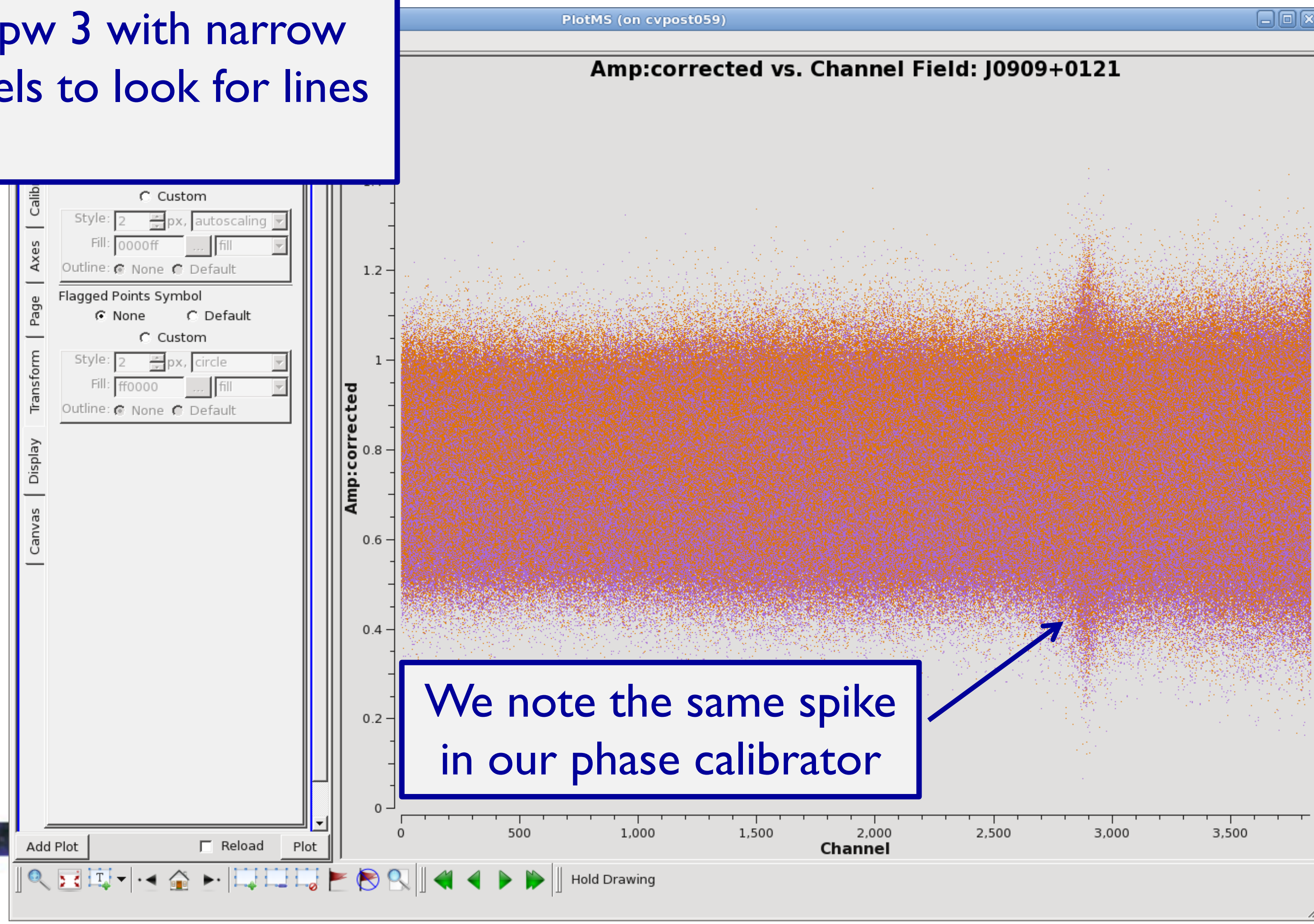


We note a spike in our bandpass calibrator

Inspection: Spectral Windows with Narrow Channels

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="channel",
   yaxis="amp", ydatacolumn="corrected", field="0,1,2,3",
   avgdata=True, avgchannel=" ", avgtime="1e6", coloraxis="corr",
   iteraxis="field", spw="3", avgantenna=True)
```

We inspect the spw 3 with narrow (488 kHz) channels to look for lines or spikes.

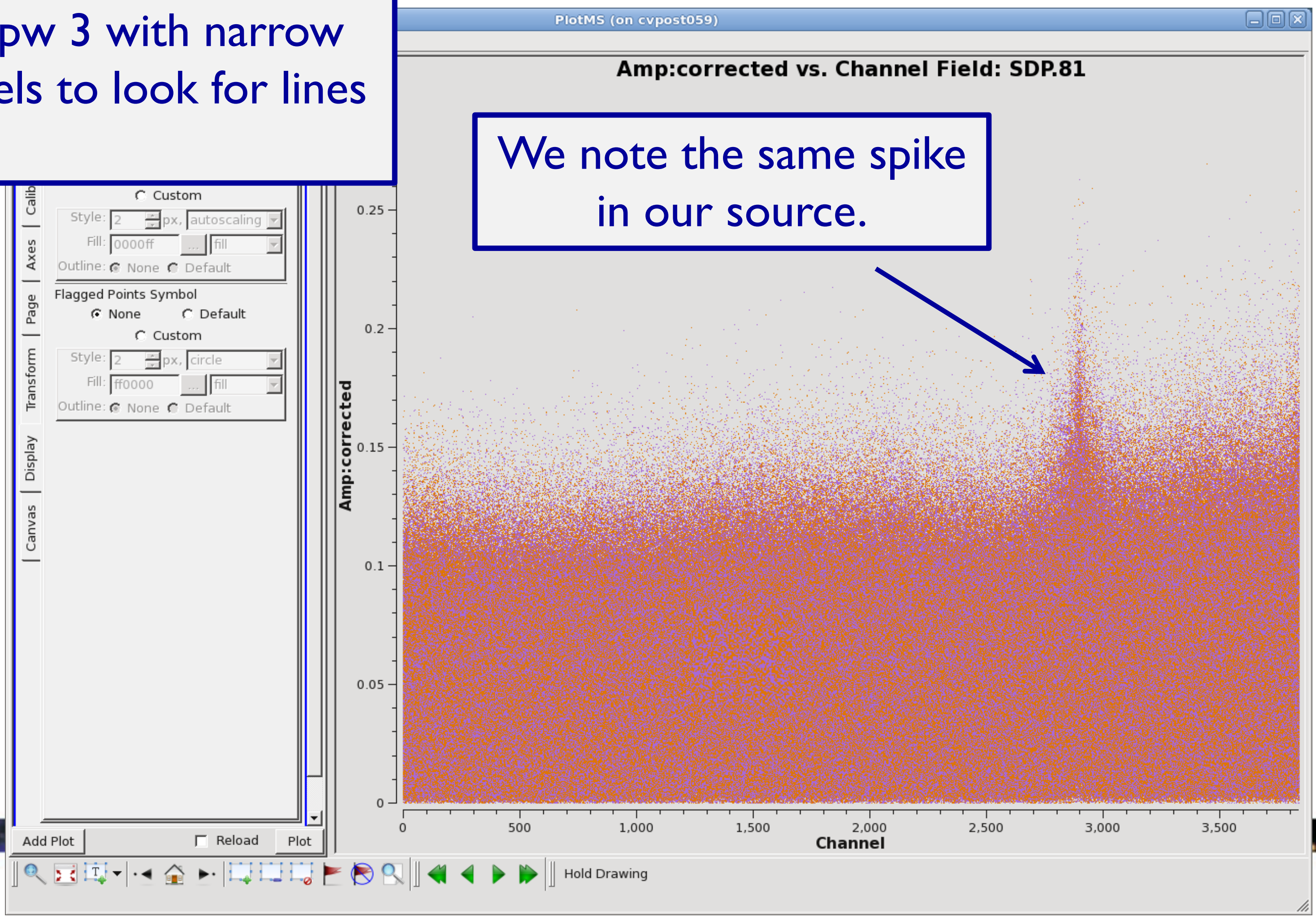


We note the same spike in our phase calibrator

Inspection: Spectral Windows with Narrow Channels

```
plotms(vis="SDP81_B4_uncalibrated.ms.split", xaxis="channel",
   yaxis="amp", ydatacolumn="corrected", field="0,1,2,3",
   avgdata=True, avgchannel=" ", avgtime="1e6", coloraxis="corr",
   iteraxis="field", spw="3", avgantenna=True)
```

We inspect the spw 3 with narrow (488 kHz) channels to look for lines or spikes.

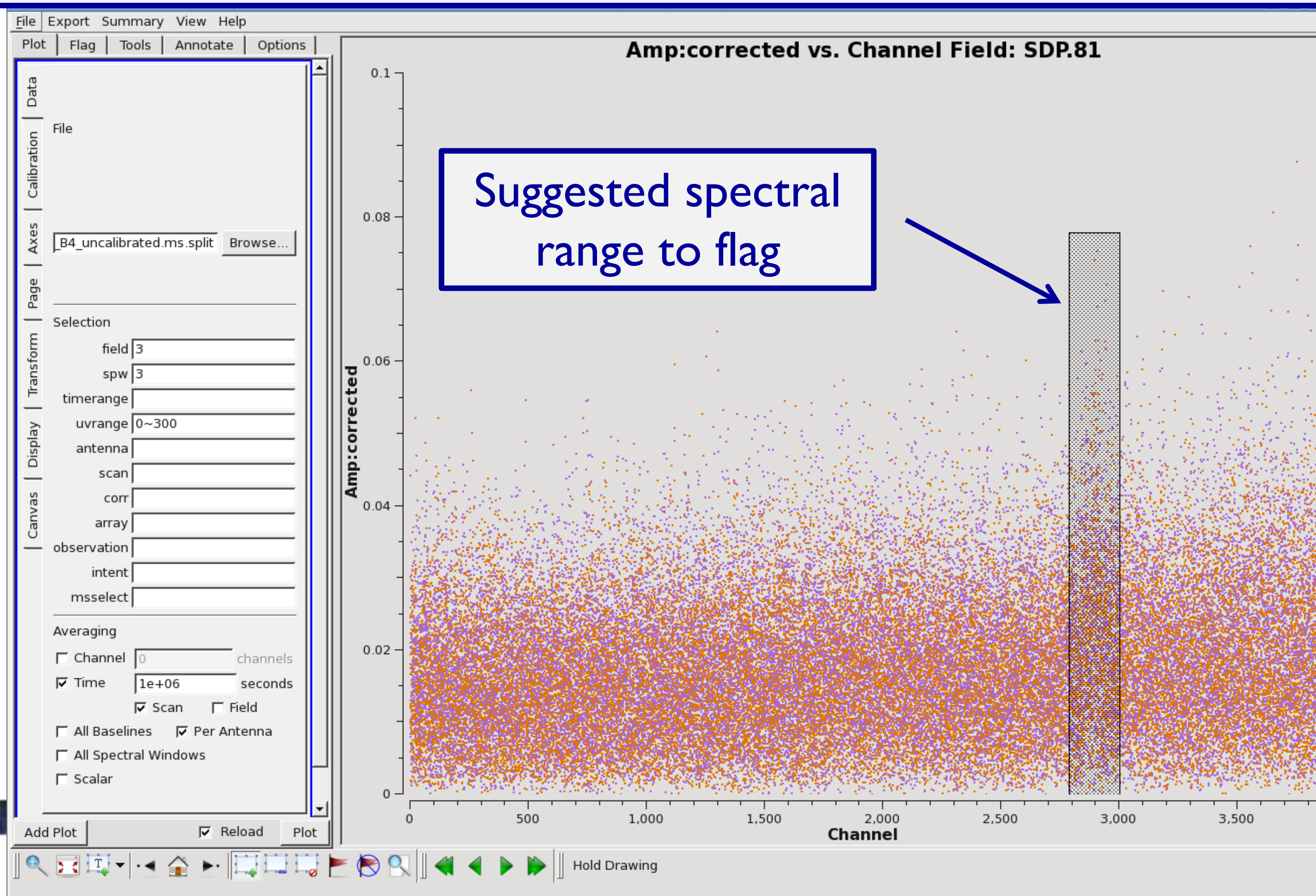


We note the same spike in our source.

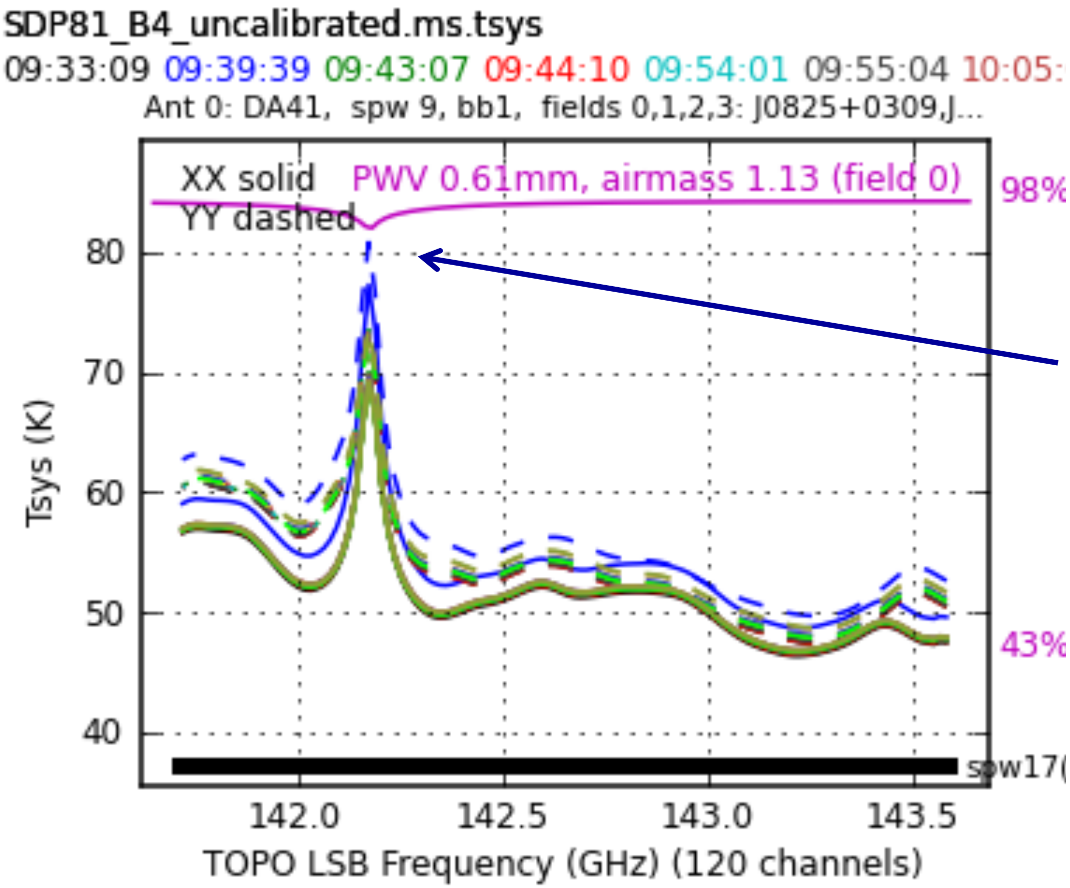
Flag the spike we see in all of our targets.

```
plotms(vis=”SDP81_B4_uncalibrated.ms.split", xaxis=”channel",
   yaxis=“amp", ydatacolumn="corrected", field="0,1,2,3”,
   avgdata=True, avgchannel=“ ", avgtime="1e6", coloraxis=”corr”,
   iteraxis=“field”, spw=“3”, avgbaseline=True)
```

Averaging over short baselines makes this feature easier to discern (and thus flag).



Note: We already know where this feature comes from!



Looking back at our Tsys plots (made when applying initial corrections to the data), we see a dip in the atmospheric transmission which highlights an absorption feature in the atmosphere at that frequency. This coincides with a peak in Tsys and with the spike in our data.

Define your Data Flags

Now take some time to inspect the data yourself and look for any additional issues that may need flagging. We have noted some recommendations at the end of the calibration.py script.

Once you have identified the data you want to flag, enter those flagging commands at the earlier (marked) point in the calibration.py script before *Bandpass Calibration* but after *Getting Oriented and Initial Flagging.*

An example: (flagging the atmospheric line in spw 3)

```
flagdata(vis="SDP81_B4_uncalibrated.ms.split",
         mode="manual",
         spw="3:1400~1500",
         flagbackup=False)
```

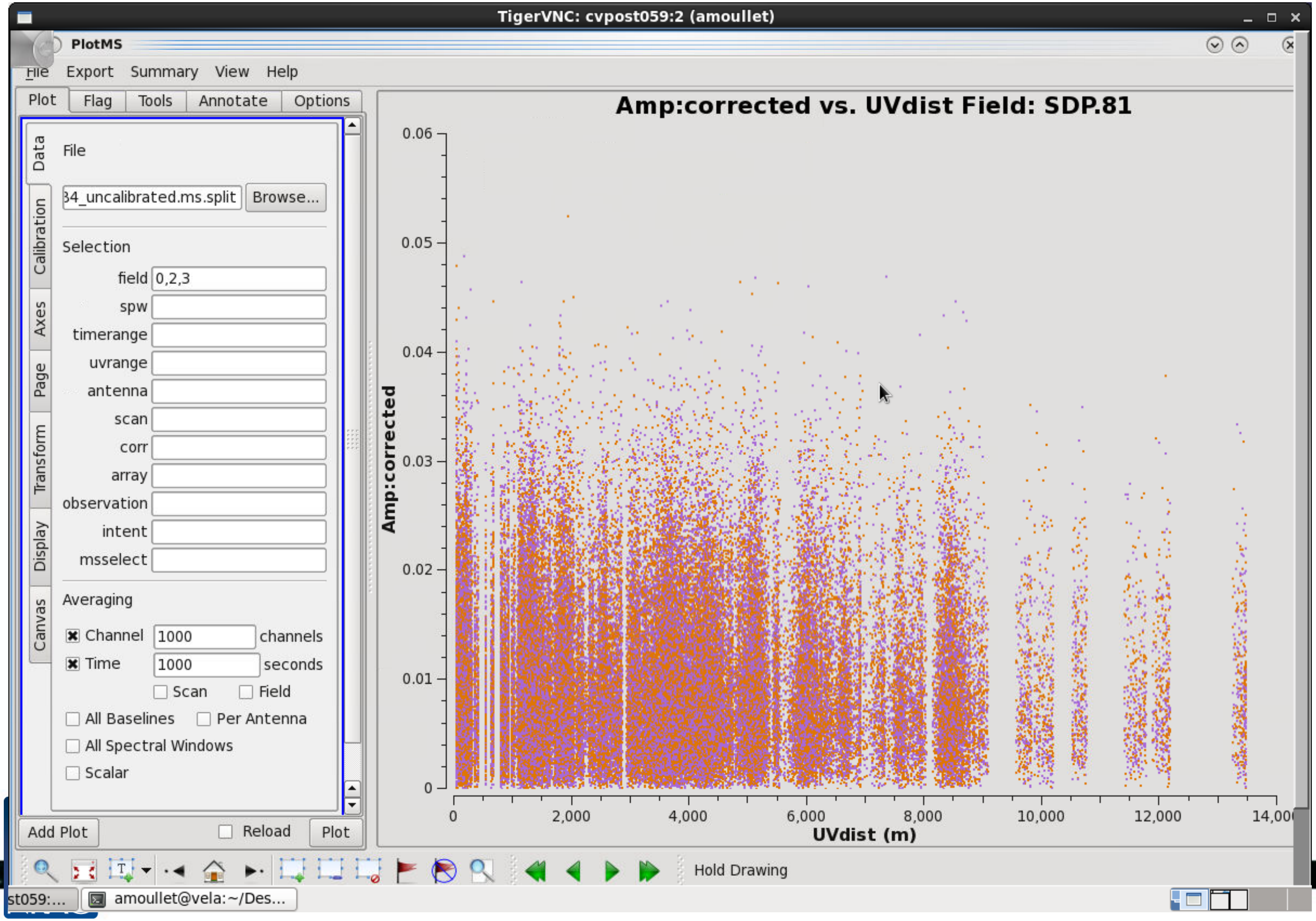
Redo calibration after flagging

- If you have flagged calibrator data, you must re-run through the entirety of the calibration after flagging. By flagging problematic data, we improve all of our solutions for our bandpass, gain, etc.
- Place any data flag commands in the flagging section in the calibration.py script before the bandpass calibration is run
- Then execute the script in its entirety (aka "end-to-end").
  - either by entering each command at the casa prompt as we have been doing or by executing the script as a whole via:

```
execfile("calibration.py")
```

- **In the interest of time, we won't re-run the script today, and instead will move on to the imaging section**

A look at the final calibrated data



Final Steps

Several iterations of inspection, defining flags, and re-calibration can be performed. Typically after one is satisfied by the calibrated data quality, it is recommended to split out the corrected column of the data to a new measurement set. **We will not do it in this tutorial in the interest of saving space.** For future reference, this is how splitting out the correct column can be done:

```
split(vis="SDP81_B4_uncalibrated.ms.split",
      outputvis="SDP81_B4_calibrated.ms.split",
      datacolumn="corrected",
      keepflags=True)
```

To free space on your machine, please remove SDP81_B4_uncalibrated.ms.split from your Calibration directory when you are done with the calibration.

```
os.system("rm -fr SDP81_B4_uncalibrated.ms.split")
```

Outline

- Short introduction to CASA and the Python interface
  - How to use tasks
  - What is a measurement set?
- The Flow of Calibration
- Overview of your Directory
  - Data preparation and set up
  - Getting oriented with your data
- Data Calibration
- Data Inspection and Flagging
- **Basic Imaging**

Basic Imaging

Introduction to deconvolution in CASA (clean)

Introduction to various imaging methods available in CASA

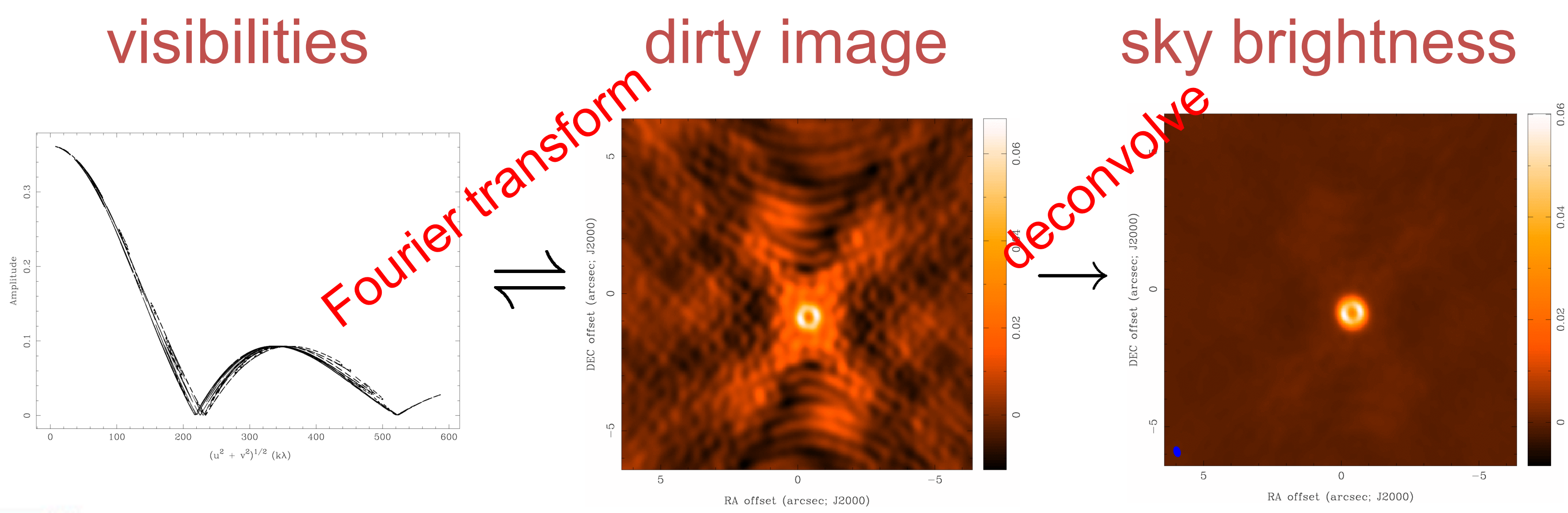
ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Robert C. Byrd Green Bank Telescope
Very Long Baseline Array

How to analyze (imperfect) interferometer data?

- image plane analysis
  - dirty image $T^D(x,y)$ = Fourier transform { $V(u,v)$ }
  - deconvolve $b(x,y)$ from $T^D(x,y)$ to determine (model of) $T(x,y)$

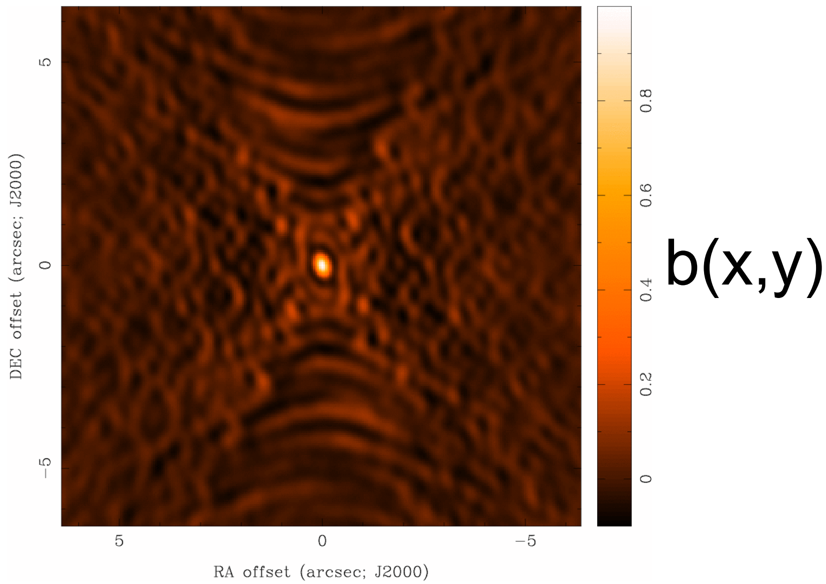


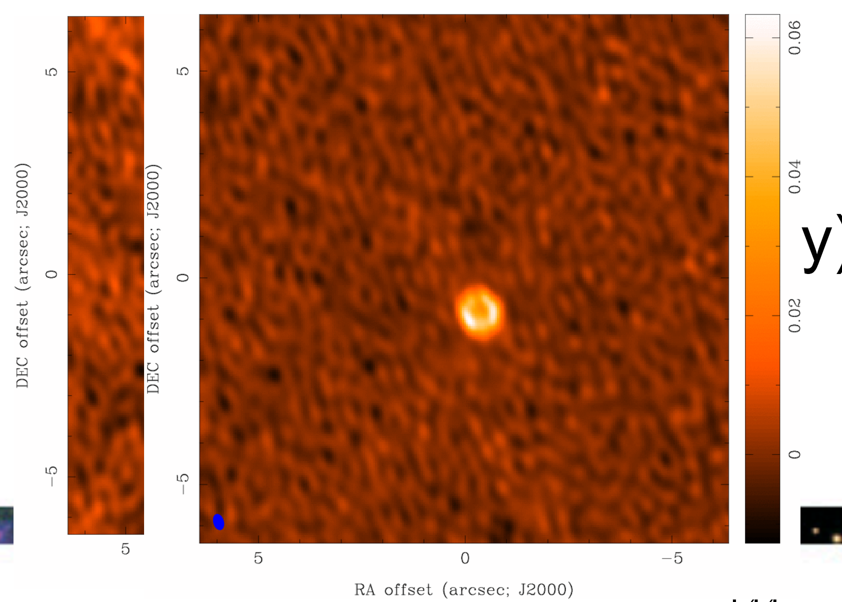
Basic CLEAN Algorithm

① Initialize a *residual* map to the dirty map

1. Start loop
2. Identify strongest feature in *residual* map as a point source
3. Add this point source to the clean component list
4. Convolve the point source with b(x,y) and subtract a fraction g (the loop gain) of that from *residual* map
5. If stopping criteria not reached, do next iteration

② Convolve *Clean component* (cc) list by an estimate of the main lobe of the dirty beam (the “Clean beam”) and add *residual* map to make the final “restored” image





Basic CLEAN Algorithm (cont)

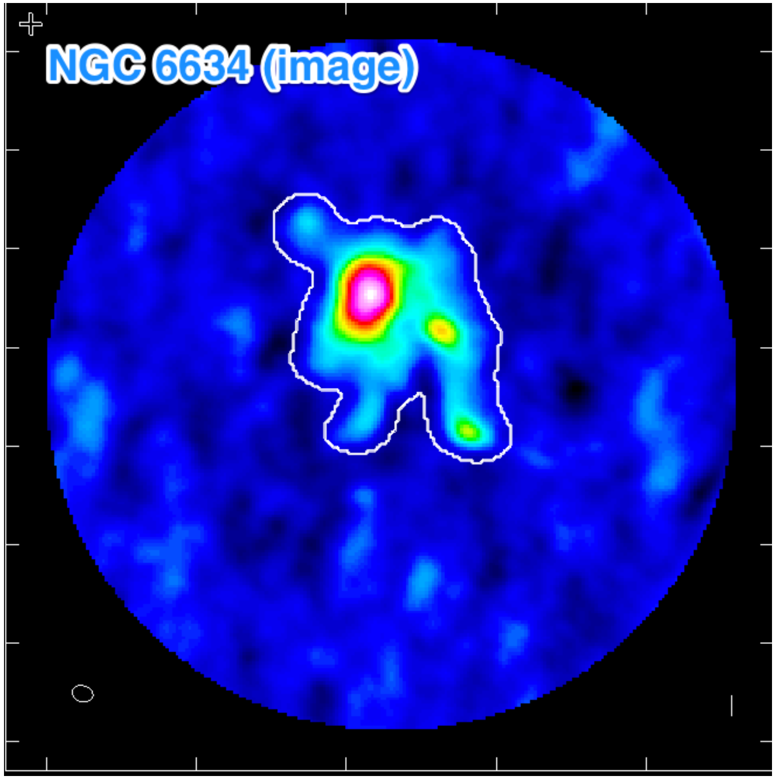
- stopping criteria
  - *residual* map max < multiple of rms (when noise limited)
  - *residual* map max < fraction of dirty map max (dynamic range limited)
  - max number of clean components reached (no justification)
- loop gain
  - good results for g ~ 0.1 to 0.3
  - lower values can work better for smoother emission, g ~ 0.05
- easy to include *a priori* information about where to search for clean components ("clean boxes")

A few notes on clean boxes

- Because we do not fully sample the uv-plane in our imaging, there is generally no unique solution to the deconvolution process
- We use clean 'boxes', or masks, to identify regions of the image or cube with real emission
- Clean boxes are a way to create the best possible model for your source – particularly sources with complex emission
- As a first step, include bright features in your mask, drawing a close contour around the emission
- For cubes, you can mask channel-by-channel, or all channels
- As tclean progresses, strong residuals that do not appear to be due to sidelobes (i.e., do not disappear in subsequent cycles) can be added iteratively
- Be careful when masking – adding a mask around noise or beam sidelobes can create features in your final image that are not real

Automasking (auto-multithresh) in tclean

- Algorithm developed by A. Kepley, T. Tsutsumi (+Yoon, Indebetouw, Brogan)
  - parameterized in terms of fundamental image parameters (S/N, fraction of beam, sidelobe level) ⇒ instrument independent
  - Masks are re-calculated every major cycle within tclean ⇒ follows evolution of image
- **Available in tclean since CASA 5.1**
  - usemask='auto-multithresh'
- Deployed in ALMA Cycle 5 pipeline
- CASA guide: https://casaguides.nrao.edu/index.php?title=Automasking_Guide



NRAO

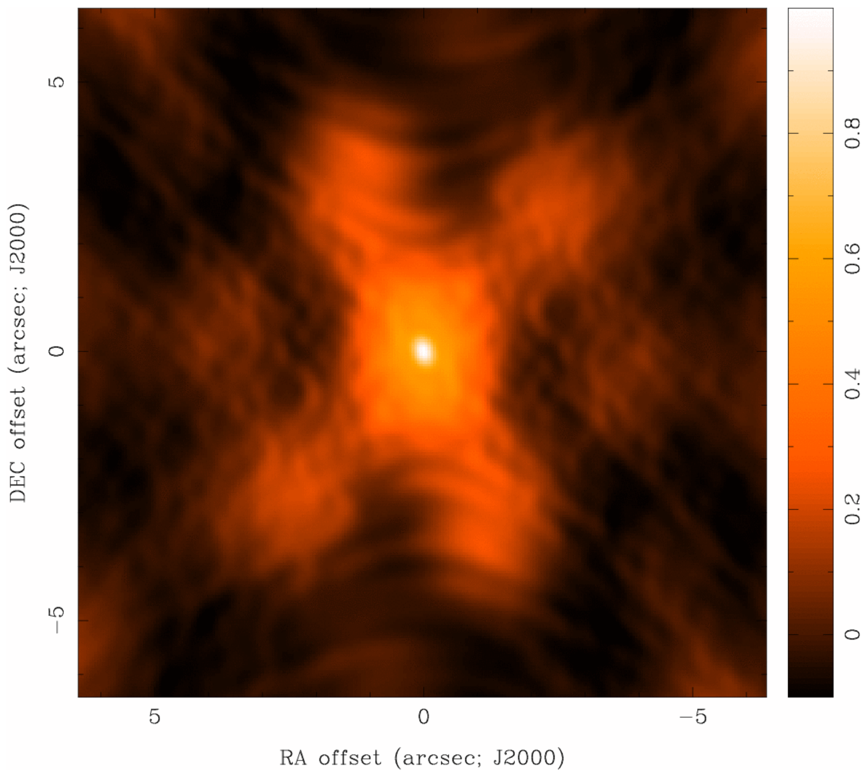
Dirty Beam Shape and Weighting

Each visibility point is given a weight in the imaging step

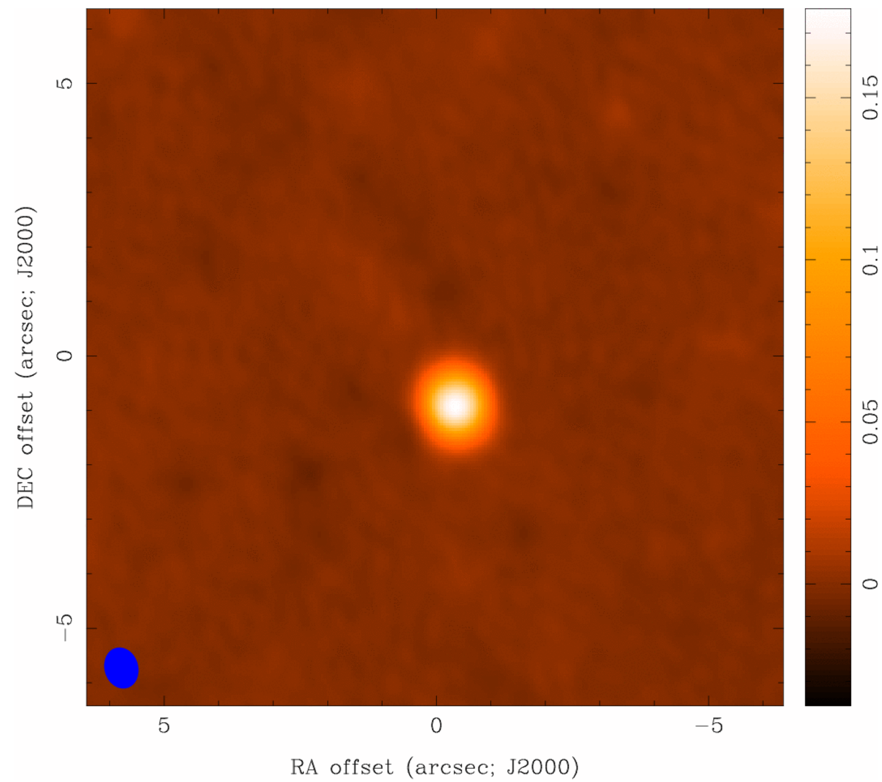
- Natural
  - Weights inversely proportional to noise variance
  - Best point-source sensitivity; poor beam characteristics
- Uniform
  - Weights inversely proportional to noise variance and sampling density (longer baseline are given higher weight than in natural)
  - Best resolution; poorer noise characteristics
- Briggs (Robust)
  - A graduated scheme using the parameter *robust*
  - In CASA, set *robust* from -2 (~ uniform) to +2 (~ natural)
  - *robust* = 0 often a good choice

Imaging Results

Natural Weight Beam

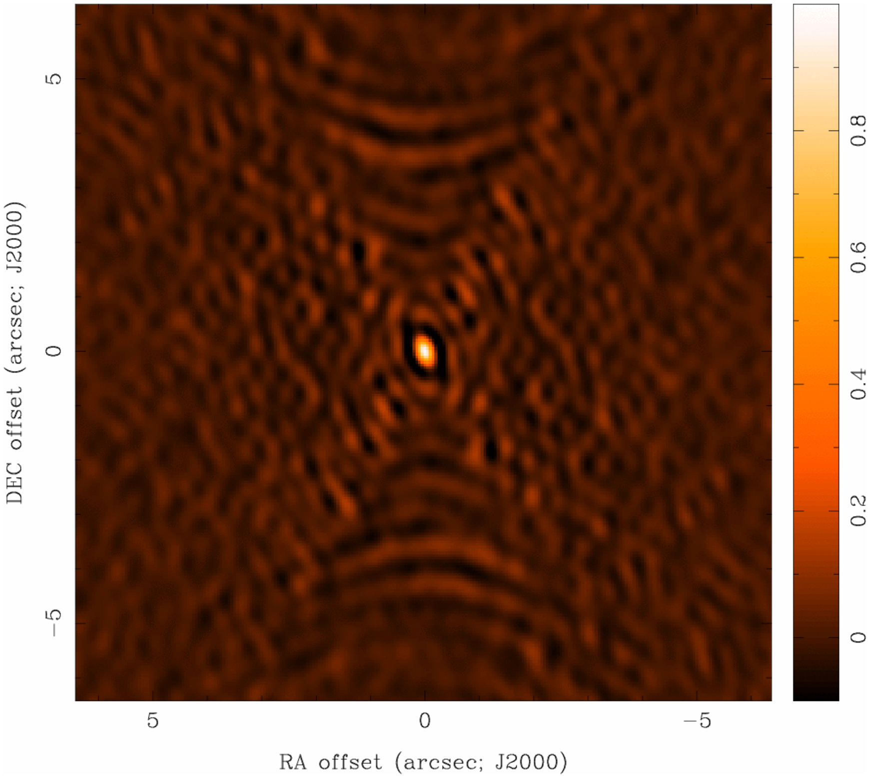


CLEAN image

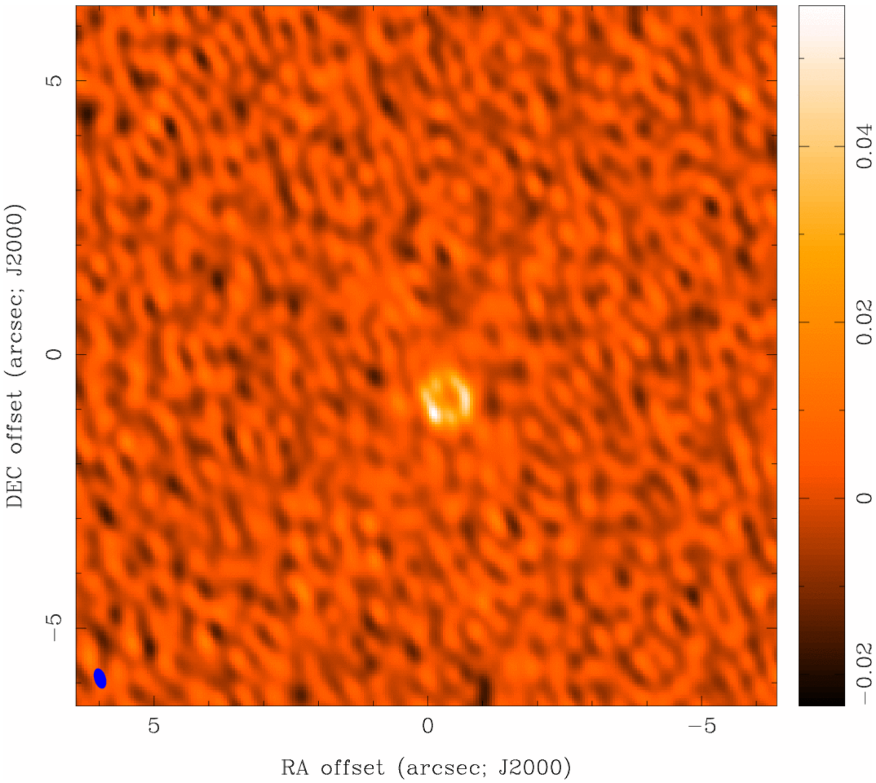


Imaging Results

Uniform Weight Beam

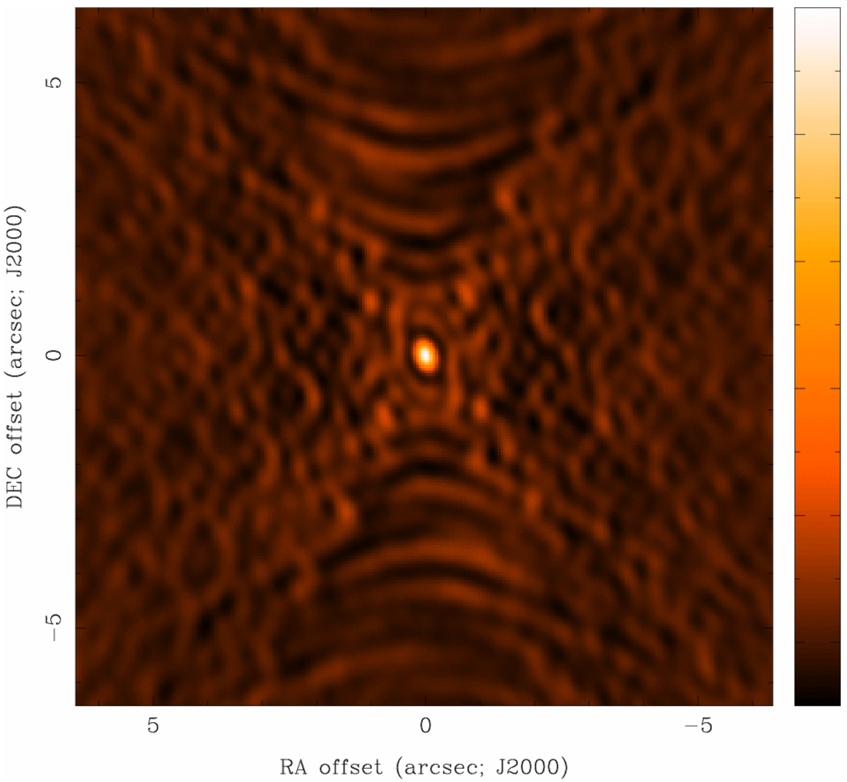


CLEAN image

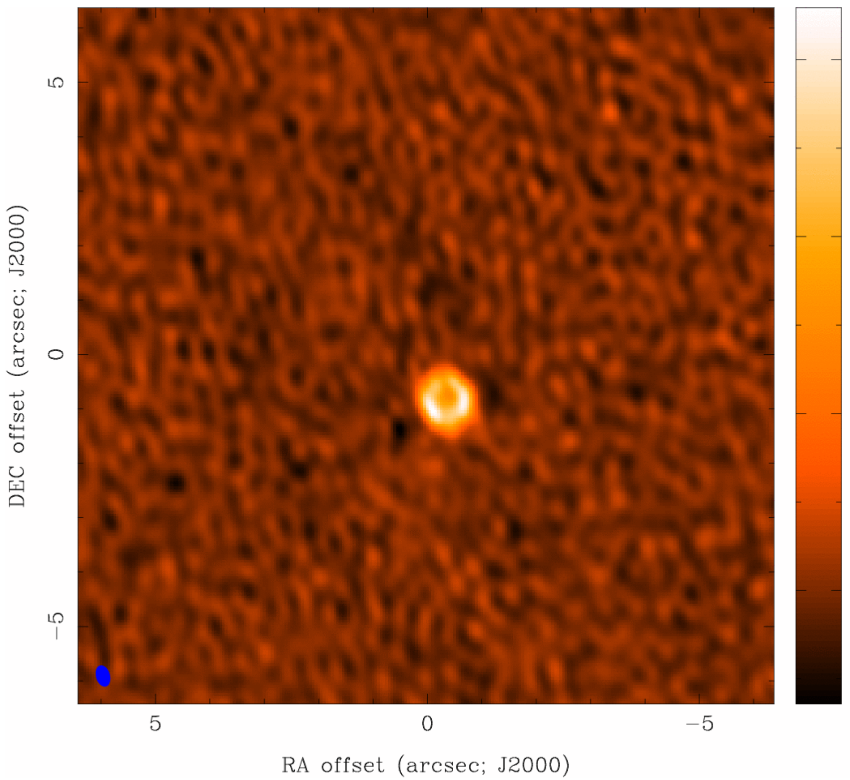


Imaging Results

Robust=0 Beam



CLEAN image



tclean in CASA:

IPython: rfriesen/SIS18

File Edit View Search Terminal Help

```
CASA <1>: inp tclean
--------> inp(tclean)
#  tclean :: Radio Interferometric Image Reconstruction
vis                 =         ''         #  Name of input visibility file(s)
selectdata          =       True         #  Enable data selection parameters
     field          =         ''         #  field(s) to select
     spw            =         ''         #  spw(s)/channels to select
     timerange      =         ''         #  Range of time to select from data
     uvrange        =         ''         #  Select data within uvrange
     antenna        =         ''         #  Select data based on antenna/baseline
     scan           =         ''         #  Scan number range
     observation    =         ''         #  Observation ID range
     intent         =         ''         #  Scan Intent(s)

datacolumn          = 'corrected'        #  Data column to image(data,corrected)
imagename           =         ''         #  Pre-name of output images
imsize              =      [100]         #  Number of pixels
cell                = ['1arcsec']        #  Cell size
phasecenter         =         ''         #  Phase center of the image
stokes              =        'I'         #  Stokes Planes to make
projection          =      'SIN'         #  Coordinate projection (SIN, HPX)
startmodel          =         ''         #  Name of starting model image
specmode            =      'mfs'         #  Spectral definition mode (mfs,cube,cubedata)
     reffreq        =         ''         #  Reference frequency

gridder             = 'standard'         #  Gridding options (standard, wproject, widefield,
                                         #   mosaic, awproject)
     vptable        =         ''         #  Name of Voltage Pattern table
     pblimit        =        0.2         #  >PB gain level at which to cut off
                                         #   normalizations

deconvolver         =   'hogbom'         #  Minor cycle algorithm
                                         #   (hogbom,clark,multiscale,mtmfs,mem,clarkstokes)
```

Basic Image Parameters: Pixel Size and Image Size

- pixel size
  - should satisfy $\Delta x < 1/(2\ u_{max})$ $\Delta y < 1/(2\ v_{max})$
  - in practice, 3 to 5 pixels across the main lobe of the dirty beam

- image size
  - Consider FWHM of primary beam (e.g. ~ 20" at Band 7)
  - Be aware that sensitivity is not uniform across the primary beam
  - Use mosaicing to image larger targets
  - Not restricted to powers of 2

* if there are bright sources in the sidelobes, they will be aliased into the image (need to make a larger image)

Largest Angular Scale

Band	Frequency (GHz)	Primary beam (")	Range of Scales (") C32-1	C32-9
3	84-116	72 - 52	4.2 - 24.6	0.7 - 15.1
6	211-275	29 - 22	1.8 - 10.7	0.3 - 6.6
7	275-373	22 - 16	1.2 - 7.1	0.2 - 4.4
9	602-720	10 – 8.5	0.6 - 3.6	0.1 - 2.2

- **Range** from synthesized beam to maximum angular scale (MAS)
- **Smooth** structures larger than LAS begin to be resolved out.
- All flux on scales larger than λ/B_{min} (~2 x MAS) completely resolved out.

NRAO

Basic Imaging

Since 12 executions of the SDP.81 observations were made, ordinarily the next steps would be to repeat the calibration steps we just performed for one execution for the remaining eleven. In the interest of time, we have already done this and combined the 12 executions for you. In your Imaging directory you should have:

SDP.81_Band4.ms

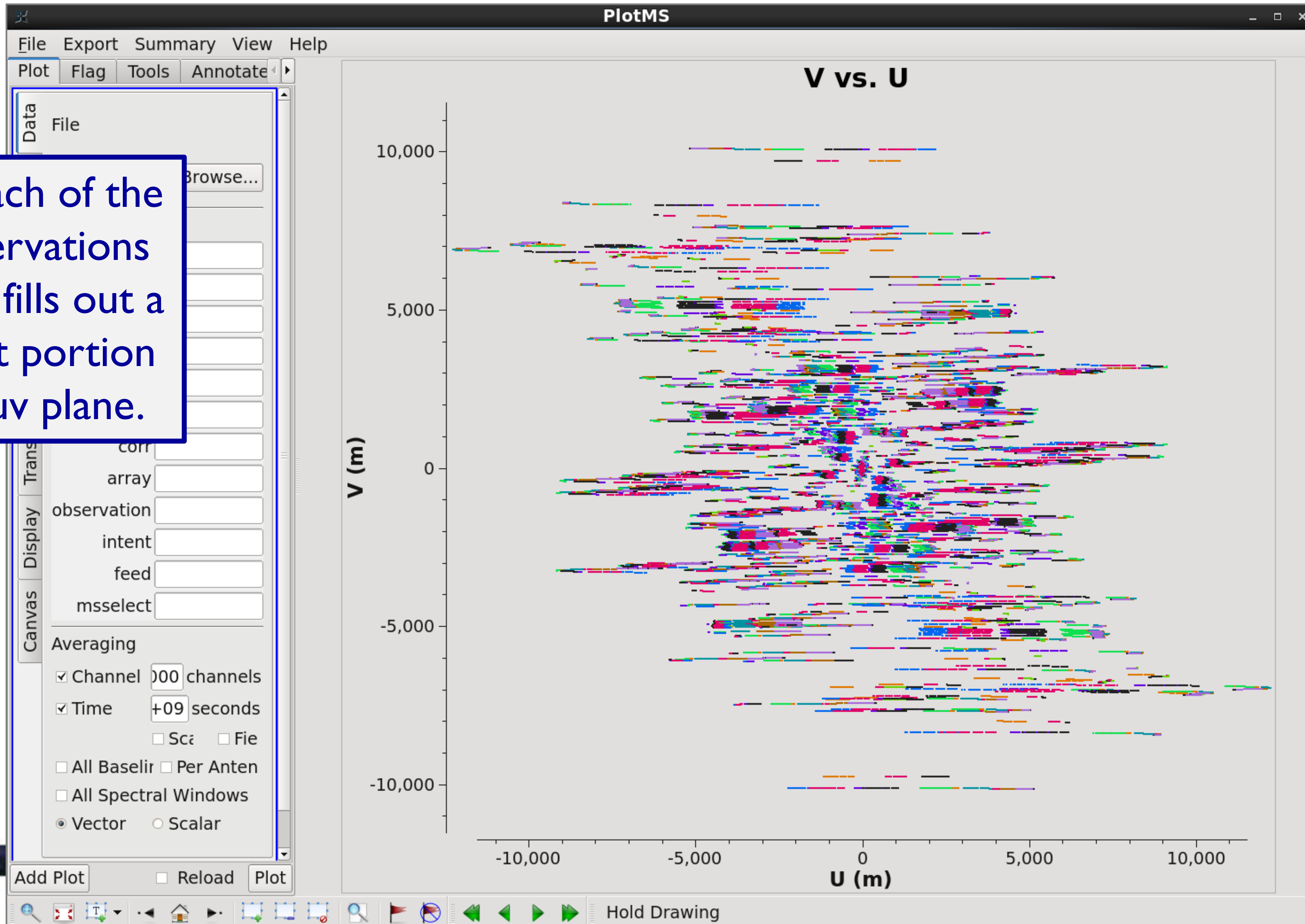
We will now work through the steps noted in the imaging script provided (imaging.py).

Orient yourself with the calibrated measurement set:

```
listobs("SDP.81_Band4.ms")
```

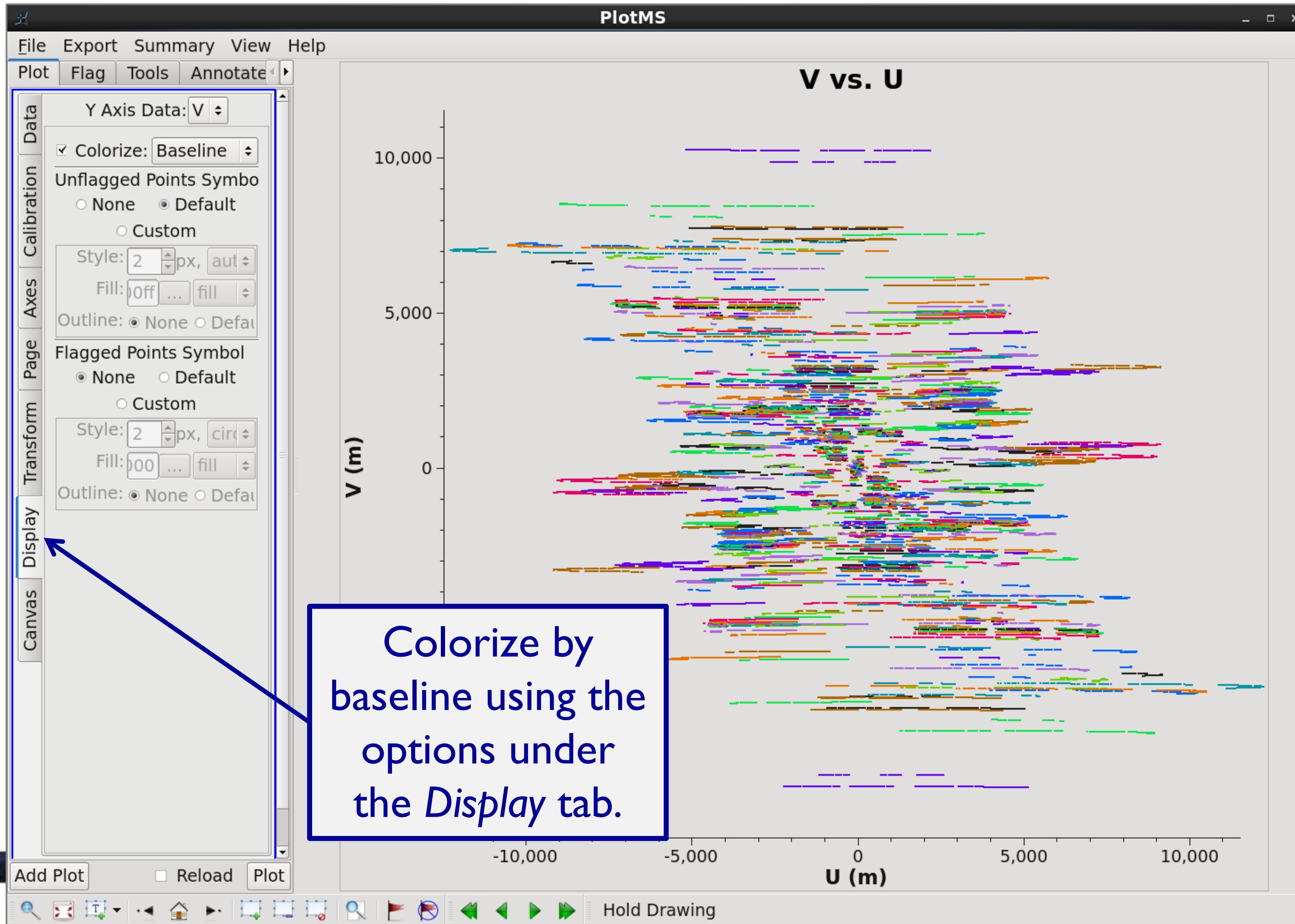
Check your Fourier Plane Coverage

```
plotms(vis='SDP.81_Band4.ms', xaxis='u', yaxis='v',
        avgchannel='10000', avgspw=False, avgtime='1e9', avgscan=False,
        coloraxis="observation")
```



Note: Each of the 12 observations (colors) fills out a different portion of the uv plane.

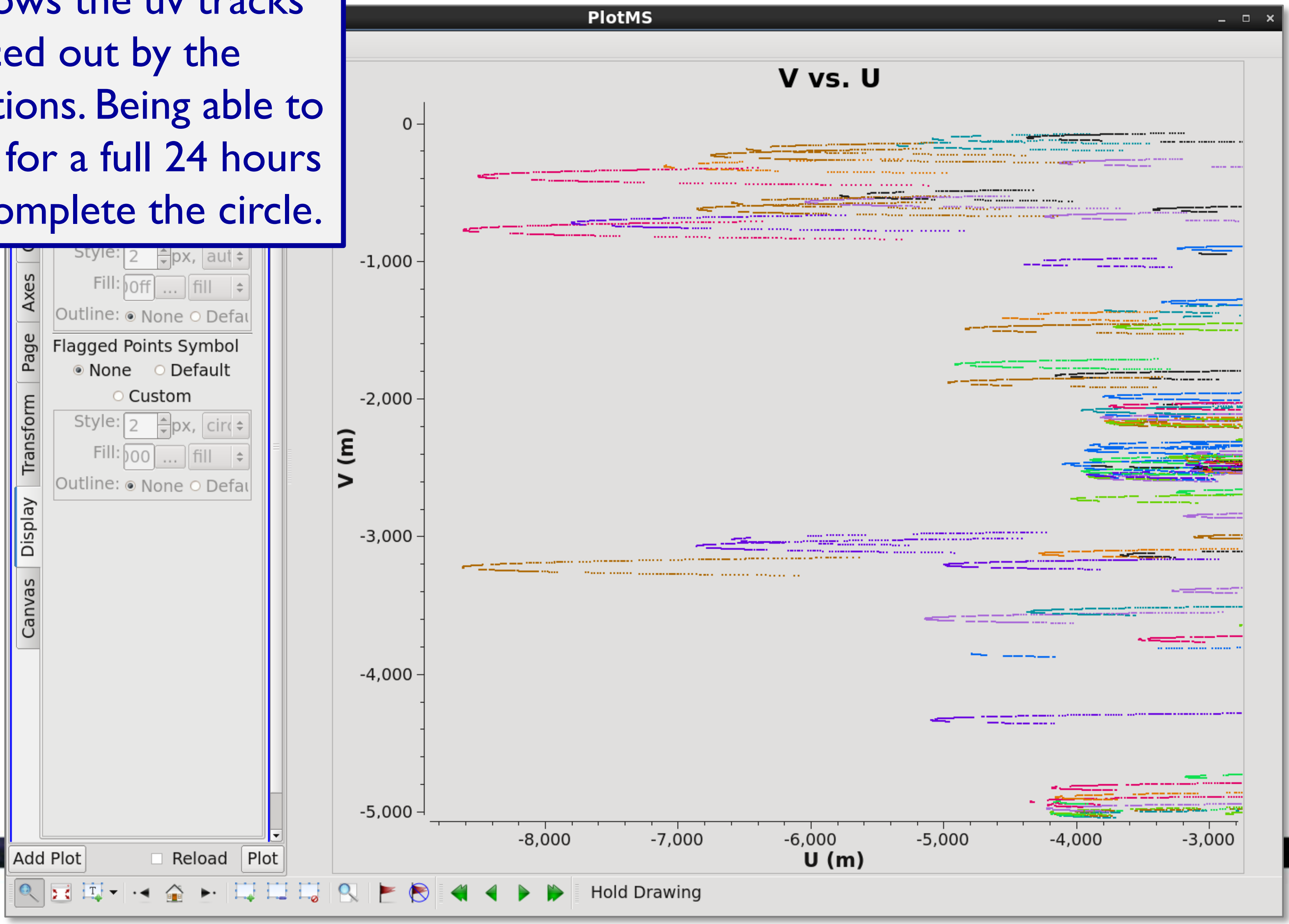
Check your Fourier Plane Coverage



Colorize by baseline using the options under the *Display* tab.

Check your Fourier Plane Coverage

This zoom on the previous plot shows the uv tracks traced out by the observations. Being able to observe for a full 24 hours would complete the circle.



Imaging the Bandpass Calibrator

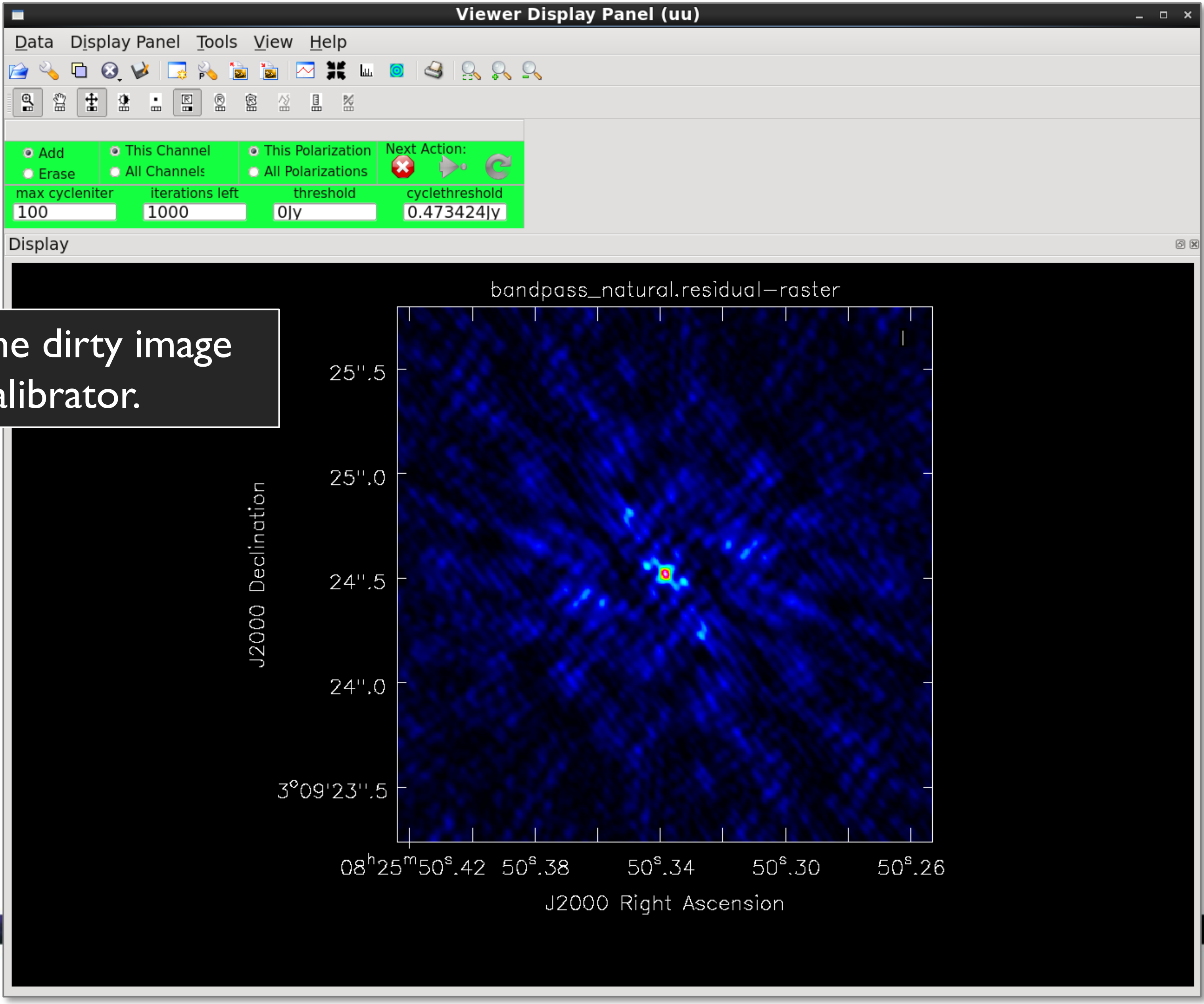
Image the Bandpass Calibrator: Natural

Just for illustrative purposes, let's start by imaging a bright, point-like source like our bandpass calibrator.

```
os.system("rm -rf bandpass_natural.*")
tclean(vis="bandpass.ms",
       imagename="bandpass_natural",
       field="0", spw="",
       specmode="mfs", deconvolver='hogbom',gridder='standard',
       imsize=[512,512], cell=["0.005arcsec"],
       weighting="natural", threshold="0mJy",
       niter=10000, interactive=True)
```

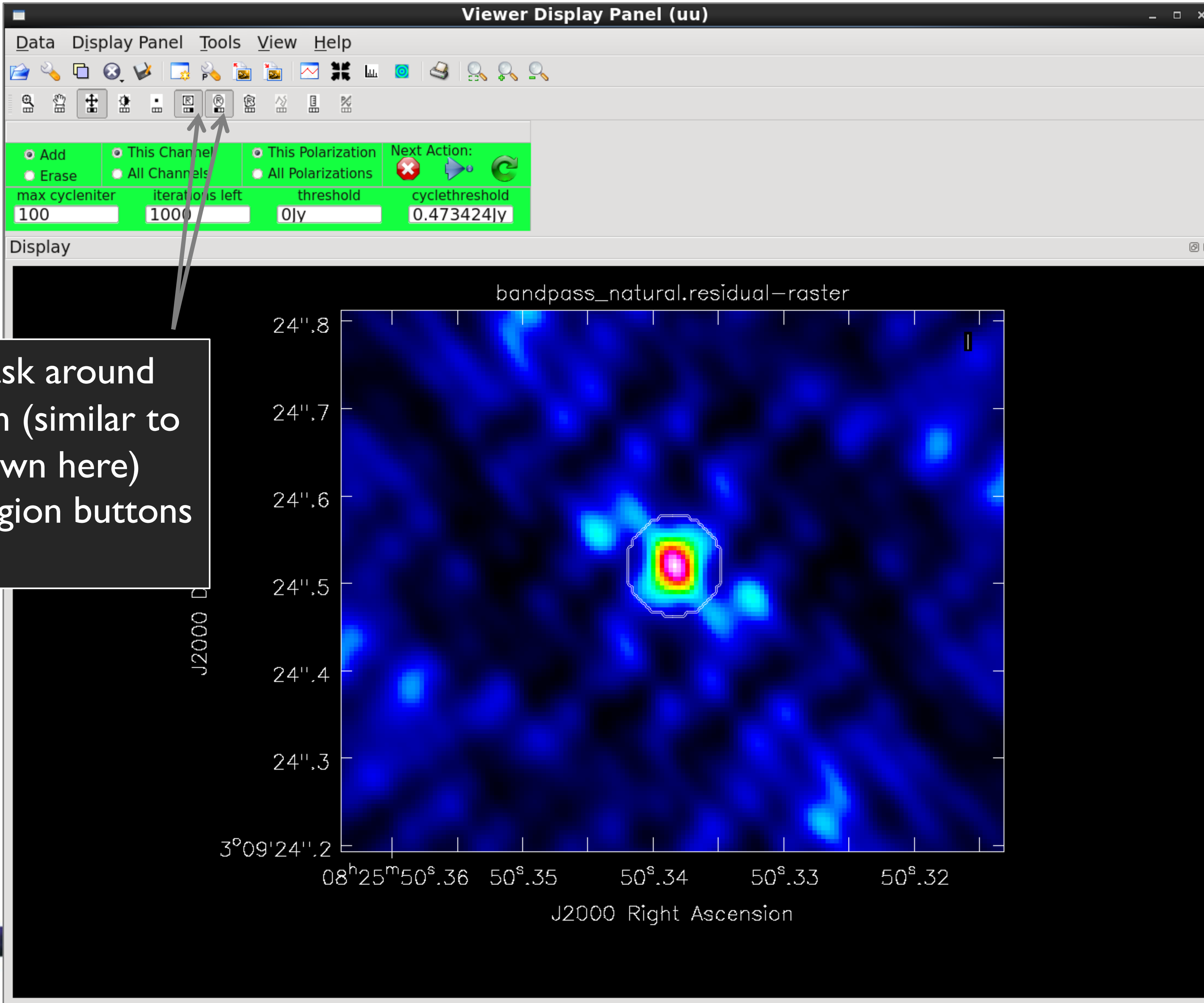
Running tclean will bring up the following interactive window …

Image the Bandpass Calibrator: Natural



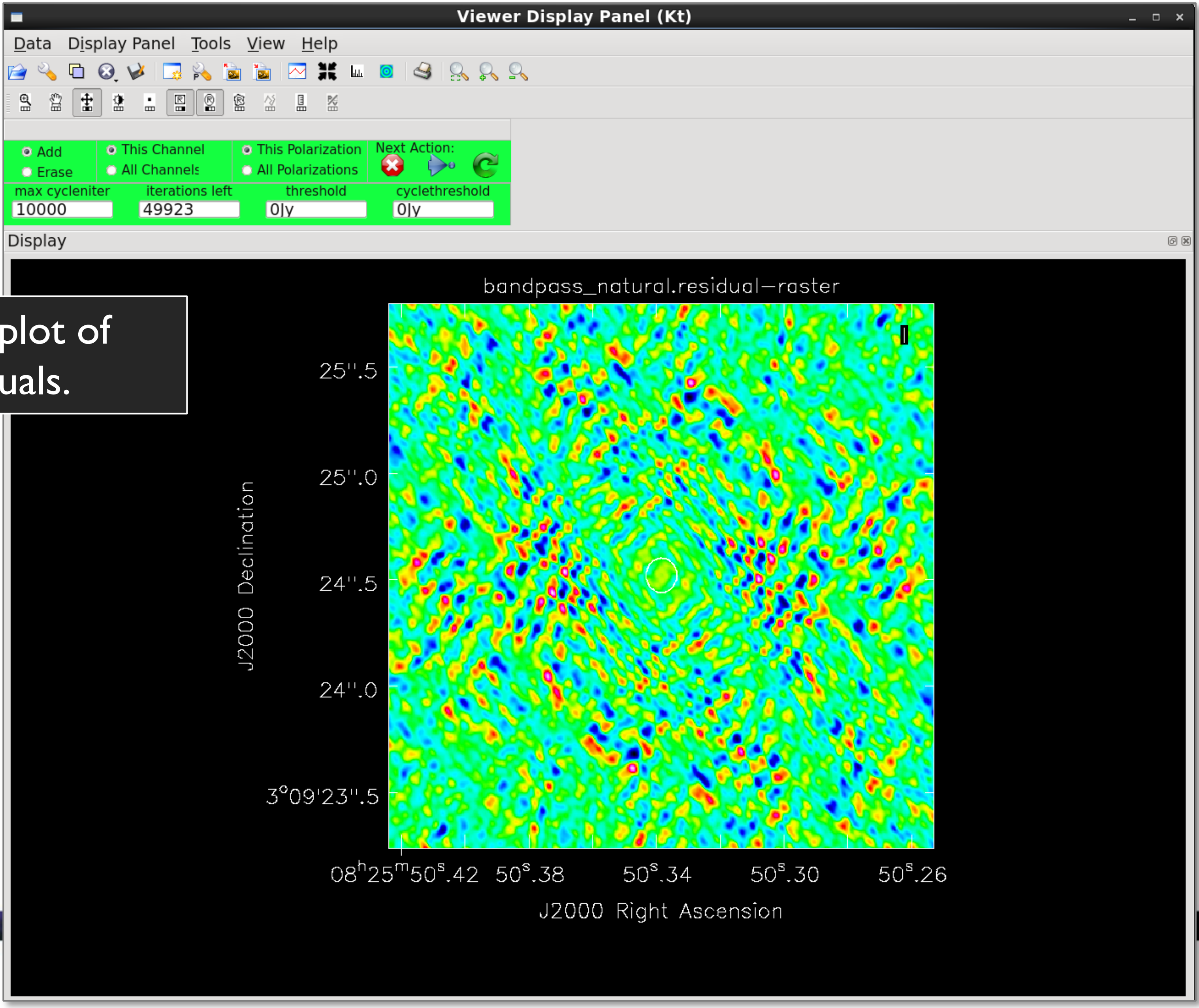
This is the dirty image of our calibrator.

Image the Bandpass Calibrator: Natural



Define a mask around the emission (similar to the one shown here) using the region buttons above.

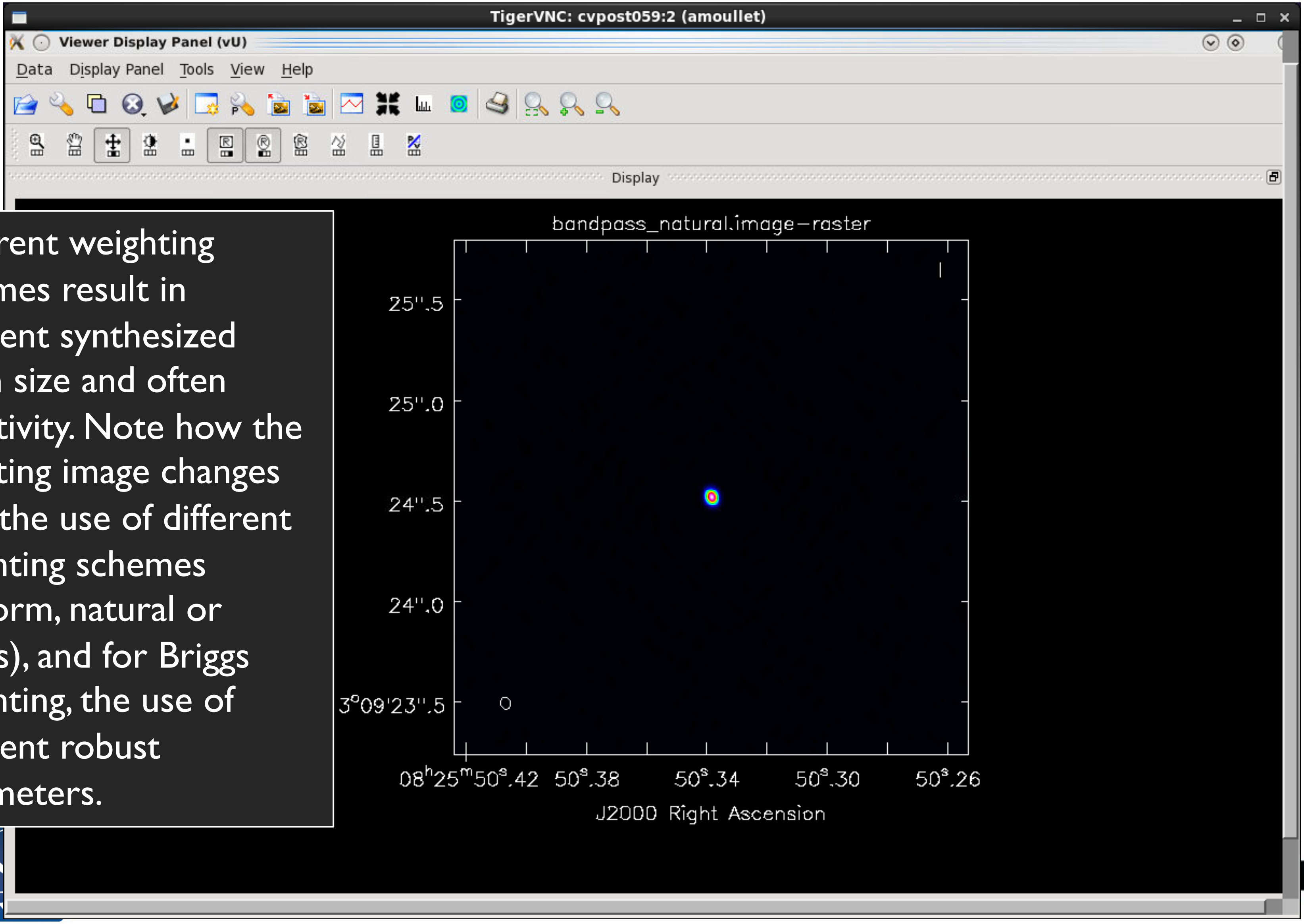
Image the Bandpass Calibrator: Natural



This is a plot of the residuals.

Image the Bandpass Calibrator: Natural

View the resulting clean image: `viewer("bandpass_natural.image")`



Different weighting schemes result in different synthesized beam size and often sensitivity. Note how the resulting image changes with the use of different weighting schemes (uniform, natural or briggs), and for Briggs weighting, the use of different robust parameters.

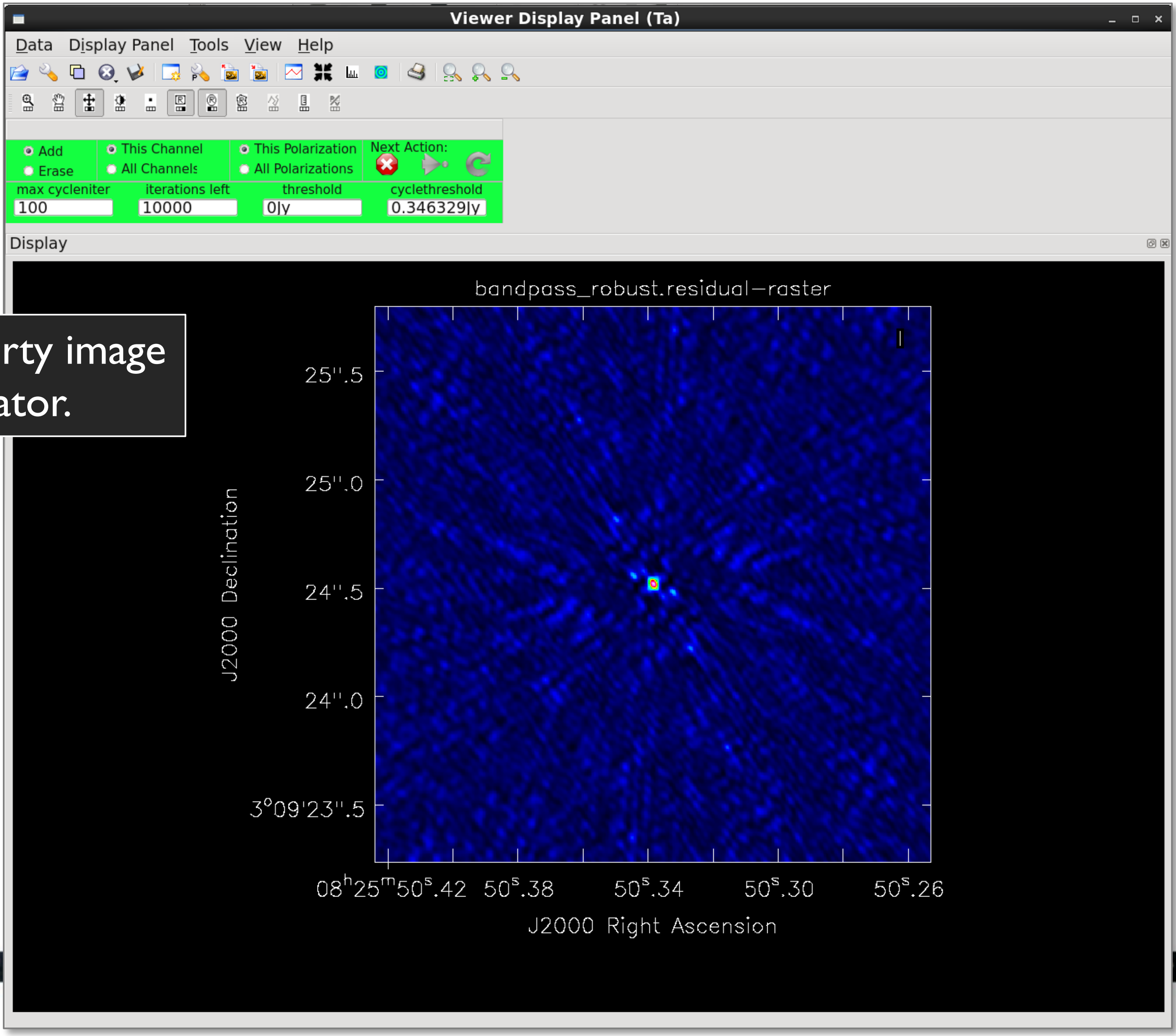
Image the Bandpass Calibrator: Briggs

Now image the bandpass calibrator using a Briggs weighting scheme:

```
os.system("rm -rf bandpass_robust.*")
tclean(vis="bandpass.ms",
       imagename="bandpass_robust",
       field="0", spw="",
       specmode="mfs", deconvolver='hogbom', gridder='standard',
       imsize=[512,512], cell=["0.005arcsec"],
       weighting="briggs", robust=0.0,
       threshold="0mJy",
       niter=10000, interactive=True)
```

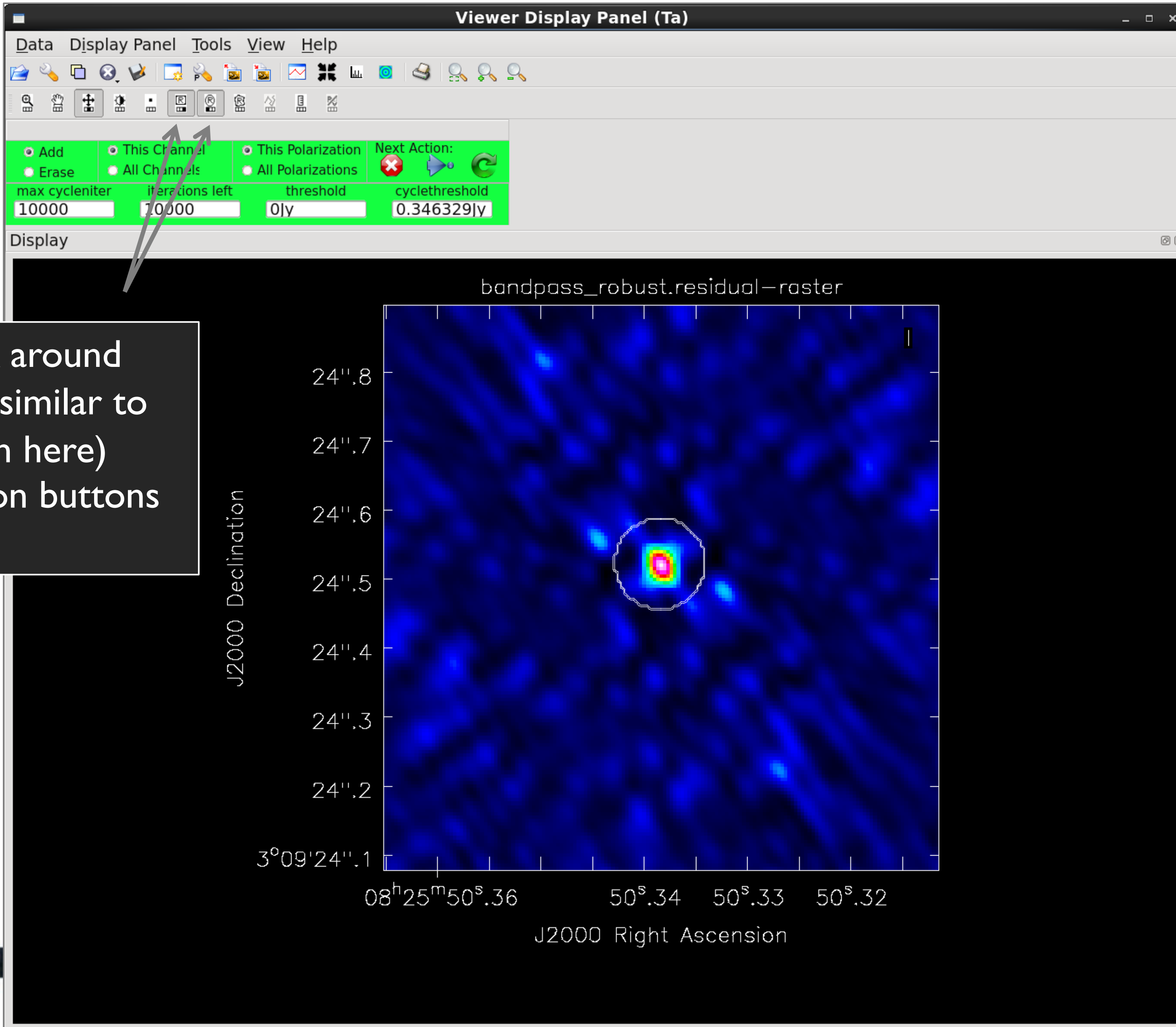
Running tclean will bring up the following interactive window …

Image the Bandpass Calibrator: Briggs



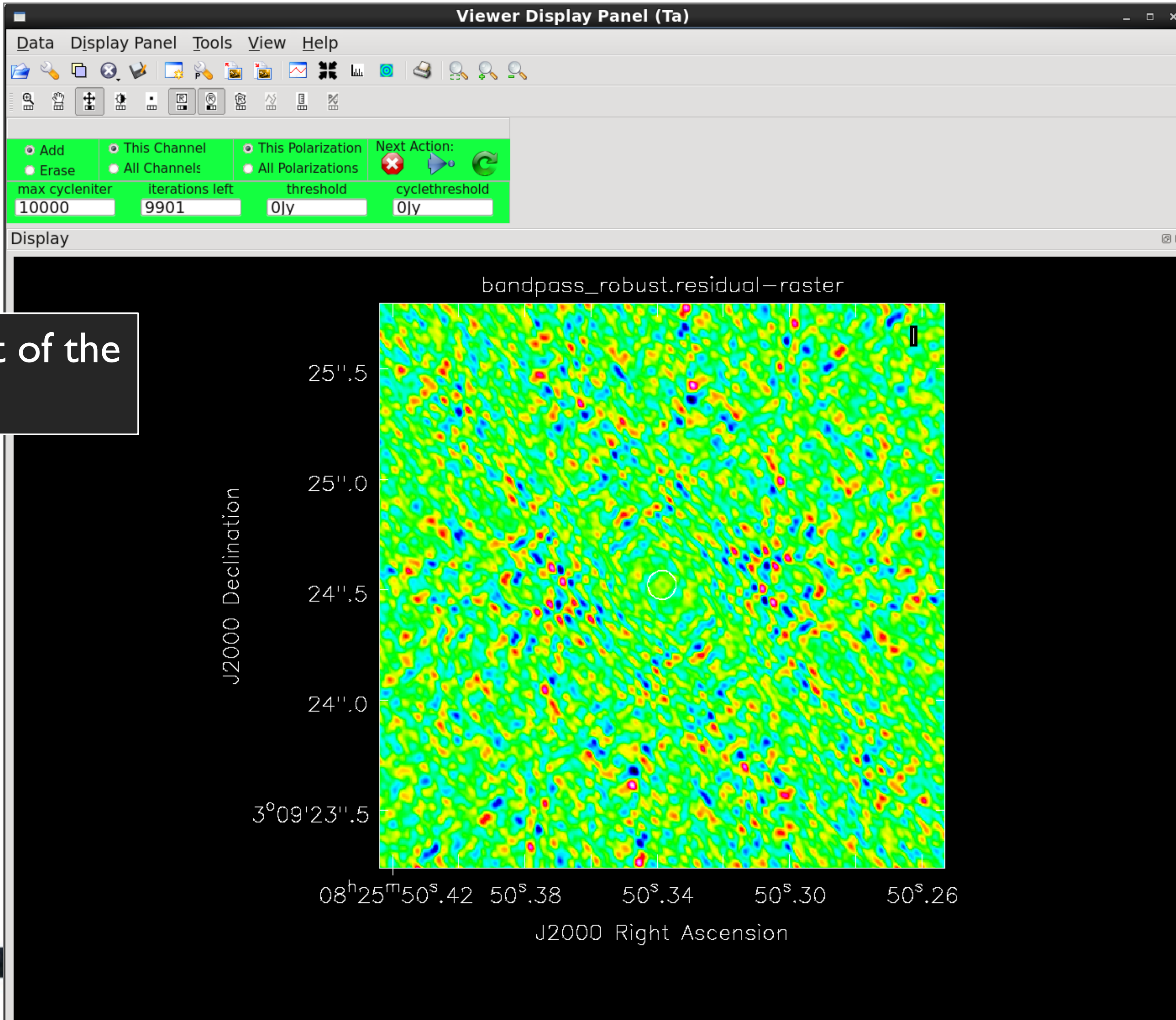
This is the dirty image of our calibrator.

Image the Bandpass Calibrator: Briggs



Define a mask around the emission (similar to the one shown here) using the region buttons above.

Image the Bandpass Calibrator: Briggs



This is a plot of the residuals.

Image the Bandpass Calibrator: Briggs

View the resulting clean image: `viewer("bandpass_robust.image")`

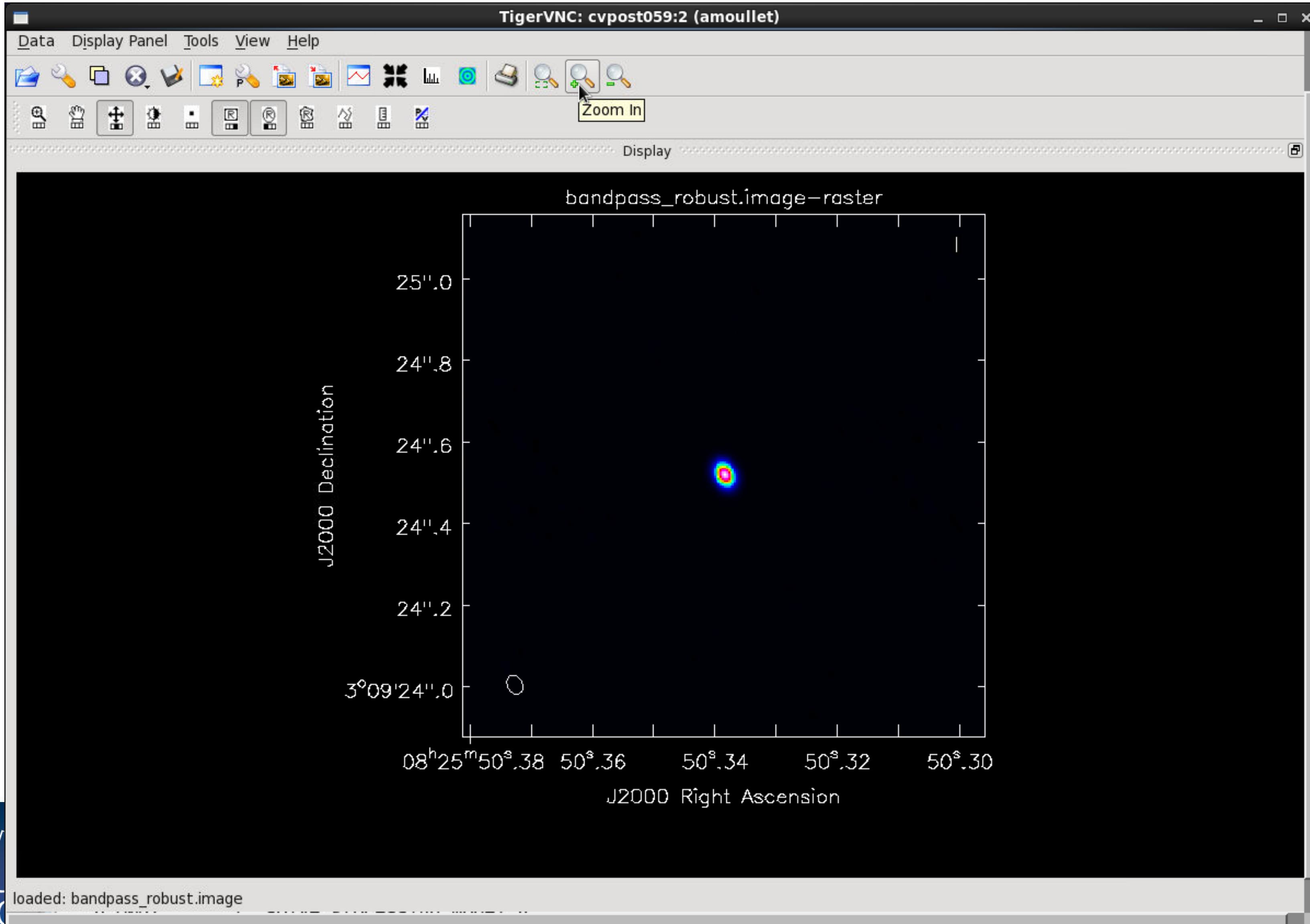


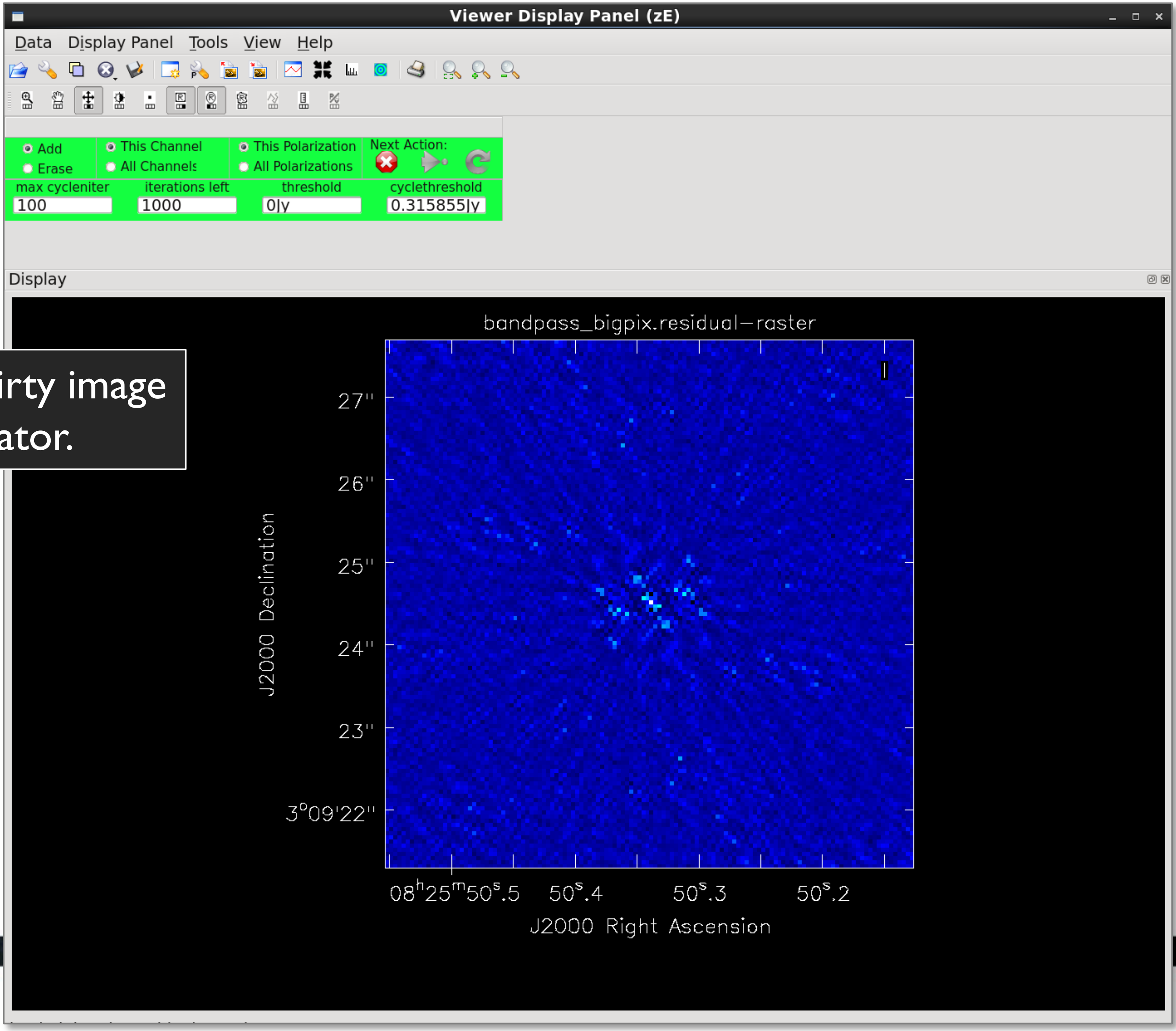
Image the Bandpass Calibrator: Large Pixels

What happens when we image the bandpass calibrator using a larger pixel size?

```
os.system("rm -rf bandpass_bigpix.*")
tclean(vis="bandpass.ms",
      imagename="bandpass_bigpix",
      field="0", spw="",
      specmode="mfs", deconvolver='hogbom', gridder='standard',
      imsize=[128,128], cell=["0.05arcsec"],
      weighting="briggs", robust=-1,
      threshold="0mJy",
      niter=10000, interactive=True)
```

Running tclean will bring up the following interactive window …

Image the Bandpass Calibrator: Large Pixels



This is the dirty image of our calibrator.

Image the Bandpass Calibrator: Large Pixels

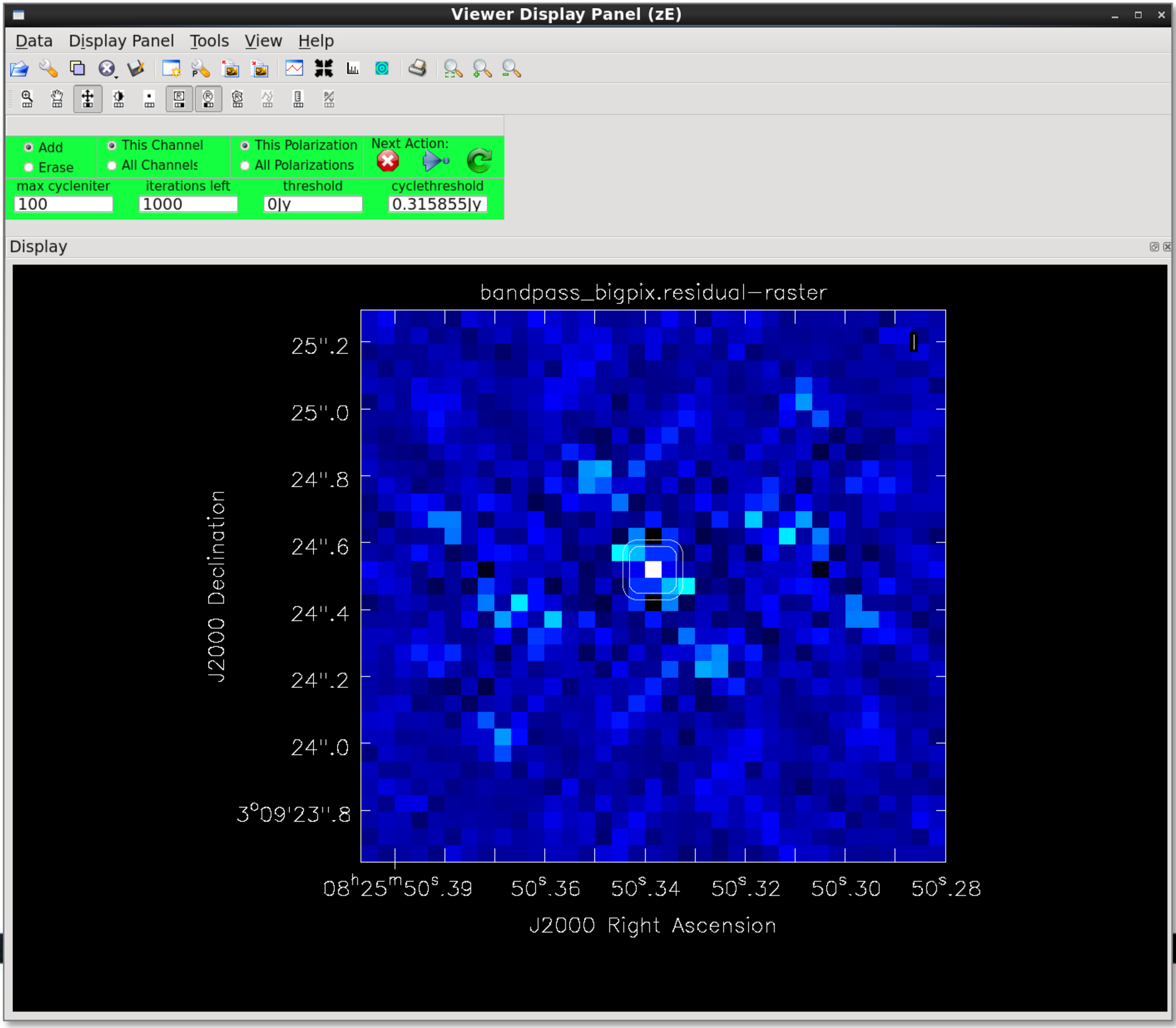
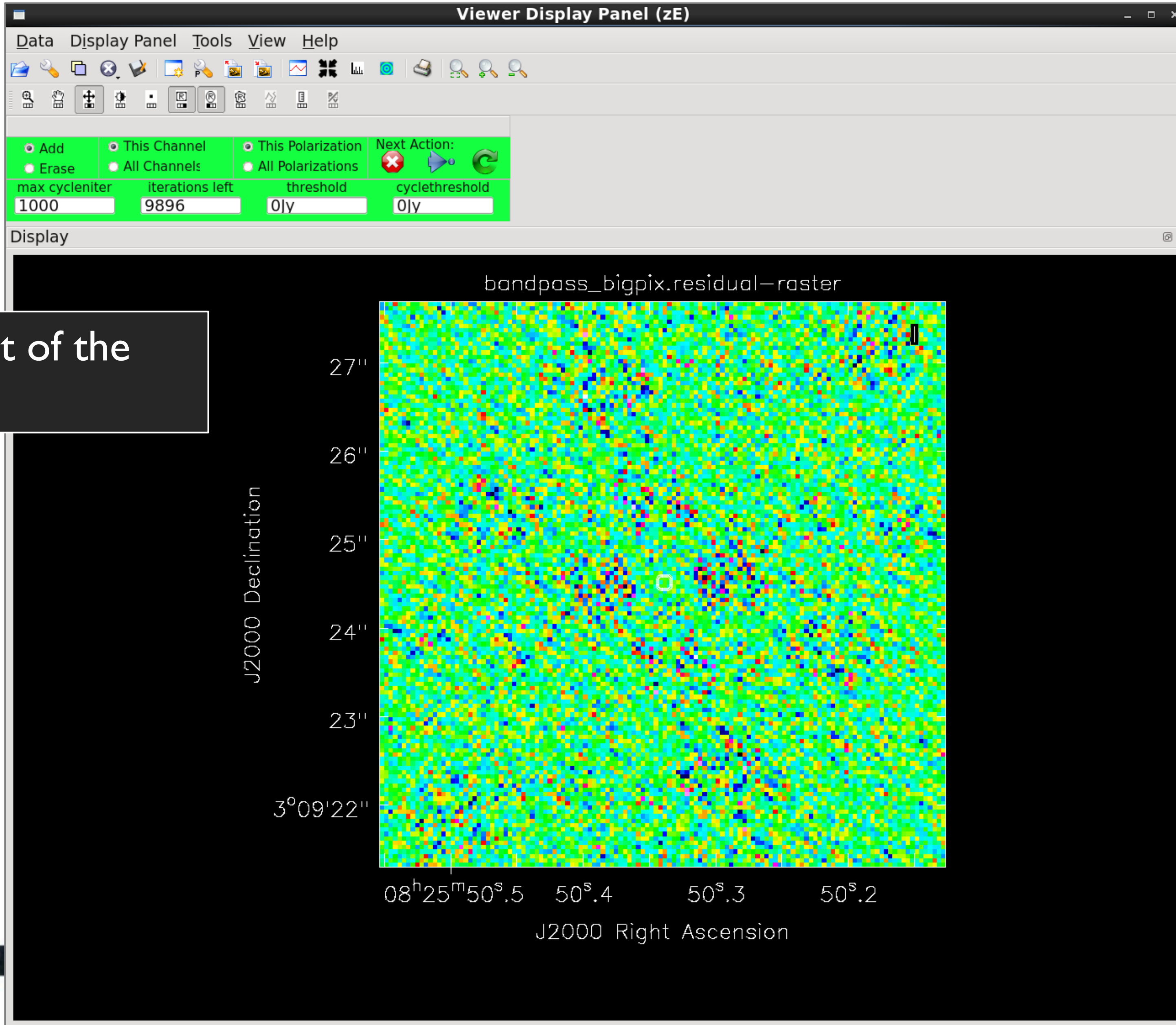


Image the Bandpass Calibrator: Large Pixels



This is a plot of the residuals.

Image the Bandpass Calibrator: Large Pixels

View the resulting clean image: `viewer("bandpass_bigpix.image")`

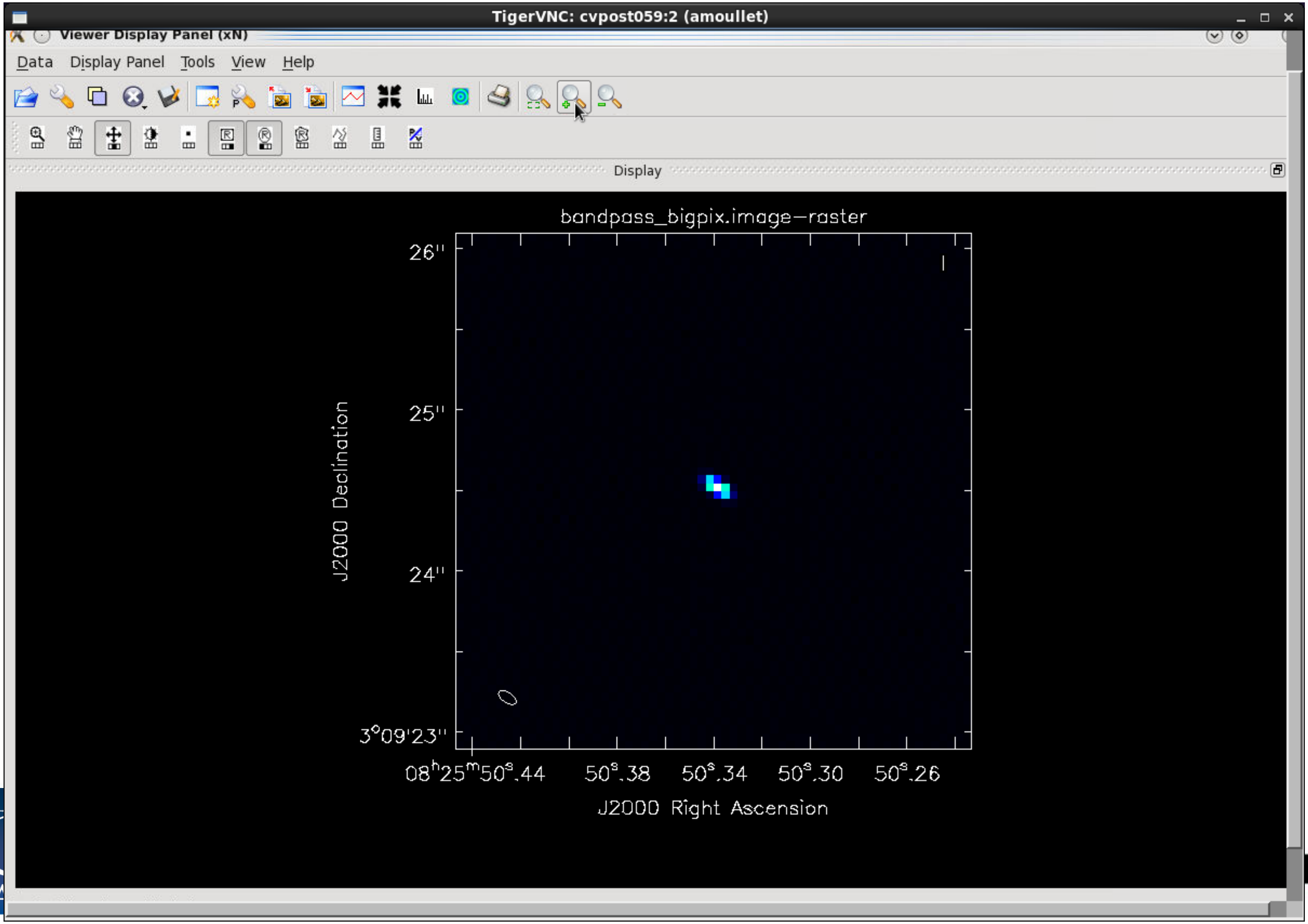
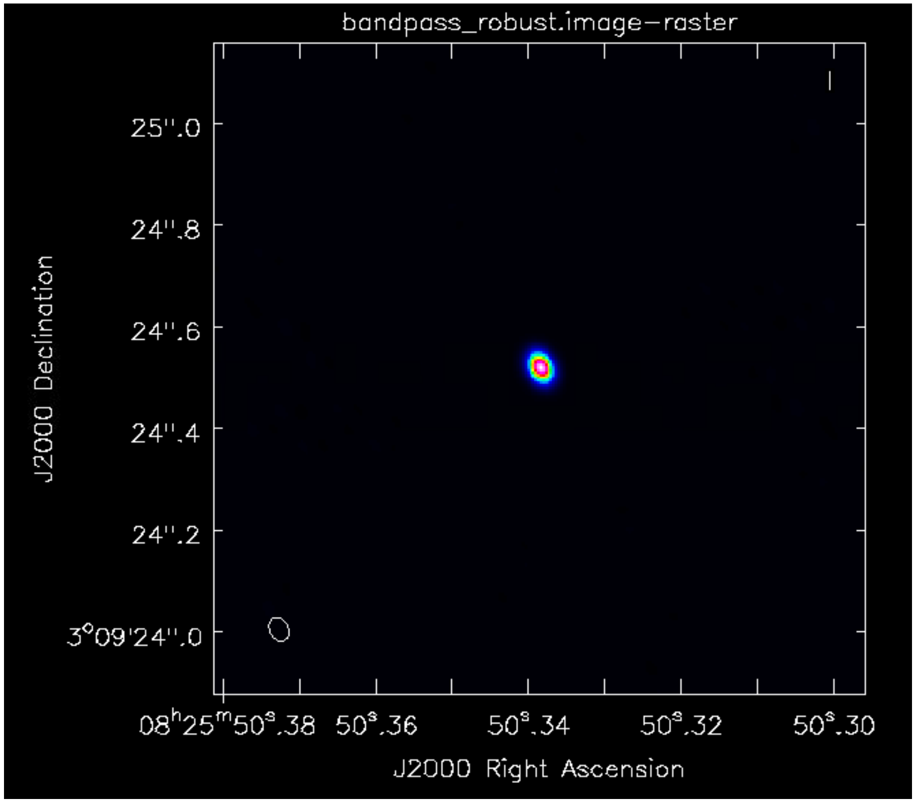


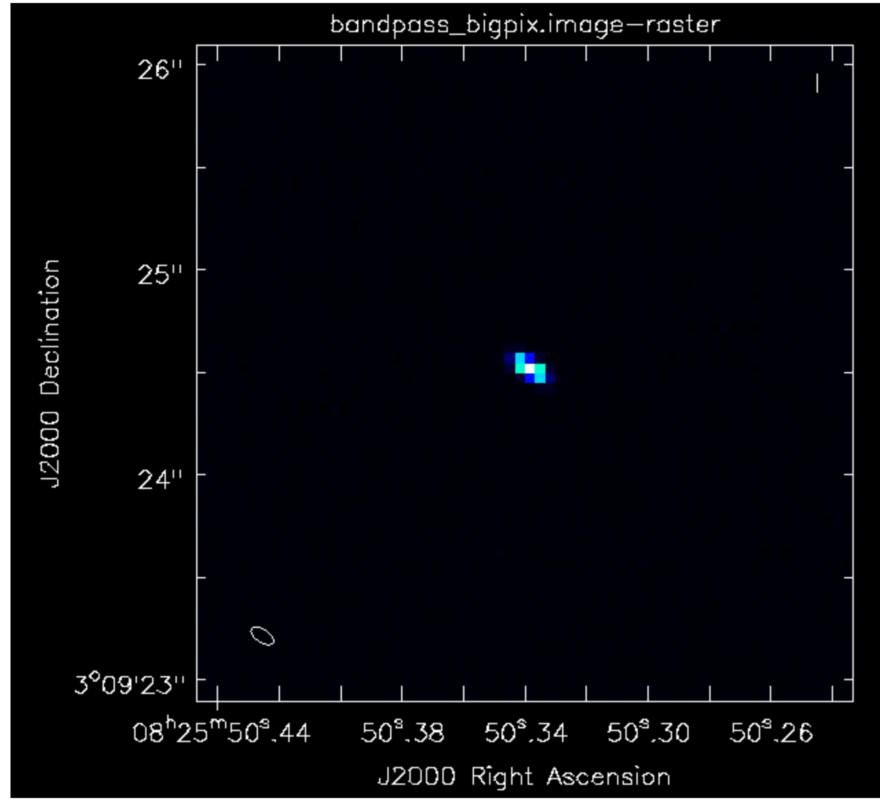
Image the Bandpass Calibrator: Comparison

Image of bandpass calibrator cleaned with robust weighting scheme

Small Pixels



Large Pixels



Imaging the SDP.81 Continuum

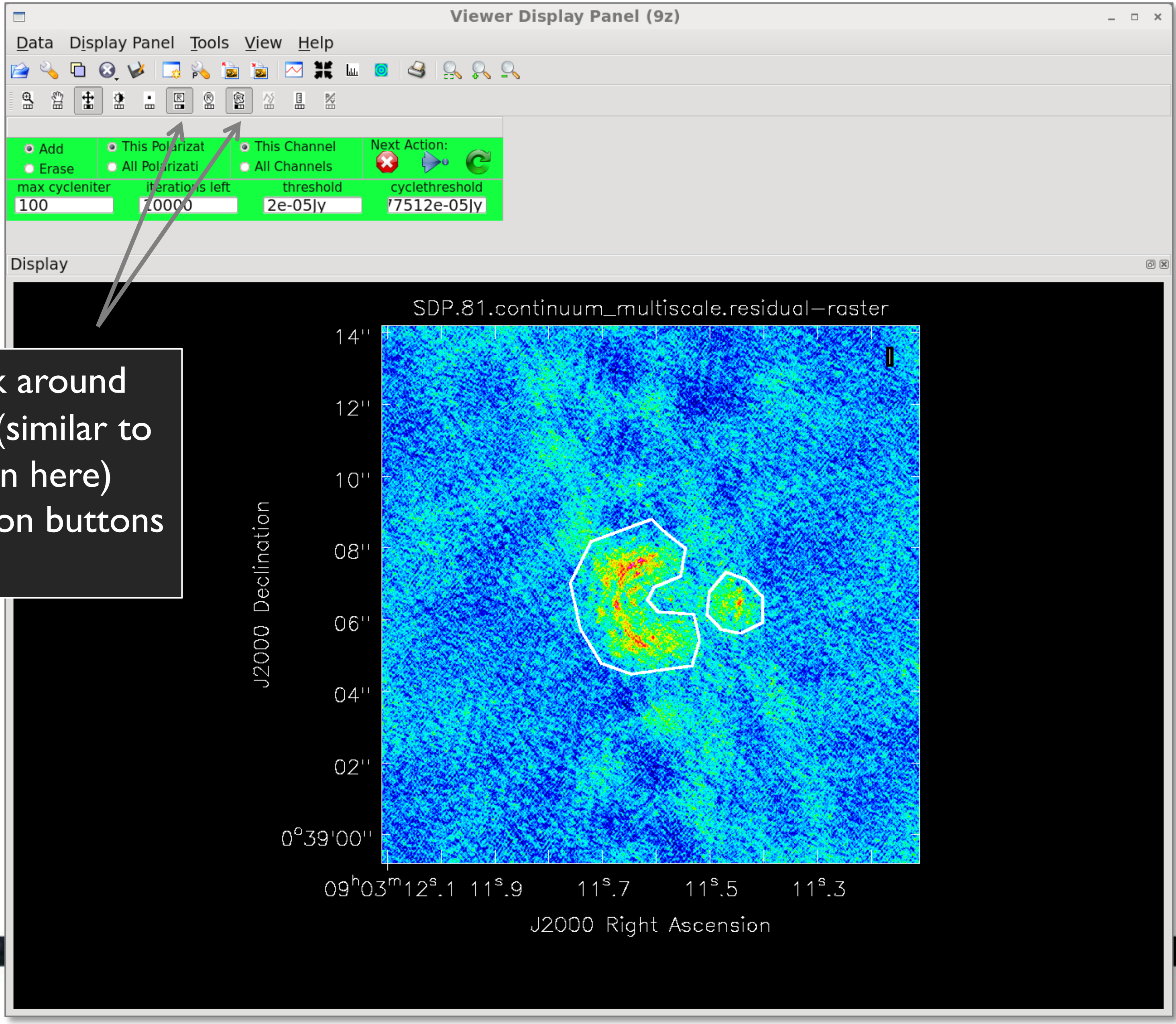
Image the SDP.81 Continuum

We will image the continuum emission in SDP.81 using a multiscale clean. For more information on multiscale cleaning, see the information/references in your imaging.py script.

```
os.system("rm -rf SDP.81.continuum_multiscale.*")
tclean(vis="SDP.81.Band4_continuum.ms",
      imagename="SDP.81.continuum_multiscale",
      spw="", field="SDP*",
      specmode="mfs", gridder="standard", deconvolver="multiscale",
      imsize=1500, cell="0.01arcsec",
      scales=[0,5,15,45],
      interactive=True, mask="",
      weighting="briggs", robust=1.0,
      niter=10000, threshold="0.02mJy")
```

Running tclean will bring up the following interactive window …

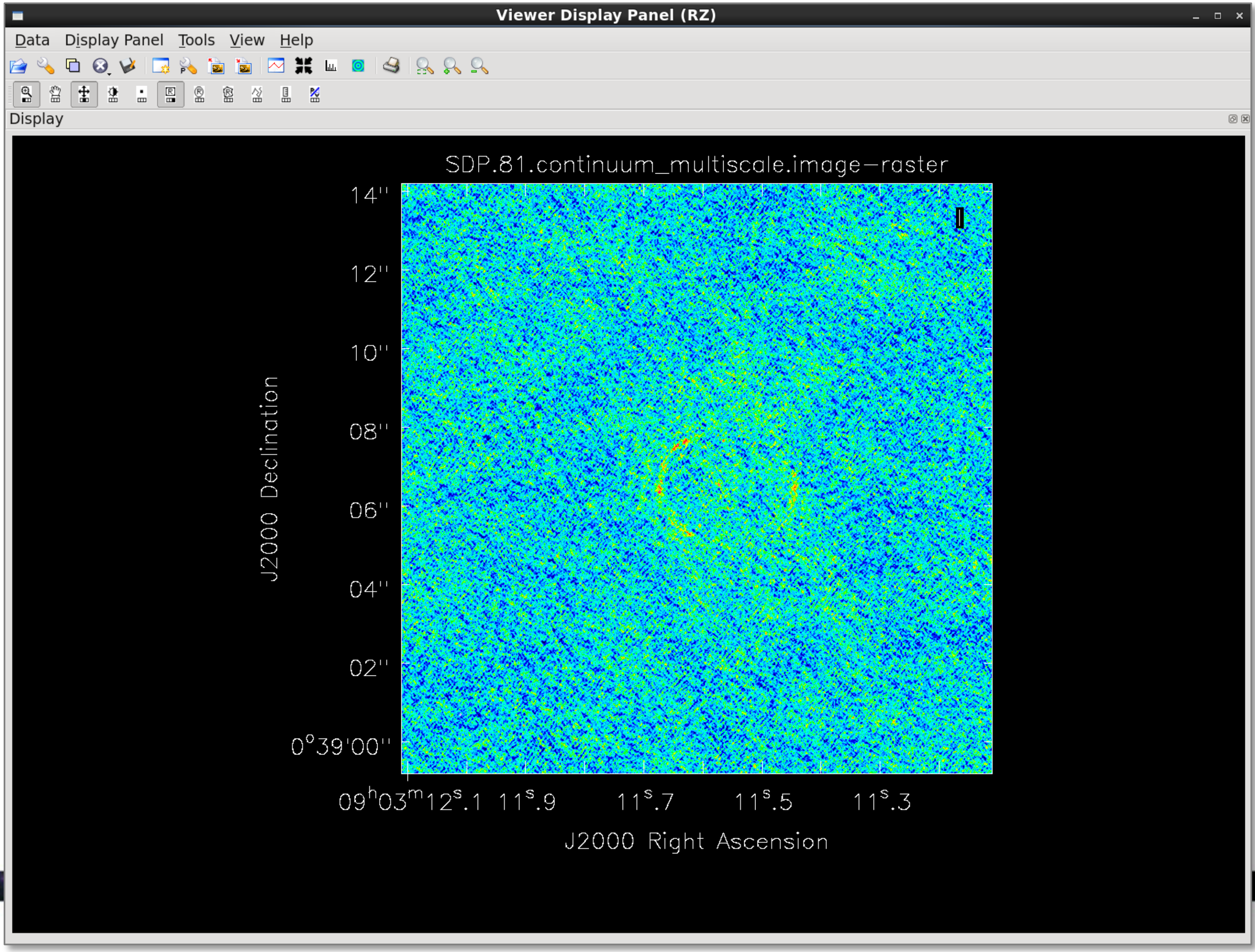
Image the SDP.81 Continuum



Define a mask around the emission (similar to the one shown here) using the region buttons above.

Image the SDP.81 Continuum

View the resulting clean image: `viewer("SDP.81.continuum_multiscale.image")`



Output of *clean*

Minimally:

- SDP.81.continuum_multiscale.pb

  Relative sky sensitivity - shows the primary beam response
- SDP.81.continuum_multiscale.image

  Cleaned and restored image
- SDP.81.continuum_multiscale.mask

  Clean "boxes" shows where you cleaned
- SDP.81.continuum_multiscale.model

  Clean components - the model used by clean (in Jy/pixel)
- SDP.81.continuum_multiscale.psf

  Dirty beam - shows the synthesized beam
- SDP.81.continuum_multiscale.residual

  Residual shows what was left after you cleaned
  (the "dirty" part of the final image)

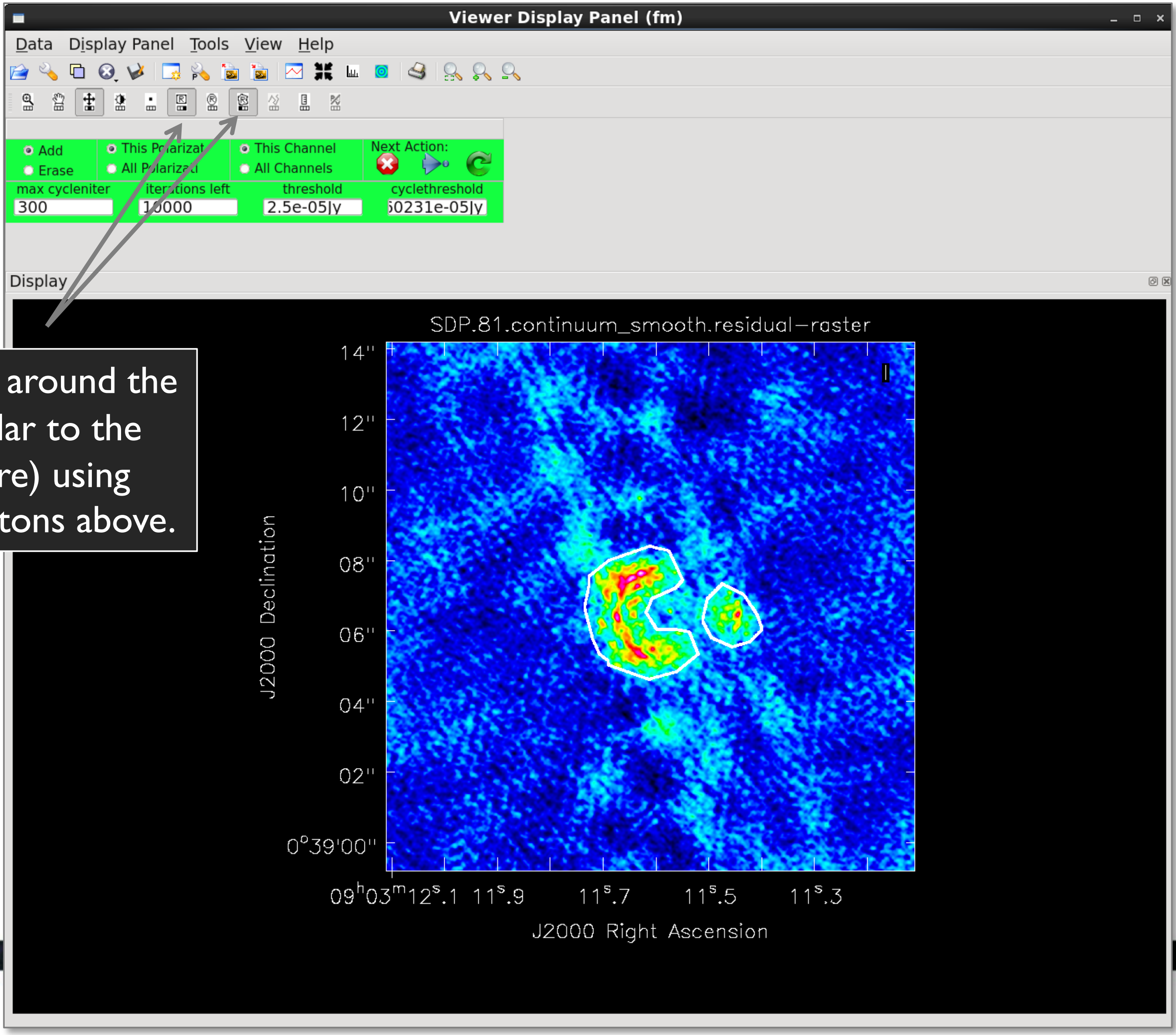
Image the SDP.81 Continuum

Since some emission is still resolved out at this angular resolution, we can image the target while tapering the uv data at long baselines to emphasize and recover more of the extended emission.

```
os.system("rm -rf SDP.81.continuum_smooth.*")
tclean(vis="SDP.81.Band4_continuum.ms",
       imagename="SDP.81.continuum_smooth",
       spw="", field="SDP*",
       specmode="mfs", gridder="standard", deconvolver="multiscale",
       imsize=1500, cell="0.01arcsec",
       scales=[0,5,15,45],
       interactive=True, mask="",
       weighting="briggs", robust=1.0,
       uvtaper=["1000klambda"],
       niter=10000, threshold="0.025mJy")
```

Running tclean will bring up the following interactive window …

Image the SDP.81 Continuum



Define a mask around the emission (similar to the one shown here) using the region buttons above.

Image the SDP.81 Continuum

View the resulting clean image: `viewer("SDP.81.continuum_smooth.image")`

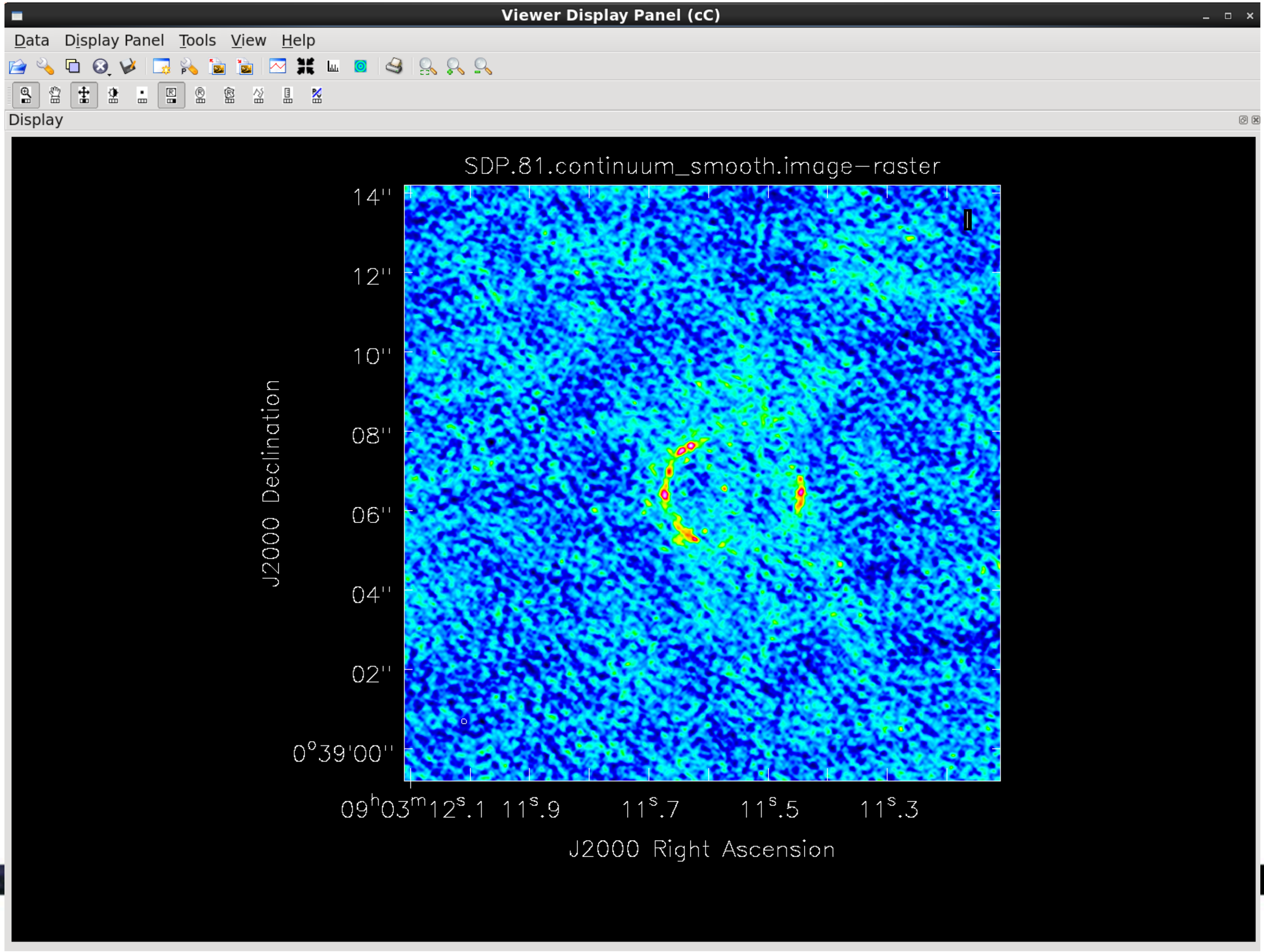
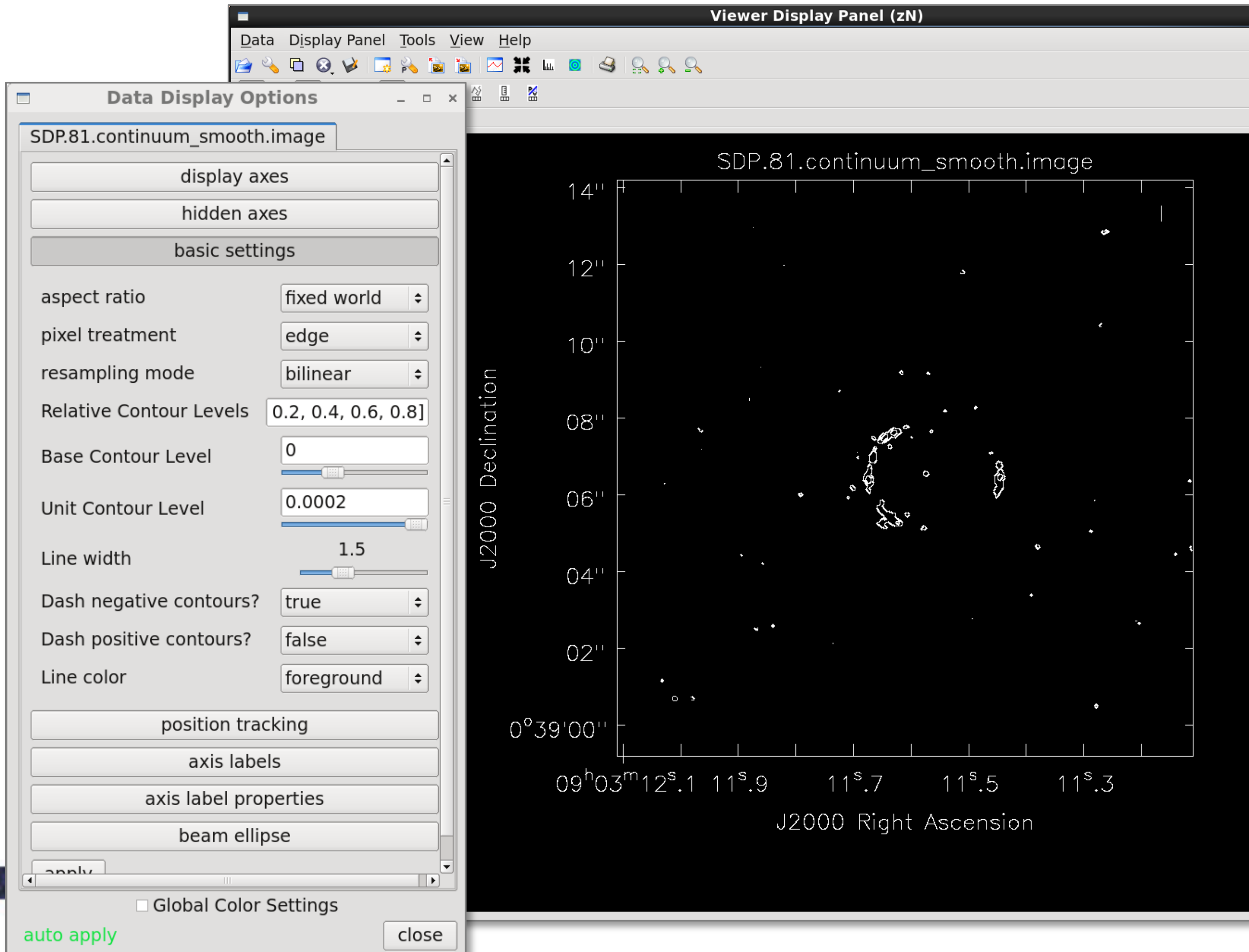


Image the SDP.81 Continuum

View the resulting clean image as a contour plot:

```
viewer("SDP.81.continuum_smooth.image","contour")
```

Adjust contour levels using Data → Adjust Data Display under Basic Settings



Imaging the SDP.81 CO Line

Image the SDP.81 CO Line

The spectral line we will image is CO(5-4) at z = 3.042 (redshifted to 142.57 GHz). To do this, we need to subtract the continuum and split off the line data.

Here, this step has been done for you, as it can take a while.

The spectral windows containing continuum vs line emission are:

```
spw_cont =
   '0~2,4~6,8~10,12~14,16~18,20~22,24~26,28~30,32~34,36~38,40~42,44~46'
spw_line = '3,7,11,15,19,23,27,31,35,39,43,47'
```

Split the spectral line data into a separate measurement set:

```
os.system('rm -rf SDP.81_Band4_COline.ms')
split(vis='SDP.81_Band4.ms',outputvis='SDP.81_Band4_COline.ms',
spw=spw_line,datacolumn='data')
```

Perform the continuum subtraction:

```
os.system("rm -rf SDP.81_Band4_COline.ms.contsub")
uvcontsub(vis="SDP.81_Band4_COline.ms", fitorder=1,
   fitspw="0~11:5~45:170~187")
```

NRAO

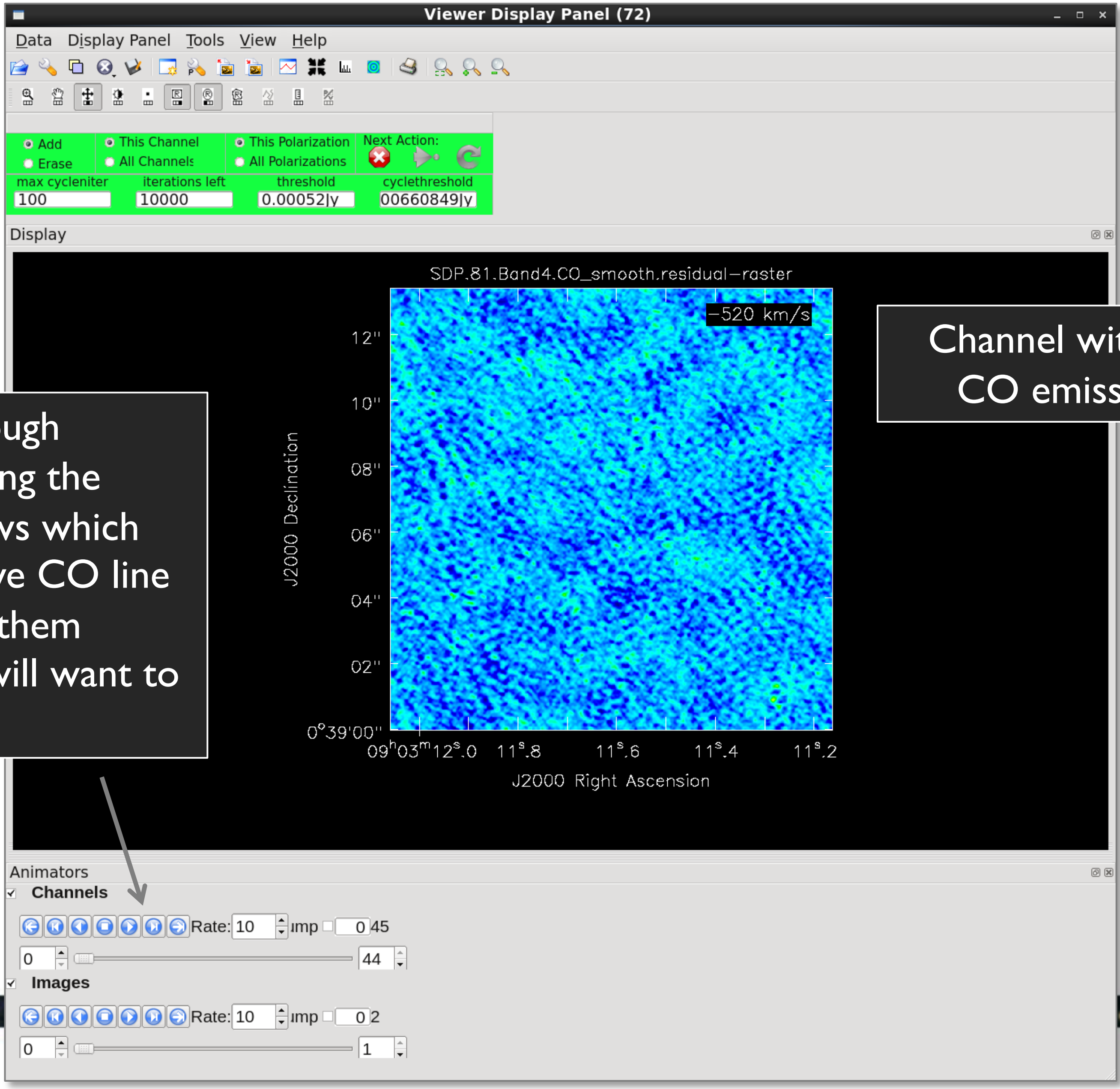
Image the SDP.81 CO Line

Image the CO line emission in SDP.81:

```
os.system("rm -rf SDP.81.CO_smooth.*")
tclean(vis="SDP.81.Band4_COline.ms.contsub",
      imagename="SDP.81.CO_smooth",
      mask="",
      specmode="cube", gridder="standard",
      deconvolver="multiscale",
      imsize=672, cell="0.02arcsec",
      start="-520km/s",width="21km/s",nchan=45,
      outframe="LSRK",restfreq="142.5700GHz",
      scales=[0,5,15,45],
      interactive=True,
      restoringbeam="common",
      weighting="briggs", robust=1.0,
      uvtaper=["1000klambda"],
      niter=10000, threshold="0.52mJy")
```

Running tclean will bring up the following interactive window …

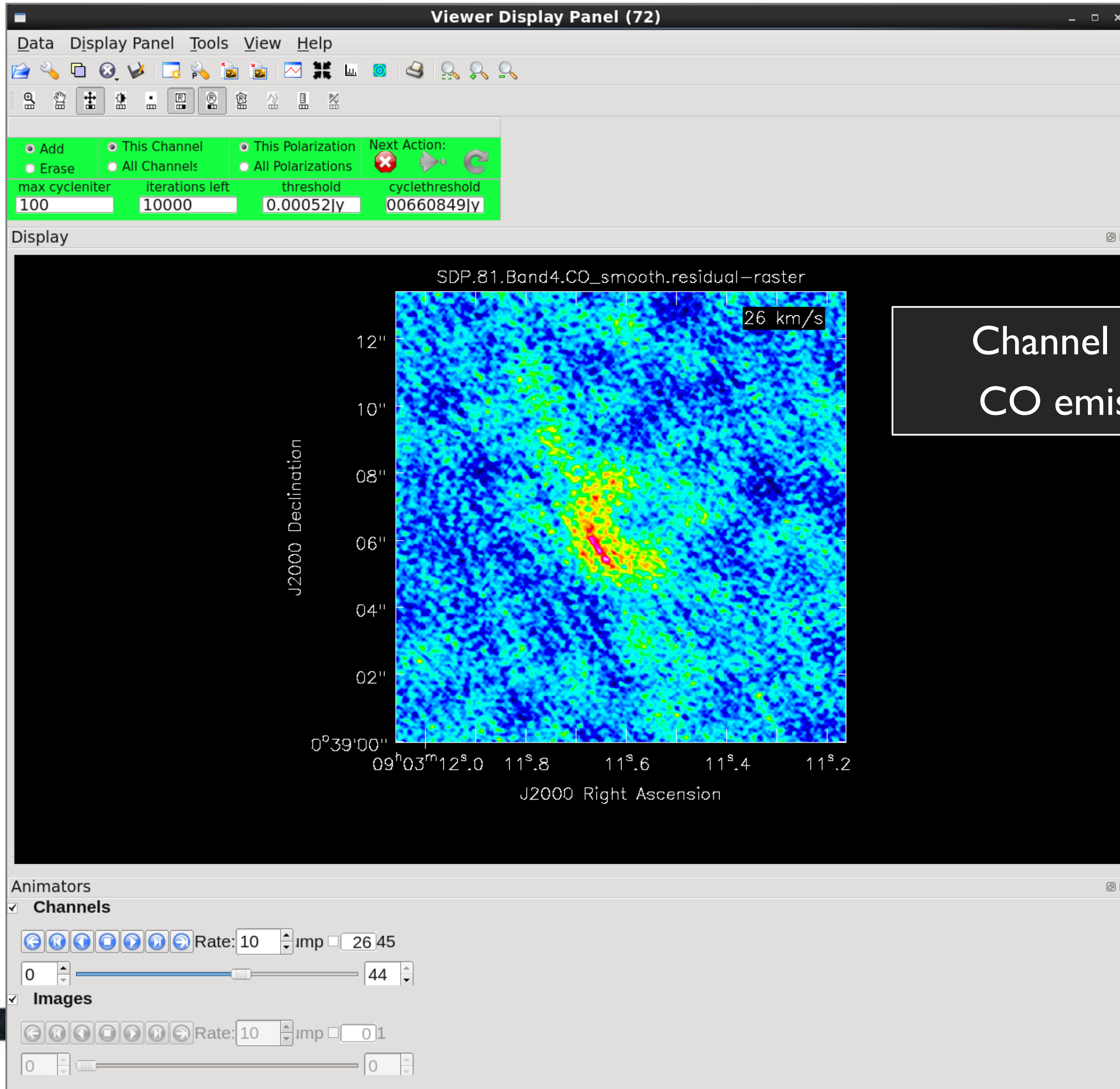
Image the SDP.81 CO Line



Channel with no CO emission

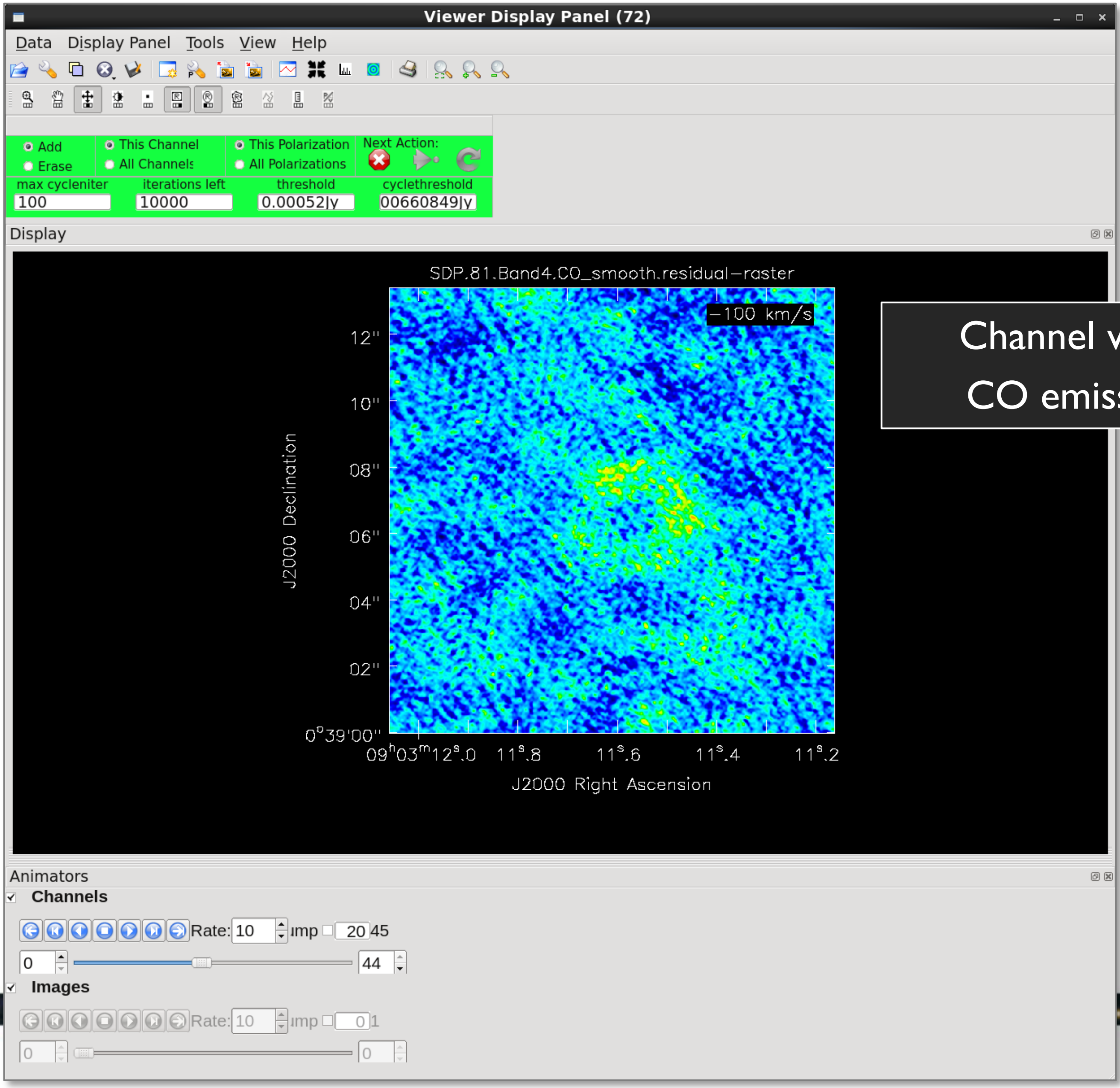
Moving through channels using the arrows shows which channels have CO line emission in them (which we will want to mask.).

Image the SDP.81 CO Line



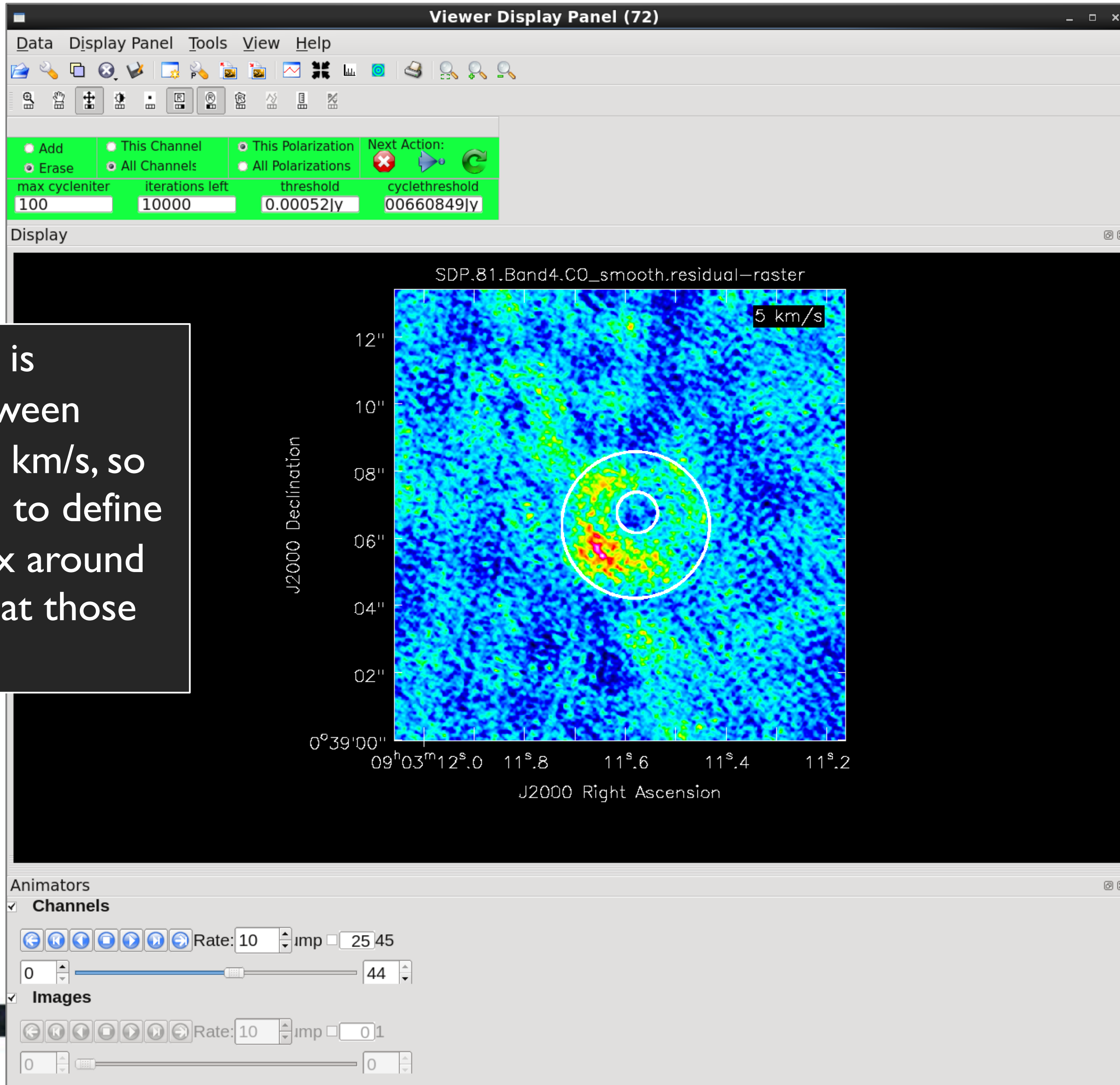
Channel with CO emission

Image the SDP.81 CO Line



Channel with CO emission

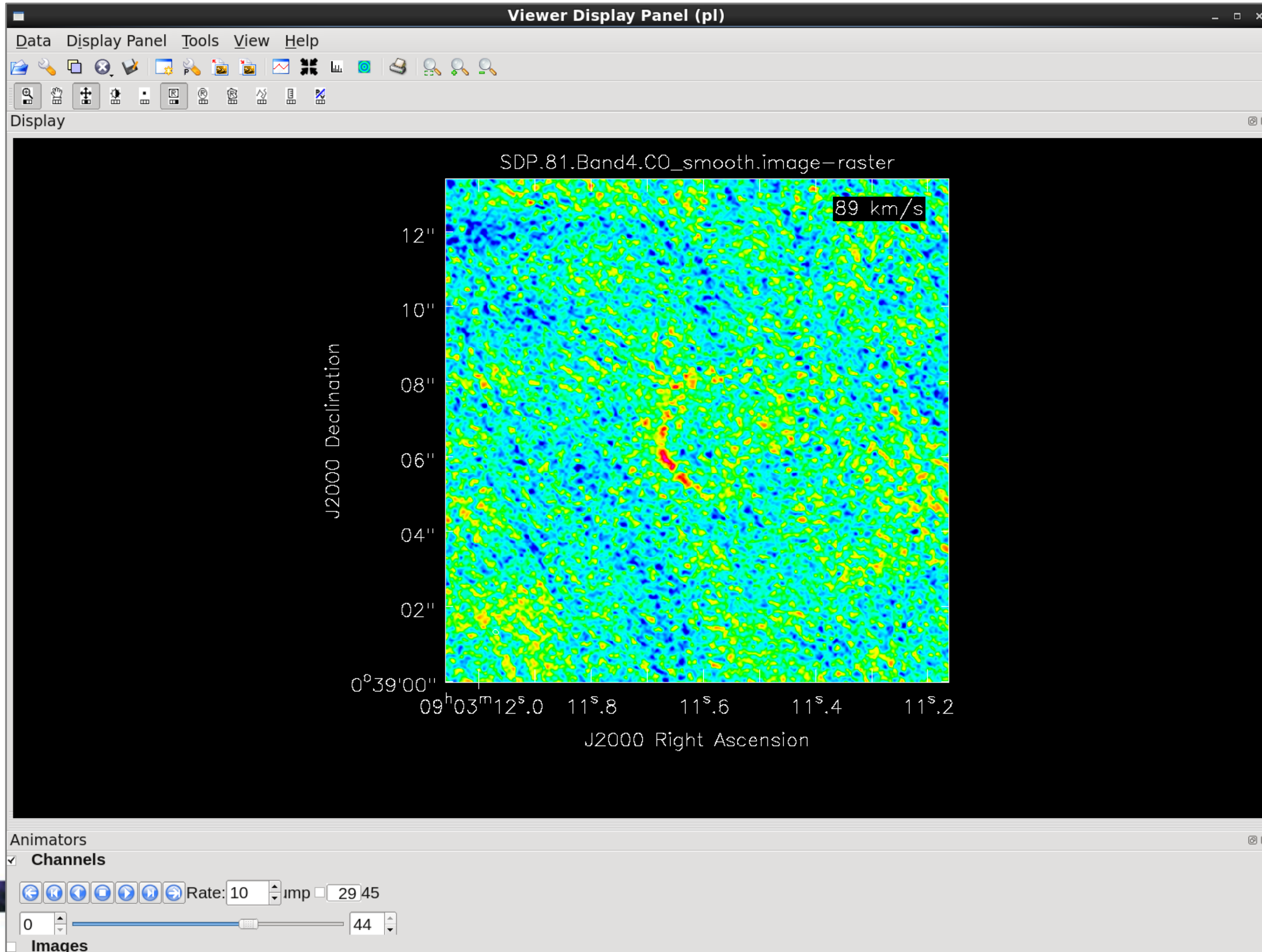
Image the SDP.81 CO Line



CO emission is detected between +200 to -400 km/s, so we only need to define a cleaning box around the emission at those channels.

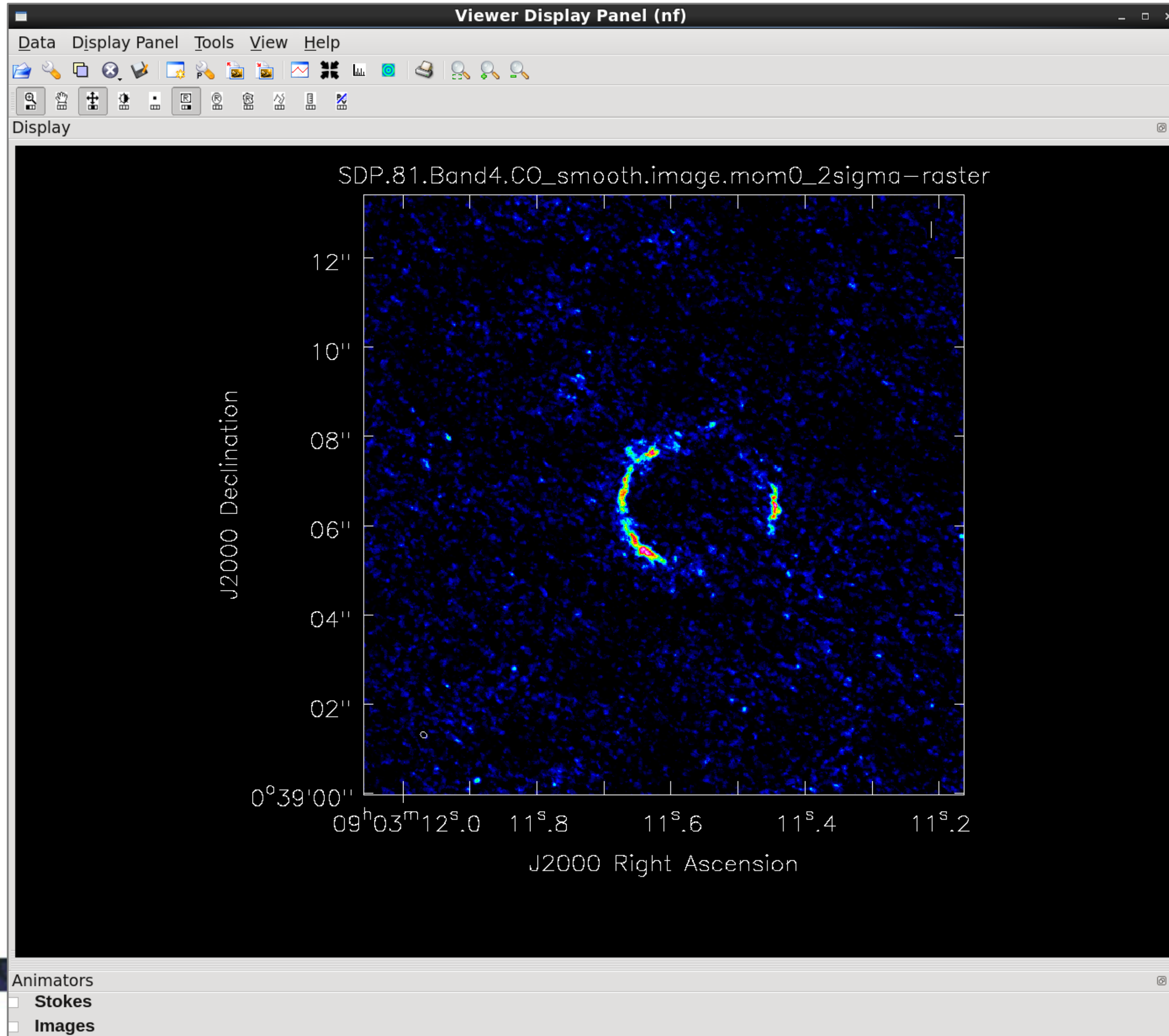
Image the SDP.81 CO Line

View the resulting clean image: `viewer("SDP.81.Band4.CO_smooth.image")`



Find the SDP.81 CO Line integrated intensity

And view: `viewer("SDP.81.Band4.CO_smooth.mom0_2sigma.image")`



And you're done!

You have calibrated one execution of a Band 4 observation of the gravitationally lensed galaxy SDP.81 and imaged the galaxy's continuum and CO line emission.

Extra slides

Expandable Parameters

- Boldface parameters have subparameters that unfold when main parameter is set

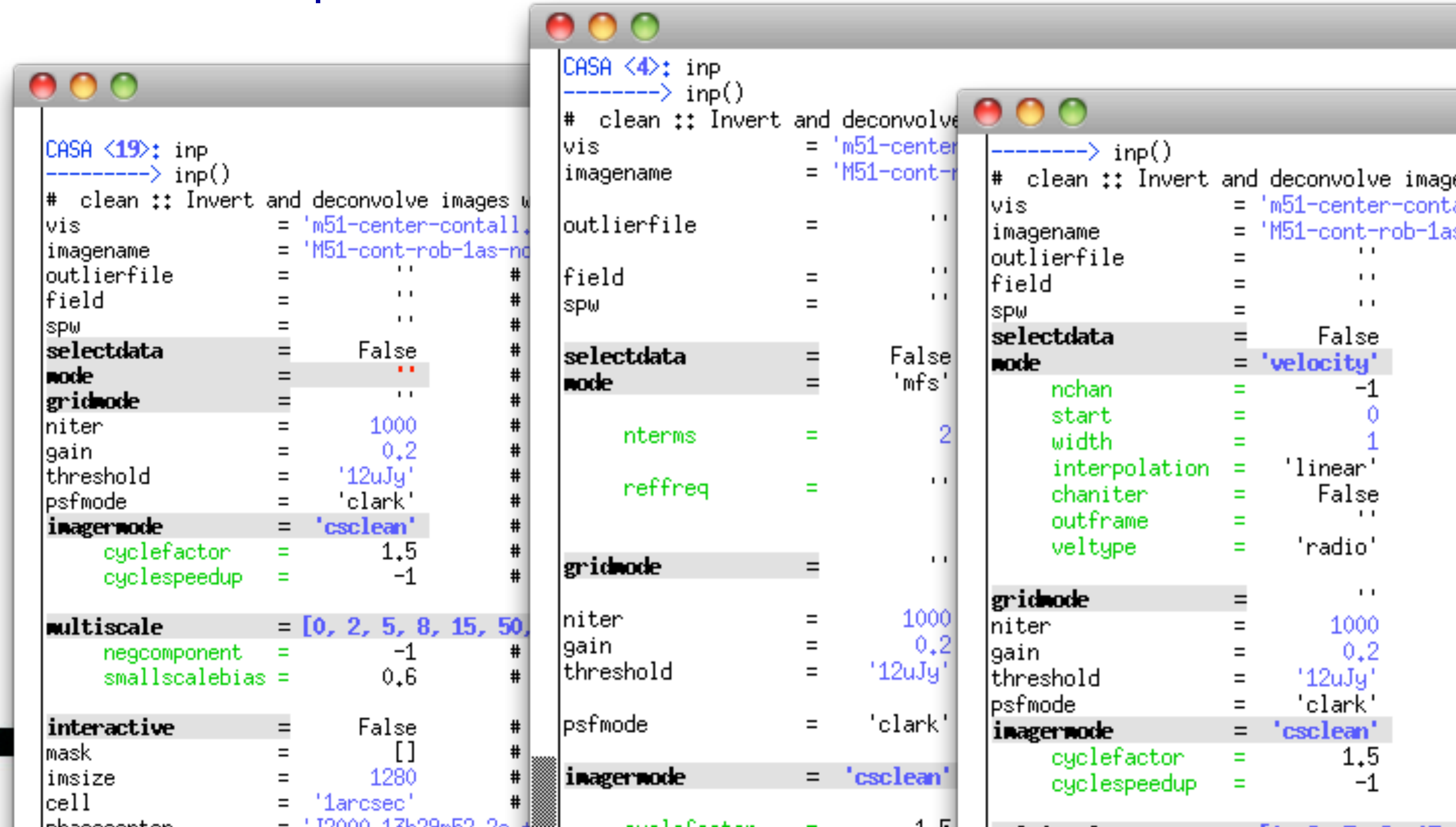


Image the SDP.81 CO Line

```
plotms("SDP.81_Band4_Coline.ms",yaxis="amp",xaxis="channel",
        avgtime="1e8",coloraxis="spw",restfreq="142.5700GHz",
        freqframe="LSRK",transform=True,avgantenna=True,avgscan=True)
```