Introduction to CASA, Calibration & Basic Imaging

ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

May 16, 2014

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**
  - **Discussion of “tools” will be done separately**
- The Flow of Calibration
- Key CASA tasks for data reduction/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 (based on ASAP)
- 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…

CASA Startup

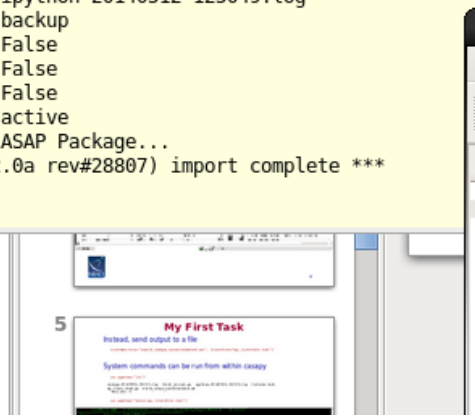
`$ casapy  (or simply “casa”)`

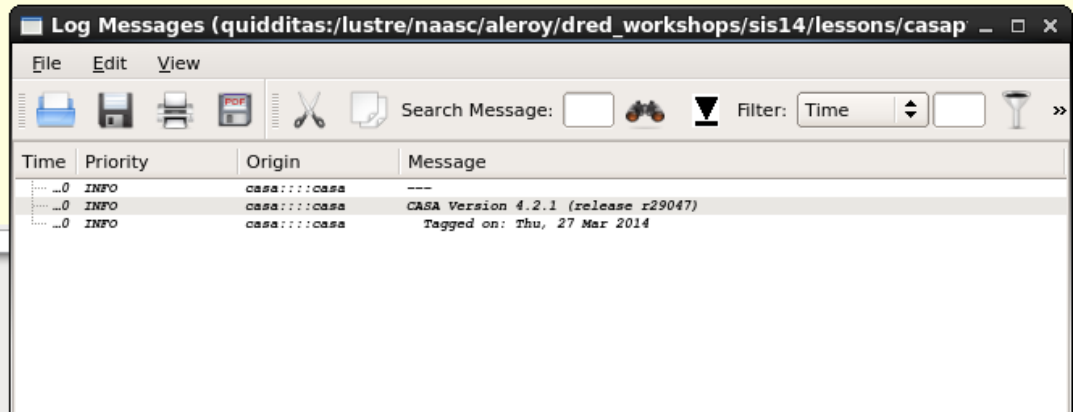
```
bash-4.1$ casa

=======================================
The start-up time of CASA may vary
depending on whether the shared libraries
are cached or not.
=======================================

CASA Version 4.2.1 (r29048)
  Compiled on: Tue 2014/04/01 19:49:27 UTC
---------------------------------------------------------------------
    For help use the following commands:
    tasklist               - Task list organized by category
    taskhelp               - One line summary of available tasks
    help taskname          - Full help for task
    toolhelp               - One line summary of available tools
    help par.parametername - Full help for parameter name
---------------------------------------------------------------------
Activating auto-logging. Current session state plus future input saved.
Filename       : ipython-20140512-125049.log
Mode           : backup
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (4.2.0a rev#28807) import complete ***

CASA <2>:
```





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
    - ipython.log – ipython command history
    - casapyTIME.log – casa logger messages
  - numbered input/output
  - history/searching

Basic Python tips

- to run a python ".py" script:

  `execfile('<scriptname>')`

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

Some python specialties:

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

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`

Default

New Info Customize Bookmarks Close

```
CASA <2>: tasklist
--------> tasklist()
Available tasks, organized by category (experimental tasks in parenthesis):

Import/Export       Information   Data Editing  Display/Plotting
-------------       -----------   ------------  ----------------
importvla           imhead        concat        clearplot
importfits          imstat        fixvis        plotants
importuvfits        listcal       flagautocorr  plotcal
exportfits          listhistory   flagdata      plotms
exportuvfits        listobs       flagmanager   plotxy
(importasdm)        listvis       plotms        viewer
(importgmrt)        vishead       plotxy        (viewerconnection)
                    visstat


Data Manipulation   Calibration   Imaging       Modelling
-----------------   -----------   -------       ---------
concat              accum         clean         setjy
cvel                applycal      deconvolve    uvcontsub
fixvis              bandpass      feather       uvmodelfit
hanningsmooth       blcal         ft            uvsub
split               calstat       makemask      (uvcontsub2)
uvcontsub           clearcal      (autoclean)
uvsub               cvel          (boxit)
(uvcontsub2)        fluxscale
(msmoments)         fixvis
                    gaincal
                    gencal
                    listcal
                    polcal
                    setjy
                    smoothcal
                    (fringecal)
                    (peel)


Image Analysis  Simulation  Utilities           Single Dish
--------------  ----------  ---------           (after running asap_init())
imcontsub       simdata     browsetable         --------------------------
imhead          (simdata2)  casalogger          sdaverage
imfit                       clearplot           sdbaseline
immath                      clearstat           sdcal
immoments                   csvclean            sdcoadd
imregrid                    filecatalog         sdfit
imsmooth                    find                sdflag
imstat                      help par.parameter  sdimaging
imval                       help task           sdimprocess
(specfit)                   rmtables            sdlist
                            startup             sdmath
                            taskhelp            sdplot
                            tasklist            sdsave
                            toolhelp            sdscale
                                                sdsmooth
                                                sdstat
                                                sdtpimaging
                                                (sdsim)
                                                (msmoments)
```

Find the right Task

To see list of tasks with short help:

`taskhelp`

Default

New Info Customize Bookmarks Close

```
CASA <15>: taskhelp
---------> taskhelp()
Available tasks:

accum               : Accumulate incremental calibration solutions into a calibration
applycal            : Apply calibrations solutions(s) to data
autoclean           : CLEAN an image with automatically-chosen clean regions.
bandpass            : Calculates a bandpass calibration solution
blcal               : Calculate a baseline-based calibration solution (gain or bandpas
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
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 t
cvel                : regrid an MS to a new spectral window / channel structure or fra
deconvolve          : Image based deconvolver
exportasdm          : Convert a CASA visibility file (MS) into an ALMA Science Data Mo
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:
fixvis              : Recalculates or converts (u, v, w)
flagautocorr        : Flag autocorrelations
flagcmd             : Flagging task based on flagging commands
flagdata            :  All purpose flagging task based on selections
flagdata2           :  All purpose flagging task based on selections. It allows the co
flagmanager         : Enable list, save, restore, delete and rename flag version files
fluxscale           : Bootstrap the flux density scale from standard calibrators
ft                  : Insert a source model into the MODEL_DATA column of a visibility
gaincal             : Determine temporal gains from calibrator observations
gencal              : Specify Calibration Values of Various Types
hanningsmooth       : Hanning smooth frequency channel data to remove Gibbs ringing
imcollapse          : Collapse image along one axis, aggregating pixel values along th
imcontsub           : Subtracts specified continuum channels from a spectral line data
imfit               : Fit one or more elliptical Gaussian components on an image regio
imhead              : List, get and put image header parameters
immath              : Perform math operations on images
immoments           : Compute moments from an image
```

Default | Default | Default | Default

Task Interface

examine task parameters with **`inp`** :

Default

New Info Customize Bookmarks Close

```
CASA <12>: inp
---------> inp()
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''            #  Name of input visibility file
imagename           =         ''            #  Pre-name of output images
outlierfile         =         ''            #  Text file with image names, sizes, centers for outliers
field               =         ''            #  Field Name or id
spw                 =         ''            #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False            #  Other data selection parameters
mode                =  'channel'            #  Spectral gridding type (mfs, channel, velocity, frequency)
     nchan          =         -1            #  Number of channels (planes) in output image; -1 = all
     start          =          0            #  Begin the output cube at the frequency of this channel in the MS
     width          =          1            #  Width of output channel relative to MS channel (# to average)
     interpolation  =  'linear'            #  Spectral interpolation (nearest, linear, cubic). Use nearest for
                                           #   mode=channel
     chaniter       =      False            #  Clean each channel to completion (True), or all channels each cycle (False)
     outframe       =         ''            #  velocity frame of output image

gridmode            =         ''            #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500            #  Maximum number of iterations
gain                =        0.1            #  Loop gain for cleaning
threshold           =   '0.0mJy'            #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'            #  Method of PSF calculation to use during minor cycles
imagermode          =         ''            #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []            #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False            #  Use interactive clean (with GUI viewer)
mask                =         []            #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]            #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']         #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''            #  Image center: direction or field index
restfreq            =         ''            #  Rest frequency to assign to image (see help)
stokes              =        'I'            #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'            #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False            #  Apply additional uv tapering of visibilities
modelimage          =         ''            #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']            #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False            #  Output primary beam-corrected image
minpb               =        0.2            #  Minimum PB level to use
calready            =       True            #  True required for self-calibration
async               =      False            #  If true the taskname must be started using clean(...)

CASA <13>:
```

Default | Default | Default | Default | Default

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

xterm

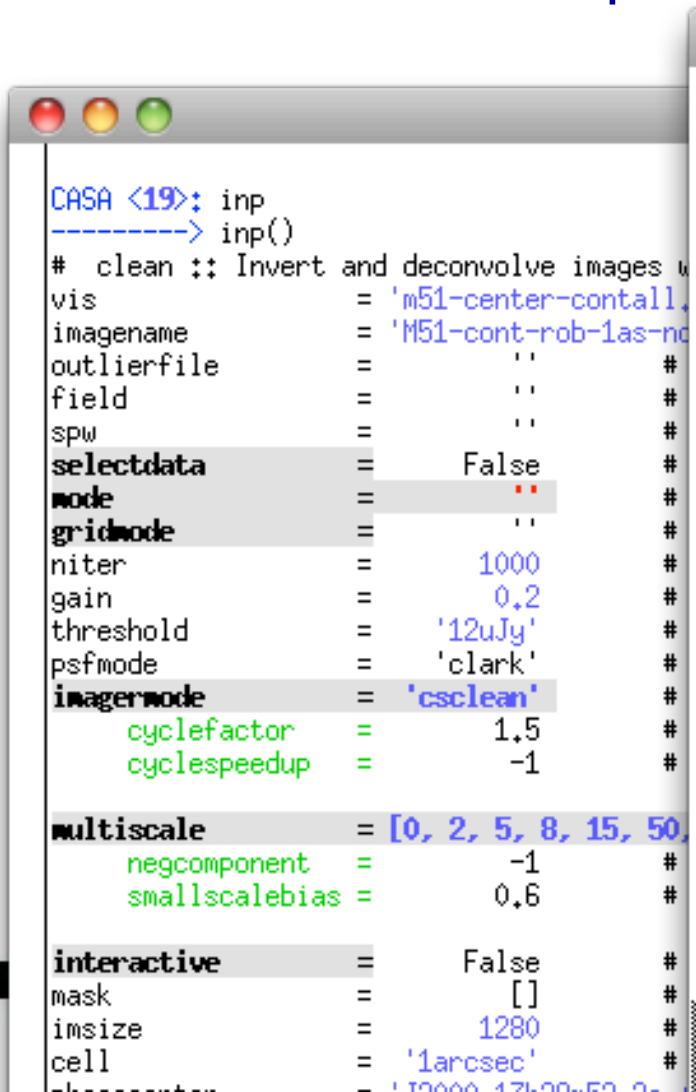
```
CASA <19>: inp
---------> inp()
#  clean :: Invert and deconvolve images with selected algorithm
vis                 = 'm51-center-contall.ms' #  Name of input visibility file
imagename           = 'M51-cont-rob-1as-noninteractive-mult-phasecenter-nterm2' #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =         ''        #  Spectral gridding type (mfs, channel, velocity, frequency)
gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =       1000        #  Maximum number of iterations
gain                =        0.2        #  Loop gain for cleaning
threshold           =     '12uJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =     'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =   'csclean'        #  Options: 'csclean' or 'mosaic', '', uses psfmode
     cyclefactor    =        1.5        #  Controls how often major cycles are done. (e.g. 5 for frequently)
     cyclespeedup   =         -1        #  Cycle threshold doubles in this number of iterations

multiscale          = [0, 2, 5, 8, 15, 50, 100] #  Deconvolution scales (pixels); [] = standard clean
     negcomponent   =         -1        #  Stop cleaning if the largest scale finds this number of neg components
     smallscalebias =        0.6        #  a bias to give more weight toward smaller scales

interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              =       1280        #  x and y image size in pixels. Single value: same for both
cell                =   '1arcsec'        #  x and y cell size(s). Default unit arcsec.
```

Expandable Parameters

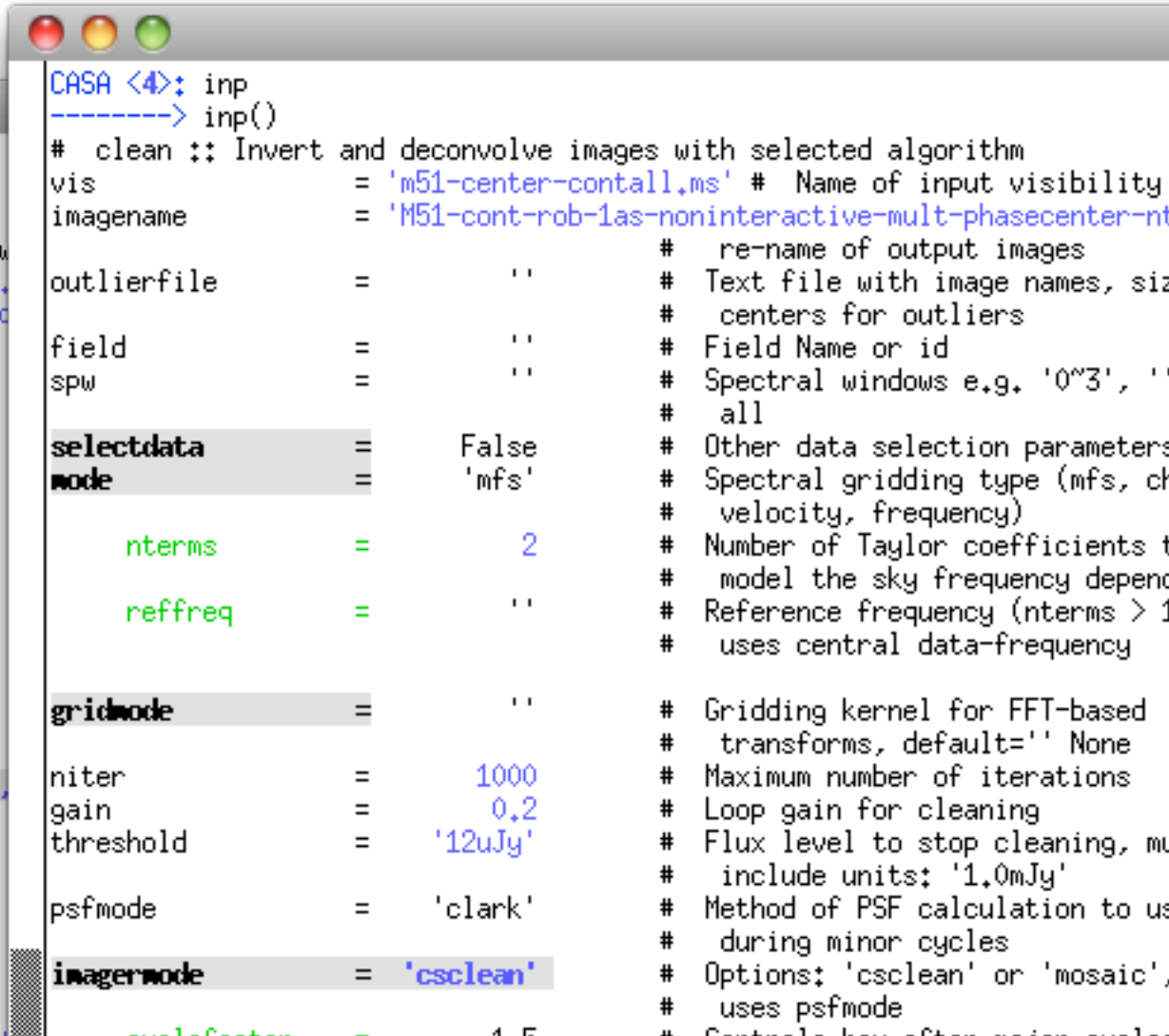
- Boldface parameters have subparameters that unfold when main parameter is set



```
CASA <19>: inp
---------> inp()
#  clean :: Invert and deconvolve images w
vis                 = 'm51-center-contall.
imagename           = 'M51-cont-rob-1as-nc
outlierfile         =         ''          #
field               =         ''          #
spw                 =         ''          #
selectdata          =      False          #
mode                =         ''          #
gridmode            =         ''          #
niter               =       1000          #
gain                =        0.2          #
threshold           =     '12uJy'          #
psfmode             =    'clark'           #
imagermode          =  'csclean'           #
     cyclefactor    =        1.5          #
     cyclespeedup   =         -1          #

multiscale          = [0, 2, 5, 8, 15, 50,
     negcomponent   =         -1          #
     smallscalebias =        0.6          #

interactive         =      False          #
mask                =         []          #
imsize              =       1280          #
cell                =   '1arcsec'          #
```

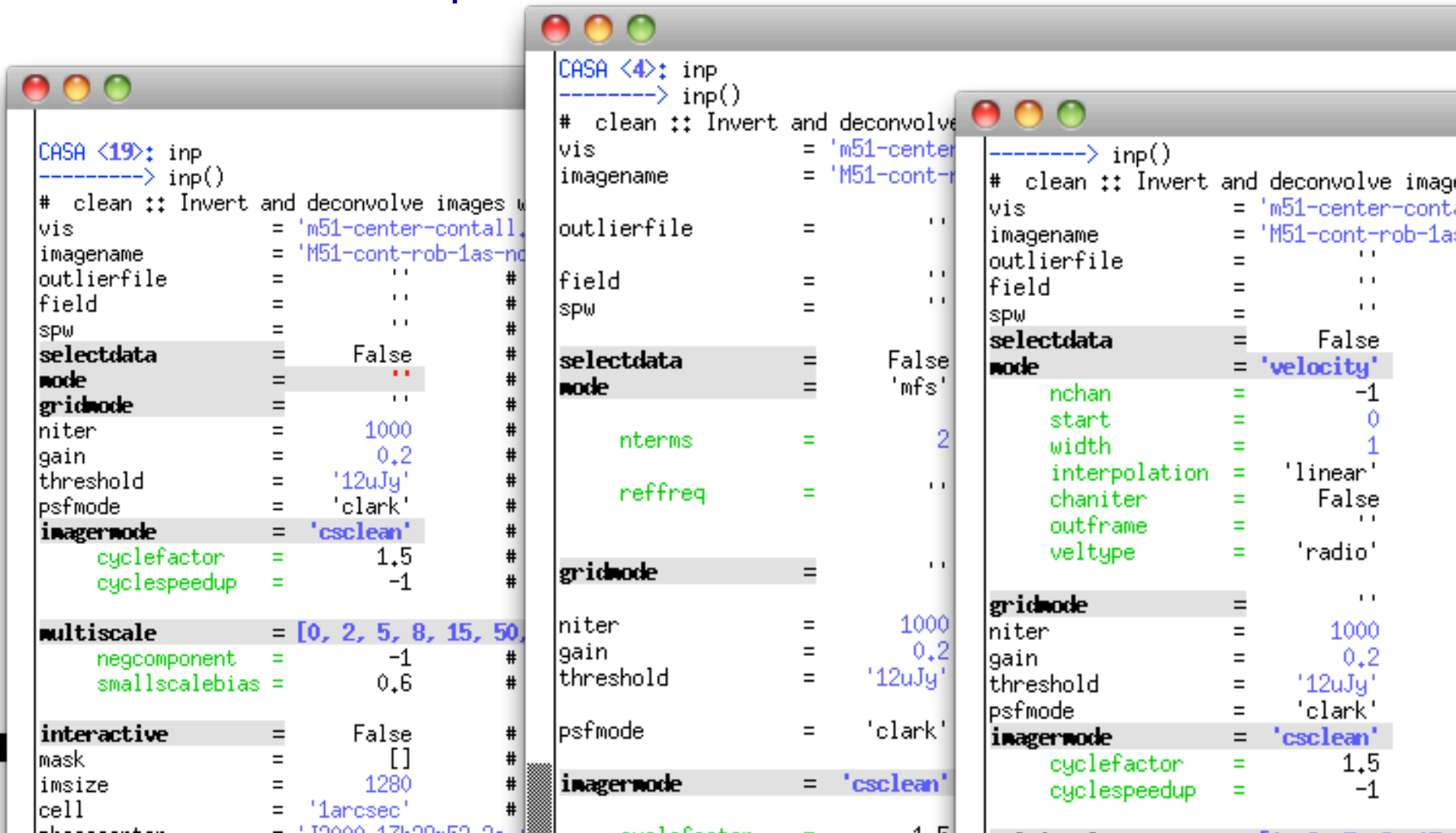


```
CASA <4>: inp
--------> inp()
#  clean :: Invert and deconvolve images with selected algorithm
vis                 = 'm51-center-contall.ms' #  Name of input visibility
imagename           = 'M51-cont-rob-1as-noninteractive-mult-phasecenter-nt
                                              #   re-name of output images
outlierfile         =         ''            #  Text file with image names, siz
                                              #   centers for outliers
field               =         ''            #  Field Name or id
spw                 =         ''            #  Spectral windows e.g. '0~3', ''
                                              #   all
selectdata          =      False            #  Other data selection parameters
mode                =      'mfs'            #  Spectral gridding type (mfs, ch
                                              #   velocity, frequency)
     nterms         =          2            #  Number of Taylor coefficients t
                                              #   model the sky frequency depend
     reffreq        =         ''            #  Reference frequency (nterms > 1
                                              #   uses central data-frequency

gridmode            =         ''            #  Gridding kernel for FFT-based
                                              #   transforms, default='' None
niter               =       1000            #  Maximum number of iterations
gain                =        0.2            #  Loop gain for cleaning
threshold           =     '12uJy'           #  Flux level to stop cleaning, mu
                                              #   include units: '1.0mJy'
psfmode             =    'clark'            #  Method of PSF calculation to us
                                              #   during minor cycles
imagermode          =  'csclean'            #  Options: 'csclean' or 'mosaic',
                                              #   uses psfmode
```

Expandable Parameters

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



Parameter Checking

sanity checks of parameters in inp :

xterm

```
CASA <17>: inp
---------> inp()
#  clean :: Invert and deconvolve images with selected algor
vis                 = 'm51-center-contall.ms' #  Name of inp
imagename           = 'M51-cont-rob-1as-noninteractive-mult-                Pre-name of output images
outlierfile         =         ''        #  Text file with im                rs for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                = 'Monty Python'    #  Spectral gridding type (mfs, channel, velocity, frequency)
gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =       1000        #  Maximum number of iterations
gain                =        0.2        #  Loop gain for cleaning
threshold           =     '12uJy'       #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =     'clark'       #  Method of PSF calculation to use during minor cycles
imagermode          =   'csclean'       #  Options: 'csclean' or 'mosaic', '', uses psfmode
     cyclefactor    =        1.5        #  Controls how often major cycles are done. (e.g. 5 for frequently)
     cyclespeedup   =         -1        #  Cycle threshold doubles in this number of iterations

multiscale          = [0, 2, 5, 8, 15, 50, 100] #  Deconvolution scales (pixels); [] = standard clean
     negcomponent   =         -1        #  Stop cleaning if the largest scale finds this number of neg components
     smallscalebias =        0.6        #  a bias to give more weight toward smaller scales

interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              =       1280        #  x and y image size in pixels. Single value: same for both
cell                =   '1arcsec'       #  x and y cell size(s). Default unit arcsec.
phasecenter         = 'J2000 13h29m52.2s +47d12m30s' #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
```

erroneous values in red

Help on Tasks

In-line help:

`help clean` (or **`pdoc clean`**)

xterm

```
Help on clean task:

Invert and deconvolve images with selected algorithm
       The clean task has many options:

        1)  Make 'dirty' image and 'dirty' beam (psf)
        2)  Multi-frequency-continuum images or spectral channel imaging
        3)  Full Stokes imaging
        4)  Mosaicking of several pointings
        5)  Multi-scale cleaning
        6)  Widefield cleaning
        7)  Interactive clean boxing
        8)  Use starting model (eg from single dish)


      vis -- Name(s) of input visibility file(s)
              default: none;
              example: vis='ngc5921.ms'
                       vis=['ngc5921a.ms','ngc5921b.ms']; multiple MSes
      imagename -- Pre-name of output images:
              default: none; example: imagename='m2'
              output images are:
                m2.image: cleaned and restored image
                       With or without primary beam correction
                m2.psf: point-spread function (dirty beam)
                m2.flux:  relative sky sensitivity over field
                m2.flux.pbcoverage:  relative pb coverage over field
                                     (gets created only for ft='mosaic')
                m2.model: image of clean components
```

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.

```
clean(vis='input.ms', imagename='galaxy',selectvis=T,
  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

Measurement Set

- CASA stores u-v data in directories called "Measurement Sets"
  TO DELETE THEM USE rmtables("my_table.gcal")

- 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.

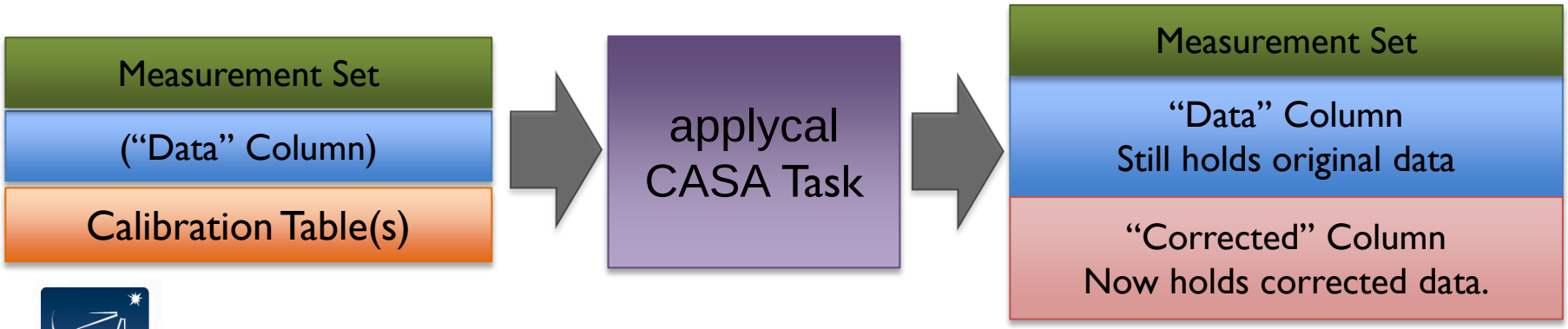
Calibration Tables

- 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")

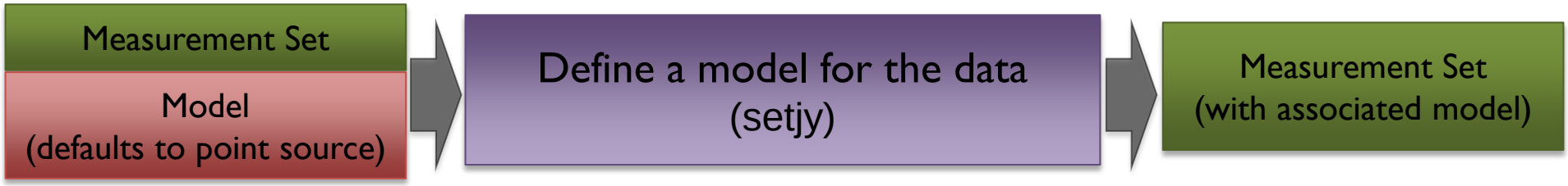
- These are created by calibration tasks:
  E.G., gaincal, bandpass, gencal

- Applied via "applycal" to the data column and saved as corrected.



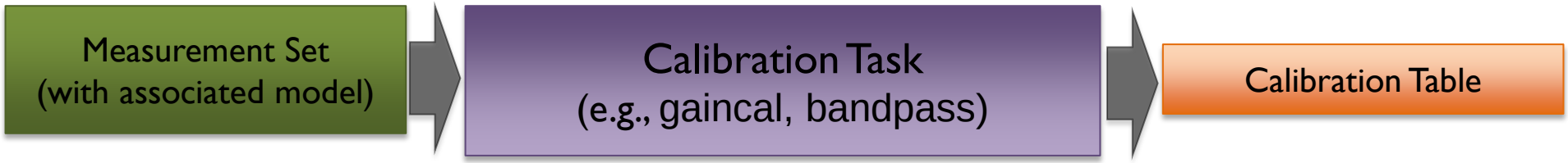
Basic Calibration Flow

Define what the telescope SHOULD have seen.



Basic Calibration Flow

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

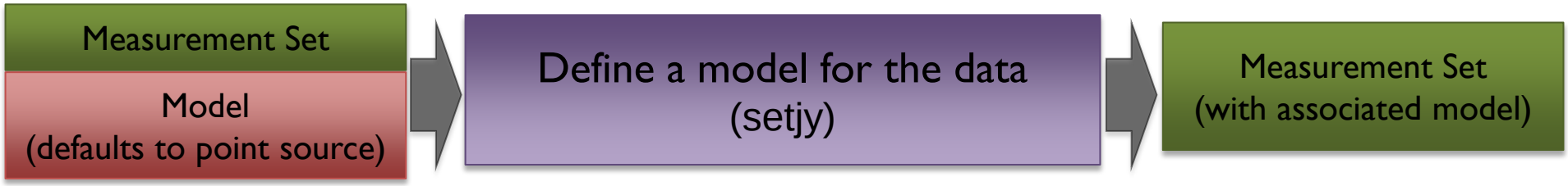


Basic Calibration Flow

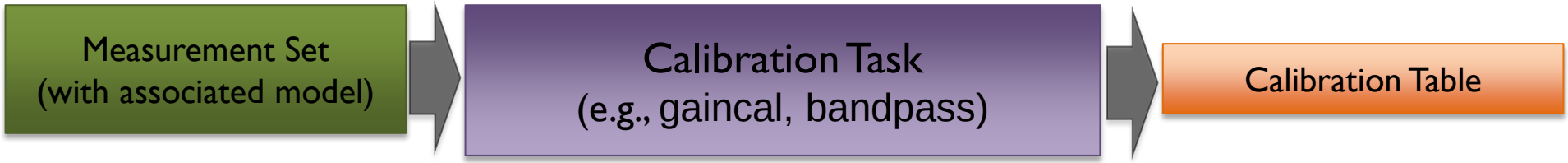
Apply these corrections to derive the corrected (calibrated) data.

Basic Calibration Flow

Define what the telescope SHOULD have seen.



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



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

Outline

- Short introduction to CASA and the Python interface
  - Discussion of “tools” will be done separately
- The Flow of Calibration
- **Key CASA tasks for data reduction/calibration**
- Data Inspection and Flagging
- Basic Imaging

ALMA Online Calibration

- 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

- Water Vapor Radiometer (WVR) – phase delay due to atmosphere
  - Key to correct short-timescale phase variations
  - Phase calibration, variable with time

These are provided by the observatory (eventually applied online).

- Apply them as first step (or start with provided pre-applied versions)
- In either case, inspect these tables to learn about data quality
- ***The datasets associated with this tutorial already have these corrections applied***
- Make sure reference antenna is "well-behaved"

Possible Flagging and Calibration Recipe

- (Apply online calibrations for water vapor and Tsys)
- EXAMINE bandpass/flux calibrator(s)
- FLAG bandpass/flux calibrators
- APPLY bandpass/flux calibration to itself

Iterate

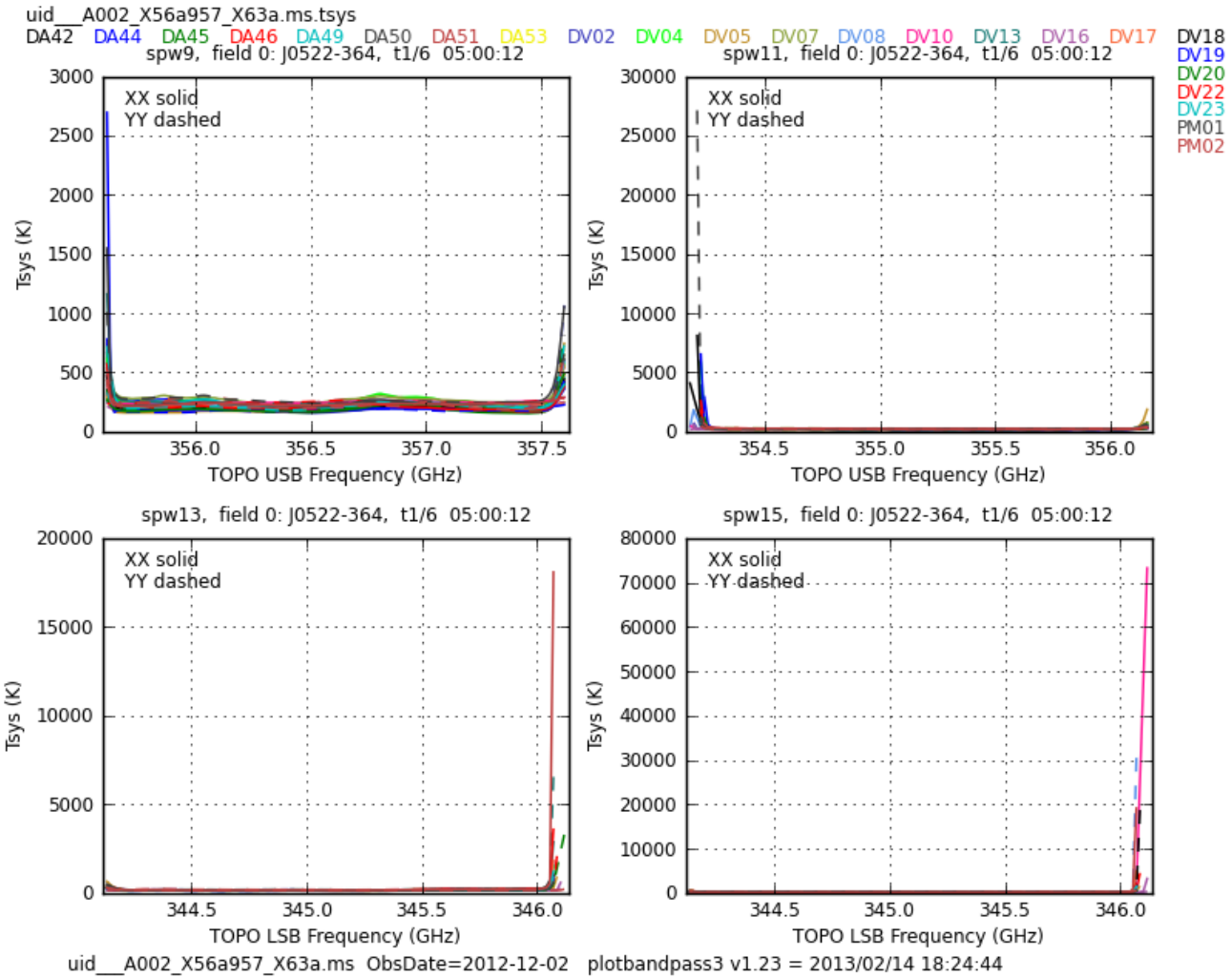
- APPLY bandpass/flux cal to gain cal sources
- EXAMINE gain calibration sources
- FLAG gain calibration sources
- APPLY gain calibration to itself

Iterate

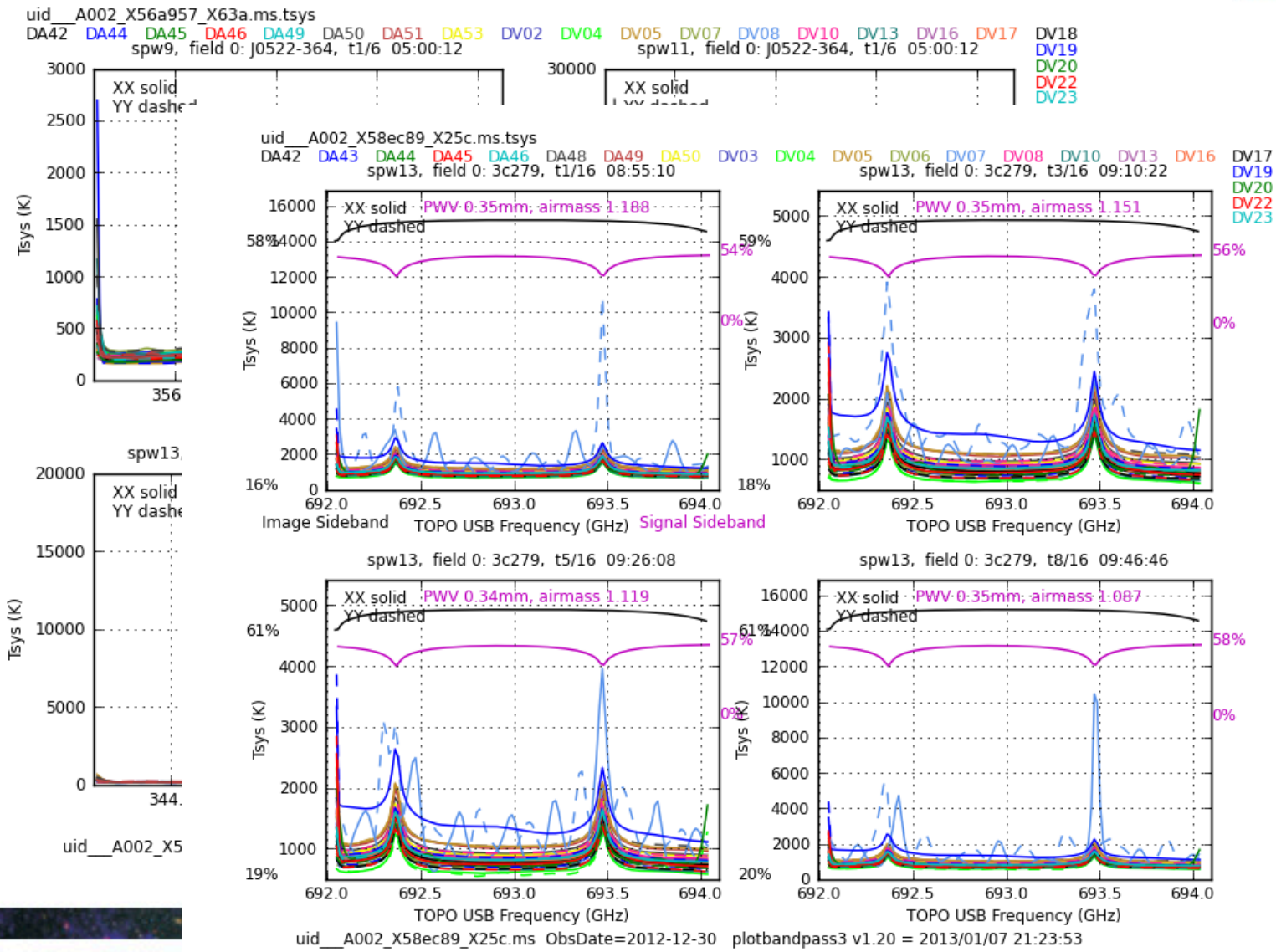
- APPLY bandpass/flux/gain cal to targets
- EXAMINE targets
- FLAG targets

Repeat as necessary

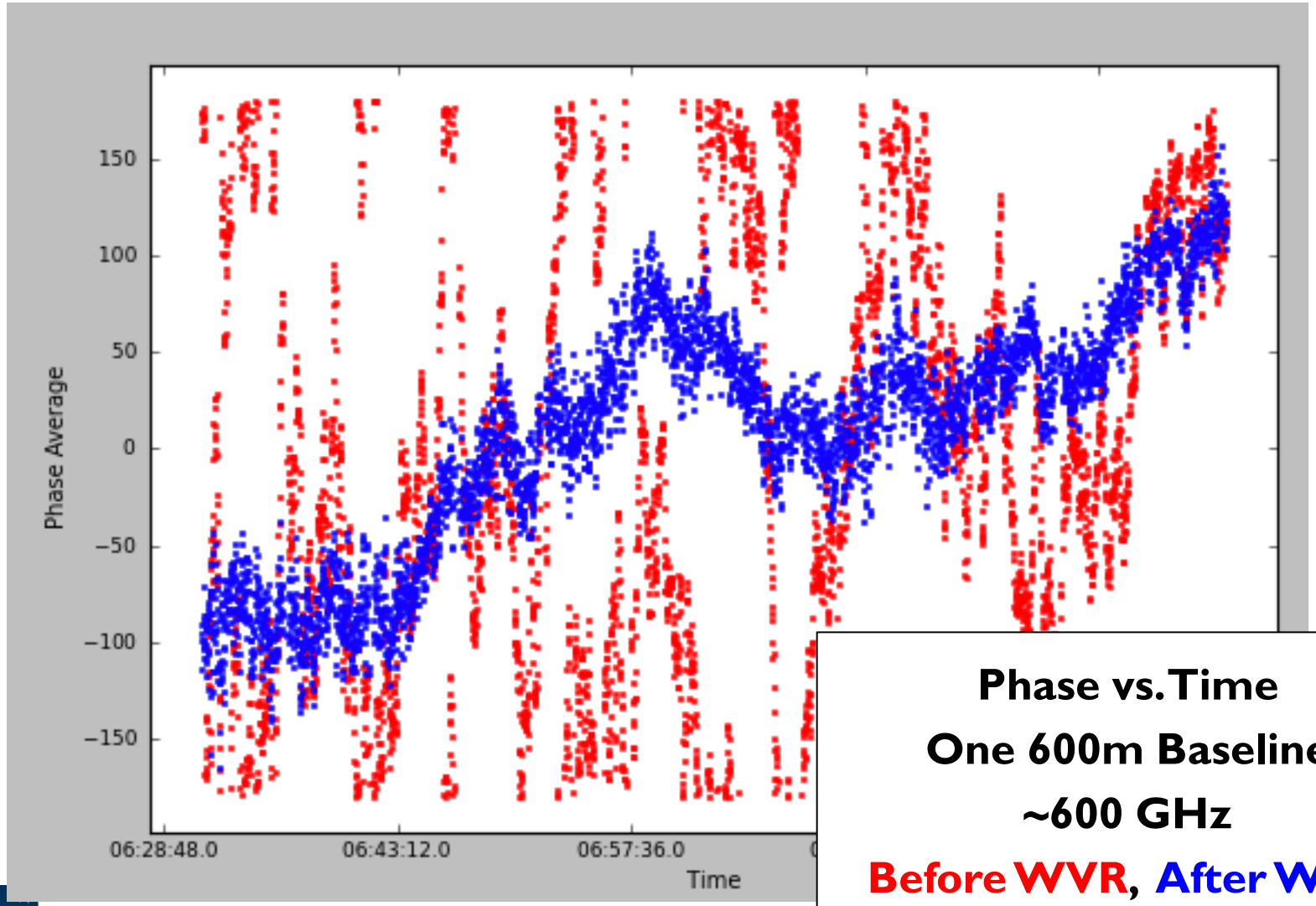
ALMA Online Calibration



ALMA Online Calibration



ALMA Online Calibration



Phase vs. Time

One 600m Baseline

~600 GHz

Before WVR, After WVR

Key Tasks for Calibration

CASA
Common Astronomy Software Applications

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

NRAO

Schematic Calibration

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

Schematic Calibration

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.

“My first task” and Orientation to the Data

ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

May 16, 2014

Atacama Large Millimeter/submillimeter Array

Expanded Very Large Array

Robert C. Byrd Green Bank Telescope

Very Long Baseline Array

My first task…

Let's get started with the data reduction…

Under the sis14 directory you will find a directory called /lessons. Inside /lessons there is a README file that outlines the directory structure for the entire tutorial.

We are going to work directly under the /lessons directory step by step so if a dataset gets corrupted, you can easily copy over a new copy from the /working_data directory. That is why every script starts with a:

```
os.system("rm -rf sis14_twhya_uncalibrated.ms")
```

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

```
casapy
```

Copy the data over from the working directory:

```
os.system("cp -r ../../working_data/sis14_twhya_uncalibrated.ms .")
```

My first task

View a list of available tasks

```
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
importfits	listcal	flagmanager	hanningsmooth
importfitsidi	listfits	msview	imhead
importuvfits	listhistory	plotms	msmoments
importvla	listobs	plotxy	partition
(importevla)	listpartition		plotms
(importgmrt)	listvis		plotxy
	plotms		split
	plotuv		testconcat
	plotxy		uvcontsub
	vishead		virtualconcat
	visstat		vishead
	(asdmsummary)		(mstransform)

.
.
.

My First Task

View available inputs to the listobs task

```
inp listobs
```

```
#  listobs :: List the summary of a data set in the logger or in a file
vis                 =         ''       #  Name of input visibility file (MS)
selectdata          =       True       #  Data selection parameters
    field           =         ''       #  Field names or field index numbers: ''==>all, field='0~2,3C286'
    spw             =         ''       #  spectral-window/frequency/channel
verbose             =       True
listfile            =         ''       #  Name of disk file to write output: ''==>to terminal
listunfl            =      False       #  List unflagged row counts? If true, it can have significant negative performance impact.
cachesize           =         50       #  EXPERIMENTAL. Maximum size in megabytes of cache in which data structures can be held.
async               =      False       #  If true the taskname must be started using listobs(...)
```

Set the visibility and review the modified inputs

```
vis = 'sis14_twhya_uncalibrated.ms'
inp listobs
```

```
#  listobs :: List the summary of a data set in the logger or in a file
vis                 = 'sis14_twhya_uncalibrated.ms' #  Name of input visibility file (MS)
selectdata          =       True       #  Data selection parameters
    field           =         ''       #  Field names or field index numbers: ''==>all, field='0~2,3C286'
    spw             =         ''       #  spectral-window/frequency/channel
    antenna         =         ''       #  antenna/baselines: ''==>all, antenna ='3,VA04'
    timerange       =         ''       #  time range: ''==>all,timerange='09:14:0~09:54:0'
                                                                    .
                                                                    .
                                                                    .
```

My First Task

Run the listobs task

go

Log Messages (multivac32:/lustre/naasc/nbrunett/sis14/lessons/first_task/casapy-20140506-152315.log) (on multivac32)

File Edit View

Search Message: Filter: Time

```
Tim Priority Origin   Message
    INFO    …::::casa+ ##########################################
    INFO    …::::casa+ ##### Begin Task: listobs            #####
    INFO    …s::::casa listobs(vis="sis14_twhya_uncalibrated.ms",selectdata=True,spw="",field="",
    INFO    …::::casa+         antenna="",uvrange="",timerange="",correlation="",scan="",
    INFO    …::::casa+         intent="",feed="",array="",observation="",verbose=True,
    INFO    …::::casa+         listfile="",listunfl=False,cachesize=50)
    INFO    …::summary ================================================================================
    INFO    …:summary+            MeasurementSet Name:  /lustre/naasc/nbrunett/sis14/lessons/first_task/sis14_twhya_uncalibrated.ms      MS Version 2
    INFO    …:summary+ ================================================================================
    INFO    …:summary+    Observer: cqi     Project: uid://A002/X327408/X6f
    INFO    …:summary+ Observation: ALMA
    INFO    …::summary Data records: 113282       Total integration time = 5641.63 seconds
    INFO    …:summary+    Observed from   19-Nov-2012/07:37:00.0   to   19-Nov-2012/09:11:01.6 (UTC)
    INFO    …::summary
    INFO    …:summary+    ObservationID = 0         ArrayID = 0
    INFO    …:summary+   Date        Timerange (UTC)          Scan  FldId FieldName             nRows     SpwIds   Average Interval(s)    ScanIntent
    INFO    …:summary+   19-Nov-2012/07:36:57.0 - 07:39:13.1     4      0 J0522-364               5060   [0]   [6.05] [CALIBRATE_BANDPASS#ON_SOURCE, CALIBRATE_PHASE#ON_SOURCE, CAL
    INFO    …:summary+                07:44:45.2 - 07:47:01.2     7      2 Ceres                   5060   [0]   [6.05] [CALIBRATE_AMPLI#ON_SOURCE, CALIBRATE_PHASE#ON_SOURCE, CALIBRA
    INFO    …:summary+                07:52:42.0 - 07:53:47.6    10      3 J1037-295               2760   [0]   [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
    INFO    …:summary+                07:56:23.5 - 08:02:11.3    12      5 TW Hya                 13754   [0]   [6.05] [OBSERVE_TARGET#ON_SOURCE]
    INFO    …:summary+                08:04:36.3 - 08:05:41.9    14      3 J1037-295               3000   [0]   [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
    INFO    …:summary+                08:08:09.6 - 08:13:57.3    16      5 TW Hya                 13639   [0]   [6.05] [OBSERVE_TARGET#ON_SOURCE]
    INFO    …:summary+                08:16:20.6 - 08:17:26.2    18      3 J1037-295               3000   [0]   [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
    INFO    …:summary+                08:19:53.9 - 08:25:41.7    20      5 TW Hya                 13572   [0]   [6.05] [OBSERVE_TARGET#ON_SOURCE]
    INFO    …:summary+                08:28:17.1 - 08:29:22.6    22      3 J1037-295               3000   [0]   [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
    INFO    …:summary+                08:32:00.5 - 08:37:48.2    24      5 TW Hya                 14788   [0]   [6.05] [OBSERVE_TARGET#ON_SOURCE]
    INFO    …:summary+                08:40:11.9 - 08:41:17.4    26      3 J1037-295               3000   [0]   [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
    INFO    …:summary+                08:43:45.6 - 08:49:33.4    28      5 TW Hya                 14952   [0]   [6.05] [OBSERVE_TARGET#ON_SOURCE]
```

Insert Message: Lock scro

My First Task

Instead, send output to a file

```
listobs(vis="sis14_twhya_uncalibrated.ms", listfile="my_listfile.txt")
```

System commands can be run from within casapy

```
os.system(“ls”)
```

casapy-20140506-152315.log first_script.py ipython-20140506-152318.log listobs.last
my_first_task.py sis14_twhya_uncalibrated.ms
 Out[12]: 0

```
os.system("more my_listfile.txt")
```

```
        MeasurementSet Name:  /lustre/naasc/nbrunett/sis14/lessons/first_task/sis14_twhya_uncalibrated.ms      MS Version 2
================================================================================
   Observer: cqi     Project: uid://A002/X327408/X6f
Observation: ALMA
Data records: 113282       Total integration time = 5641.63 seconds
   Observed from   19-Nov-2012/07:37:00.0   to   19-Nov-2012/09:11:01.6 (UTC)

   ObservationID = 0         ArrayID = 0
  Date        Timerange (UTC)          Scan  FldId FieldName             nRows     SpwIds   Average Interval(s)    ScanIntent
  19-Nov-2012/07:36:57.0 - 07:39:13.1     4      0 J0522-364               5060  [0]  [6.05] [CALIBRATE_BANDPASS#ON_SOURCE, CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
              07:44:45.2 - 07:47:01.2     7      2 Ceres                   5060  [0]  [6.05] [CALIBRATE_AMPLI#ON_SOURCE, CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
              07:52:42.0 - 07:53:47.6    10      3 J1037-295               2760  [0]  [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
              07:56:23.5 - 08:02:11.3    12      5 TW Hya                 13754  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
              08:04:36.3 - 08:05:41.9    14      3 J1037-295               3000  [0]  [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
              08:08:09.6 - 08:13:57.3    16      5 TW Hya                 13639  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
              08:16:20.6 - 08:17:26.2    18      3 J1037-295               3000  [0]  [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
              08:19:53.9 - 08:25:41.7    20      5 TW Hya                 13572  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
              08:28:17.1 - 08:29:22.6    22      3 J1037-295               3000  [0]  [6.05] [CALIBRATE_PHASE#ON_SOURCE, CALIBRATE_WVR#ON_SOURCE]
--More--(23%)
```

In this case, we will use os.system() instead of “!” because it is needed when scripting…

My First Task

first_script.py

```
# This is a comment


# A python command
print "Hello CASA!"


# A call to a system command
os.system("rm -rf my_script_listfile.txt")


# A CASA command
listobs(vis="sis14_twhya_uncalibrated.ms", listfile="my_script_listfile.txt")
```

Run it with execfile

```
execfile("first_script.py")
```

```
Hello CASA!
Writing output to file: my_script_listfile.txt
```

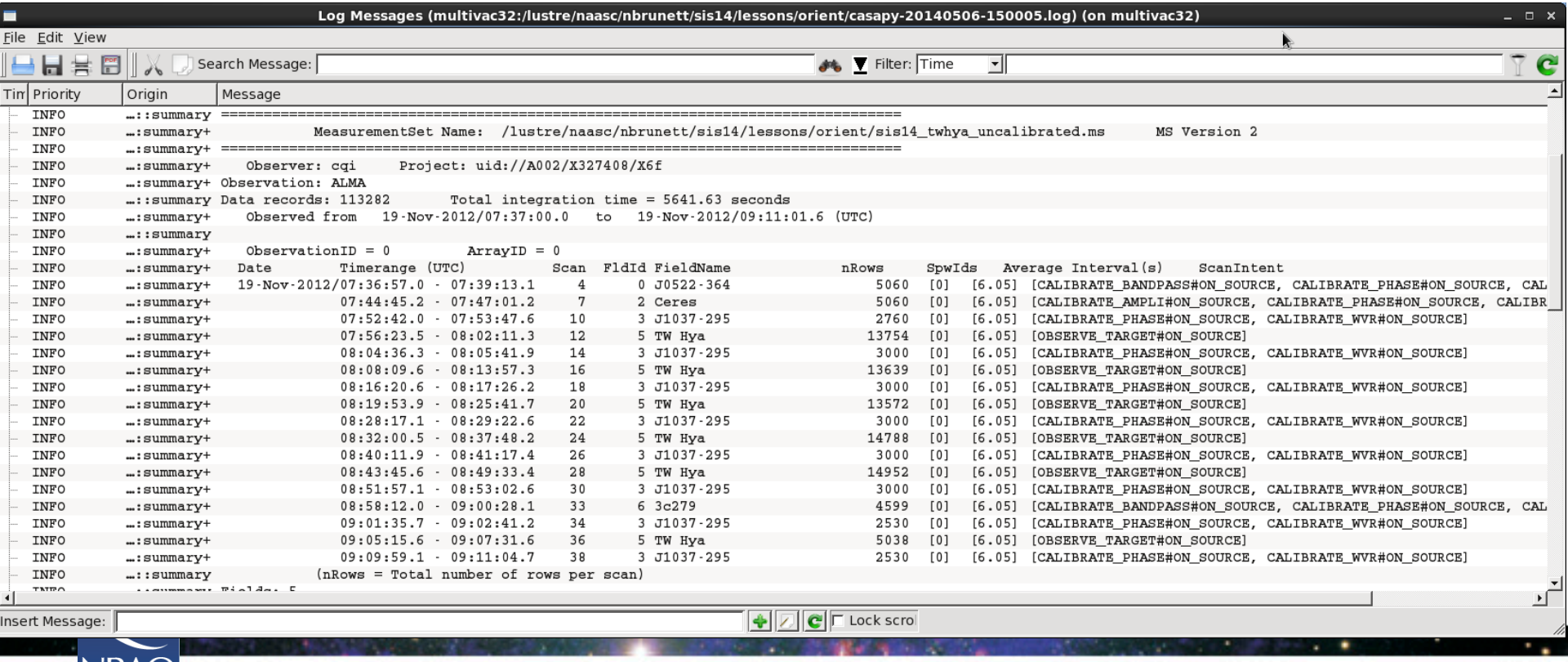
Getting Oriented

First, cd into the /orient directory and copy the data over for inspection:

```
os.system("cp -r ../../working_data/sis14_twhya_uncalibrated.ms .")
```

Run the listobs task

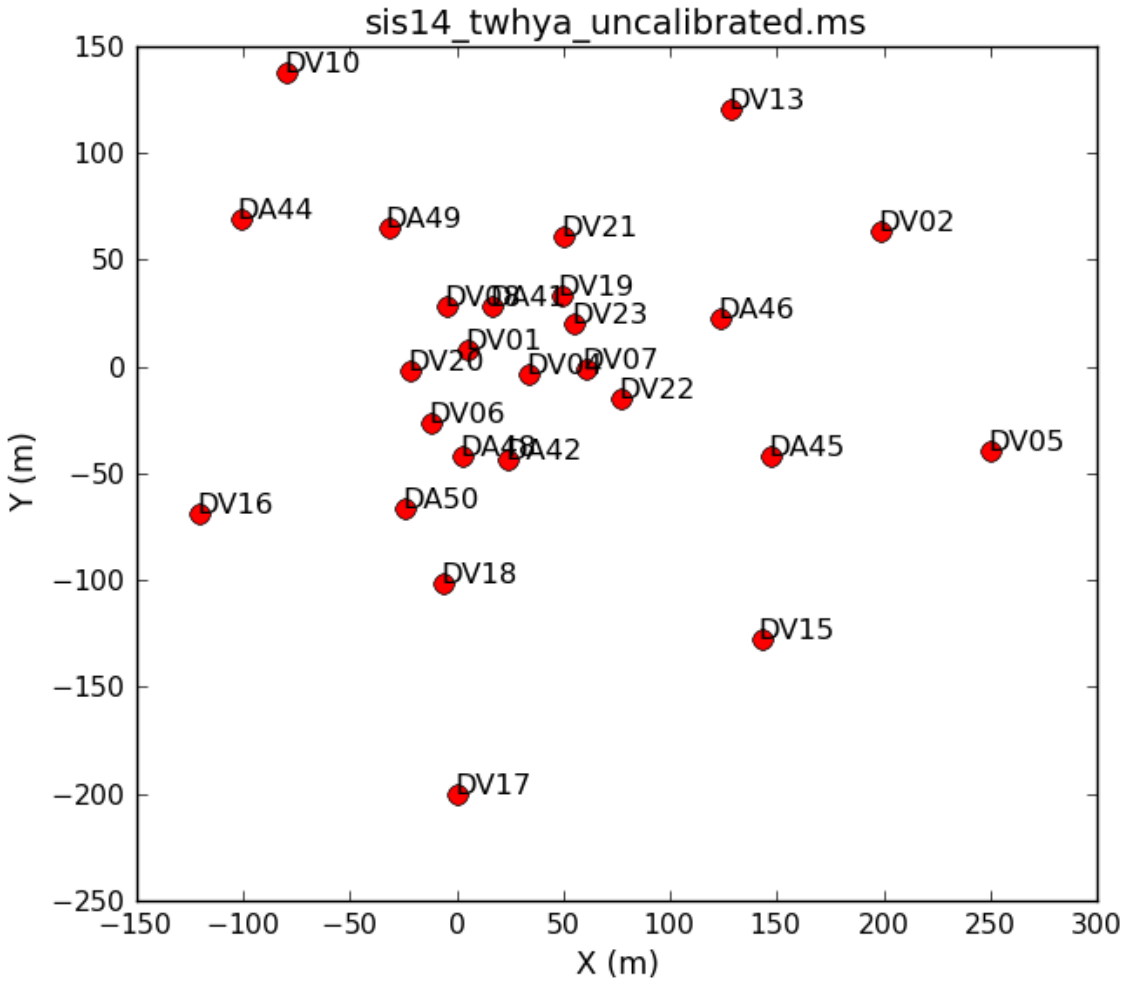
```
listobs("sis14_twhya_uncalibrated.ms")
```



Getting Oriented

Run the plotants task

```
plotants("sis14_twhya_uncalibrated.ms", figfile="plotants.png")
```



Getting Oriented

```
inp plotms
```

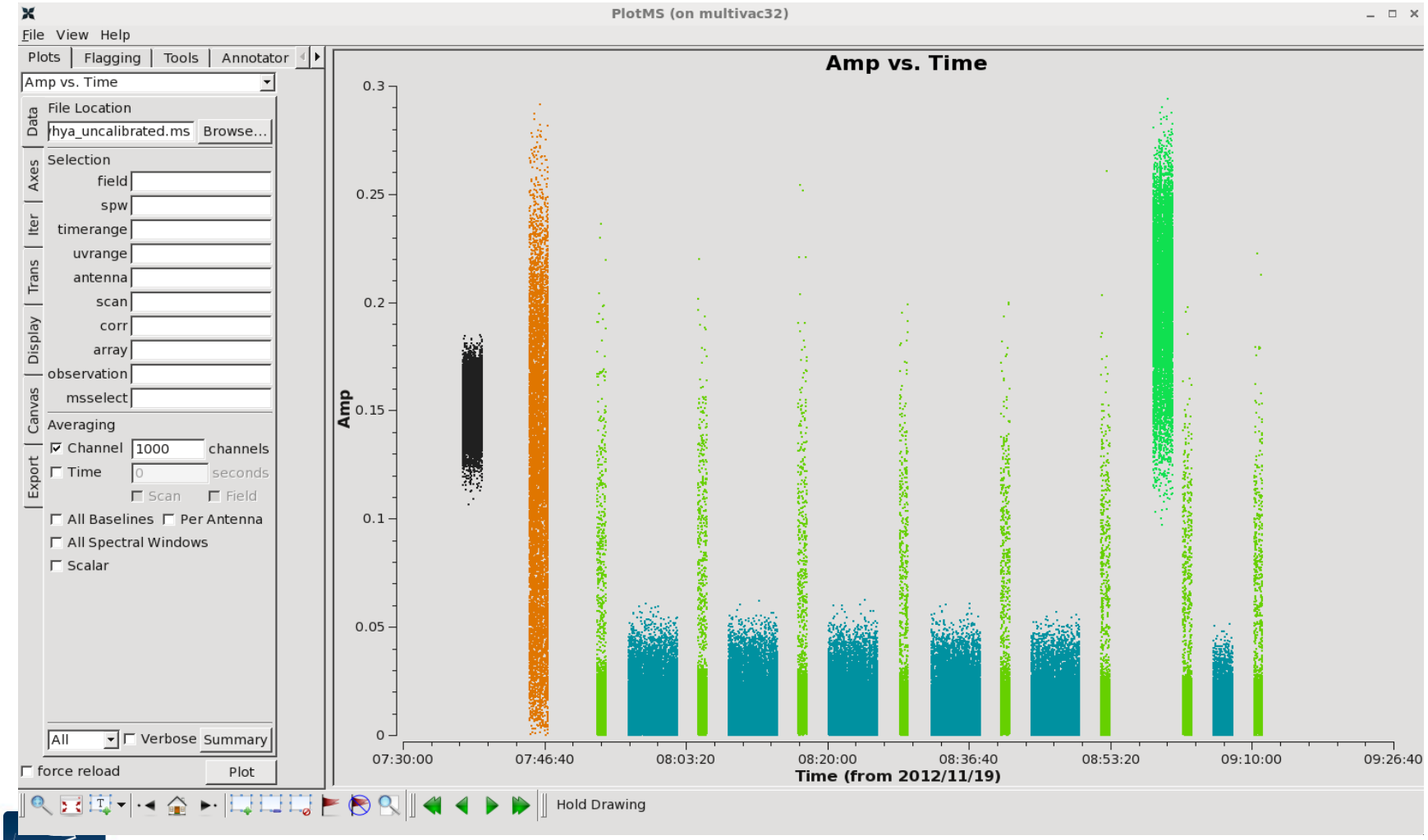
```
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 =         ''       #  input MS (or CalTable) (blank for none)
xaxis               =         ''       #  plot x-axis (blank for default/current)
yaxis               =         ''       #  plot y-axis (blank for default/current)
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     =         ''       #  Select by observation ID(s)
    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, other value in seconds)
    avgscan         =      False       #  only valid if time averaging is turned on.  average over scans?
    avgfield        =      False       #  only valid if time averaging is turned on.  average over fields?
    avgbaseline     =      False       #  average over all baselines?  (mutually exclusive with avgantenna)
    avgantenna      =      False       #  average by per-antenna?  (mutually exclusive with avgbaseline)
    avgspw          =      False       #  average over all spectral windows?
    scalar          =      False       #  Do scalar averaging?


transform           =      False       #  transform data in various ways?
extendflag          =      False       #  have flagging extend to other data points?
iteraxis            =         ''       #  the axis over which to iterate
customsymbol        =       True       #  set a custom symbol for unflagged points
    symbolshape     = 'autoscaling'     #  shape of plotted unflagged symbols
    symbolsize      =          2        #  size of plotted unflagged symbols
```

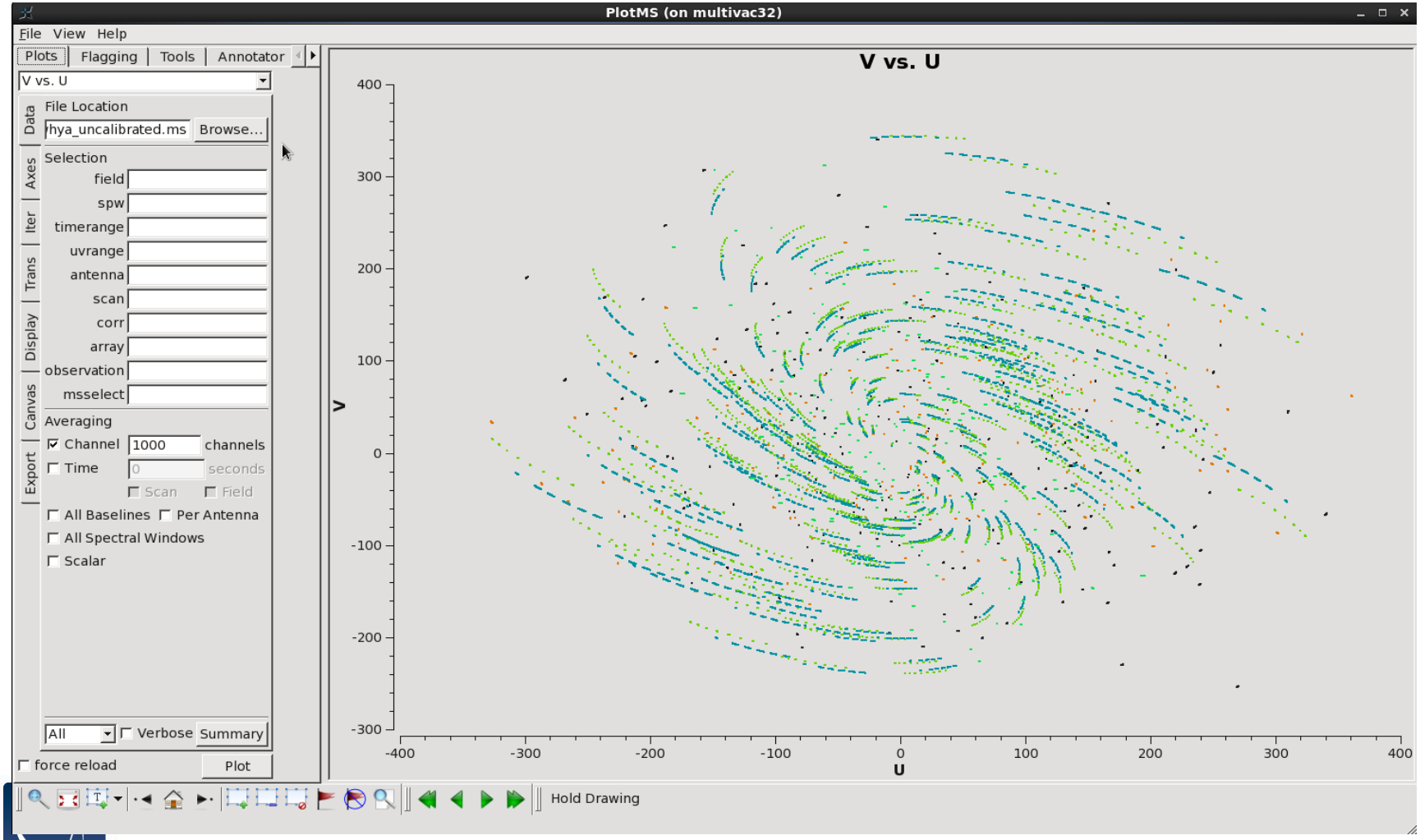
Getting Oriented

```
plotms(vis="sis14_twhya_uncalibrated.ms", xaxis="time", yaxis="amp", averagedata=T,
avgchannel="1e3", coloraxis="field")
```



Getting Oriented

```
plotms(vis="sis14_twhya_uncalibrated.ms", xaxis="time", yaxis="amp", averagedata=T,
avgchannel="1e3", coloraxis="field")
```



Bandpass, Phase and Amplitude Calibration

ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

May 16, 2014

Atacama Large Millimeter/submillimeter Array

Expanded Very Large Array

Robert C. Byrd Green Bank Telescope

Very Long Baseline Array

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.

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

Orient yourself with a listobs

Run a listobs and note the bandpass calibrators. We have two, but will work with field 0 in this data set.

- `listobs("sis14_twhya_uncalibrated.ms")`

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.cal".

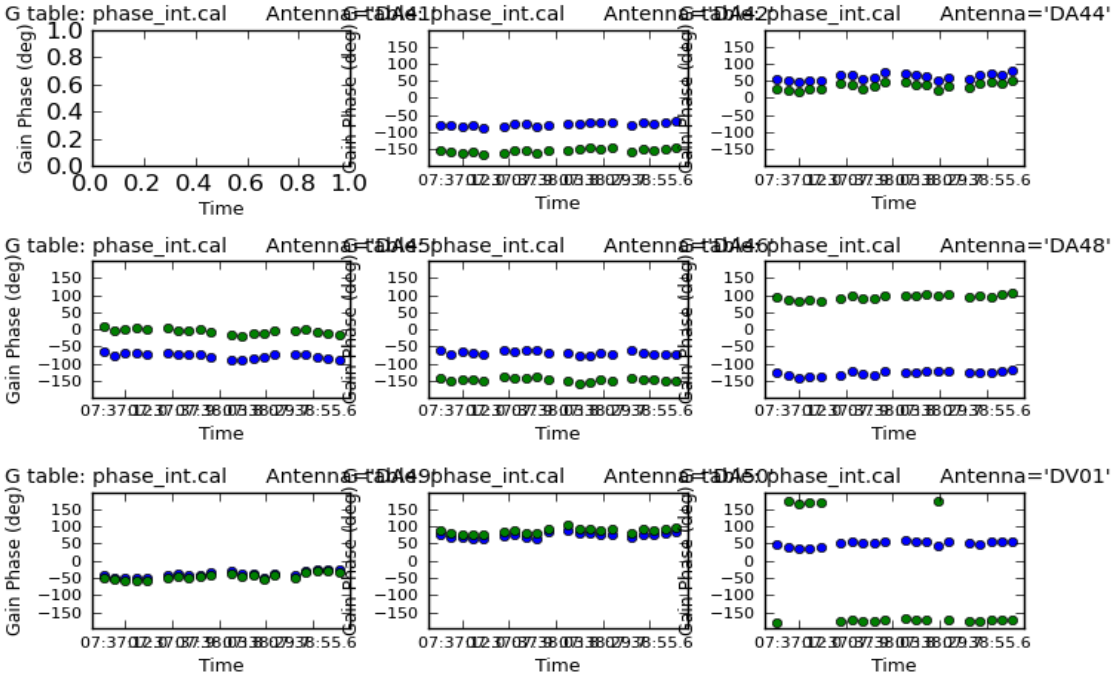
- `os.system("rm -rf phase_int.cal")`
- ```
  gaincal(vis="sis14_twhya_uncalibrated.ms",
      caltable="phase_int.cal",
      field="0",
      solint="int",
      calmode="p",
      refant="DV22",
      gaintype="G")
  ```


  ```

Plot phase vs. time

Now we 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.cal", xaxis="time",yaxis="phase", subplot=331, iteration="antenna", plotrange=[0,0,-180,180])`



First 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.

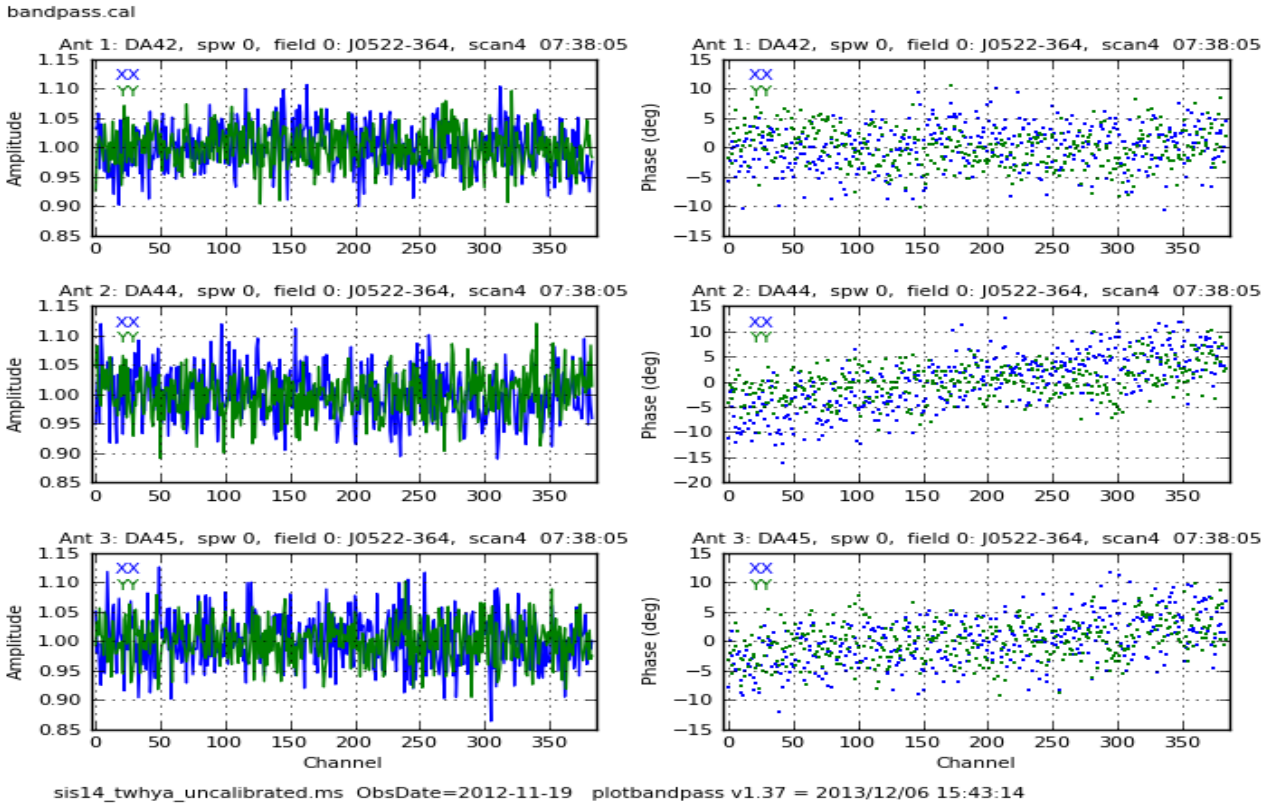
- `os.system("rm -rf bandpass.cal")`
- ```
  bandpass(vis="sis14_twhya_uncalibrated.ms",
       caltable="bandpass.cal",
       field="0",
       refant="DV22",
       solint="inf",
       combine="scan",
       solnorm=True,
       gaintable=["phase_int.cal"])
  ```


  ```

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 three antennas at a time. Cycle through the plots.

- `plotbandpass(caltable="bandpass.cal", xaxis="chan", yaxis="both",  subplot=32)`



Calibrate the bandpass

Notice how noisy the solutions are. We can also calibrate the bandpass but average 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 by setting a solution interval of 10 channels.

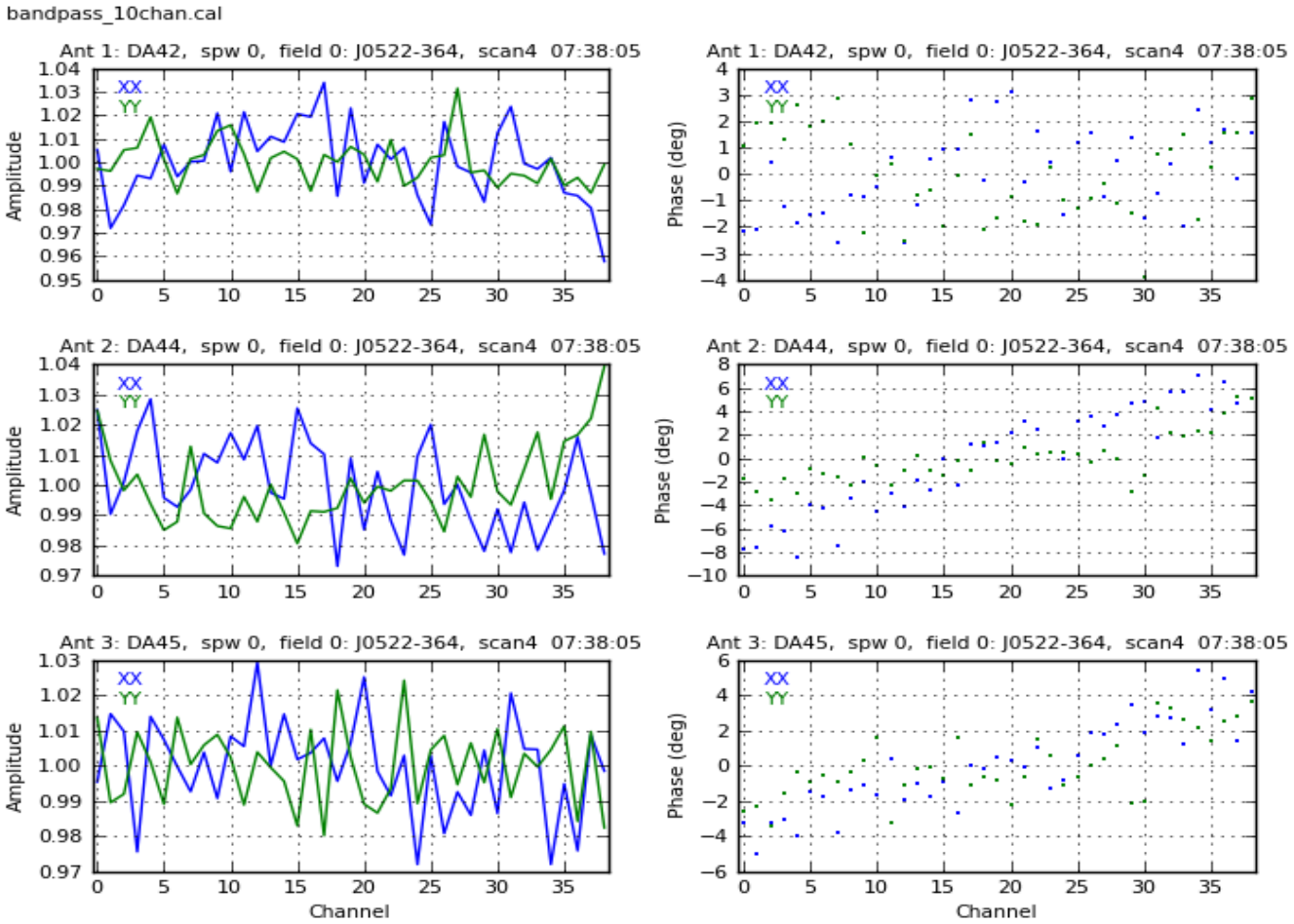
- `os.system("rm -rf bandpass_10chan.cal")`
- ```
  bandpass(vis="sis14_twhya_uncalibrated.ms",
       caltable="bandpass_10chan.cal",
       field="0",
       refant="DV22",
       solint="inf,10chan",
       combine="scan",
       solnorm=True,
       gaintable=["phase_int.cal"])
  ```


  ```

Plot solutions

Now plot these. There are less points and they are less noisy in absolute scale. Both tables seemed fine, but we will use these.

- `plotbandpass(caltable="bandpass_10chan.cal", xaxis="chan", yaxis="both", subplot=32)`



Apply the solutions

Apply the solutions - both in time and frequency - to the data using applycal. This creates a new corrected data column. Note that we will only apply these to field 0 at first and then look at the effects.

- ```
  applycal(vis="sis14_twhya_uncalibrated.ms",
       field="0",
       gaintable=["bandpass_10chan.cal", "phase_int.cal"],
       interp=["linear","linear"],
       Gainfield=["0","0"])
  ```

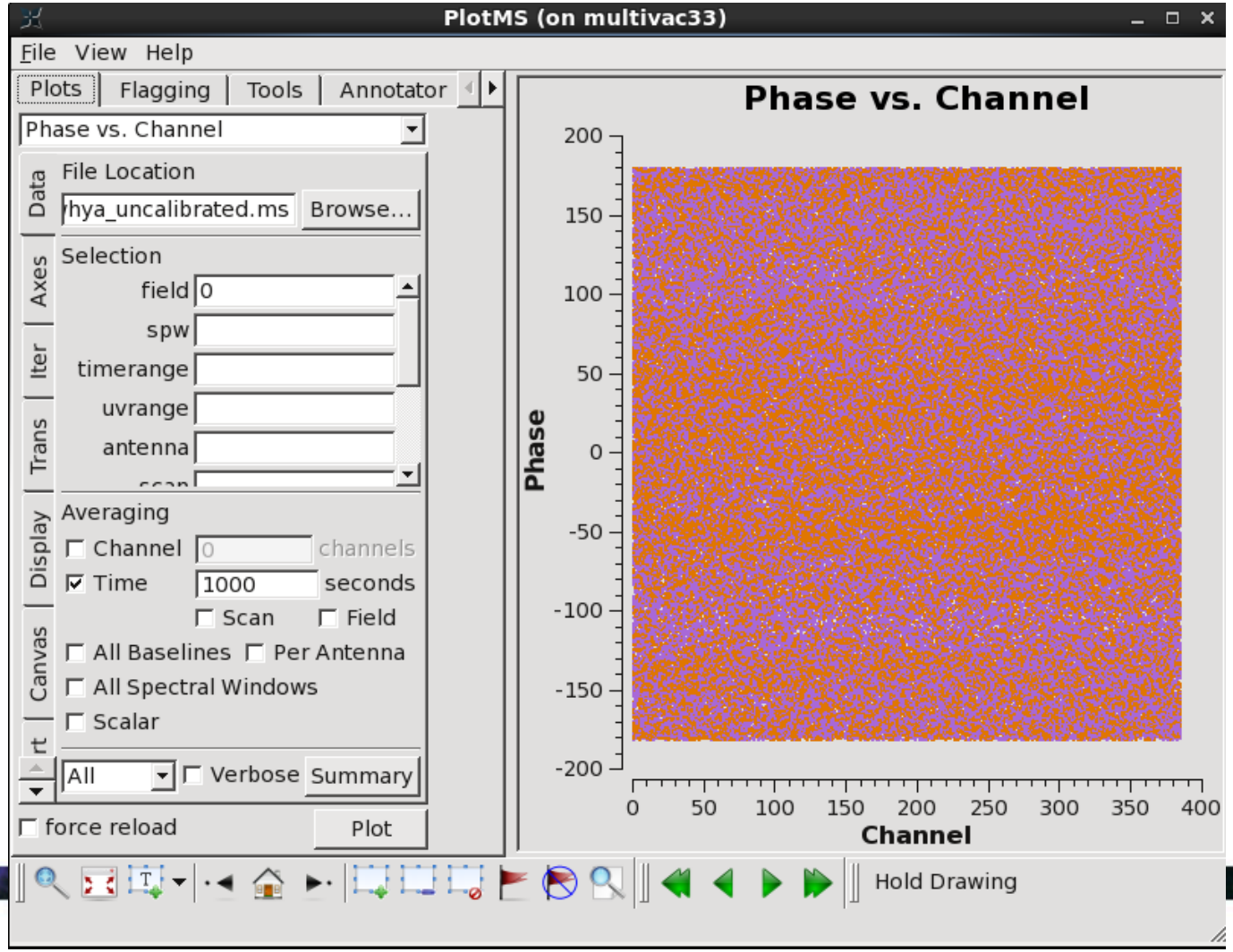
Plot the results of the calibration by comparing the dependence of phase and amplitude on channel before and after calibration.

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.


  ```

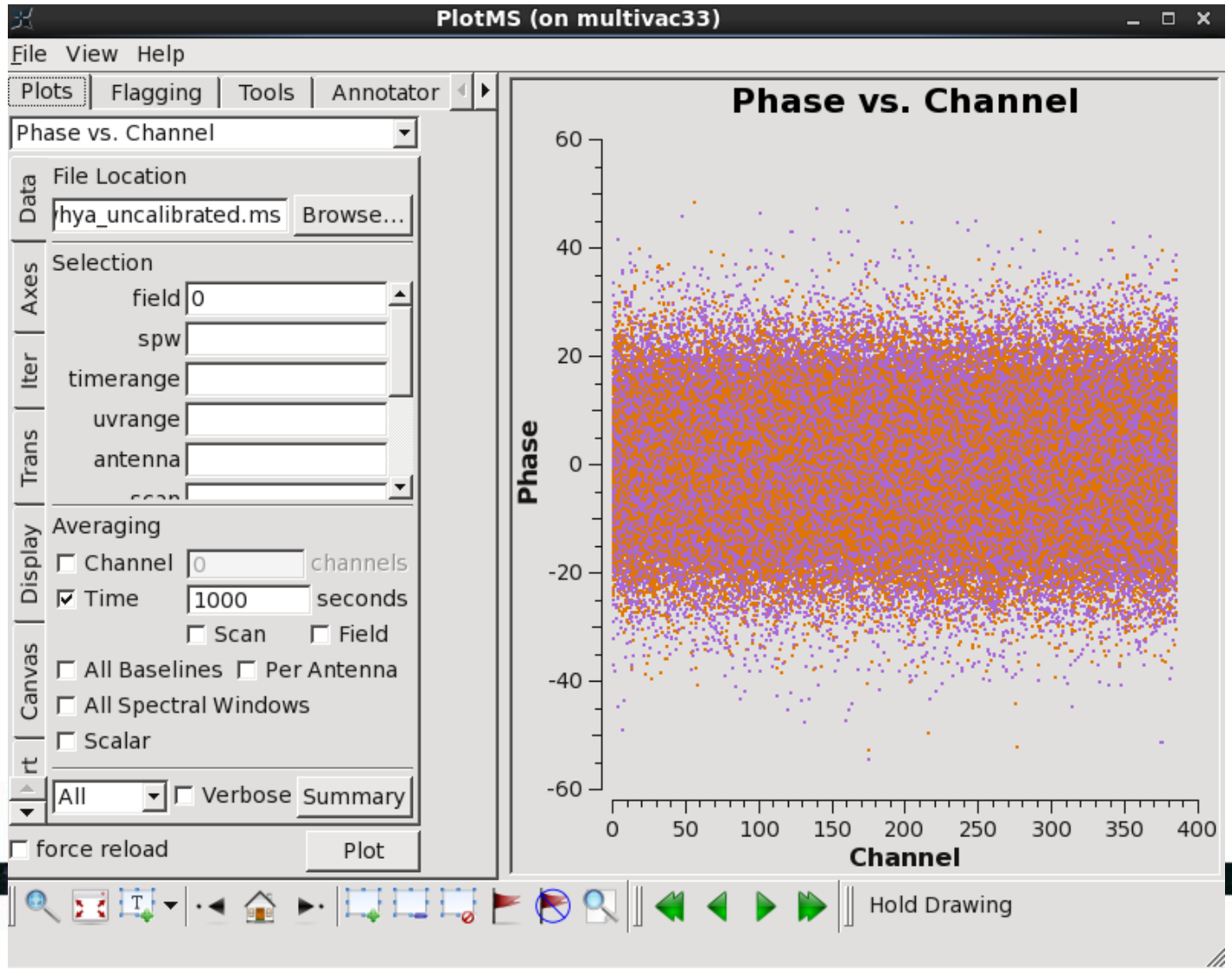
Phase vs Channel before

```
plotms(vis="sis14_twhya_uncalibrated.ms", xaxis="chan", yaxis="phase",  ydatacolumn="data",
    field="0", averagedata=T, avgtime="1e3", coloraxis="corr")
```



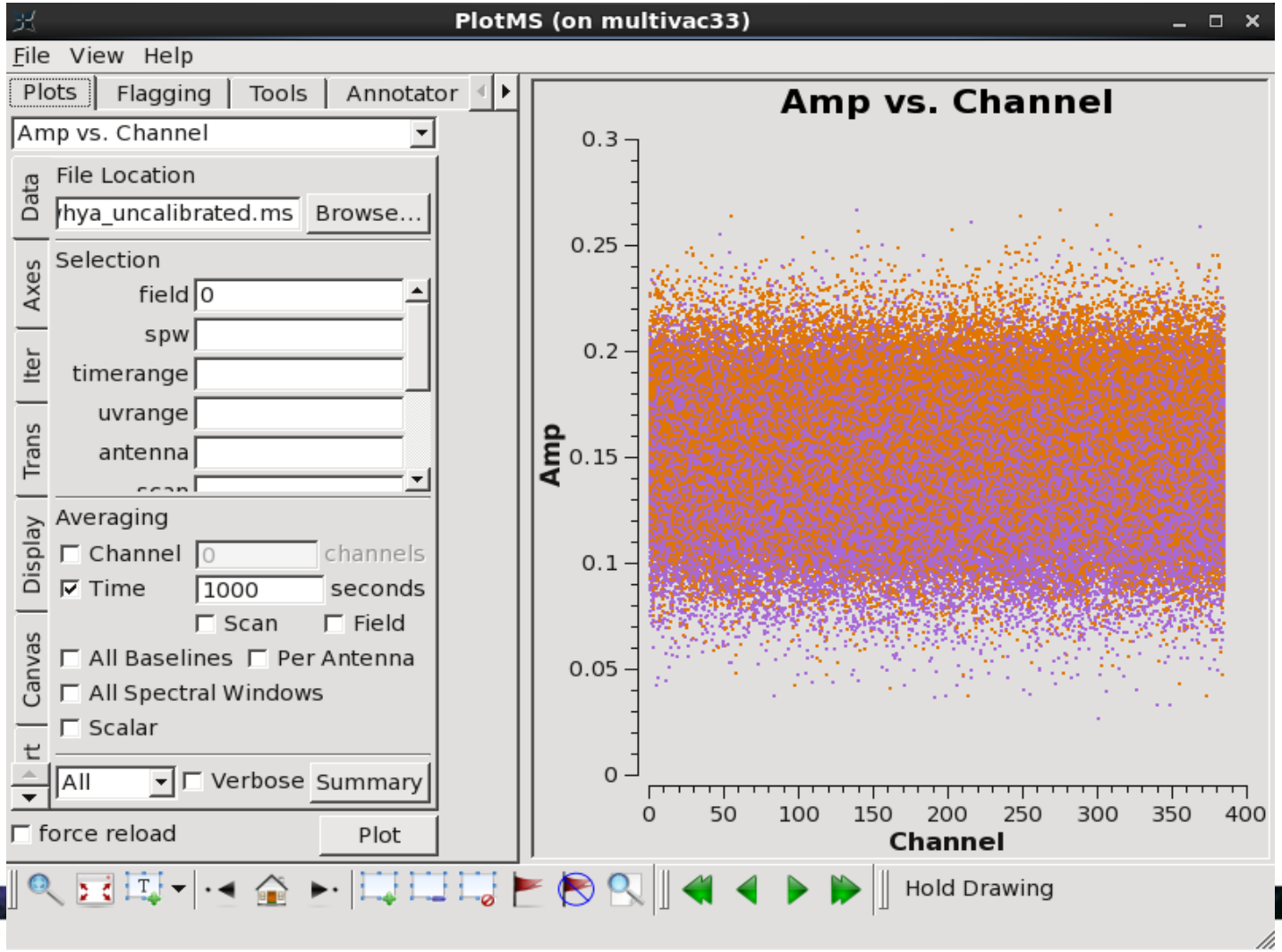
Phase vs Channel after

```
plotms(vis="sis14_twhya_uncalibrated.ms", xaxis="chan", yaxis="phase",
    ydatacolumn="corrected", field="0", averagedata=T, avgtime="1e3", coloraxis="corr")
```



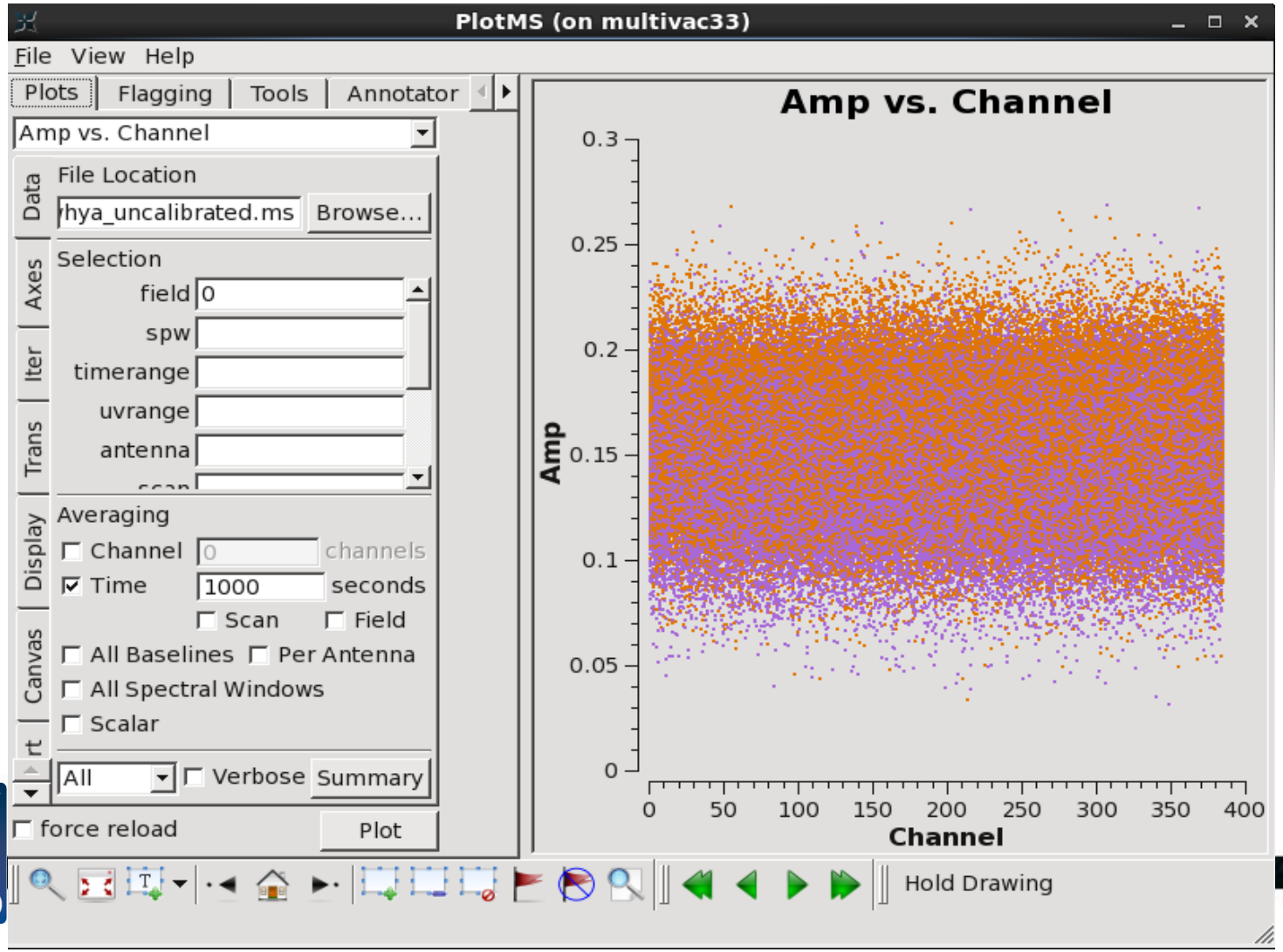
Amp vs. Chan before

- `plotms(vis="sis14_twhya_uncalibrated.ms", xaxis="chan", yaxis="amp", ydatacolumn="data",field="0", averagedata=T, avgtime="1e3",coloraxis="corr")`



Amp vs. Chan after

- `plotms(vis="sis14_twhya_uncalibrated.ms", xaxis="chan", yaxis="amp", ydatacolumn="corrected",field="0", averagedata=T, avgtime="1e3",coloraxis="corr")`



Apply bandpass solutions to the whole data set

Note a couple things.

1. This will overwrite the previous corrected data for the bandpass calibrator.
2. Without the time-dependent gain factor applied the plotms plots above will not necessarily work as well (they do okay, but that's just lucky and good quality data).

We now thing that we have removed frequency dependent effects from the whole data set and will proceed with a time-dependent calibration.

Note that we use the non-standard "calonly" command, which tells applycal not to flag data for which the calibration has failed.

```
applycal(vis="sis14_twhya_uncalibrated.ms",
    field="",
    gaintable=["bandpass_10chan.cal"],
    interp=["linear"],
    gainfield=["0"],
    applymode="calonly")
```

Split out calibrated data

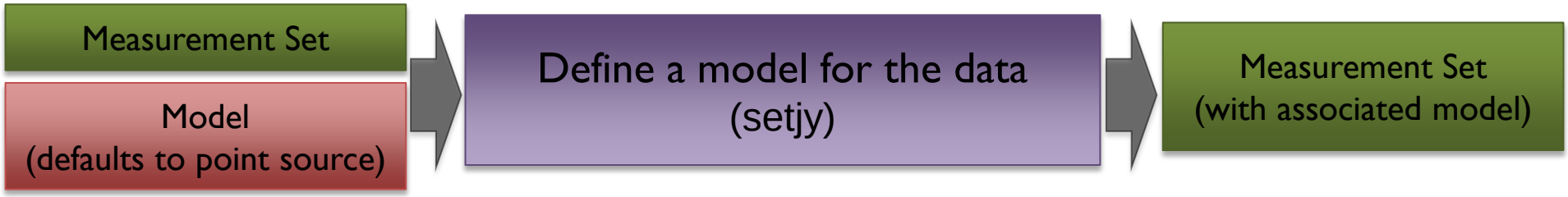
Now that we are satisfied with the bandpass calibration, we split out the bandpass calibrated data for further processing.

```
os.system("rm -rf sis14_twhya_bpcal.ms")
split(vis="sis14_twhya_uncalibrated.ms",
  datacolumn="corrected",
  outputvis="sis14_twhya_bpcal.ms",
  keepflags=False)
```

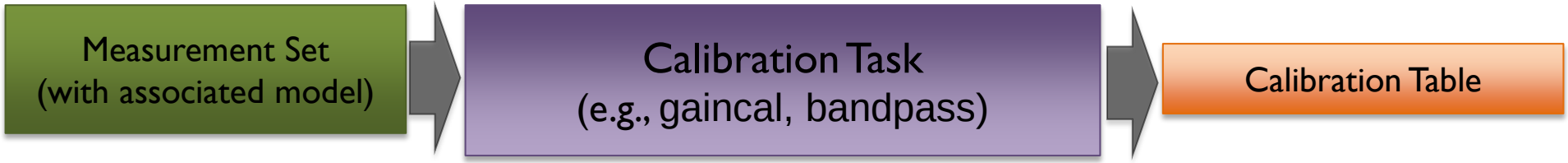
This produces one of the supplied data products (sis14_twhya_bpcal.ms) - so you can restart from a successful version of this script anytime

Reminder: Basic Calibration Flow

Define what the telescope SHOULD have seen.



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



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

Schematic Calibration

DONE

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna

ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE TIME VARIABILITY

DONE

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 (MODELLED), MONITORED QUASAR, ETC.

Apply all corrections to produce calibrated data

CASA
Common Astronomy Software Applications

NRAO

Schematic Calibration

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.

Key Tasks for Calibration

CASA
Common Astronomy Software Applications

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

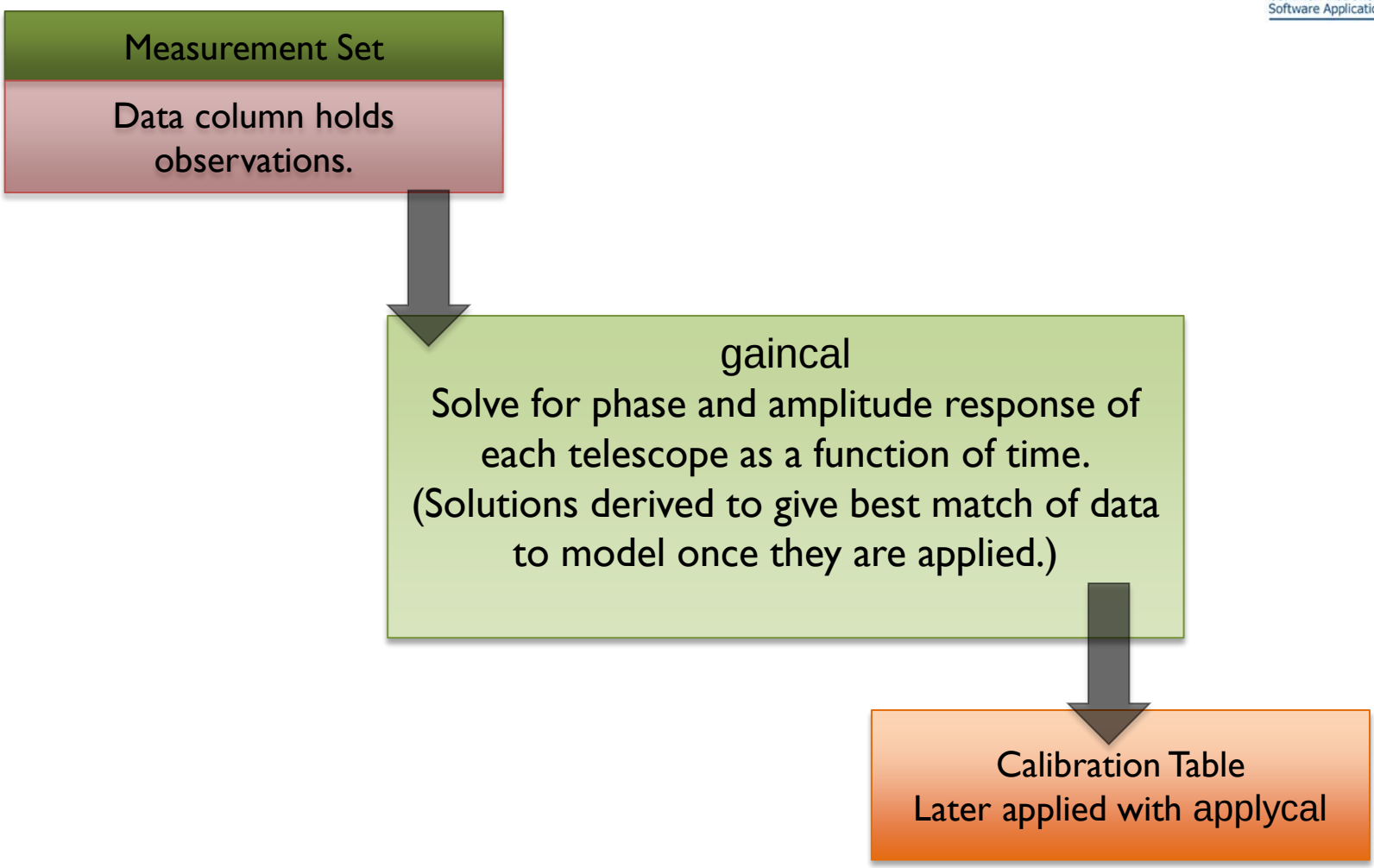
- flagdata/flagcmd/flagmanager: flag (remove) bad data
- applycal: apply calibration table(s) from previous steps
- split: split off calibrated data from your ms (for imaging!)

Inspect Your Data and Results

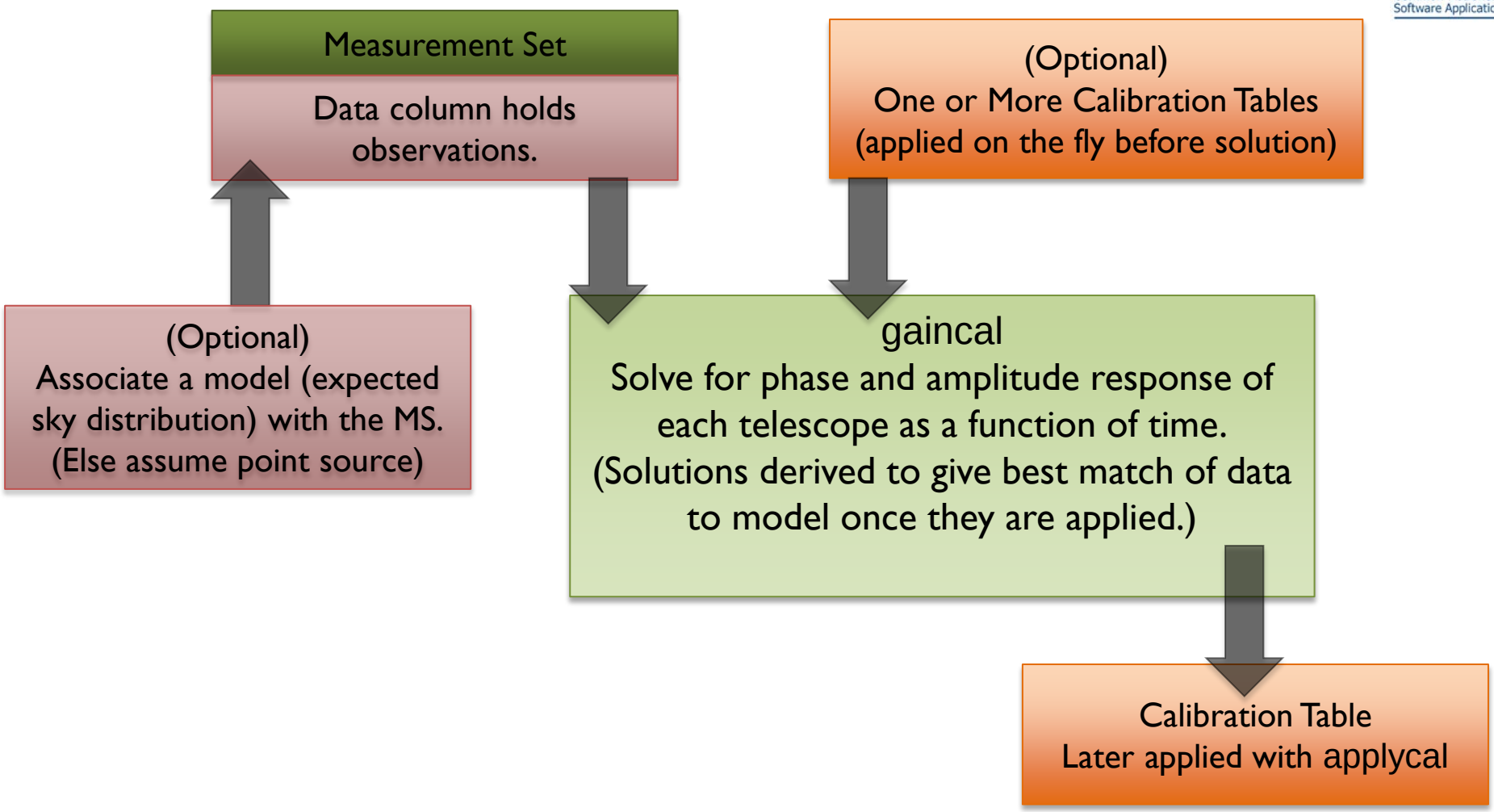
- plotms: inspect your data interactively
- plotcal: examine a calibration table

NRAO

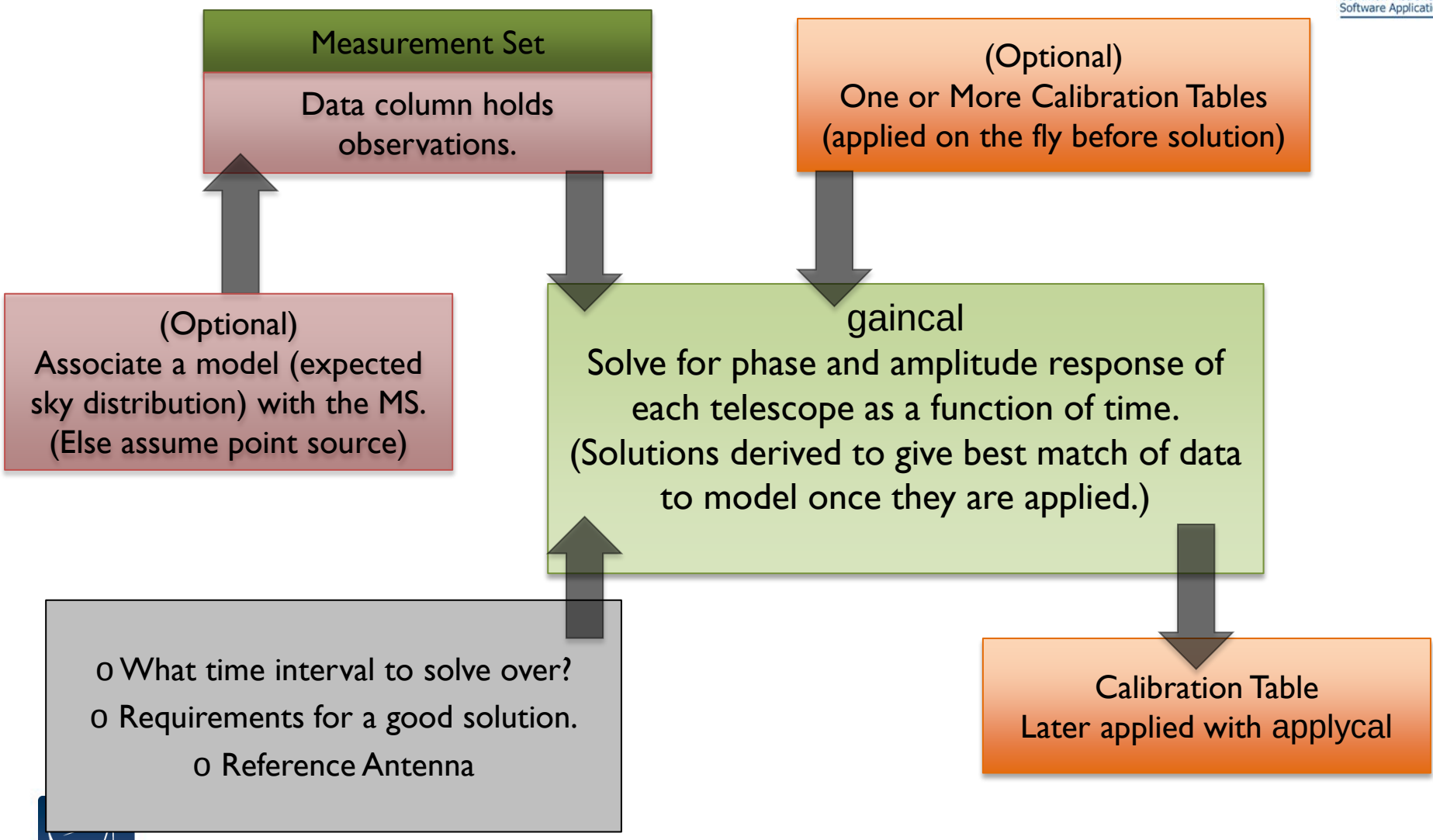
gaincal


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.)
o What time interval to solve over?
o Requirements for a good solution.
o Reference Antenna
Calibration Table
Later applied with applycal

Set Model for the Planet

First things first - we need to make sure that we have valid models in place for our data. We will work out the fluxes of the quasars later, for now it's good enough that we expect them to be point sources. However, the flux calibrator Ceres is somewhat resolved and we don't know the flux a priori. We need to read in a model from the solar system models that ship with CASA. We will use the task "setjy" and the library "Butler-JPL-Horizons 2012". With this call, we fill in the model column for Ceres.

- ```
  setjy(vis="sis14_twhya_bpcal.ms",
    field="2",
    standard="Butler-JPL-Horizons 2012",
    usescratch=True)
  ```


  ```

Phase Calibration

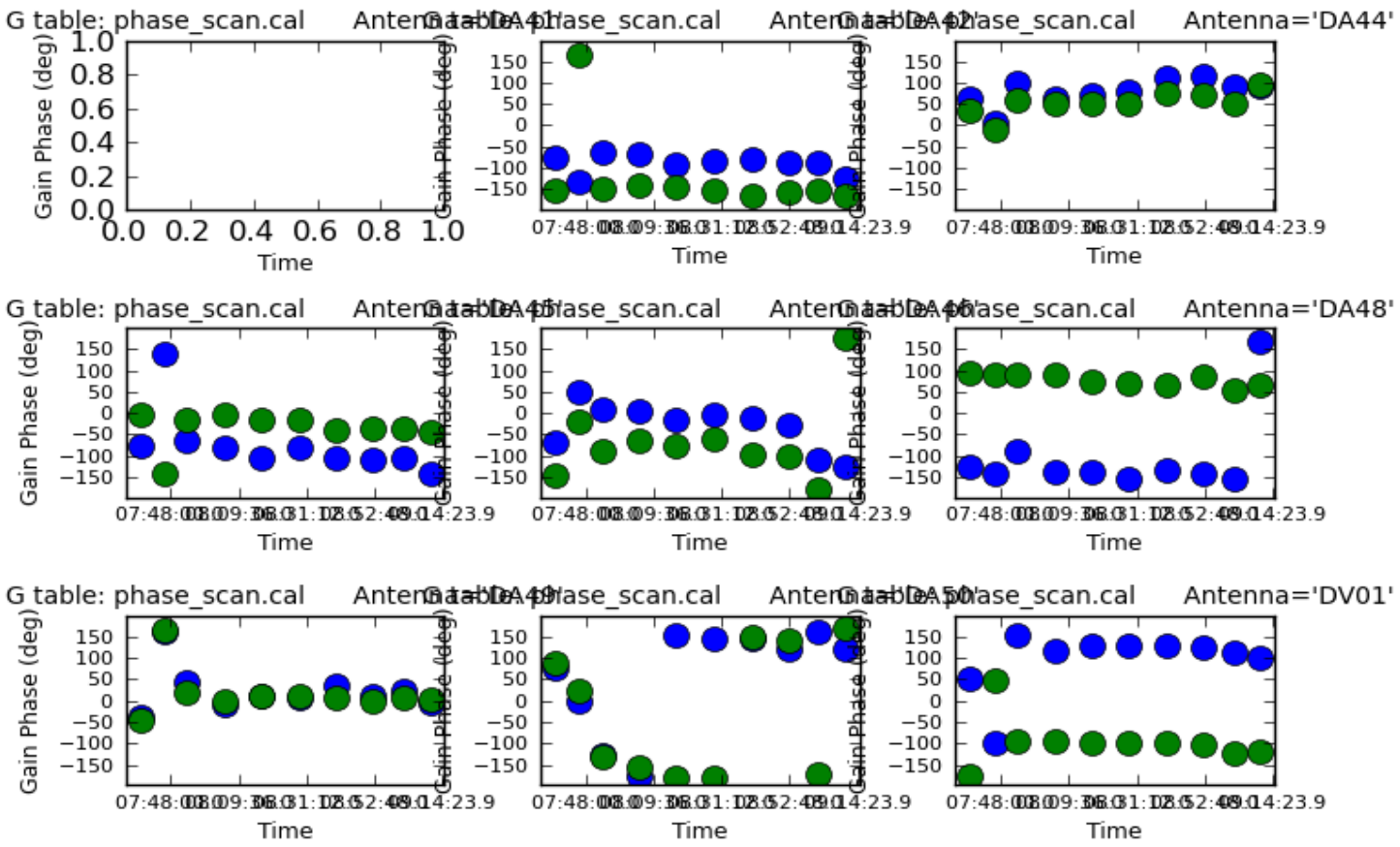
First, we calibrate the phase for each antenna for each scan. This is the right cadence to transfer to the science target, which is visited only on a ~ every-other-scan timescale.

- `os.system("rm -rf phase_scan.cal")`
- ```
  gaincal(vis="sis14_twhya_bpcal.ms",
      caltable="phase_scan.cal",
      field="0,2,3",
      solint="inf",
      calmode="p",
      refant="DV22",
      gaintype="G")
  ```


  ```

Plot resulting phase calibration

```
plotcal(caltable="phase_scan.cal",xaxis="time",yaxis="phase",subplot=331,iteration="antenna",
    plotrange=[0,0,-180,180],markersize=10, figfile="sis14_phase_scan.png")
```



Flux Calibration

Flux calibration requires estimating the flux of the secondary calibrator. We will get there by bootstrapping from the flux of the primary calibrator (in this case the planet Ceres), which is known from a well understood model.

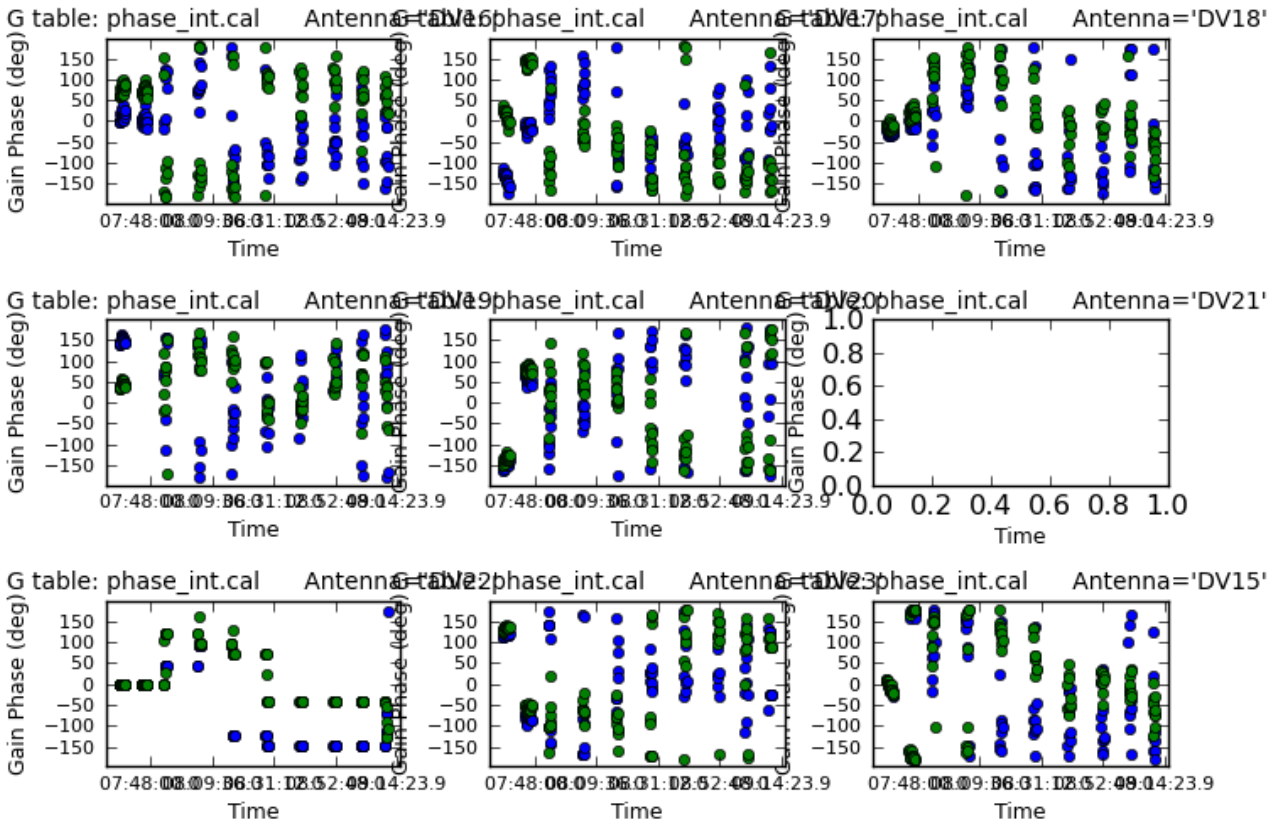
Before we begin, we want to remove any short timescale phase variation from the sources involved in the flux calibration. Do so using gaincal.

- `os.system("rm -rf phase_int.cal")`
- ```
  gaincal(vis="sis14_twhya_bpcal.ms",
      caltable="phase_int.cal",
      field="0,2,3",
      solint="int",
      calmode="p",
      refant="DV22",
      gaintype="G")
  ```


  ```

Plot on a short timescale phase calibration

```
plotcal(caltable="phase_int.cal",xaxis="time",yaxis="phase",subplot=331,
    iteration="antenna",plotrange=[0,0,-180,180], figfile="sis14_phase_int.png")
```

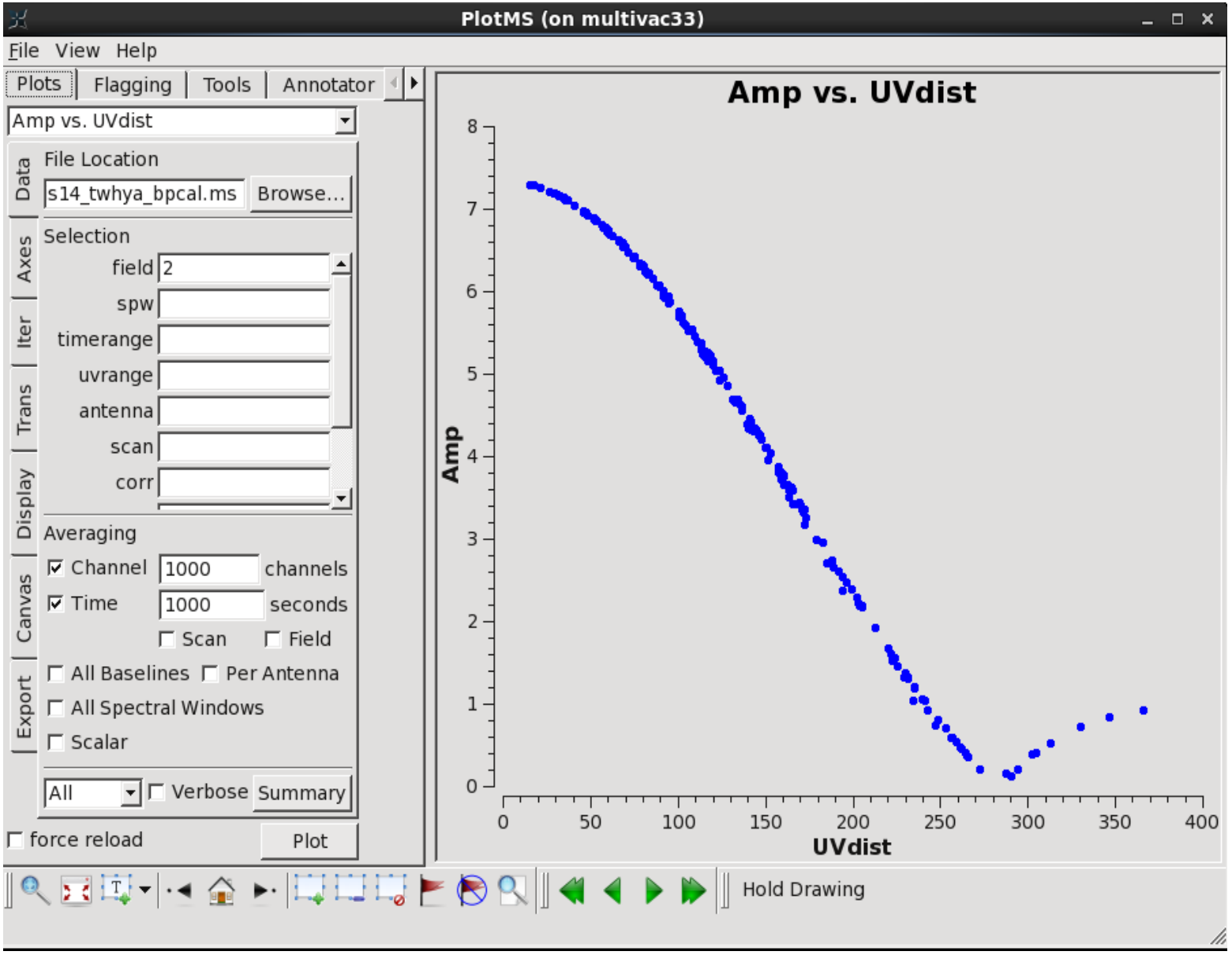


Check uv range of model

Our primary calibrator is a solar system body (Ceres). These can often be resolved, which can complicate any attempt to use them as calibrators. Best practice using these targets for flux calibration is to identify a subset of antennas or (more easily) a uv range over which the planetary disk shows a strong response. Look at the uv range of the model using plotms and try to identify such a range.

- ```
  plotms(vis="sis14_twhya_bpcal.ms",
     xaxis="uvdist",
     yaxis="amp",
     ydatacolumn="model",
     field="2",
     averagedata=T,
     avgchannel="1e3",
     avgtime="1e3")
  ```


  ```


PlotMS (on multivac33)
File View Help
Plots Flagging Tools Annotator
Amp vs. UVdist
Data
File Location
s14_twhya_bpcal.ms
Browse...
Axes
Iter
Trans
Display
Canvas
Export
Selection
field 2
spw
timerange
uvrange
antenna
scan
corr
Averaging
Channel 1000 channels
Time 1000 seconds
Scan Field
All Baselines Per Antenna
All Spectral Windows
Scalar
All Verbose Summary
force reload
Plot
Amp vs. UVdist
Amp
UVdist
0 1 2 3 4 5 6 7 8
0 50 100 150 200 250 300 350 400
Hold Drawing

Apply short timescale phase solutions

It looks like 0~150m is probably a good u-v range to be able to calibrate using Ceres. Now let's run an amplitude solution, first applying the short-timescale phase solution *only for this u-v range.

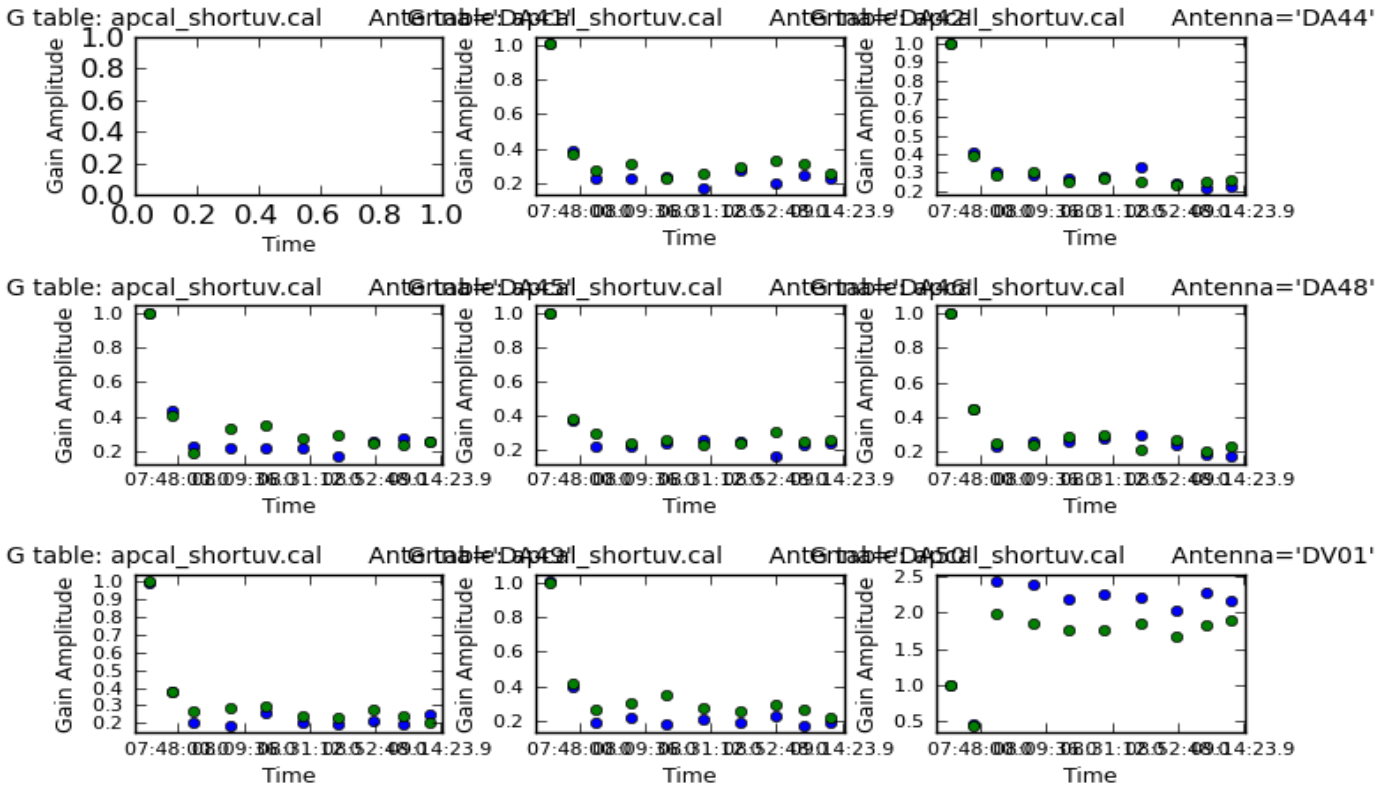
- `os.system("rm -rf apcal_shortuv.cal")`
- ```
  gaincal(vis="sis14_twhya_bpcal.ms",
      caltable="apcal_shortuv.cal",
      field="0,2,3",
      solint="inf",
      calmode="a",
      uvrange="0~150",
      gaintype="G",
      refant="DV22",
      gaintable="phase_int.cal")
  ```

Experiment and apply solutions over a longer or shorter uvrange and see what happens…


  ```

Plot calibration,amplitude vs. time for each antenna

- `plotcal(caltable="apcal_shortuv.cal", xaxis="time", yaxis="amp", subplot=331, iteration="antenna", plotrange=[0,0,0,0])`



Correct flux of calibrators

The gaincal solved for the amplitude scaling to make the data match the current model. For Ceres, 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 Ceres 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.

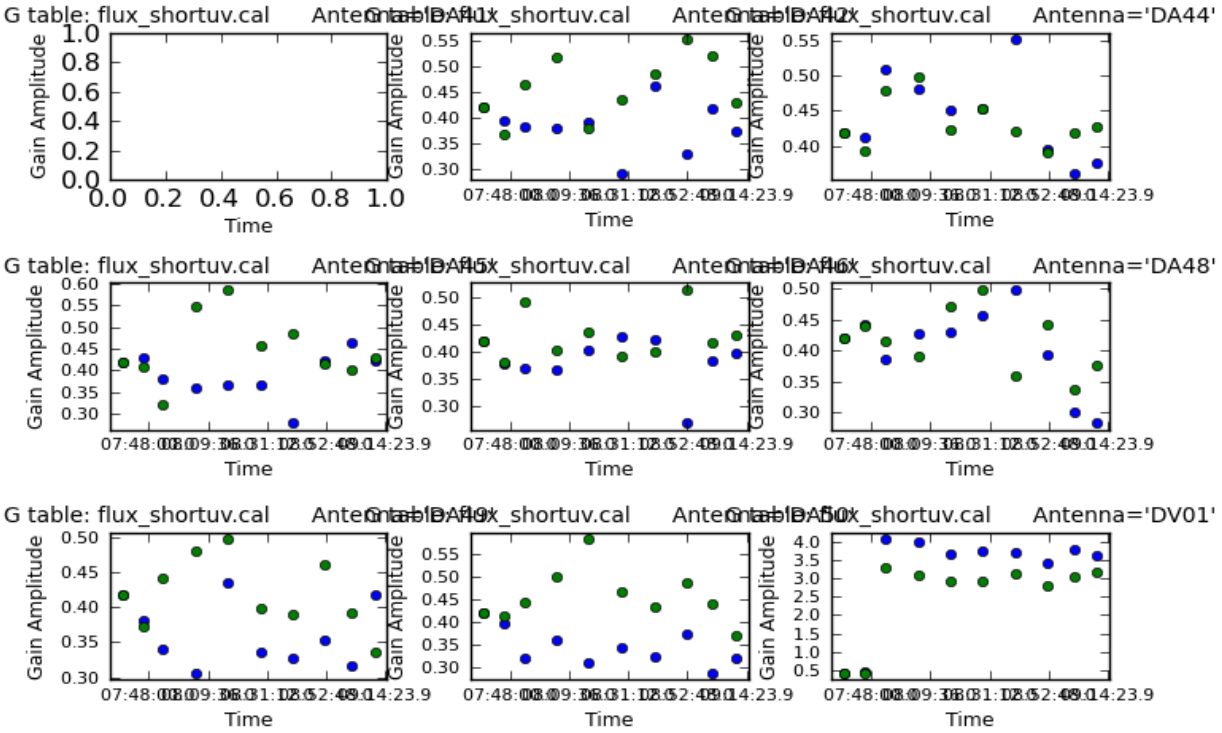
- `os.system("rm -rf flux_shortuv.cal")`
- ```
  fluxscale(vis="sis14_twhya_bpcal.ms",
        caltable="apcal_shortuv.cal",
        fluxtable="flux_shortuv.cal",
        reference="2")
  ```


  ```

Plot rescaled flux table

Plot the rescaled flux table, which now should contain the correct flux calibrations. It will not be our final amplitude table, though, because we only solved over short u-v distance baselines.

- `plotcal(caltable="flux_shortuv.cal", xaxis="time", yaxis="amp", subplot=331,iteration="antenna", plotrange=[0,0,0,0])`



Amplitude Calibration

From fluxscale, we see that we the two quasars have fluxes of ~0.65 and ~8.4 Jy. Using the task setjy, we will adjust the model of these sources to reflect these flux estimates

- ```
  setjy(vis="sis14_twhya_bpcal.ms",
    field="3",
    fluxdensity = [0.65,0,0,0],
    usescratch=True)
  ```

- ```
  setjy(vis="sis14_twhya_bpcal.ms",
    field="0",
    fluxdensity = [8.43,0,0,0],
    usescratch=True)
  ```

Amplitude solution

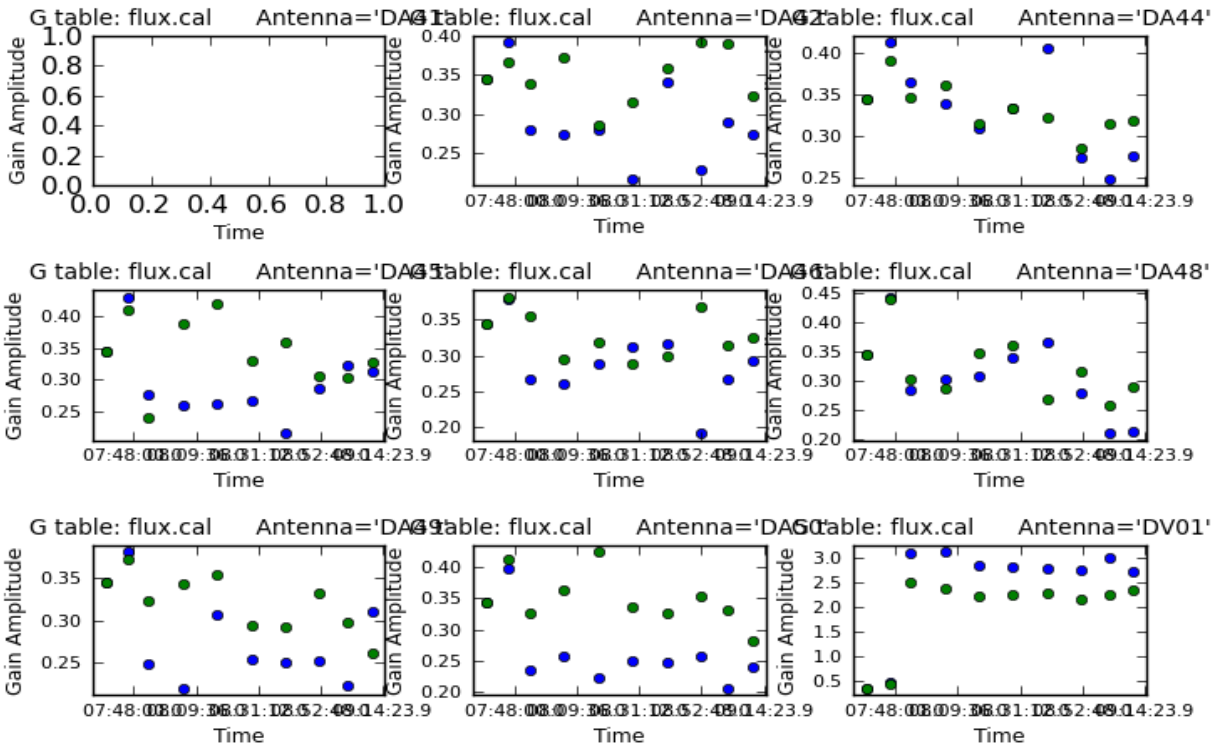
Now we have the model correct for the two quasars, which - as point sources - are useful calibrators for all u-v ranges. We can run another amplitude solution without restricting the u-v range (though note that this will push things a bit on Ceres).

```
os.system("rm -rf flux.cal")
gaincal(vis="sis14_twhya_bpcal.ms",
    caltable="flux.cal",
    field="0,2,3",
    solint="inf",
    calmode="a",
    gaintype="G",
    refant="DV22",
    gaintable="phase_int.cal")
```

Final flux calibration table

This is our final flux calibration table. Inspect the amplitude corrections for each antenna.

- `plotcal(caltable="flux.cal", xaxis="time", yaxis="amp", subplot=331,iteration="antenna", plotrange=[0,0,0,0])`



Apply Calibration

Apply our flux calibration and the (scan based) phase solution to all fields (including the science target).

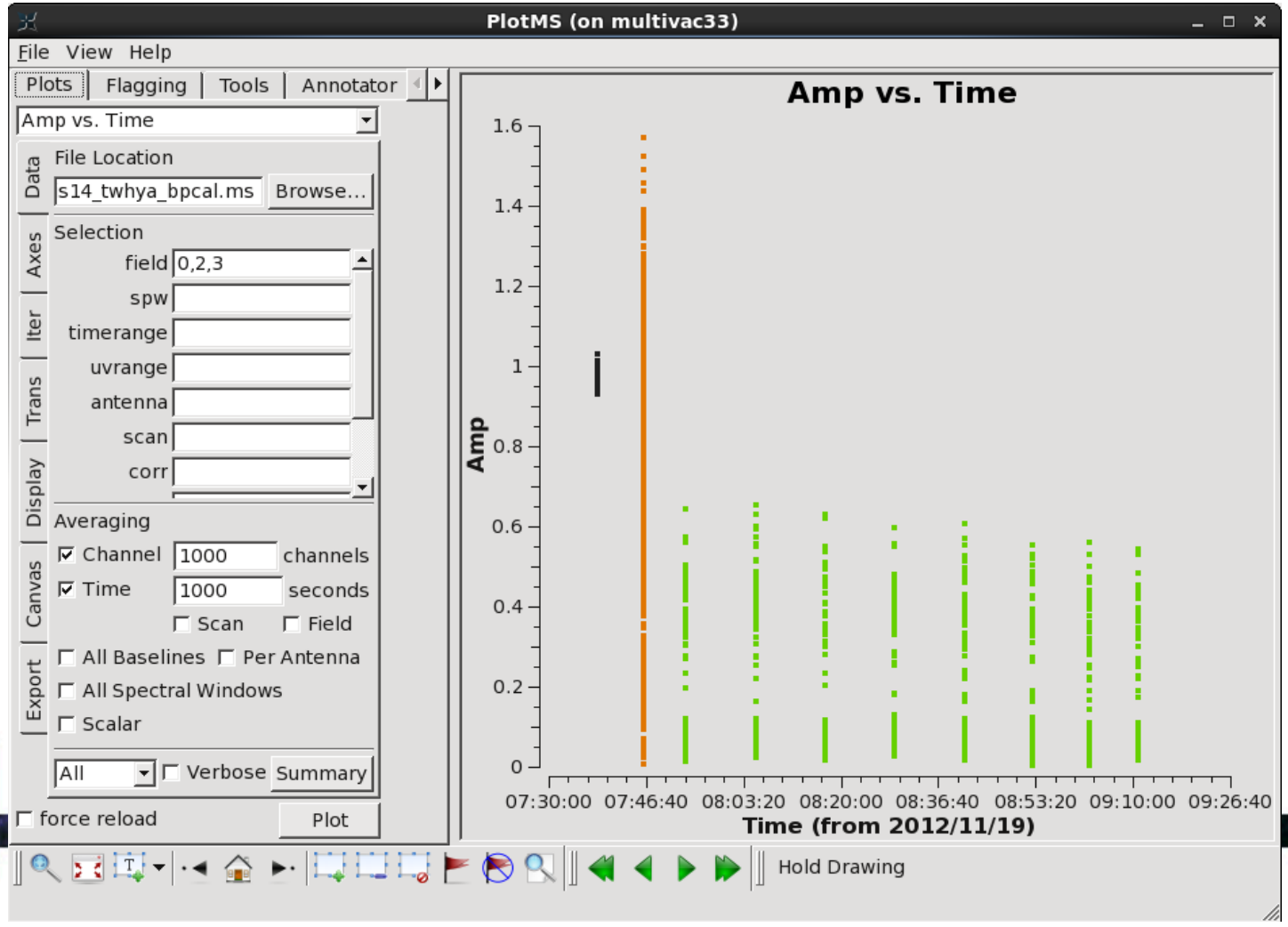
Note that we use the non-standard "calonly" command, which tells applycal not to flag data for which the calibration has failed.

```
applycal(vis="sis14_twhya_bpcal.ms",
     field="",
     gaintable=["phase_scan.cal", "flux.cal"],
     interp="linear",
     applycal="calonly")
```

Inspect the Results

Look at amplitude vs. time first in calibrated data

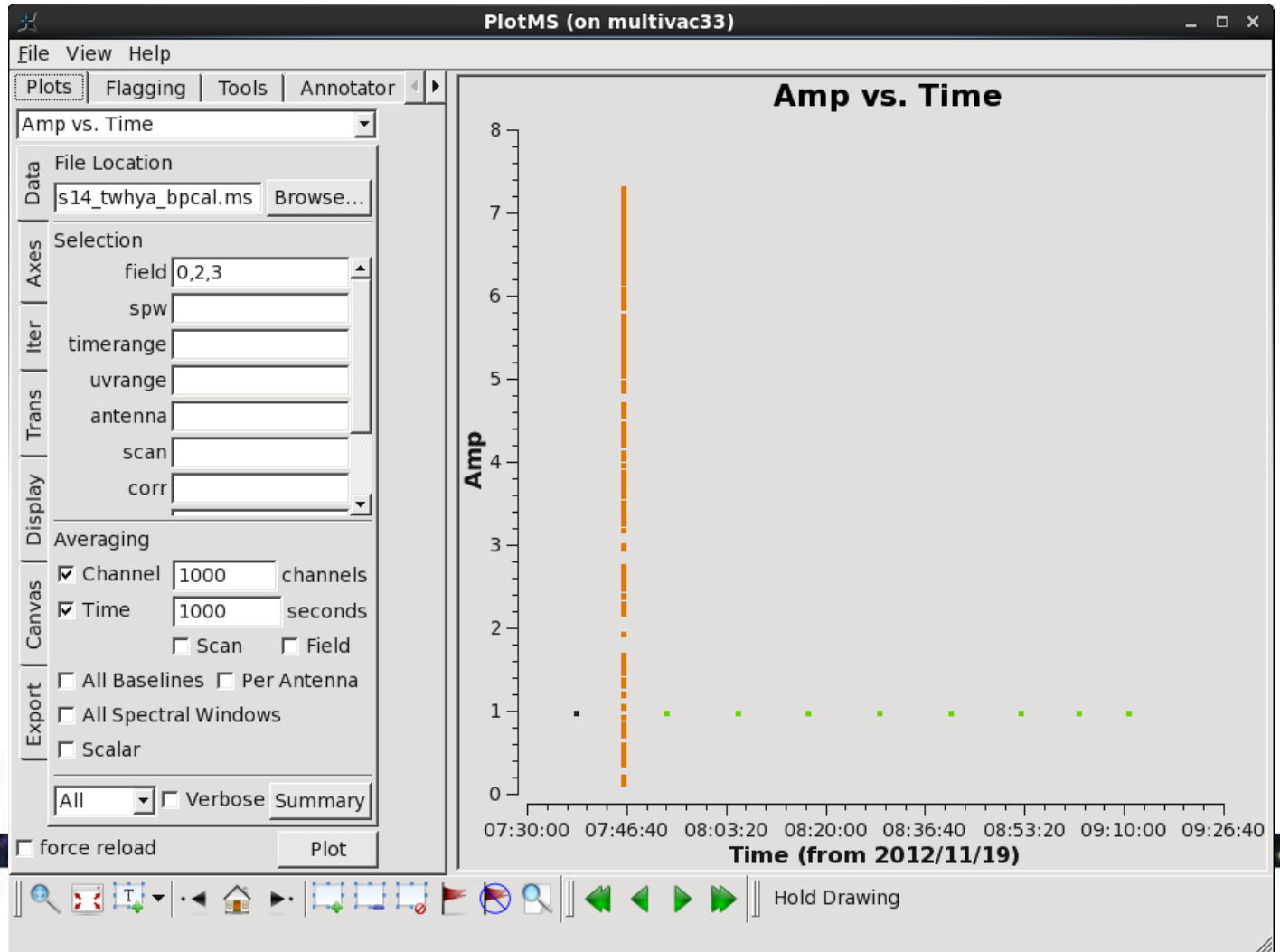
- `plotms(vis="sis14_twhya_bpcal.ms", xaxis="time", yaxis="amp", ydatacolumn="corrected", field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", coloraxis="field")`



Inspect the Results

Look at amplitude vs. time first in model

- `plotms(vis="sis14_twhya_bpcal.ms", xaxis="time", yaxis="amp", ydatacolumn="model", field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", coloraxis="field")`



Split out calibrated data

Now that we are satisfied with the gain calibration, we split out the gain calibrated data for further processing.

- `os.system("rm -rf sis14_twhya_calibrated.ms")`
- ```
  split(vis="sis14_twhya_bpcal.ms",
    outputvis="sis14_twhya_calibrated.ms",
    datacolumn="corrected",
    keepflags=False)
  ```

This produces one of the supplied data products (sis14_twhya_calibrated.ms) - so you can restart from a successful version of this script anytime.


  ```

Outline

- Short introduction to CASA and the Python interface
  - Discussion of “tools” will be done separately
- The Flow of Calibration
- Key CASA tasks for data reduction/calibration
- **Data Inspection and Flagging**
- Basic Imaging

Data Inspection and Flagging and End to End processing

ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

May 16, 2014

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, 2, and 3 here) should look like in each space. Before that however, let's walk through the plotms GUI to familiarize ourselves with the interface.

plotms

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

Can be started in the usual *casapy* interface

- inp plotms

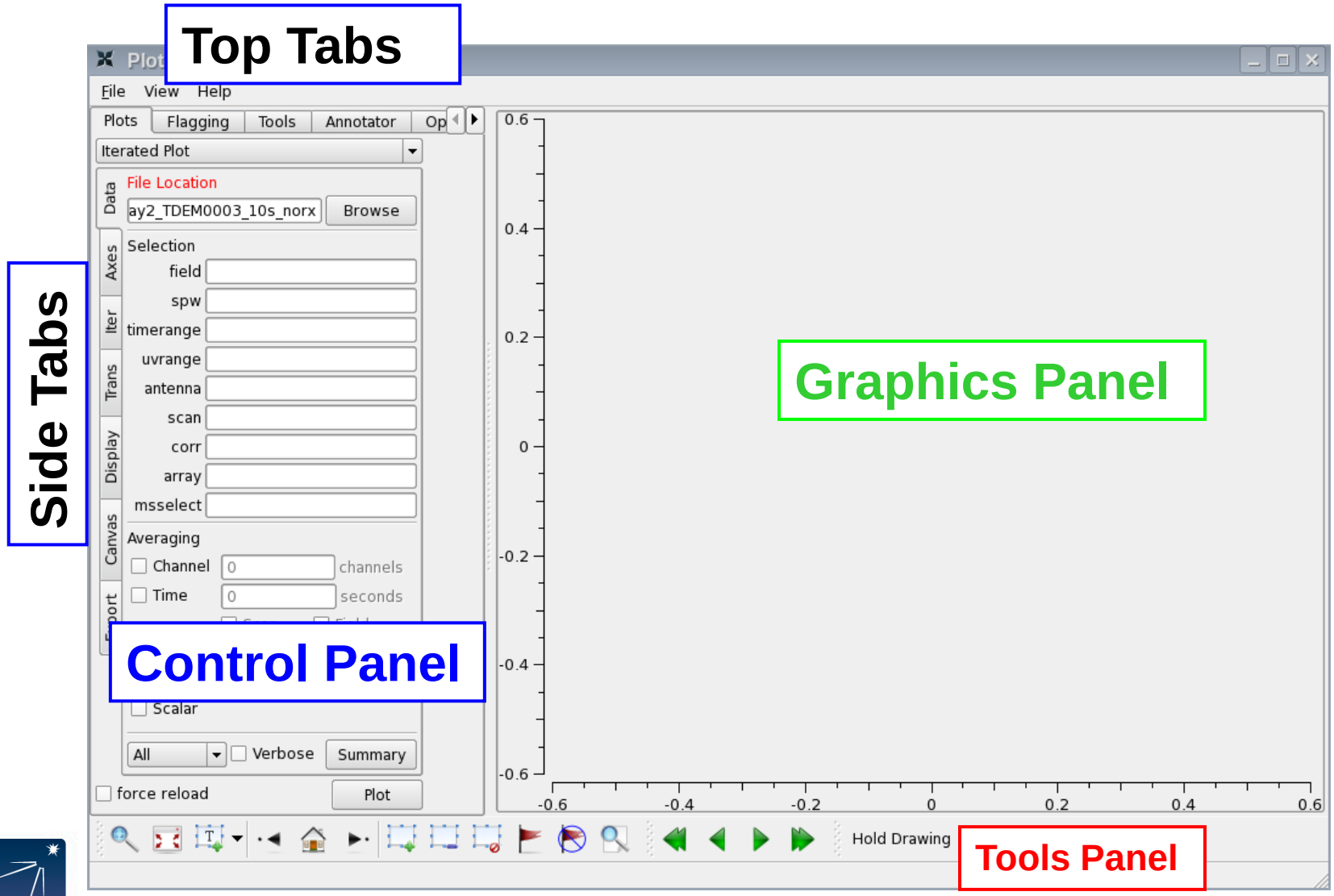
Can be fully specified in the CASA command line (e.g.):

- `plotms(vis="sis14_twhya_bpcal.ms", xaxis="time", yaxis="amp", ydatacolumn="corrected", field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", coloraxis="field")`

Also can be started directly from the unix prompt:

- `% casaplotms`

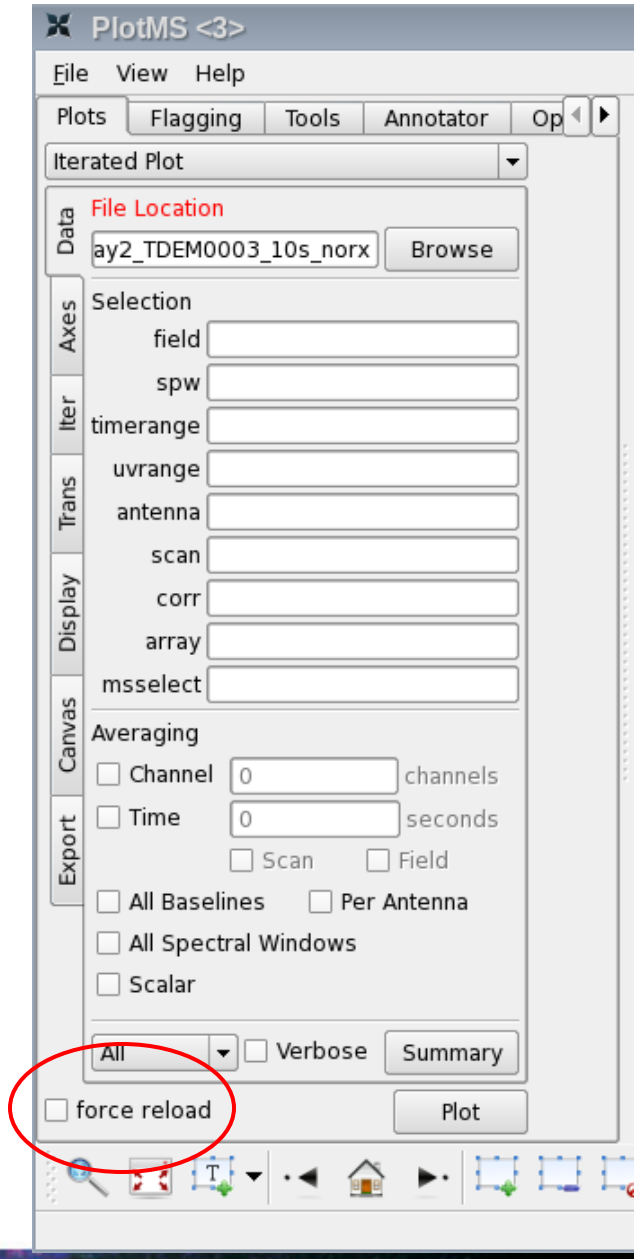
Data Review: *plotms*



Data Review: *plotms*

Control panel: Data

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

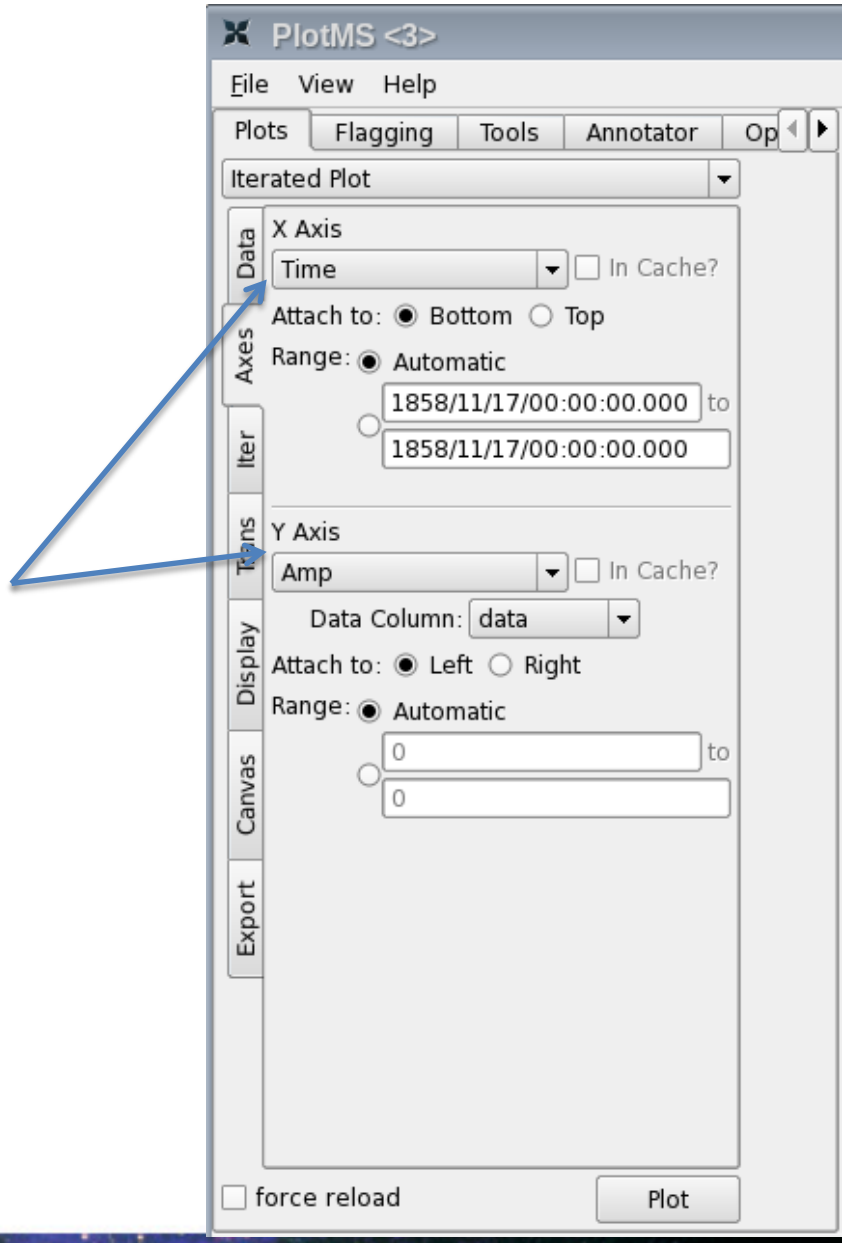


Data Review: *plotms*

Control panel: Axes

Drop down menus to select x and y axes:

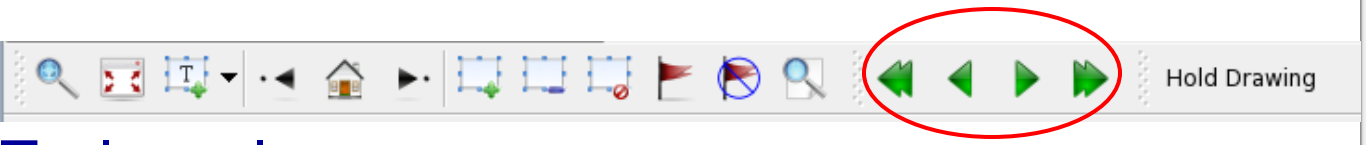
time, channel, frequency, velocity, amplitude, phase, uvdist, elevation, etc.



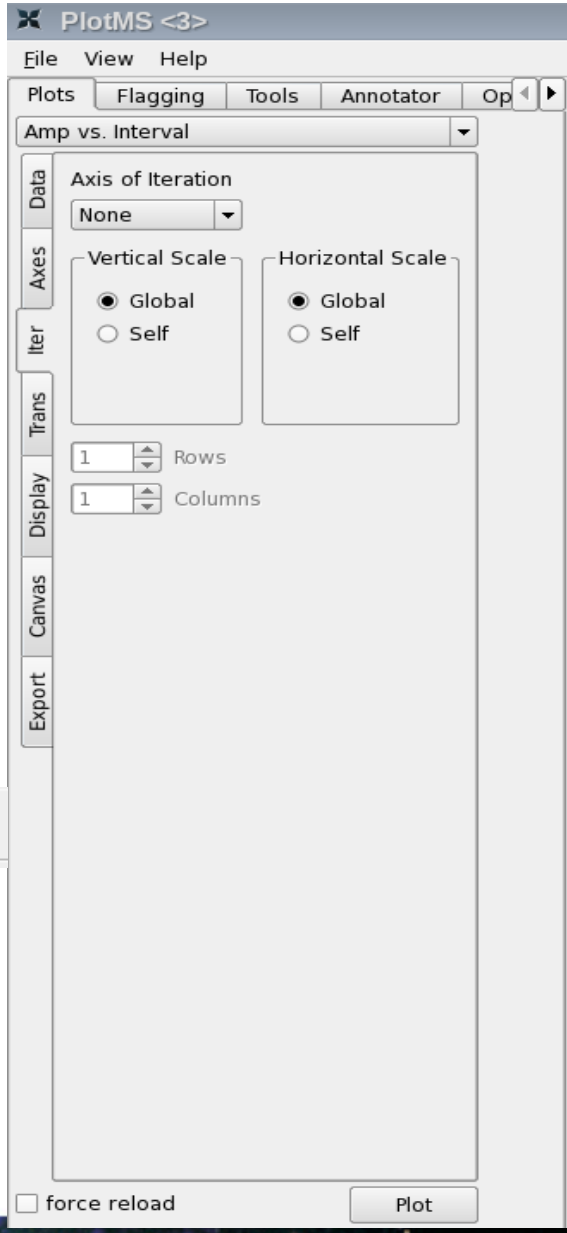
Data Review: *plotms*

Iteration

Scan
Field
Spw
Baseline
Antenna



Tool panel

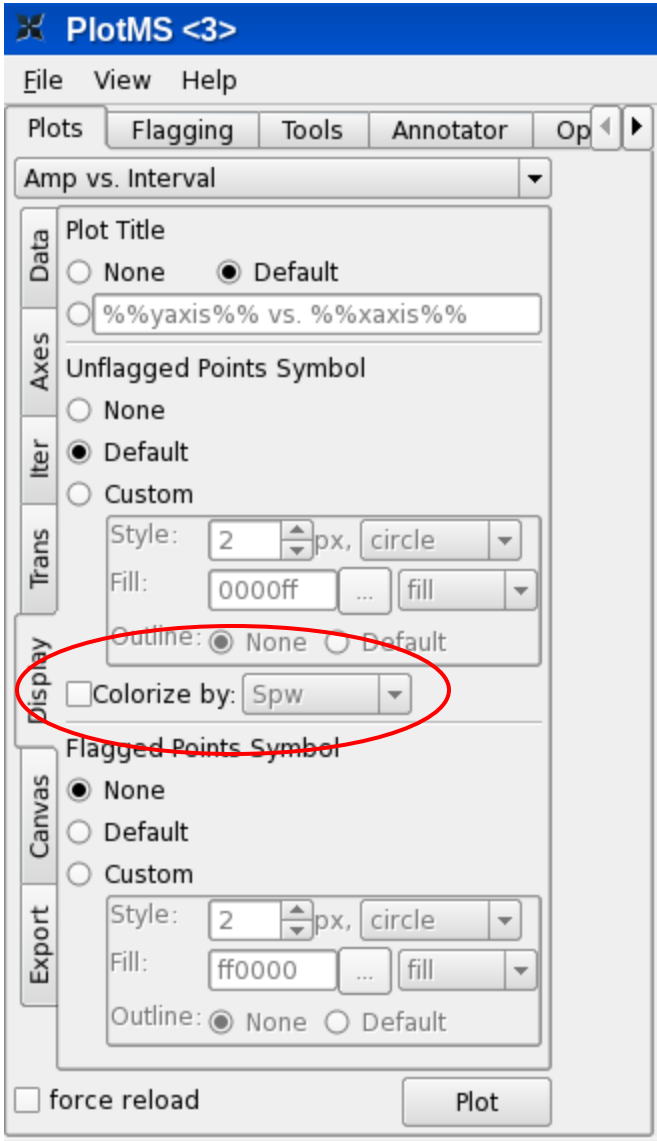


Data Review: *plotms*

Display

Colorize by:

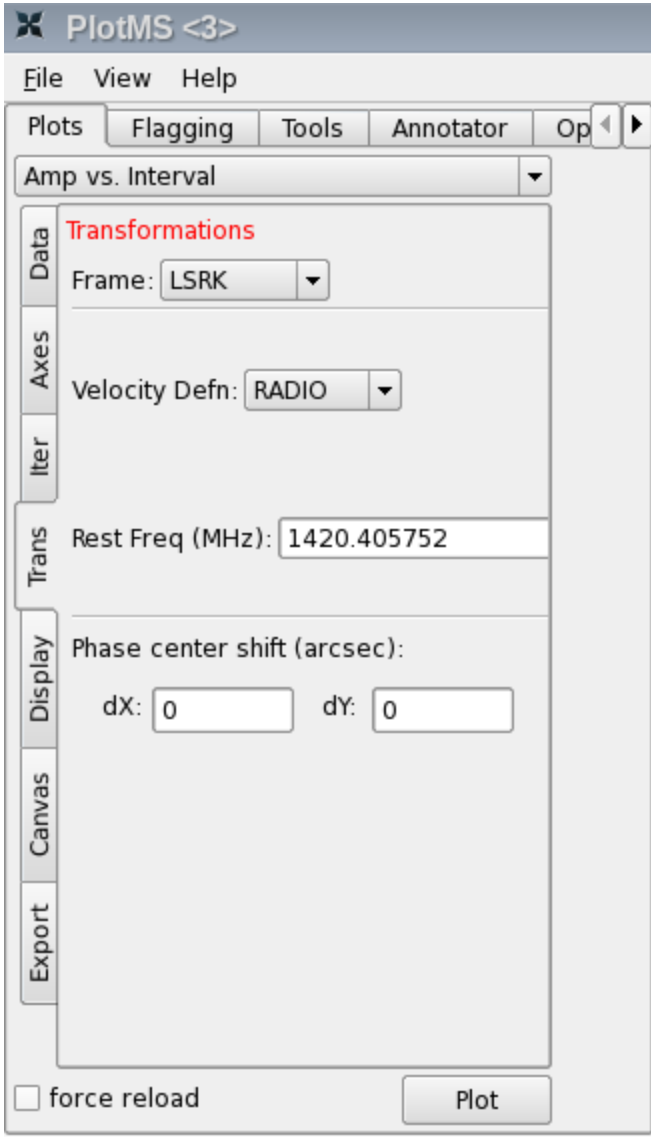
- Scan
- Field
- Spw
- Antenna1
- Antenna2
- Baseline
- Channel
- Correlation



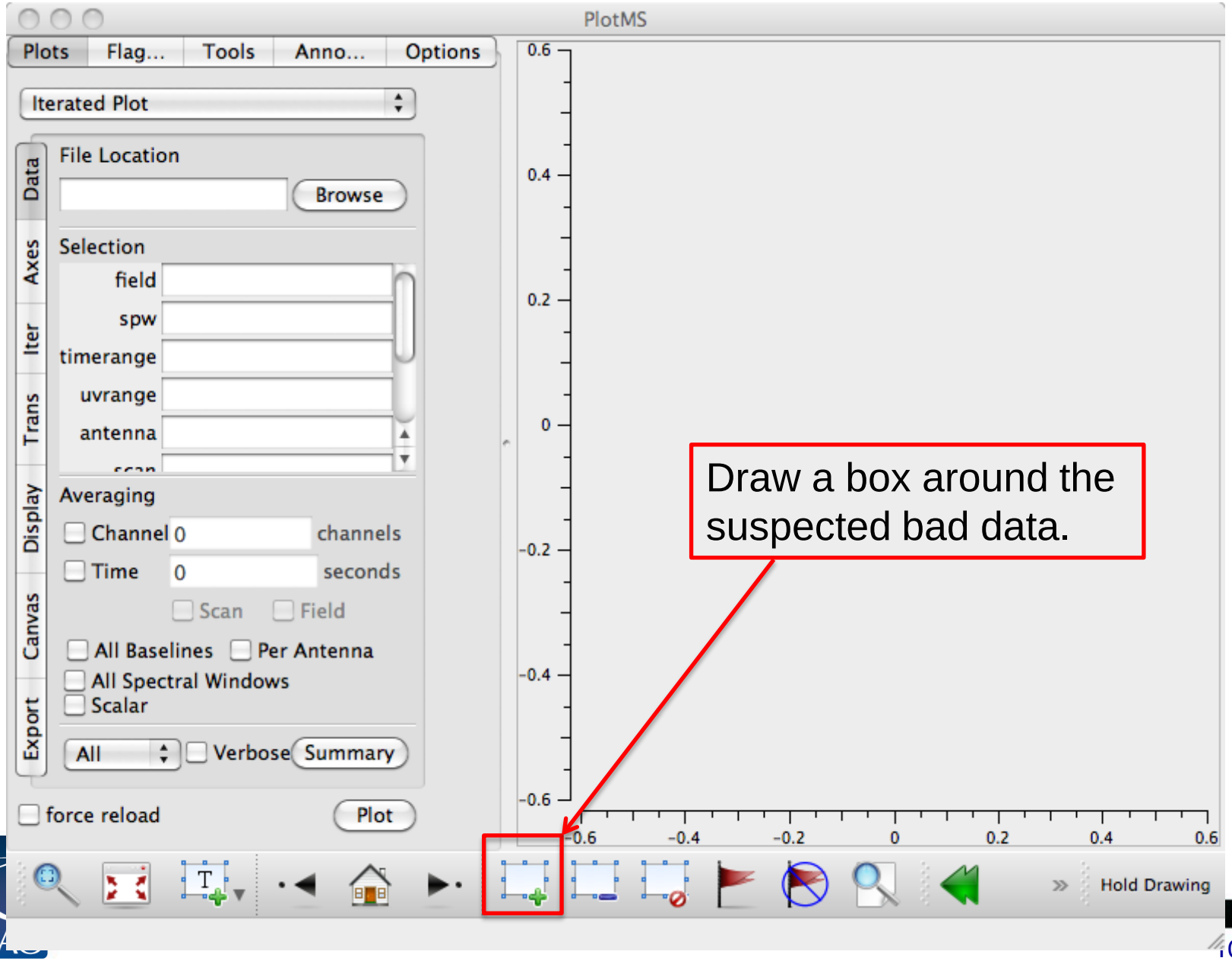
Data Review: *plotms*

Transformations

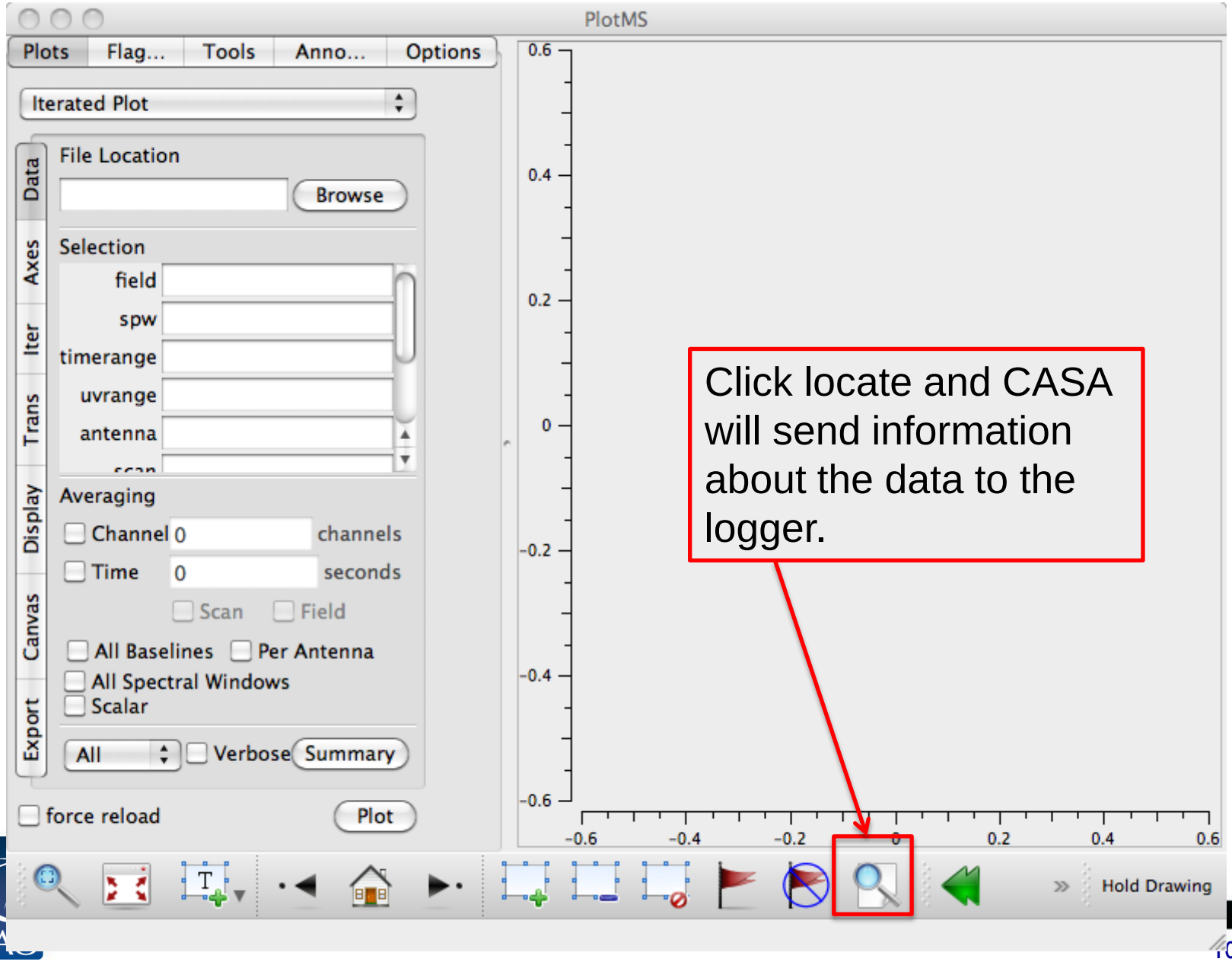
Frame: TOPO, GEO, BARY, LSRK, LSRD, etc..



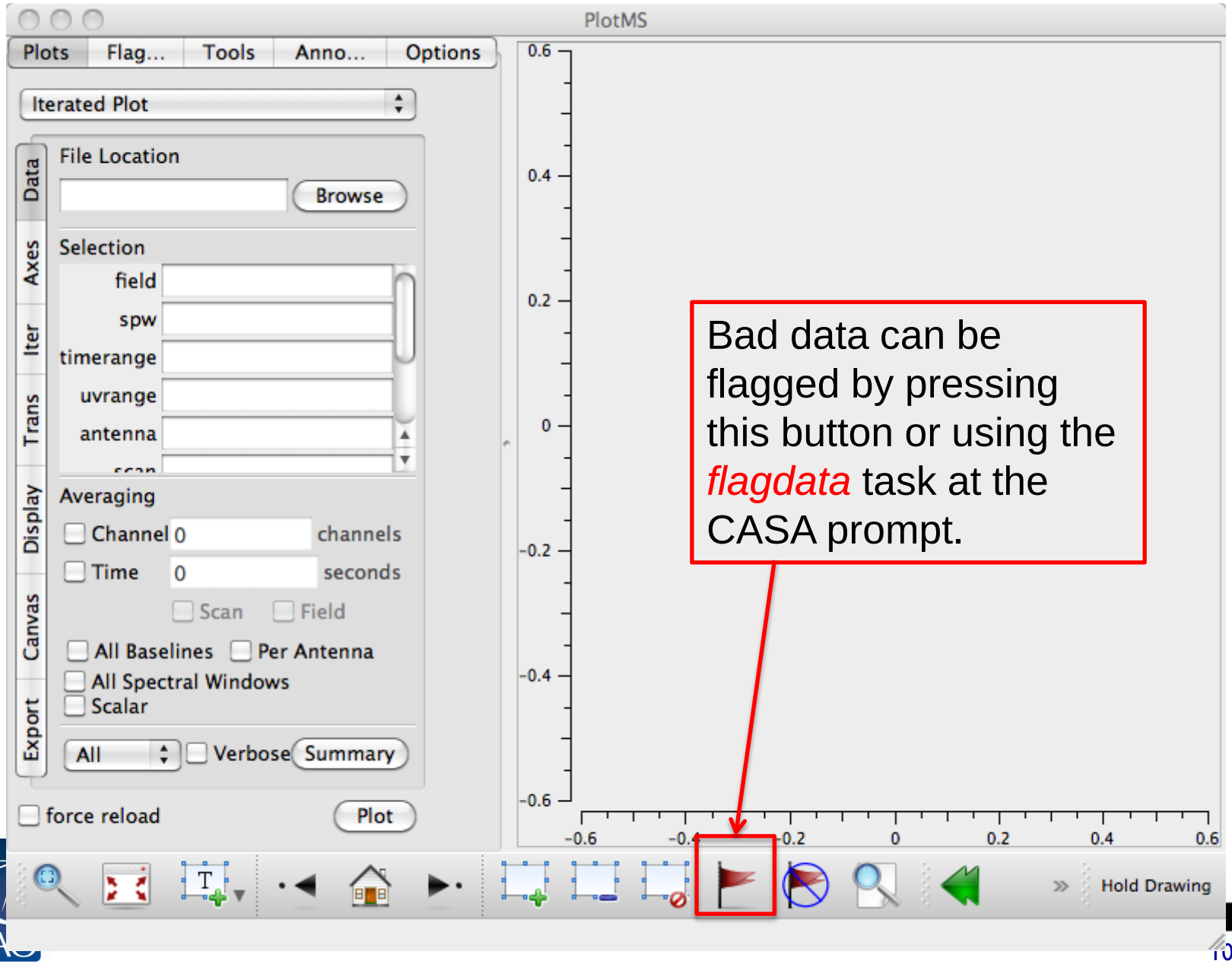
Flagging: Locating Bad Data - *plotms*



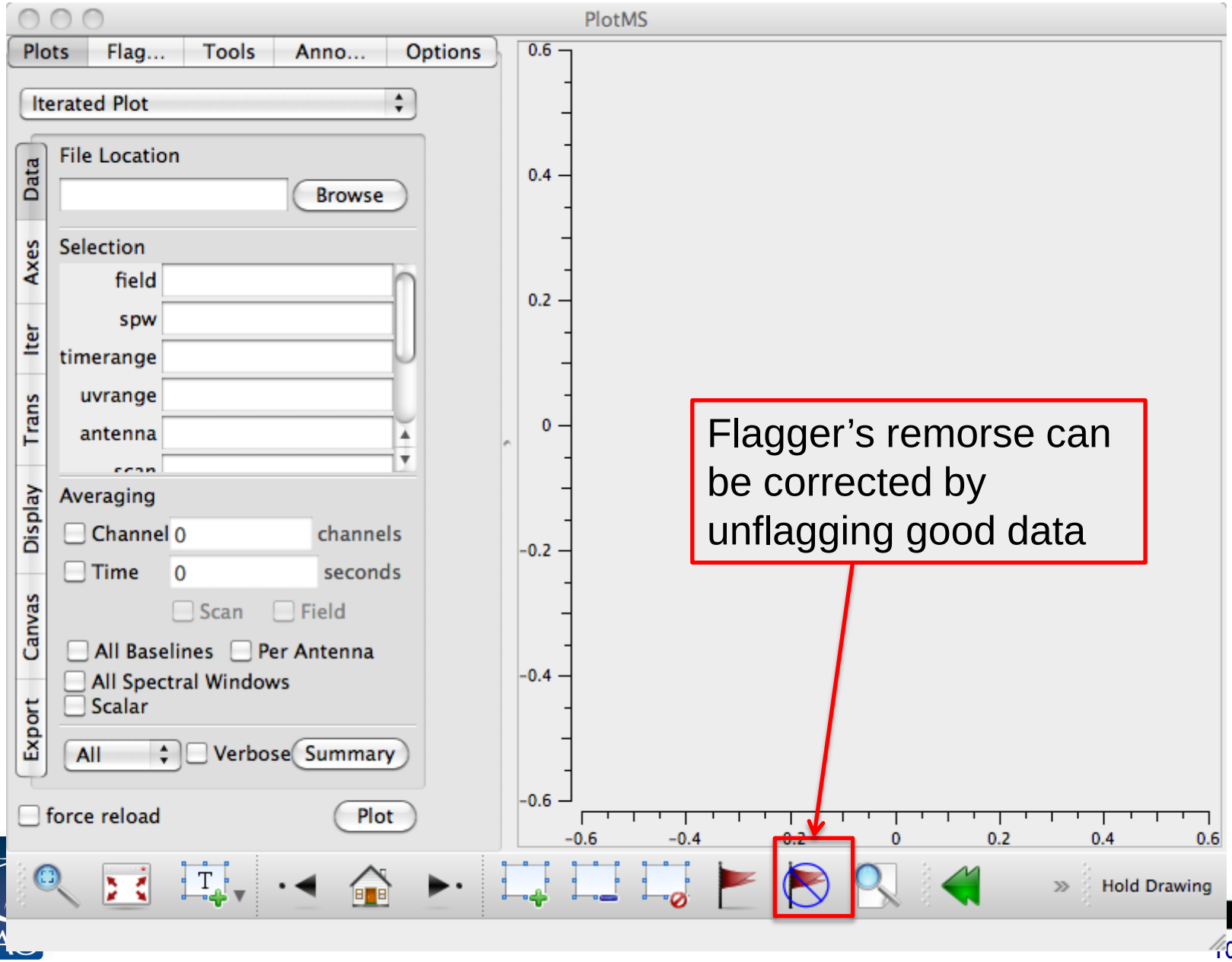
Flagging: Locating Bad Data - *plotms*



Flagging: Locating Bad Data - *plotms*



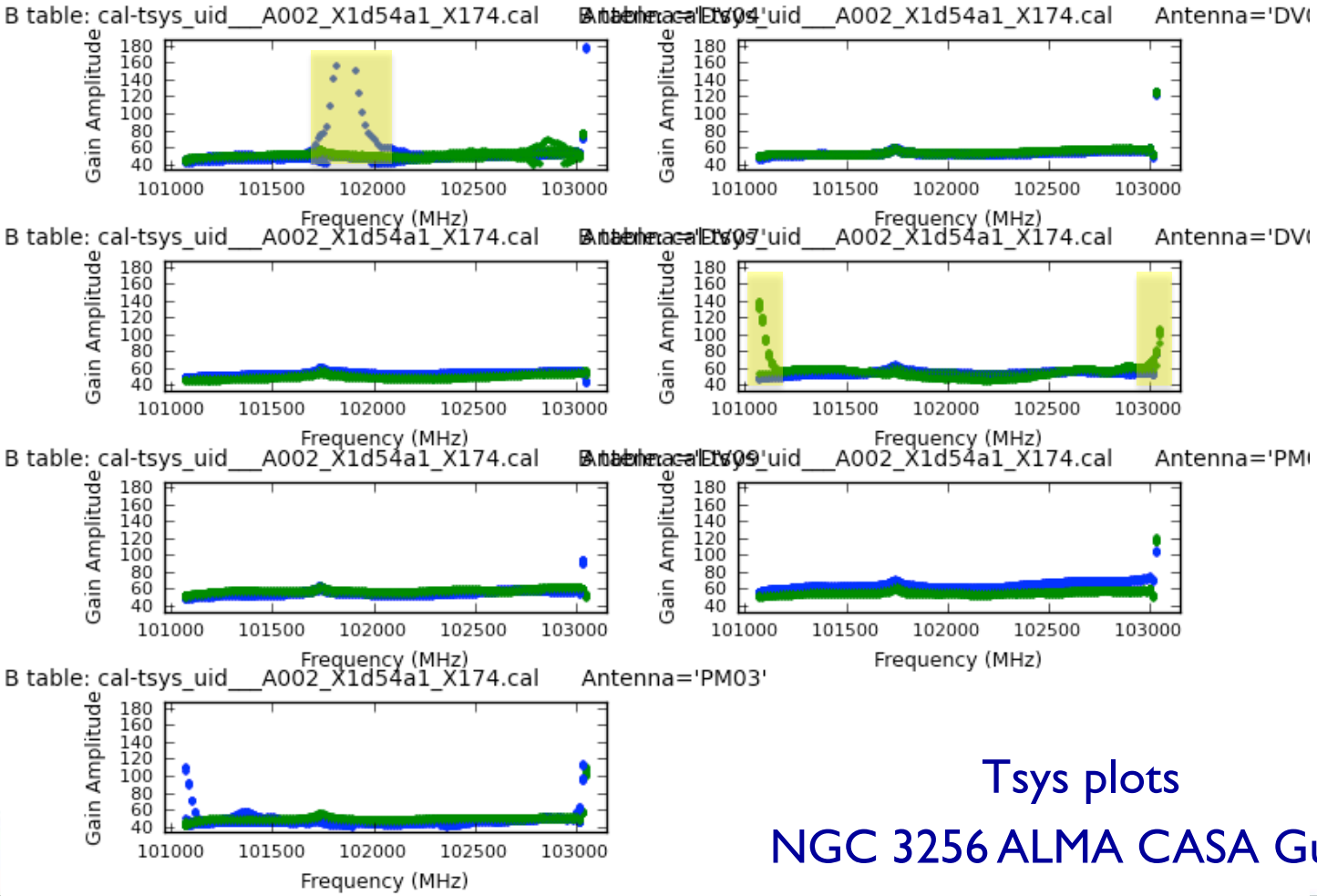
Flagging: Locating Bad Data - *plotms*



Flagging: Initial Flagging

- 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

Flagging:
Initial Flagging



Tsys plots

NGC 3256 ALMA CASA Guide

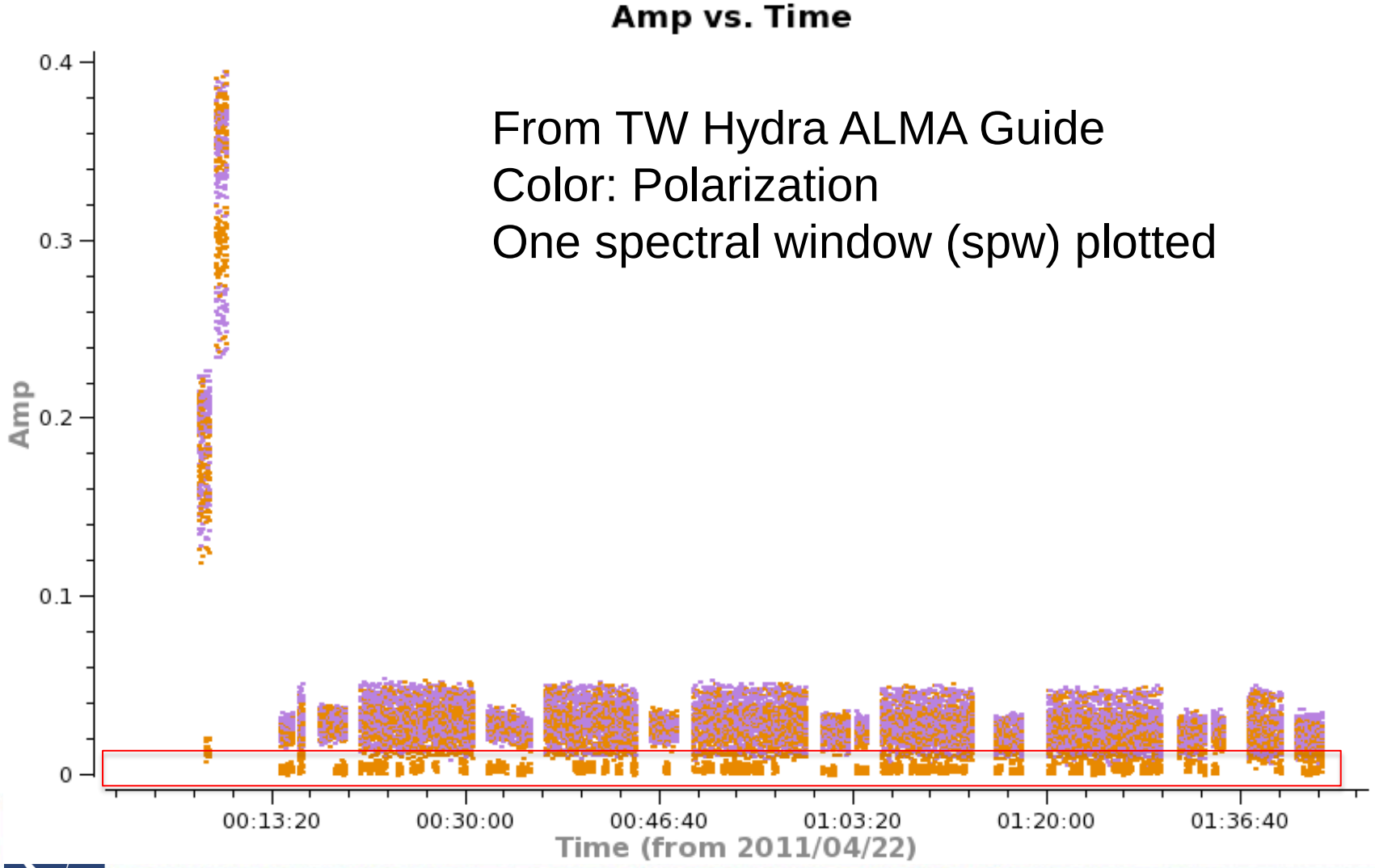
Flagging: What to Look For

- Plots of amplitude and phase vs. time and frequency
- 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: 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

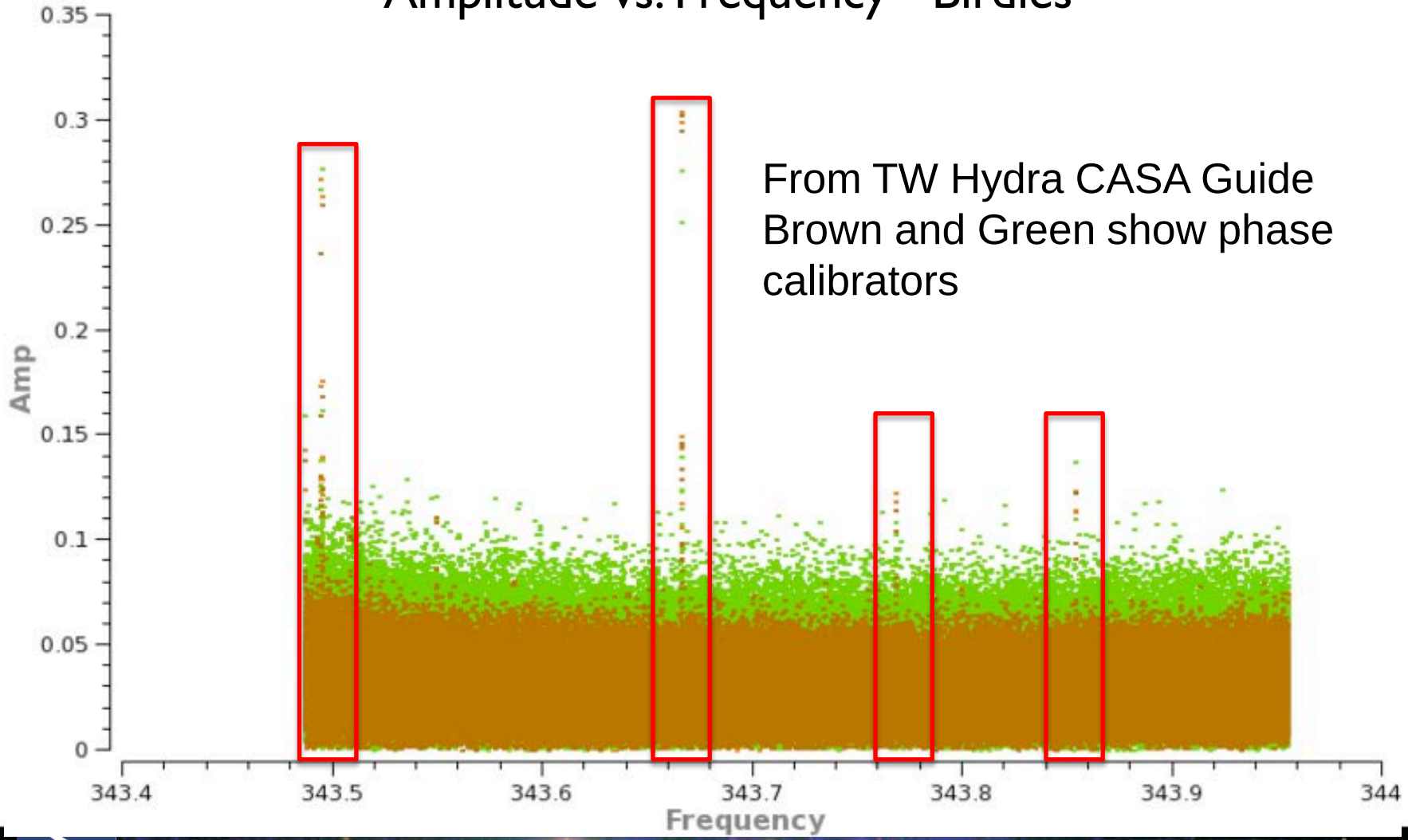
Flagging: What to Look For



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

Flagging:
What to Look For

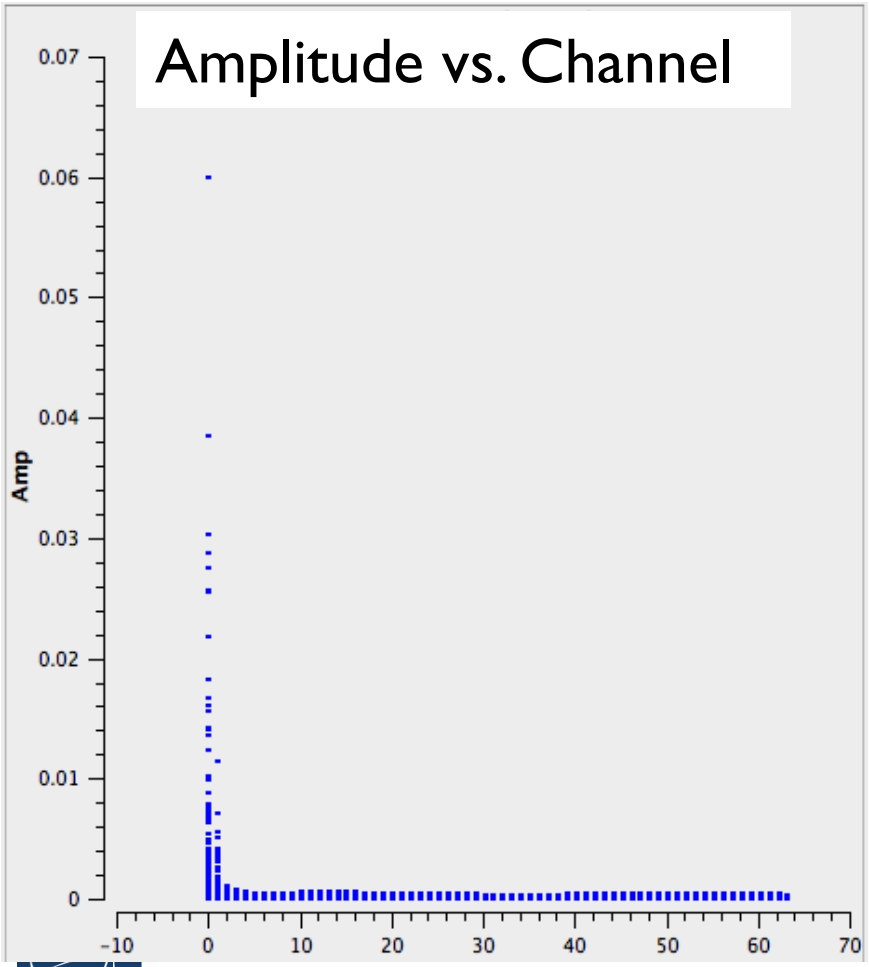
Amplitude vs. Frequency - Birdies

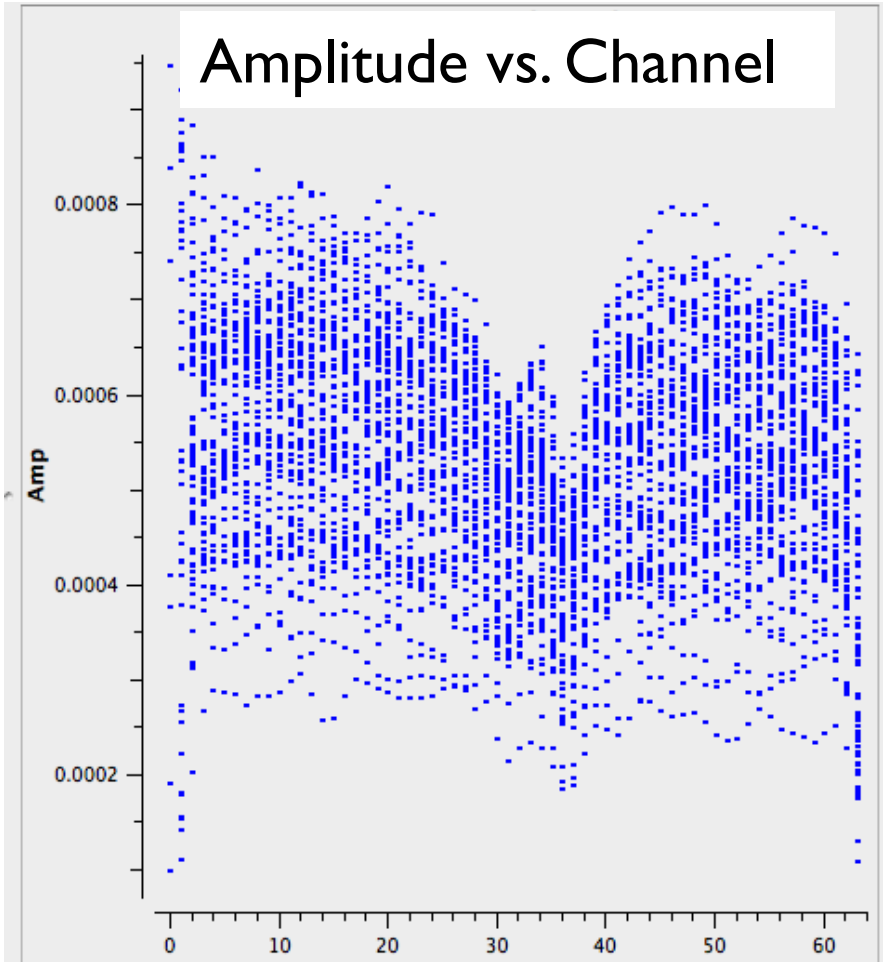


From TW Hydra CASA Guide
Brown and Green show phase calibrators

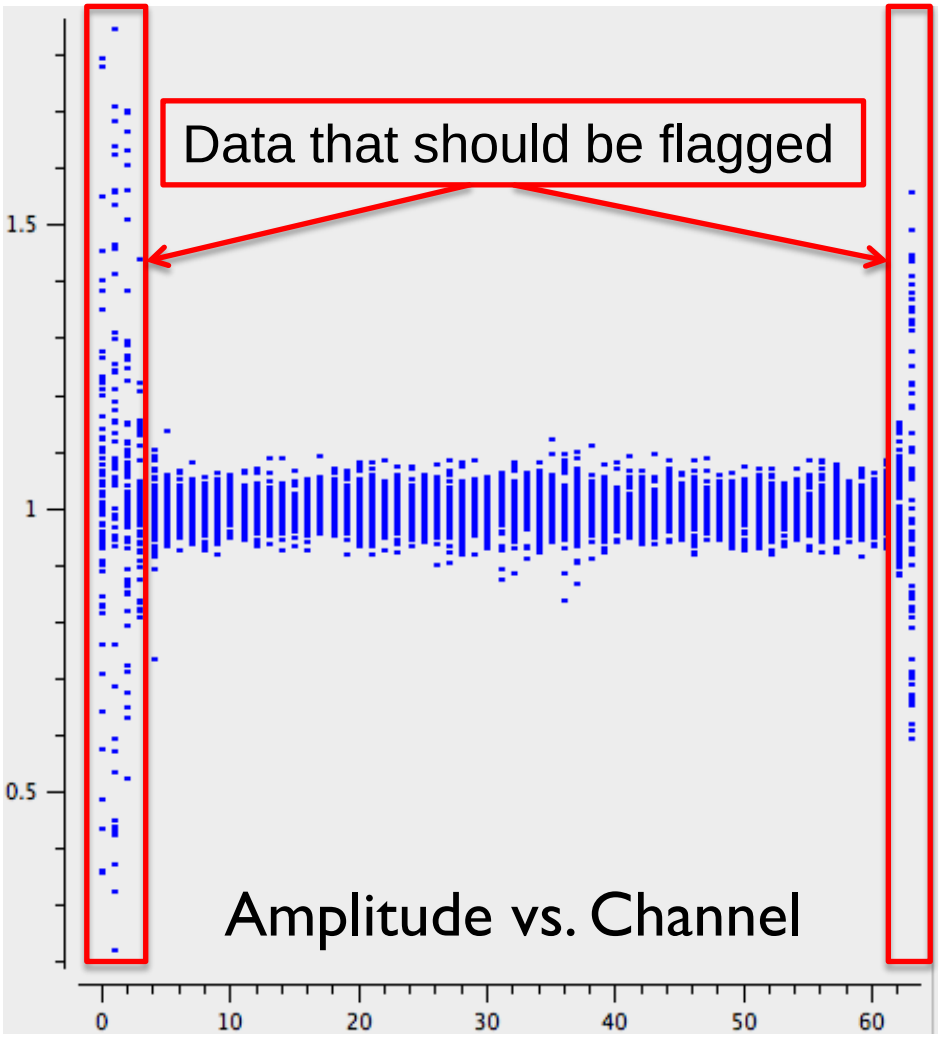
Flagging: What to Look For

Edge Channels



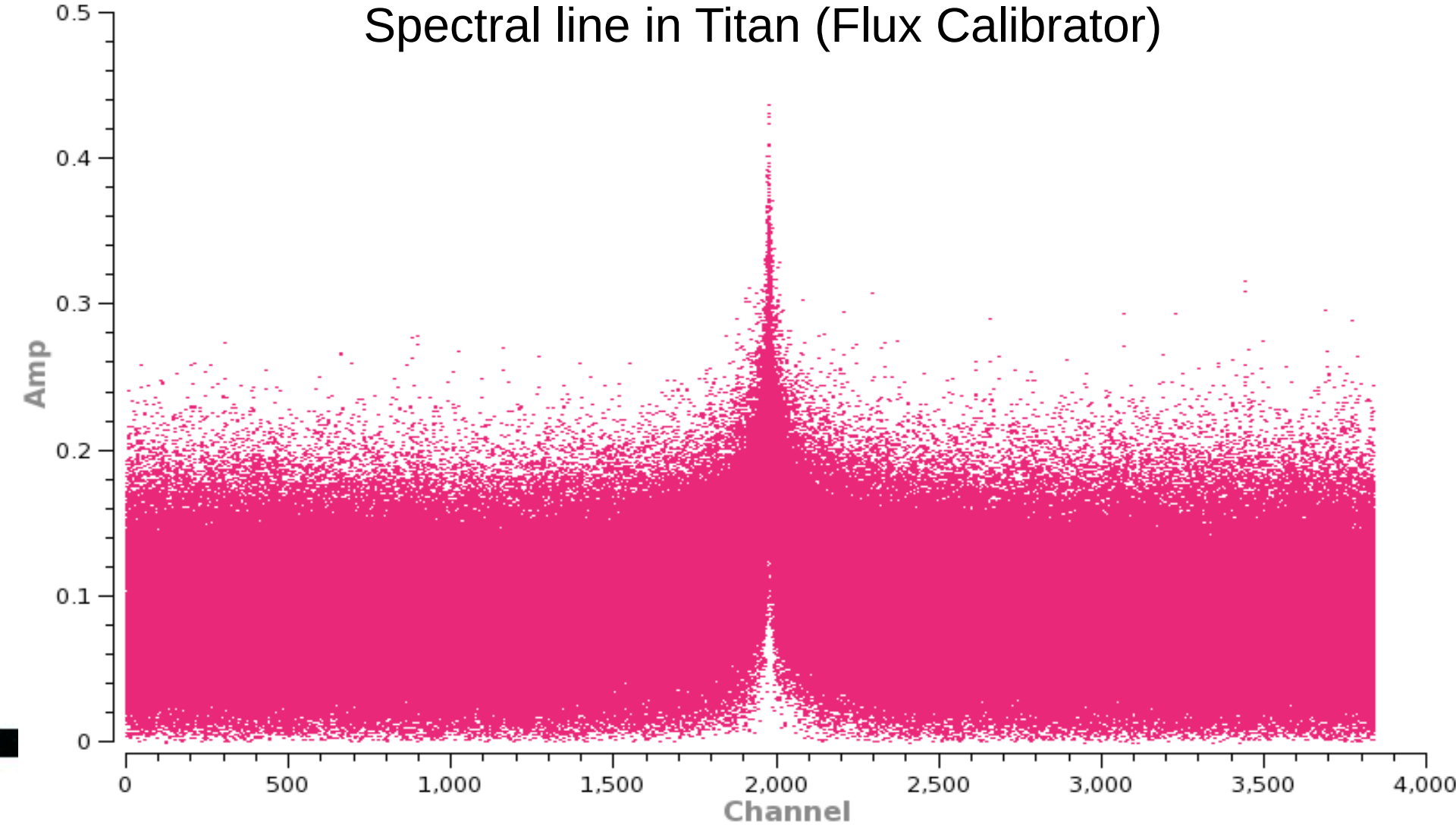


Flagging: What to Look For

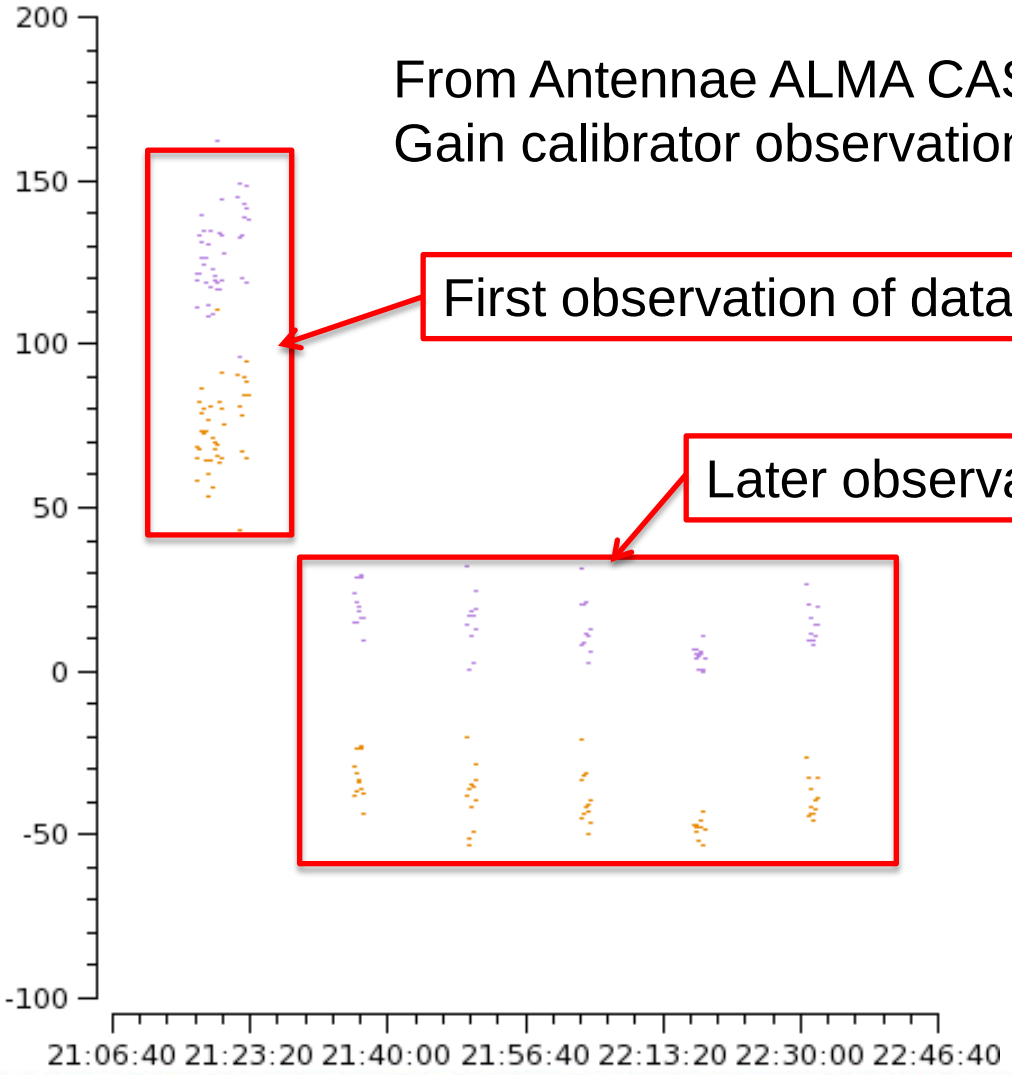


Flagging:
What to Look For

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

First observation of data

Later observations of data

Sage Advice

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

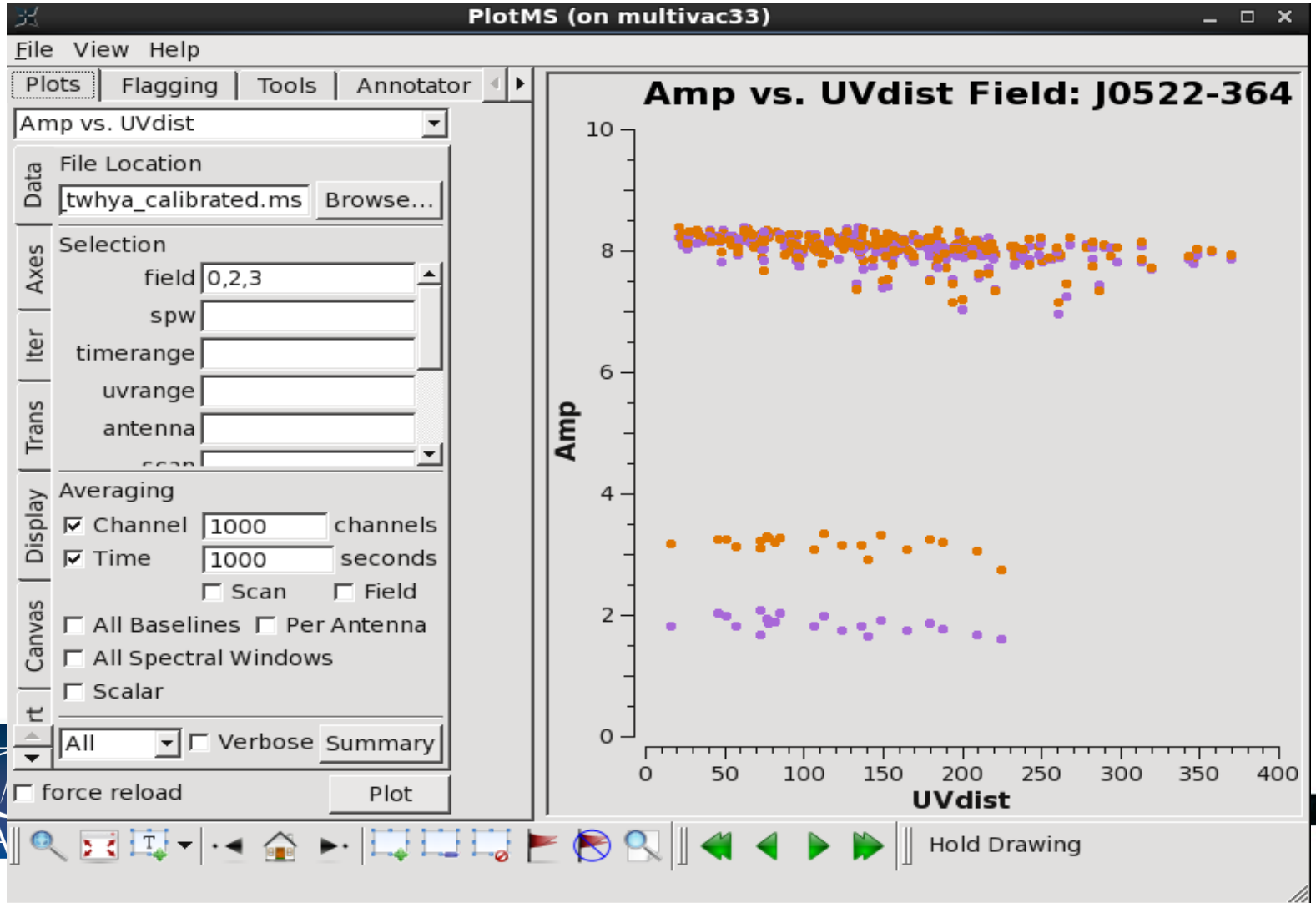
Inspect your Data

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 here that the *science* data are not generally shown and will still need to be flagged.

Start with plots of amplitude and phase vs. uv distance. For point sources we expect flat amplitude and zero phases for these plots.

Amp vs. UVdis

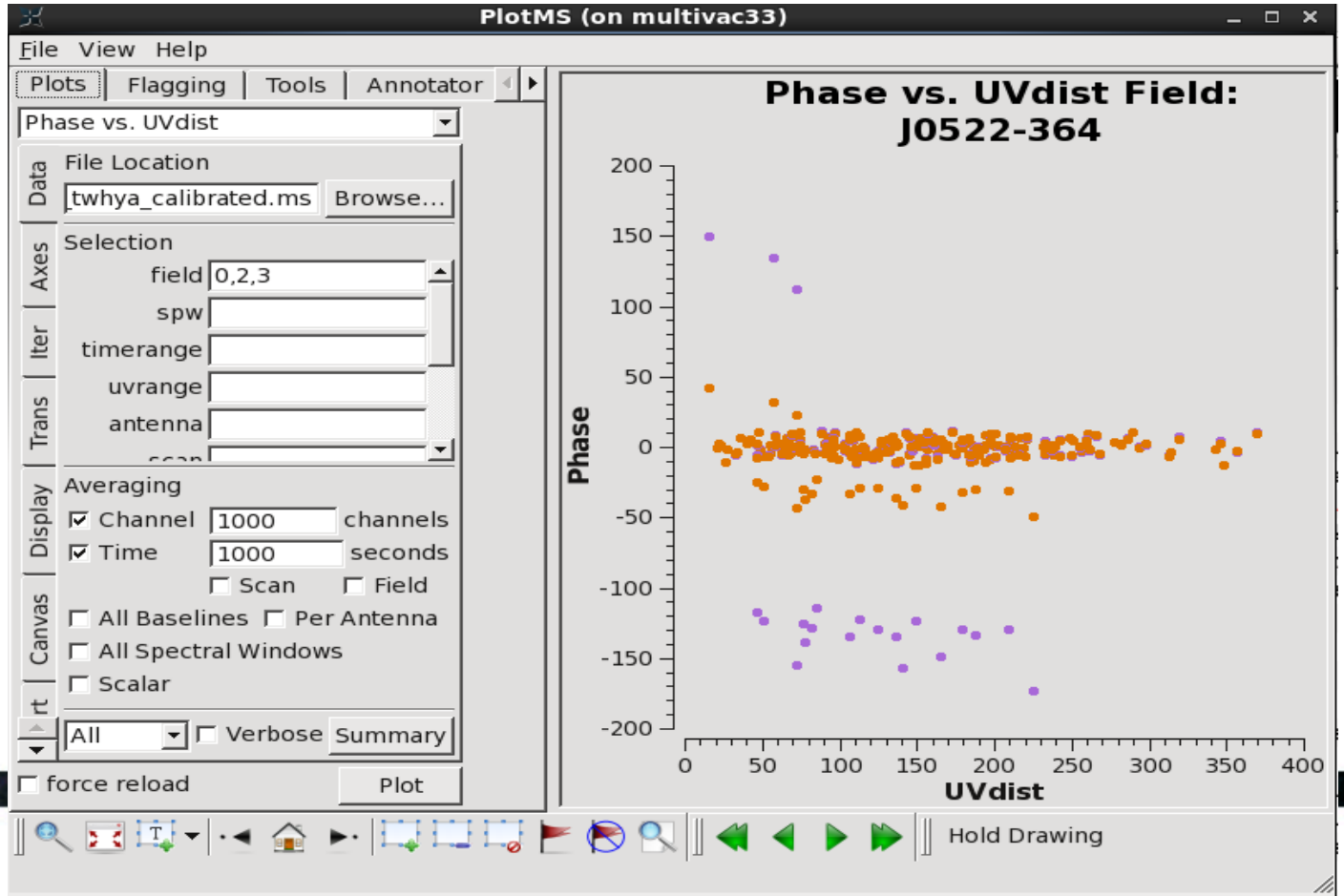
```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="uvdist", yaxis="amp", ydatacolumn="corrected",
    field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", iteraxis="field",
    coloraxis="corr")
```



Phase vs. UVdis

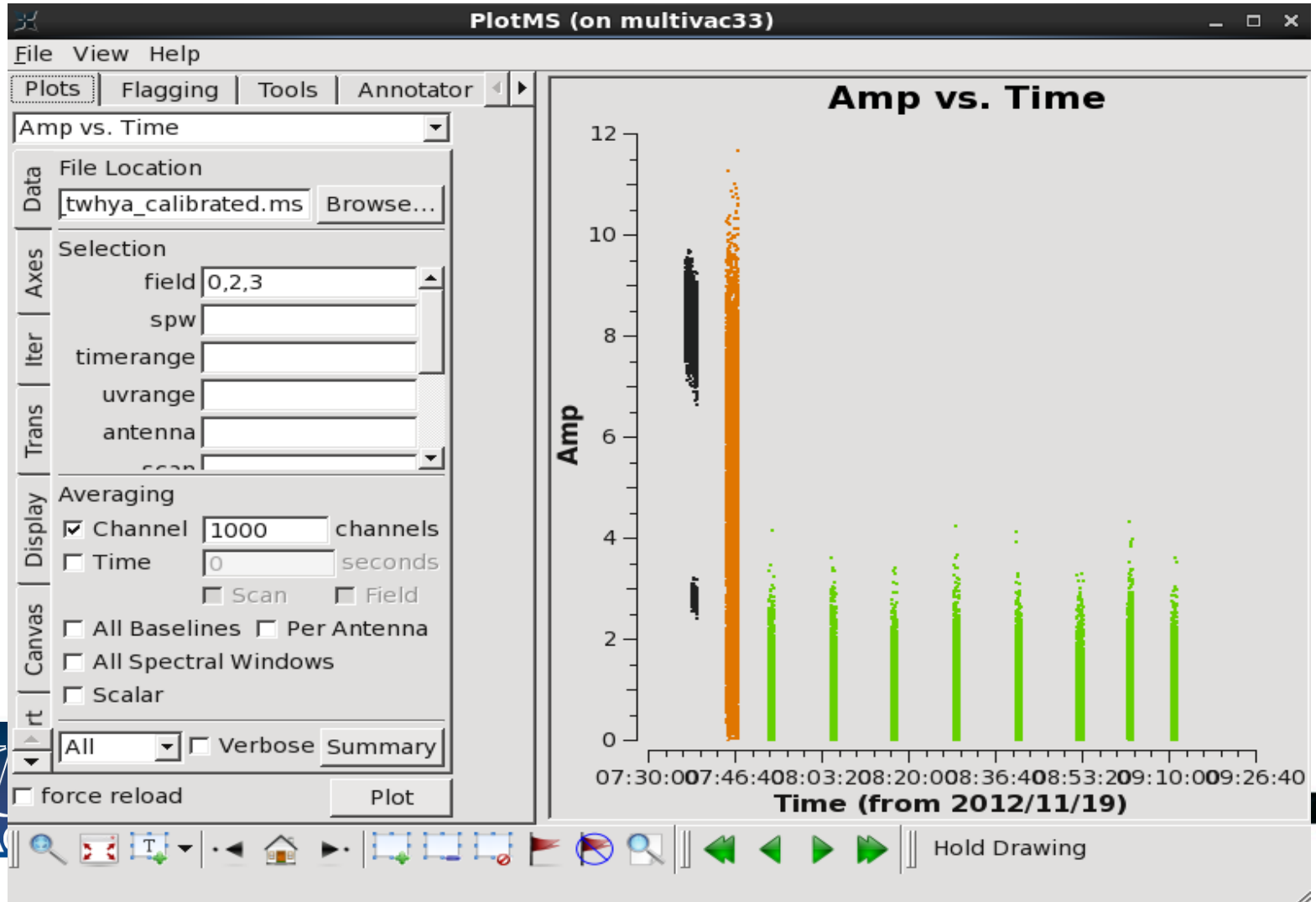
We see some outliers. Using "locate" we clearly see that DV19 is having problems for the bandpass calibrator, showing low amplitudes. Let's have a look at amplitude vs. time

- `plotms(vis="sis14_twhya_calibrated.ms", xaxis="uvdist", yaxis="phase", ydatacolumn="corrected", field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", iteraxis="field", coloraxis="corr")`



Amp vs. Time

```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="uvdist", yaxis="phase", ydatacolumn="corrected",
    field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", iteraxis="field",
    coloraxis="corr")
```



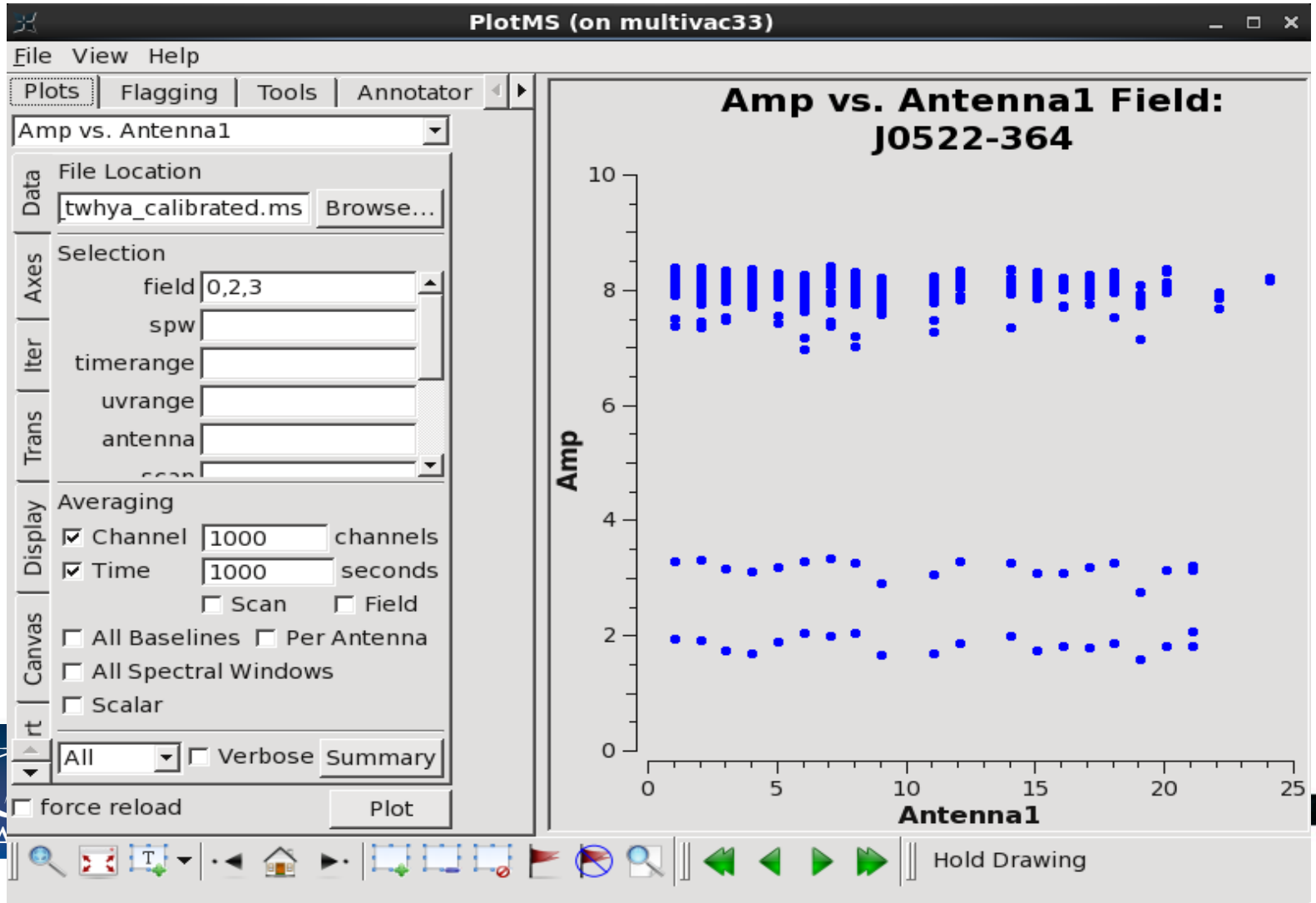
Amp vs. Time

Again, you see the same issues. We'll note that we want to flag DV19. Potentially we could flag it only on field 0, it's not totally clear that it's bad throughout the track. However, this means we will need to come up with an alternative calibration scheme for the bandpass (possibly using the other quasar). We will plan to flag DV19.

Next we will inspect phase and amplitude as a function of antenna. Each visibility (point) has two antennas associated with it, which are identified as Antenna1 and Antenna2 based on their number (the lower number is always 1). Remember here that we don't expect astrophysical signals to show up strongly as a function of antenna unless the antenna sits in a very unusual point in the array (check your plotants).

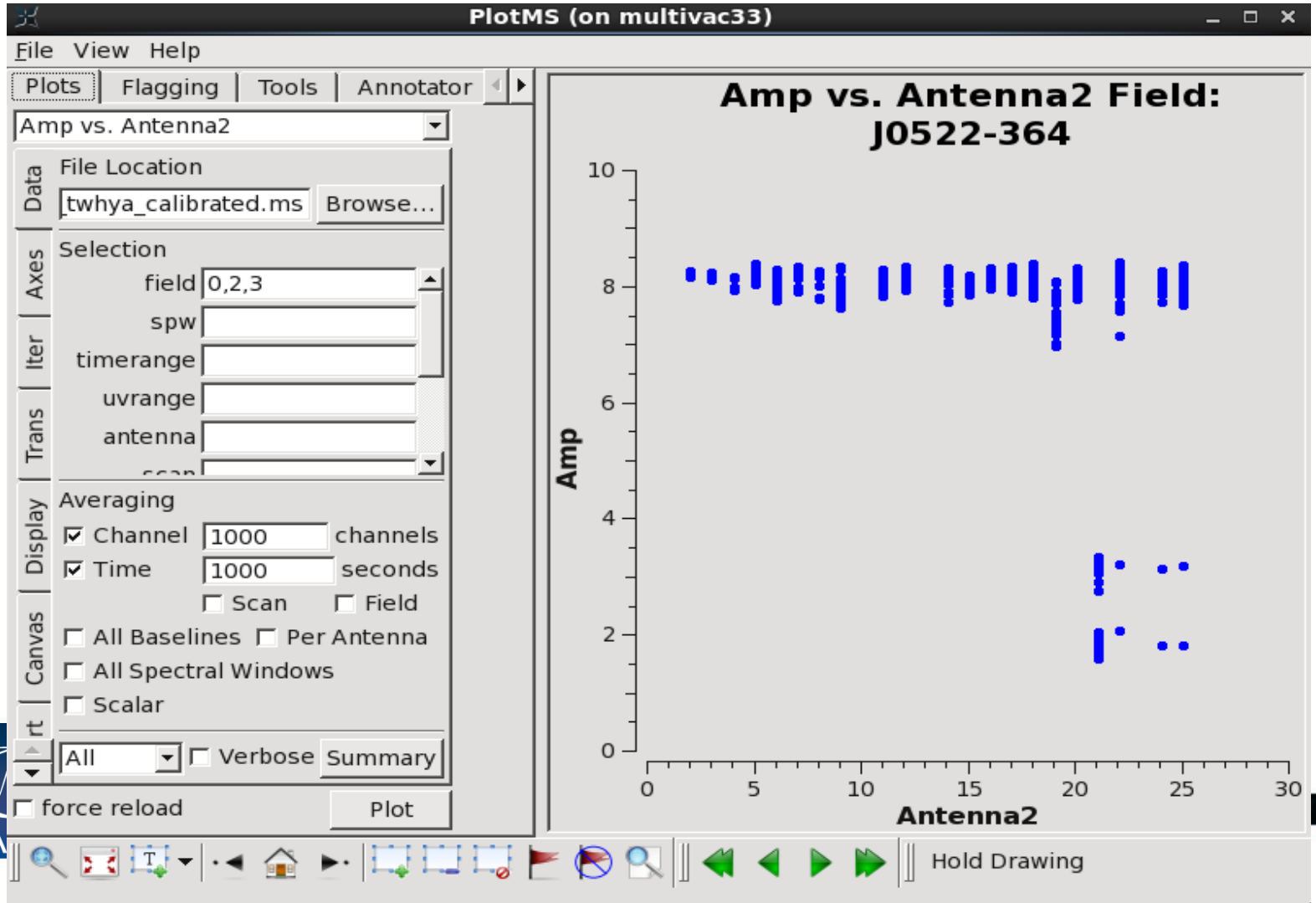
Amp vs. antenna1

```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="antenna1", yaxis="amp", ydatacolumn="corrected",
    field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", iteraxis="field",
    coloraxis="corr")
```



Amp vs. antenna2

```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="antenna2", yaxis="amp", ydatacolumn="corrected",
    field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3", iteraxis="field",
    coloraxis="corr")
```



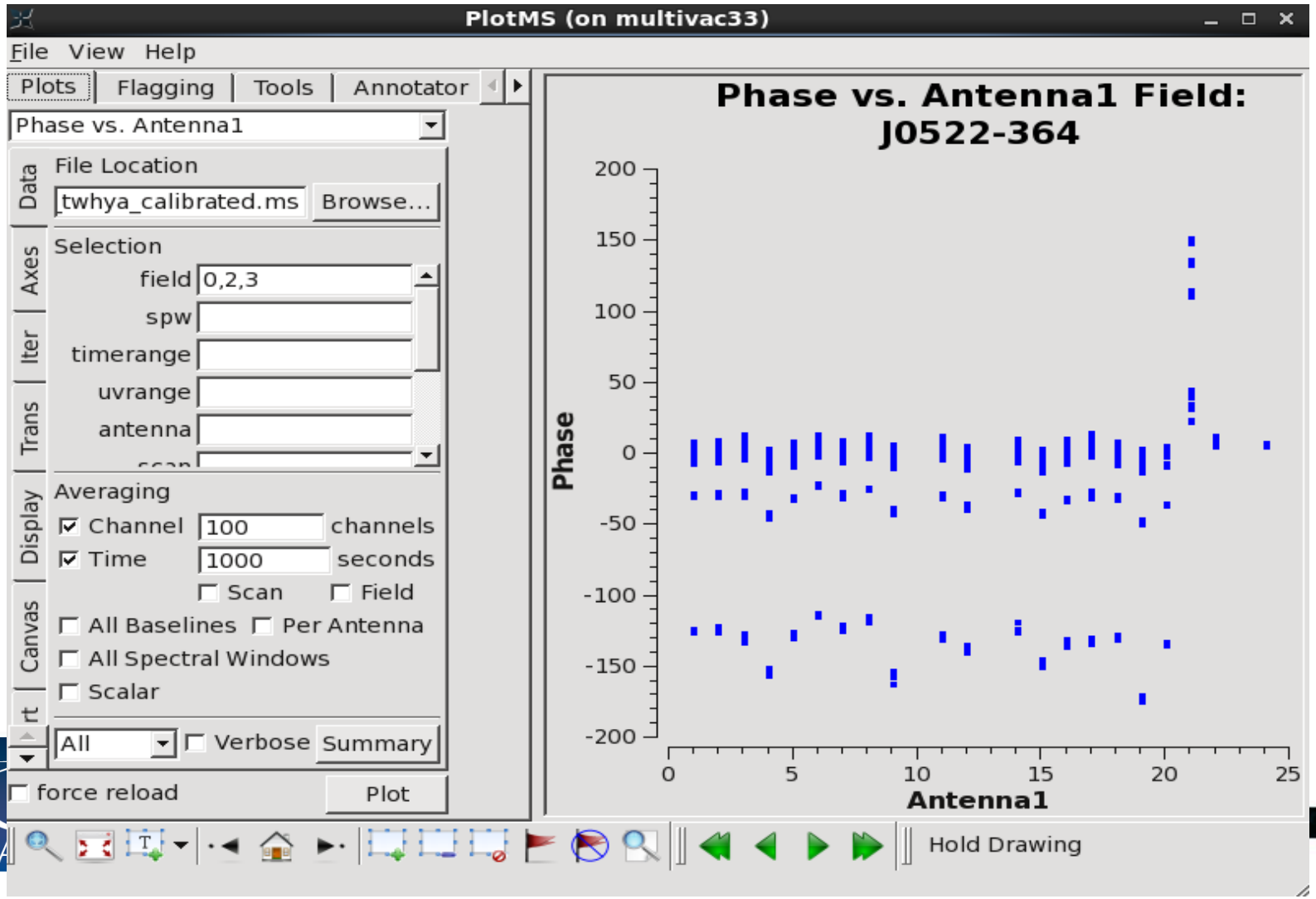
Amplitude vs. Antenna Plots…

You can see the problems with DV19 on field 0 here and you can also notice that DV01 has low amplitudes on field 3 (all amplitudes are low for this antenna). You may also notice that DV20 (antennae 22) shows some (but not all) low amplitudes, mostly on scan 30. We will note this and also plan to flag DV01.

Now look at the same plot in phase.

Phase vs. antenna1

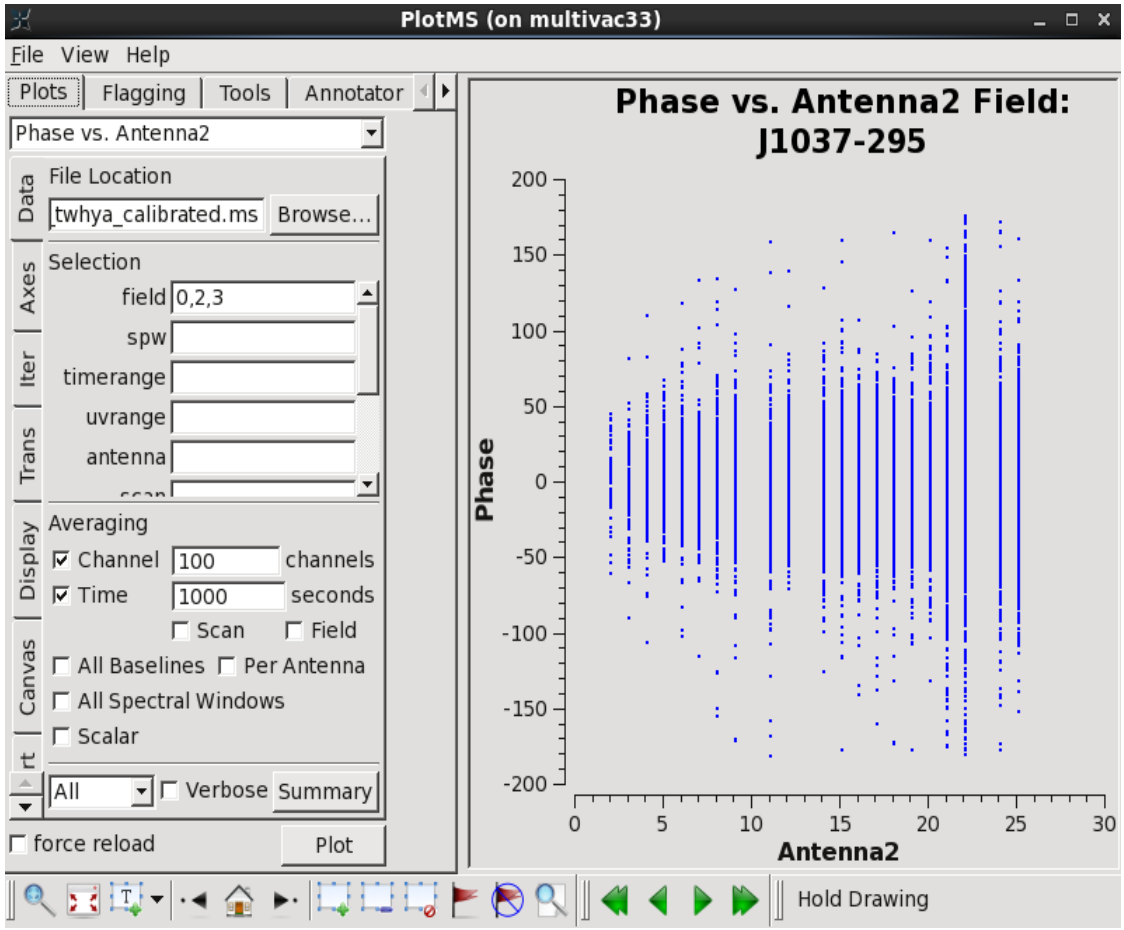
```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="antenna1", yaxis="phase",
    ydatacolumn="corrected",   field="0,2,3", averagedata=T, avgchannel="1e3", avgtime="1e3",
    iteraxis="field", coloraxis="corr")
```



Phase vs. antenna2

The issue with DV 22 is particularly evident in these phase plots on Field 3, especially looking at Antennae2 (because 22 is a high number it's more commonly found as Antennae2).

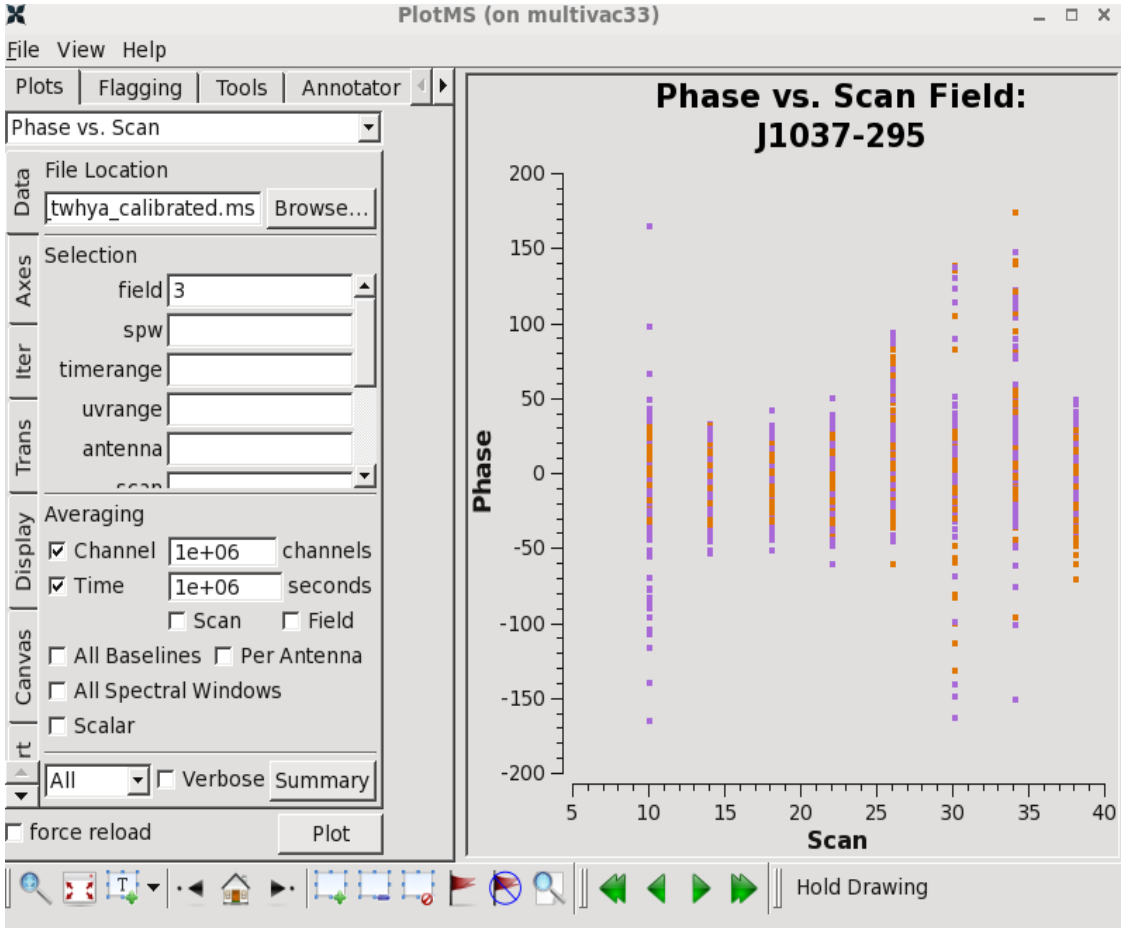
```
plotms(vis="sis14_twhya_calibrated.ms",
    xaxis="antenna2", yaxis="phase",
    ydatacolumn="corrected",
    field="0,2,3", averagedata=T,
    avgchannel="1e3", avgtime="1e3",
    iteraxis="field", coloraxis="corr")
```



Phase vs. Scan

You can see that likely DV19 is only really problematic in the first part of the track. Later on, scans 26 ~ 34, we see issues with DV20 (Antenna 22). We will plan to flag DV 20 over that range.

```
plotms(vis="sis14_twhya_calibrated.ms",
    xaxis="scan", yaxis="phase",
    ydatacolumn="data", field="3",
    antenna="", averagedata=T,
    avgchannel="1e6", iteraxis="field",
    coloraxis="corr", avgtime="1e6",
    avgscan=False)
```



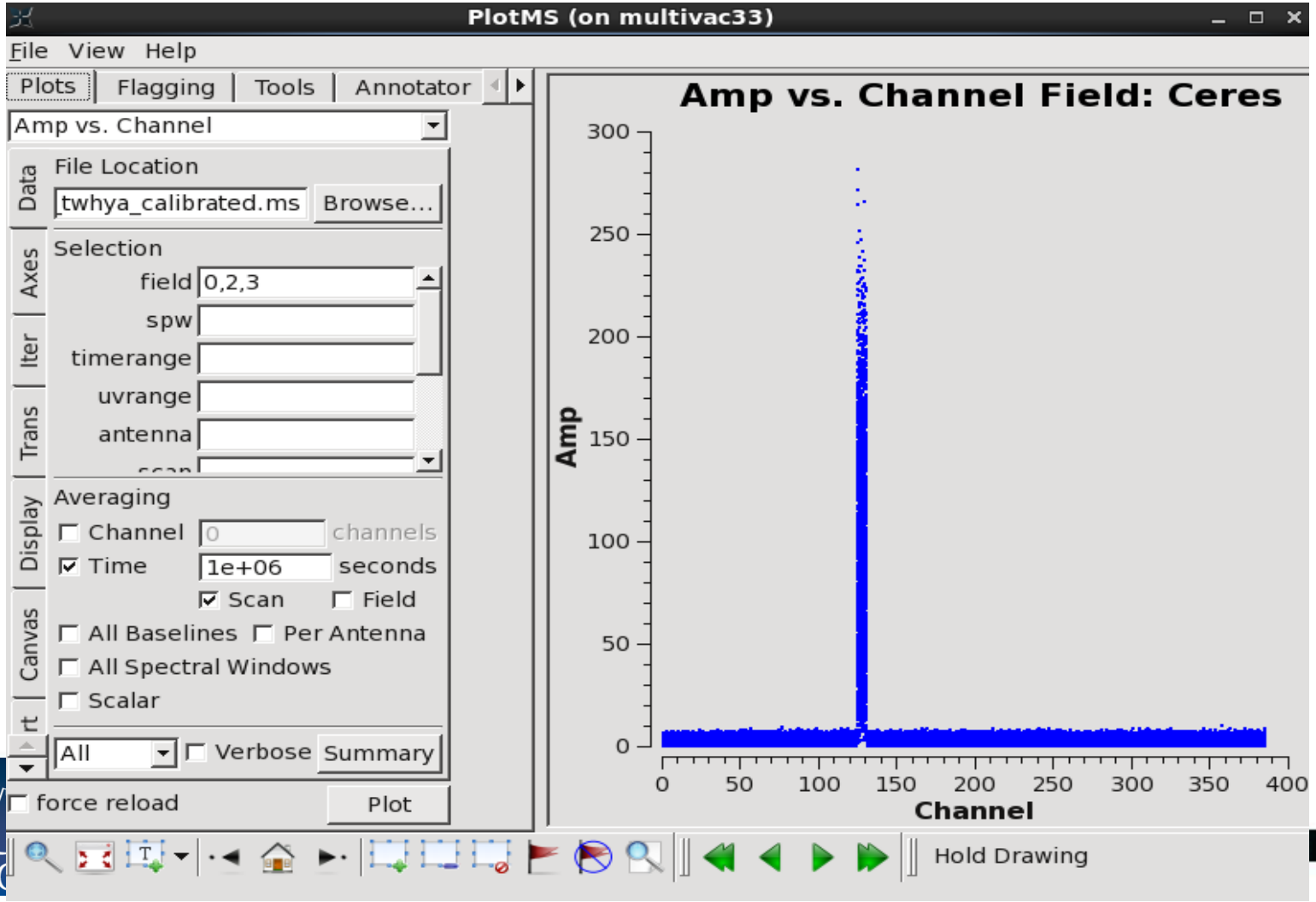
Lines or Spikes

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.

Field 2 (Ceres) shows an unexpected spike around channel 130. That seems spurious and we will want to flag this channel range.

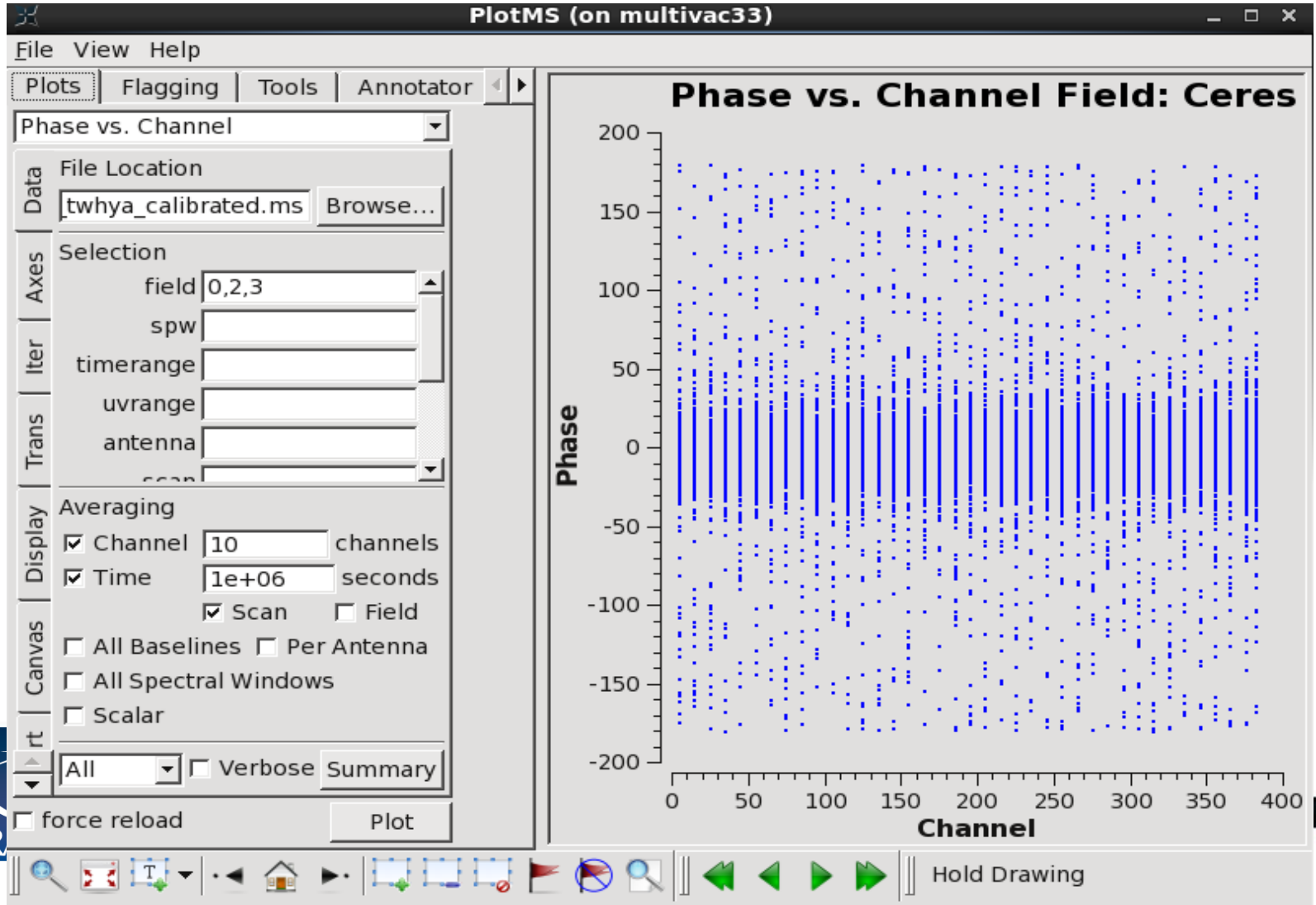
Amp vs. Channel

```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="channel", yaxis="amp", ydatacolumn="corrected",
    field="0,2,3", averagedata=T, avgtime="1e6", iteraxis="field", coloraxis="corr")
```



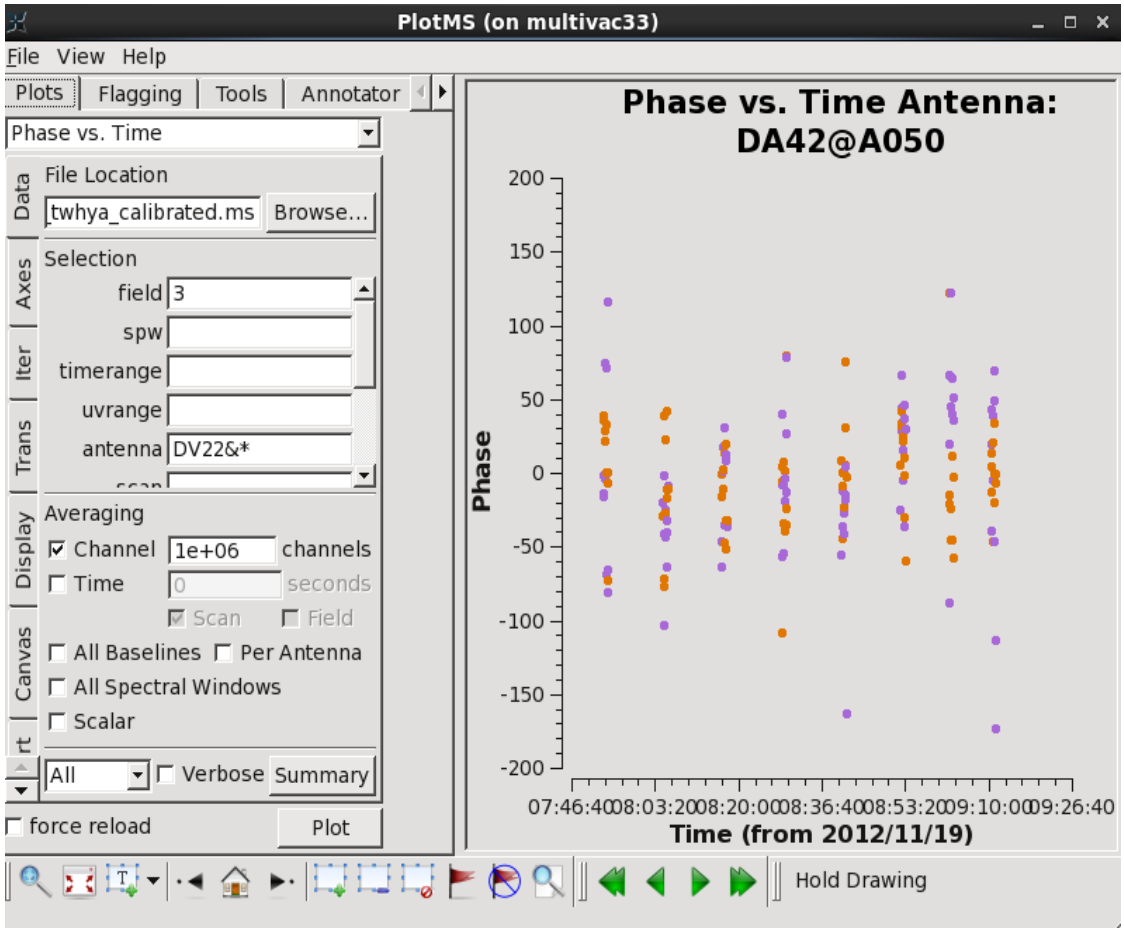
Phase vs. Channel

```
plotms(vis="sis14_twhya_calibrated.ms", xaxis="channel", yaxis="phase",
    ydatacolumn="corrected",   field="0,2,3", averagedata=T, avgtime="1e6",avgchannel="10",
    iteraxis="field", coloraxis="corr")
```



Phase vs. Time

Finally, one can look at the continuity of the phase vs. time of each antenna paired with the reference. This is a good way to really dig into the data, but can be overwhelming. The previous plots (which show all data together) are probably your best first line of defense.



- `plotms(vis="sis14_twhya_calibrated.ms", xaxis="time", yaxis="phase", ydatacolumn="data", field="3", antenna="DV22&*", averagedata=T, avgchannel="1e6", iteraxis="antenna", coloraxis="corr", avgscan=True)`

Flag your Data

We decided to flag DV19 and DV01 for all scans, DV20 for scans 26-34, and channels 124-130 on Ceres (Field 2). We do this using the flagdata command in its "manual" mode.

First flag the two antennas entirely.

- `flagdata("sis14_twhya_calibrated.ms", antenna="DV01,DV19")`

Now specify a scan range for DV20.

- ```
  flagdata("sis14_twhya_calibrated.ms",
       antenna="DV20",
       scan="27~34")
  ```

Finally, pick a field and a channel/spw range for Ceres.

- ```
  flagdata("sis14_twhya_calibrated.ms",
       field="2",
       spw="0:124~130")
  ```

Flag your Data

We could split out the flagged data here, but we would rather take the knowledge of these flags back to the beginning of the calibration process. That is the next lesson.

For now, go repeat the commands above to see the effect of your data flagging and convince yourself that these commands will remove most of the problems that you see in the data.

Run through the calibration scheme now "end-to-end" in CASA

- This final lesson before we get to imaging will step you through a realistic calibration workflow. As such, there will be little in the way of plots in this part of the presentation.
- You will start with the uncalibrated data. Then you will execute a calibration script.
- After you inspect the data - as in the last tutorial - you will apply some flagging and then rerun the calibration.
- At the end, we will split out the calibrated and flagged data for further imaging.

Run Calibration Script

The next few slides on calibration are all steps inside “calibration_script.py”. You can run this script with the commands:

- `vis = "sis14_twhya_uncalibrated"`
- `execfile("calibration_script.py")`

Calibrate without Flags

Start by removing previous calibrations

- `vis = "sis14_twhya_uncalibrated"`
- `clearcal(vis+".ms")`

Bandpass Calibration

A short-timescale phase solution

- `os.system("rm -rf phase_int_bp.cal")`
- ```
  gaincal(vis=vis+".ms",
      caltable="phase_int_bp.cal",
      field="0",
      solint="int",
      calmode="p",
      refant="DV22",
      gaintype="G")
  ```


  ```

Bandpass Calibration

Calibrate the bandpass

- ```
  os.system("rm -rf bandpass_10chan.cal")
  ```
- ```
  bandpass(vis=vis+".ms",
       caltable="bandpass_10chan.cal",
       field="0",
       refant="DV22",
       solint="inf,10chan",
       combine="scan",
       gaintable=["phase_int_bp.cal"])
  ```

Bandpass Calibration

Apply!

- ```
  applycal(vis=vis+".ms",
       gaintable=["bandpass_10chan.cal"],
       interp=["nearest"],
       gainfield=["0"])
  ```

- ```
  os.system("rm -rf "+vis+"_bpcal.ms")
  ```
- ```
  split(vis=vis+".ms",
    datacolumn="corrected",
    outputvis=vis+"_bpcal.ms",
    keepflags=False)
  ```


  ```

Set Calibrator Fluxes

Look up the model for Ceres

- ```
  setjy(vis=vis+"_bpcal.ms",
    field="2",
    standard="Butler-JPL-Horizons 2012",
    usescratch=True)
  ```

Set the model for the bandpass calibrator

- ```
  setjy(vis=vis+"_bpcal.ms",
    field="0",
    fluxdensity = [8.43,0,0,0],
    usescratch=True)
  ```

Set the model for the secondary calibrator

- ```
  setjy(vis=vis+"_bpcal.ms",
    field="3",
    fluxdensity = [0.65,0,0,0],
    usescratch=True)
  ```


  ```

Phase and Amplitude

Derive a short-timescale phase solution

- `os.system("rm -rf phase_int.cal")`
- ```
  gaincal(vis=vis+"_bpcal.ms",
      caltable="phase_int.cal",
      field="0,2,3",
      solint="int",
      calmode="p",
      refant="DV22",
      gaintype="G")
  ```


  ```

Phase and Amplitude

Calibrate the phase

- `os.system("rm -rf phase_scan.cal")`
- ```
  gaincal(vis=vis+"_bpcal.ms",
      caltable="phase_scan.cal",
      field="0,2,3",
      solint="inf",
      calmode="p",
      refant="DV22",
      gaintype="G")
  ```


  ```

Phase and Amplitude

Calibrate the amplitude

- `os.system("rm -rf amp_scan.cal")`
- ```
  gaincal(vis=vis+"_bpcal.ms",
      caltable="amp_scan.cal",
      field="0,2,3",
      solint="inf",
      calmode="a",
      refant="DV22",
      gaintype="G",
      gaintable=["phase_int.cal"])
  ```


  ```

Application

scan based applied to everything

- ```
  applycal(vis=vis+"_bpcal.ms",
       gaintable=["phase_scan.cal",  "amp_scan.cal"],
       interp=["linear",  "linear"],
       gainfield=[[],[]],
       applymode='calonly')
  ```

- Now inspect the data, following the previous lessons. Go back and review, or try a few of the same commands from those lessons.


  ```

Flagging

Here we apply the already worked-out flagging

First flag the two antennas entirely.

- ```
  flagdata("sis14_twhya_uncalibrated.ms",
       antenna="DV01,DV19")
  ```

Now specify a scan range for DV20.

- ```
  flagdata("sis14_twhya_uncalibrated.ms",
       antenna="DV20",
       scan="27~34")
  ```

Finally, pick a field and a channel/spw range for Ceres.

- ```
  flagdata("sis14_twhya_uncalibrated.ms",
       field="2",
       spw="0:124~130")
  ```


  ```

Rerun calibration with Flags

After flagging problematic data we want to rerun the entire calibration with the problem data removed (those data could affect the calibration of other antennas, so that removing them will improve the overall data quality).

- `vis = "sis14_twhya_uncalibrated"`
- `execfile("calibration_script.py")`

Split Out Calibrated Data

Finally, let's split out the calibrated, flagged data. These should now be ready for imaging.

```
os.system("rm -rf sis14_twhya_calibrated_and_flagged.ms")
split(vis="sis14_twhya_uncalibrated_bpcal.ms",
      outputvis="sis14_twhya_calibrated_and_flagged.ms",
      datacolumn="corrected",
      keepflags=False)
```

Basic Imaging

ALMA Data Reduction Tutorials

Synthesis Imaging Summer School

May 16, 2014

Atacama Large Millimeter/submillimeter Array

Expanded Very Large Array

Robert C. Byrd Green Bank Telescope

Very Long Baseline Array

Imaging in CASA

- Goals of this lesson:
  - Introduce deconvolution in CASA (clean)
  - Introduce various imaging methods available in CASA

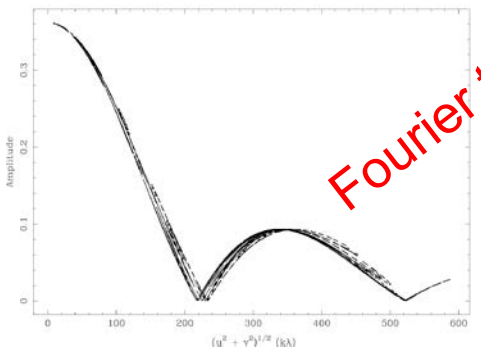
How to analyze (imperfect) interferometer data?

- 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)

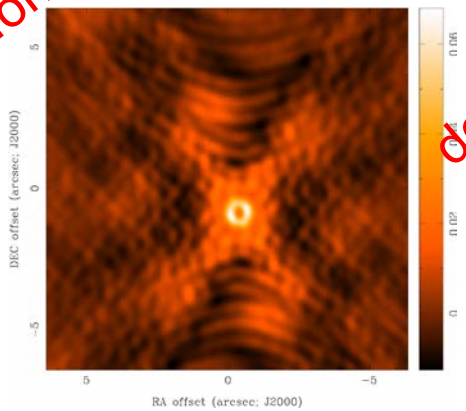
visibilities

dirty image

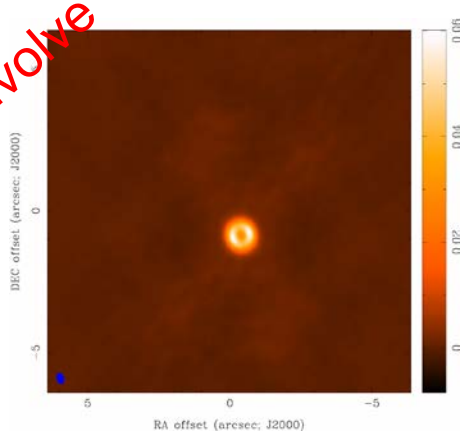
sky brightness



Fourier transform



deconvolve

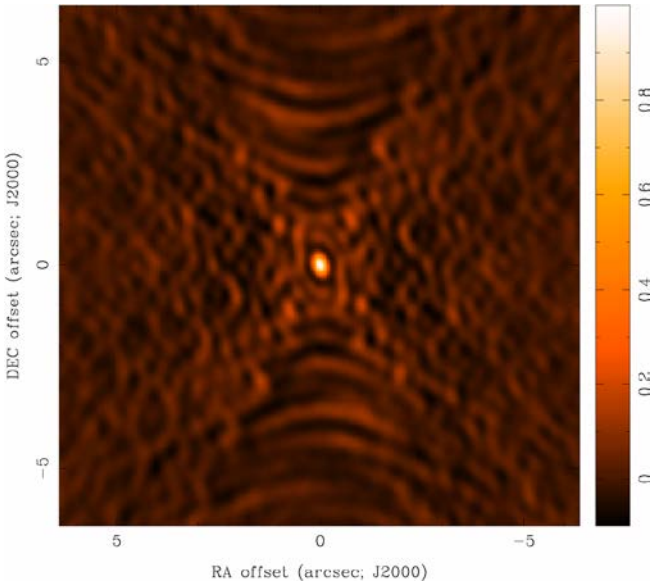


Basic CLEAN Algorithm

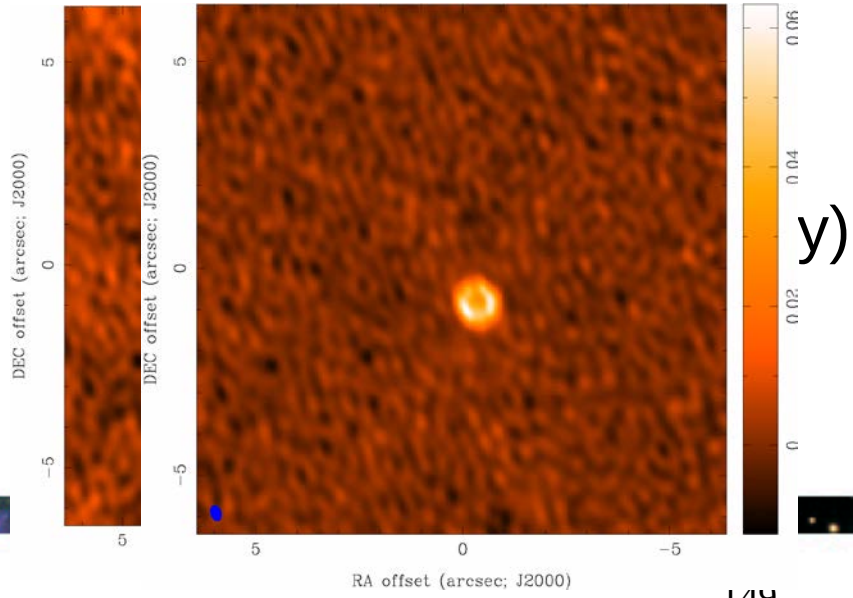
① Initialize a *residual* map to the dirty map

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



b(x,y)



y)

Basic CLEAN Algorithm (cont)

- stopping criteria
  - r*esidual* map max < multiple of rms (when noise limited)
  - r*esidual* 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")
  - very useful but potentially dangerous!

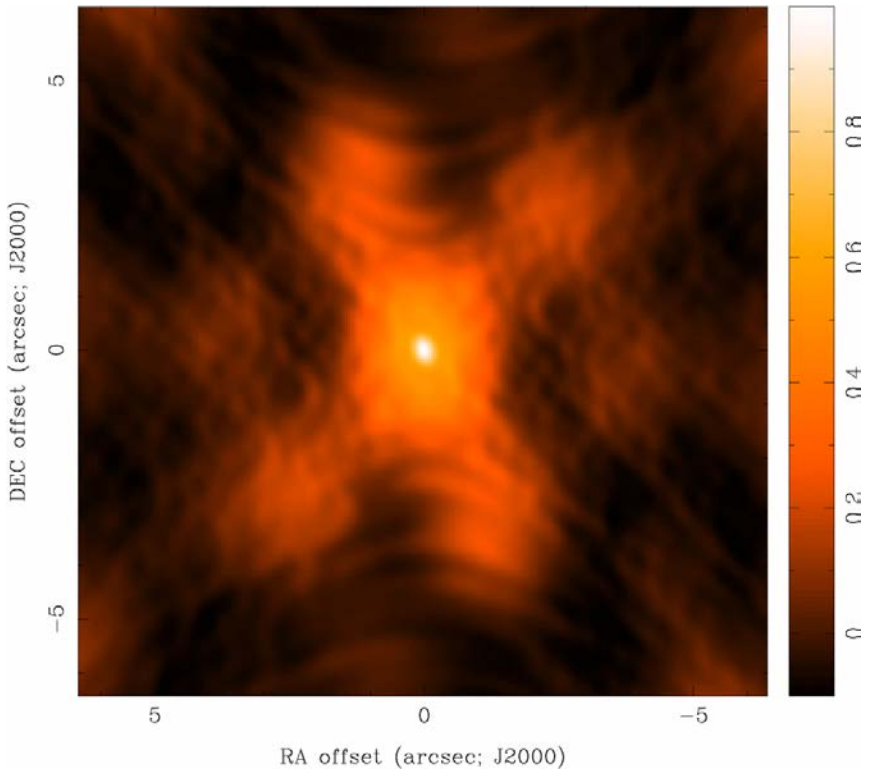
Dirty Beam Shape and Weighting

- Each visibility point is given a weight in the imaging step

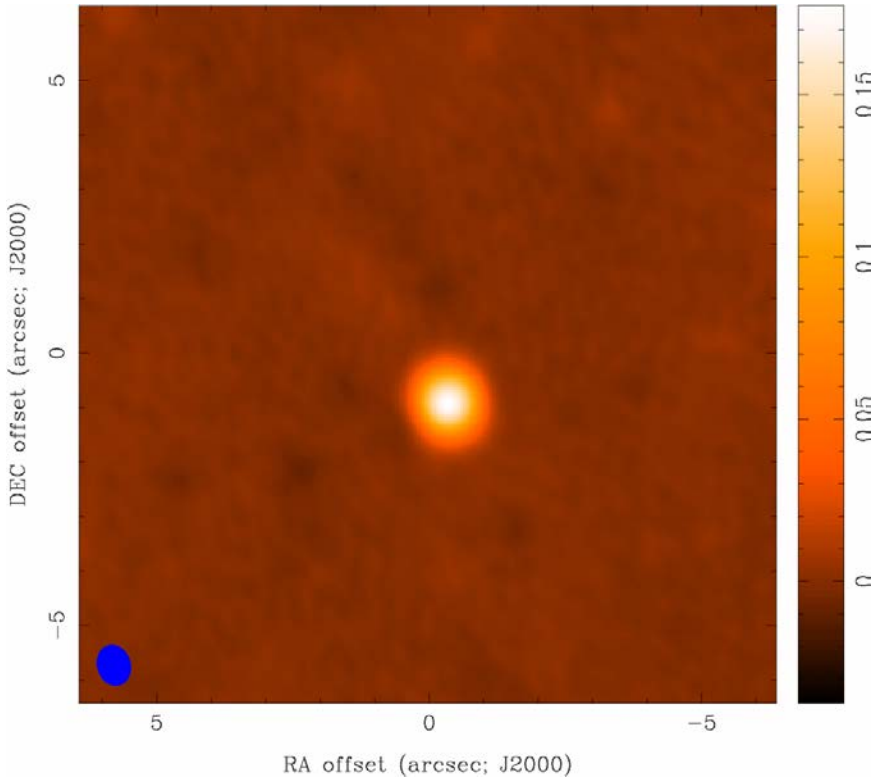
- Natural
  - Weights inversely proportional to noise variance
  - Best point-source sensitivity; poor beam characteristics
- Uniform
  - Weights inversely proportional to sampling density (longer baseline are given higher weight than in natural)
  - Best resolution; poorer noise characteristics
- Briggs (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

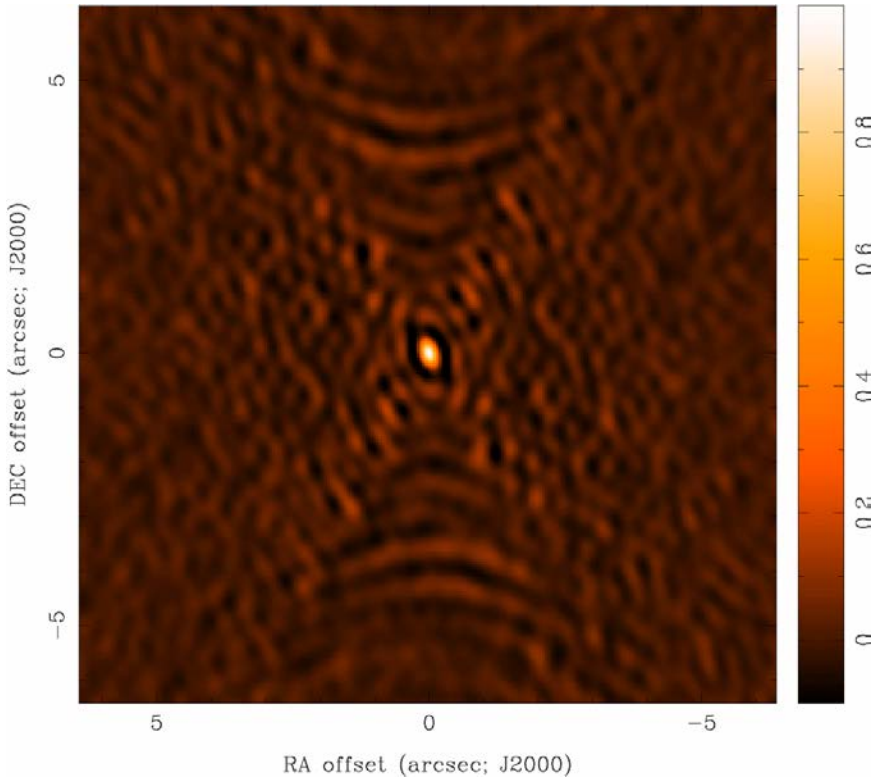


CLEAN image

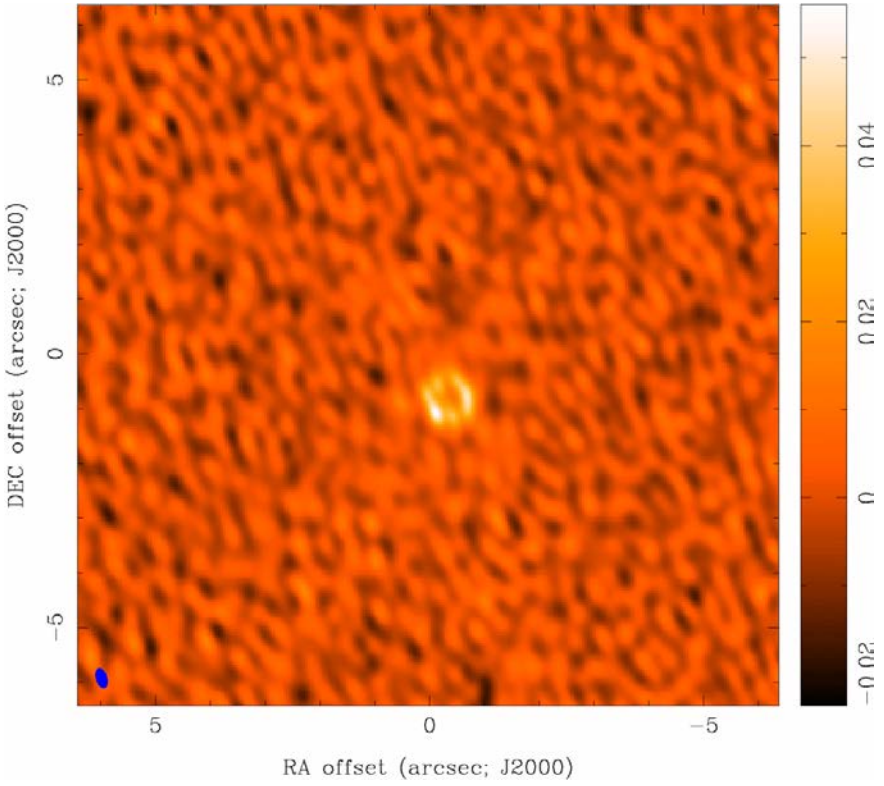


Imaging Results

Uniform Weight Beam

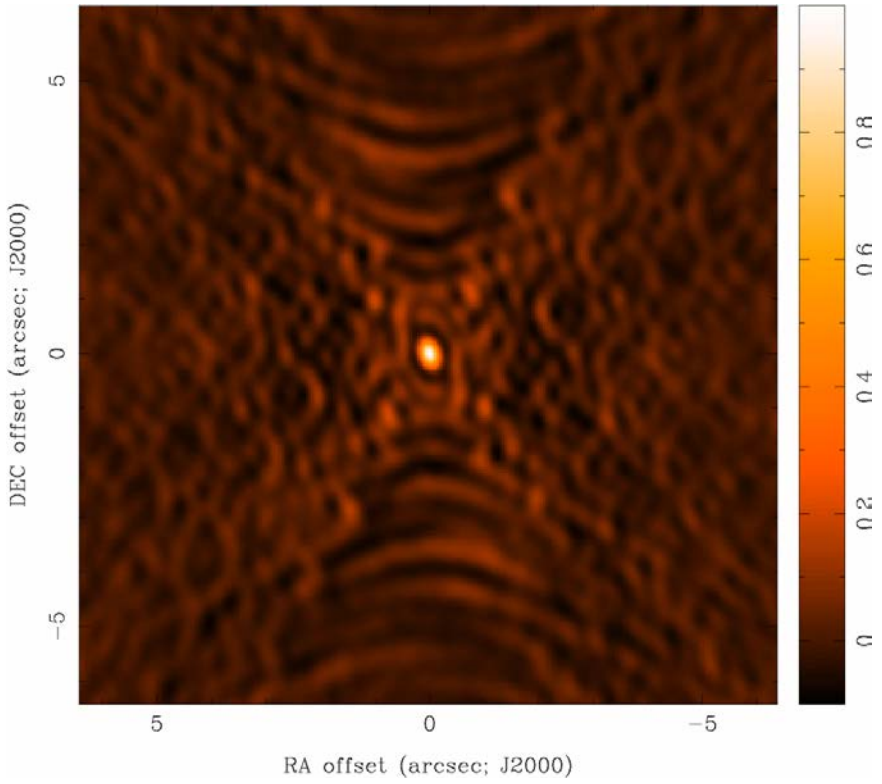


CLEAN image

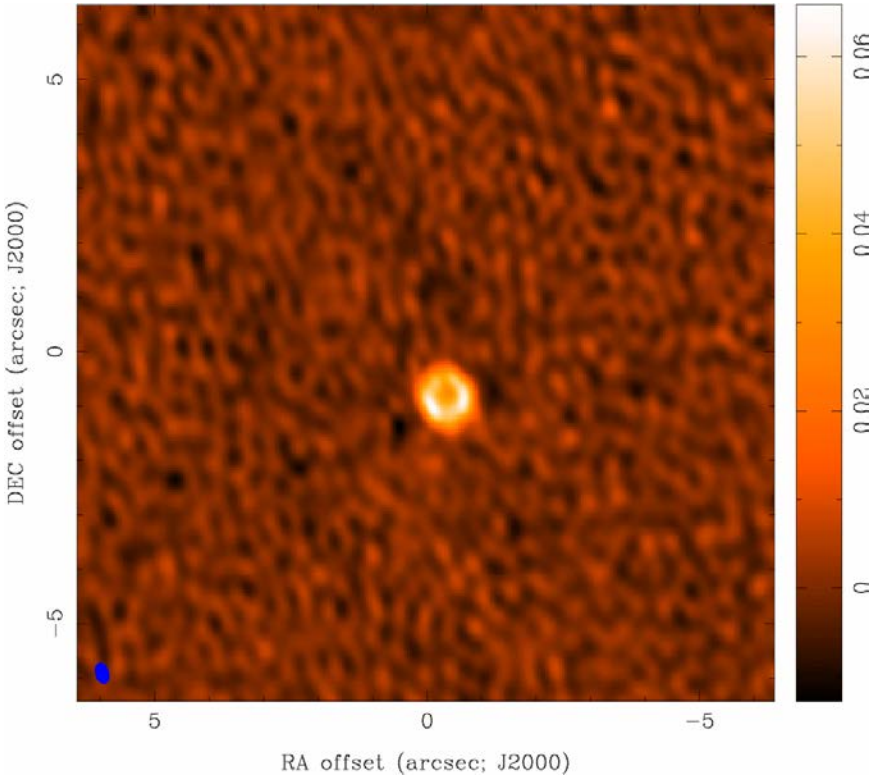


Imaging Results

Robust=0 Beam



CLEAN image



Clean in CASA:

```
CASA <13>: inp clean
---------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''            #  Name of input visibility file
imagename           =         ''            #  Pre-name of output images
outlierfile         =         ''            #  Text file with image names, sizes, centers for outliers
field               =         ''            #  Field Name or id
spw                 =         ''            #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False            #  Other data selection parameters
mode                =      'mfs'            #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1            #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''            #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''            #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500            #  Maximum number of iterations
gain                =        0.1            #  Loop gain for cleaning
threshold           =   '0.0mJy'            #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'            #  Method of PSF calculation to use during minor cycles
imagermode          =  'csclean'            #  Options: 'csclean' or 'mosaic', '', uses psfmode
     cyclefactor    =        1.5            #  Controls how often major cycles are done. (e.g. 5 for frequently)
     cyclespeedup   =         -1            #  Cycle threshold doubles in this number of iterations

multiscale          =         []            #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False            #  Use interactive clean (with GUI viewer)
mask                =         []            #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]            #  x and y image size in pixels.  Single value: same for both
cell                = ['1.0arcsec']         #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''            #  Image center: direction or field index
restfreq            =         ''            #  Rest frequency to assign to image (see help)
stokes              =        'I'            #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'            #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False            #  Apply additional uv tapering of visibilities
modelimage          =         ''            #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']            #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False            #  Output primary beam-corrected image
minpb               =        0.2            #  Minimum PB level to use
usescratch          =      False            #  True if to save model visibilities in MODEL_DATA column
allowchunk          =      False            #  Divide large image cubes into channel chunks for deconvolution
async               =      False            #  If true the taskname must be started using clean(...)
```

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)

Maximum 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 MAS begin to be resolved out.
- All flux on scales larger than λ/B_{min} (~2 x MAS) completely resolved out.

Basic Imaging

Copy the calibrated and flagged data from the working directory. This is our best version of the data.

```
os.system("cp -r ../../working_data/sis14_twhya_calibrated_flagged.ms .")
```

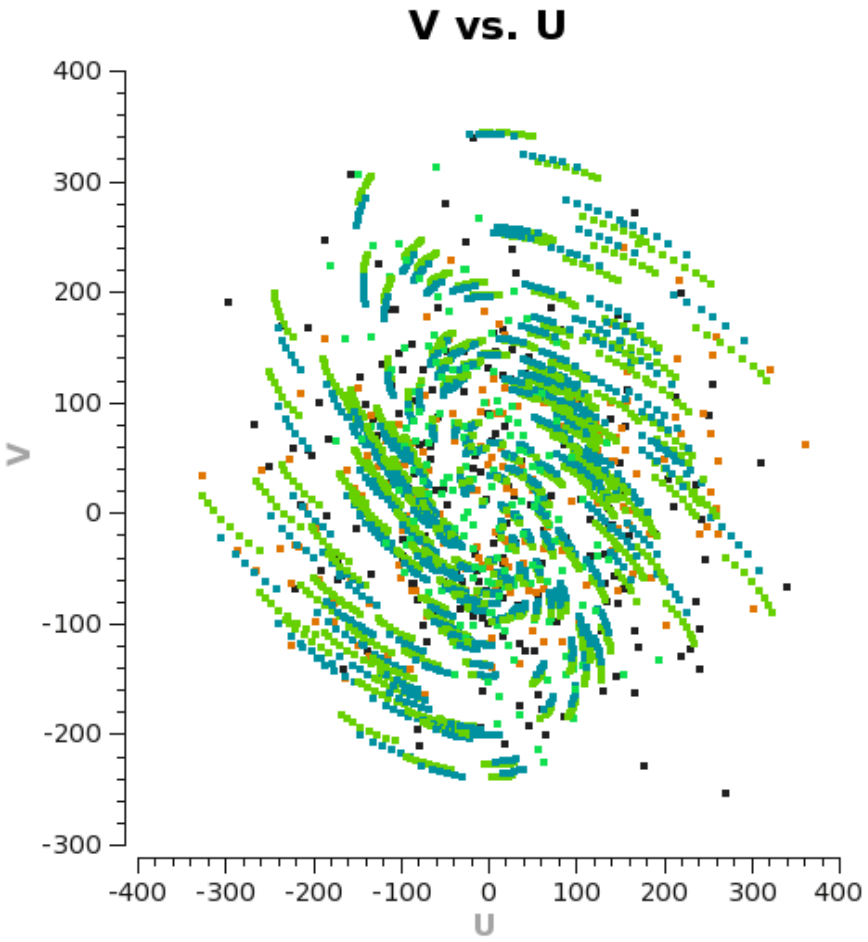
Orient yourself

```
listobs('sis14_twhya_calibrated_flagged.ms')
```

```
                                                                                .
                                                                                .
                                                                                .
INFO listobs        Fields: 5
INFO listobs          ID   Code Name            RA              Decl          Epoch   SrcId      nRows
INFO listobs          0    none J0522-364       05:22:57.984648 -36.27.30.85128 J2000   0          4200
INFO listobs          2    none Ceres           06:10:15.950590 +23.22.06.90668 J2000   2          3800
INFO listobs          3    none J1037-295       10:37:16.079736 -29.34.02.81316 J2000   3          16000
INFO listobs          5    none TW Hya          11:01:51.796000 -34.42.17.36600 J2000   4          53161
INFO listobs          6    none 3c279           12:56:11.166576 -05.47.21.52464 J2000   5          3402
INFO listobs        Spectral Windows:  (1 unique spectral windows and 1 unique polarization setups)
INFO listobs          SpwID  Name                          #Chans   Frame   Ch0(MHz)  ChanWid(kHz)  TotBW(kHz) BBC Num  Corrs
INFO listobs          0      ALMA_RB_07#BB_2#SW-01#FULL_RES   384   TOPO  372533.086      610.352    234375.0      2  XX  YY
                                                                                .
                                                                                .
                                                                                .
```

Basic Imaging

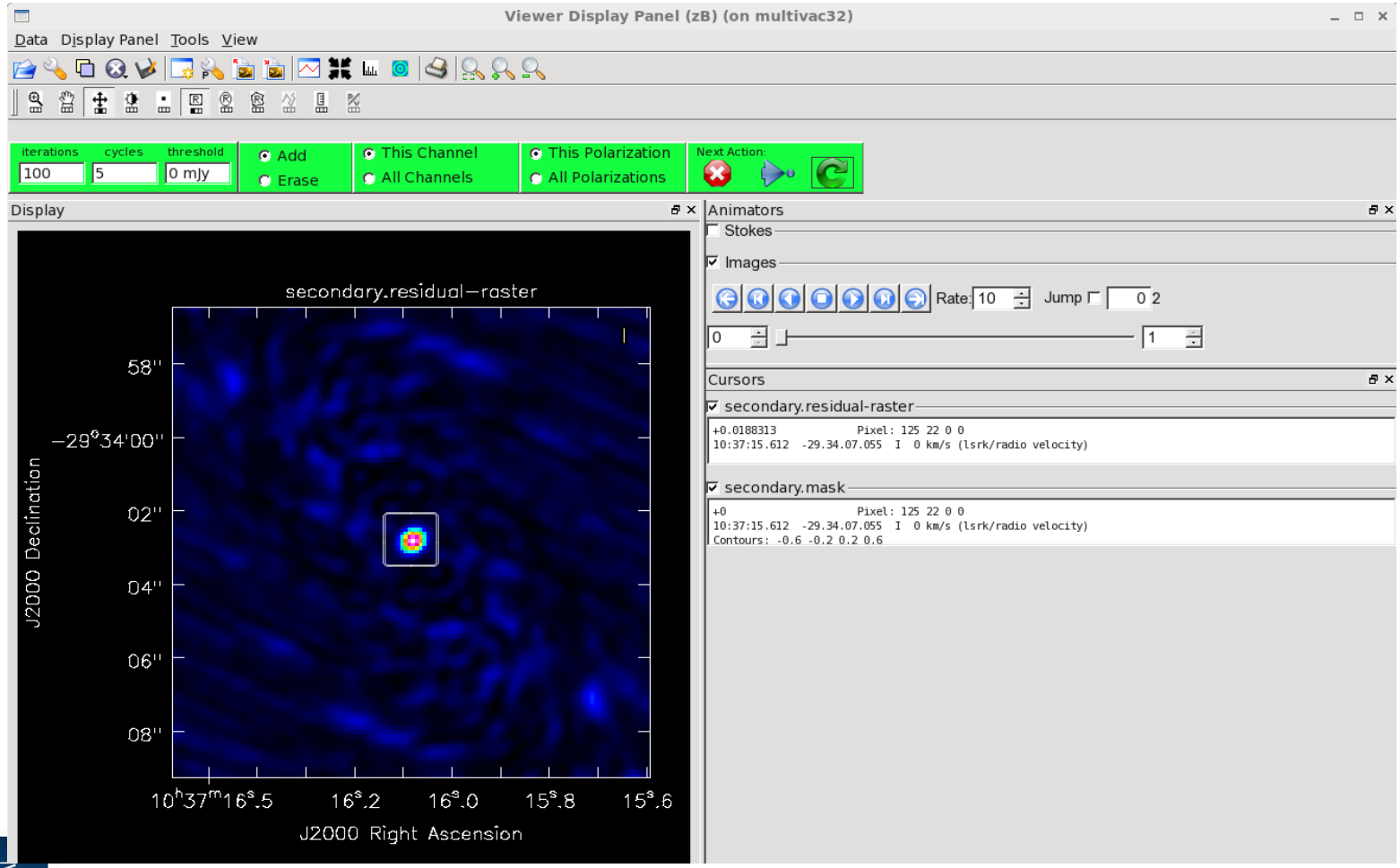
```
plotms(vis='sis14_twhya_calibrated_flagged.ms', xaxis='u', yaxis='v',
       avgchannel='10000', avgspw=False, avgtime='1e9', avgscan=False,
       coloraxis="field")
```



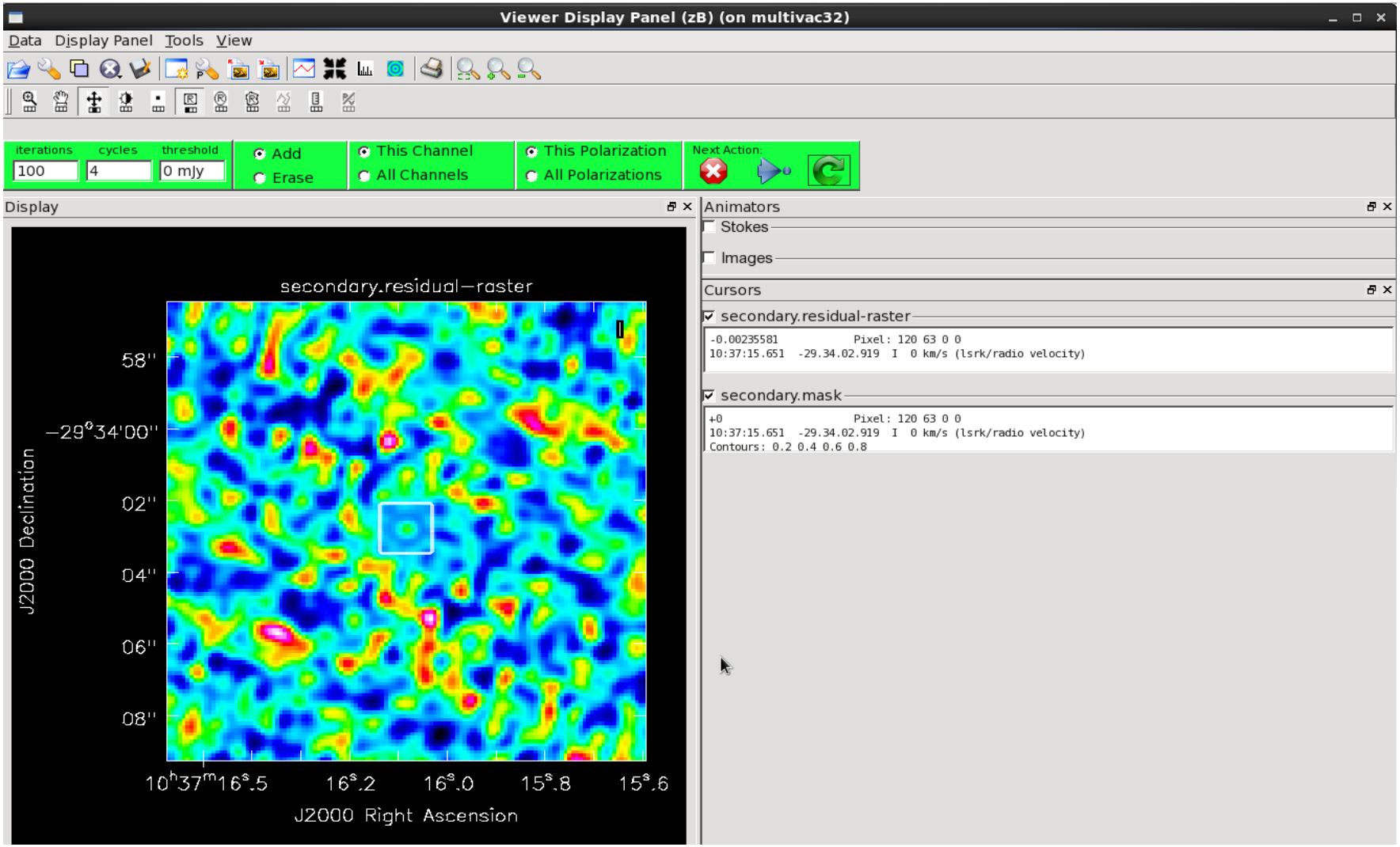
Plot the u-v coverage

Basic Imaging

```
clean(vis='sis14_twhya_calibrated_flagged.ms', imagename='secondary', field='3',
      spw='', mode='mfs', nterms=1, imsize=[128,128], cell=['0.1arcsec'], weighting='natural',
      threshold='0mJy', interactive=True)
```



Basic Imaging



Clean Residuals

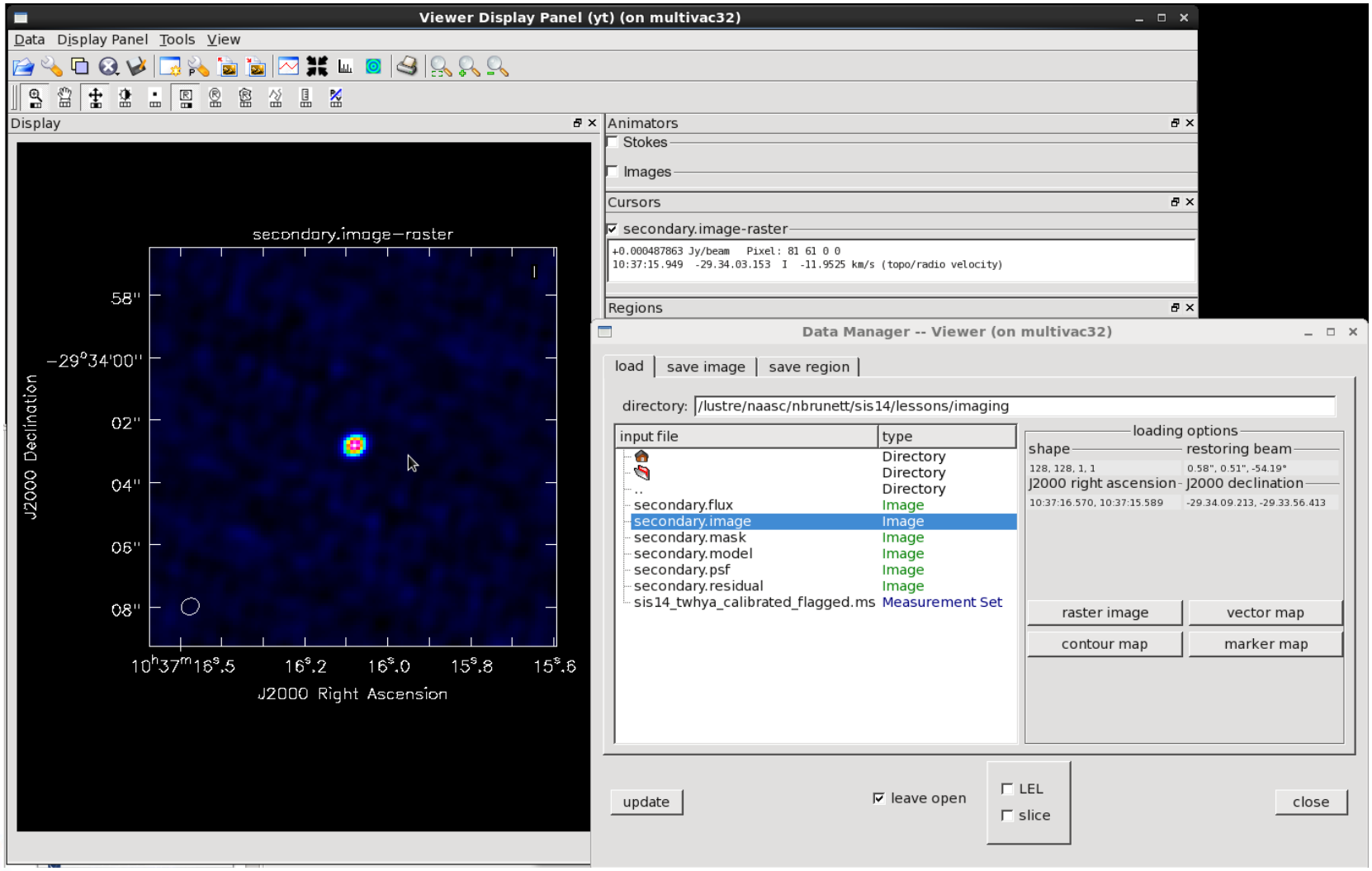
Output of *clean*

Minimally:

• my_image.flux	Relative sky sensitivity - shows the primary beam response
• my_image.image	Cleaned and restored image
• my_image.mask	Clean "boxes" shows where you cleaned
• my_image.model	Clean components - the model used by clean (in Jy/pixel)
• my_image.psf	Dirty beam - shows the synthesized beam
• my_image.residual	Residual shows what was left after you cleaned (the "dirty" part of the final image)

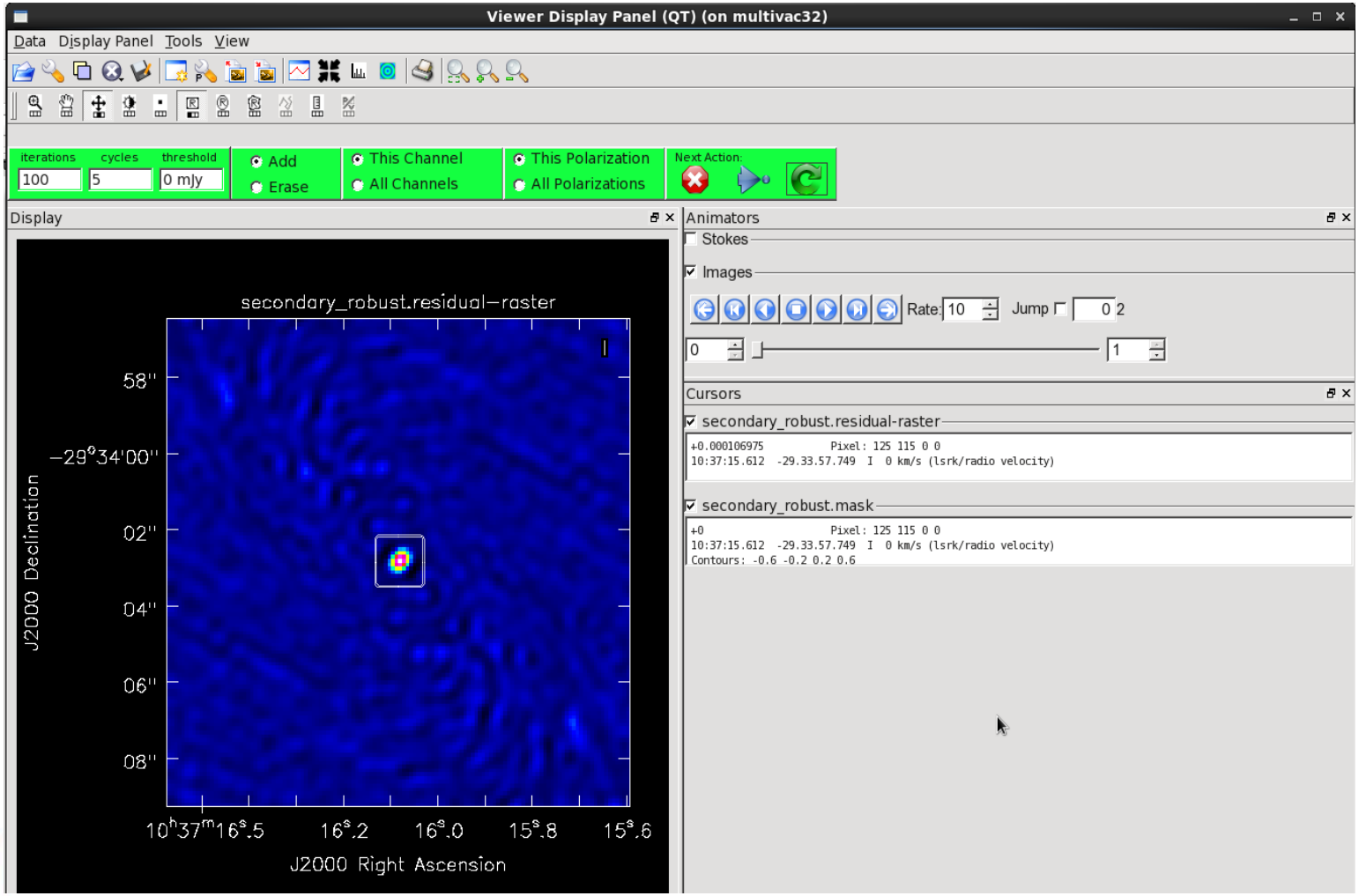
Basic Imaging

```
imview("secondary.image")
```

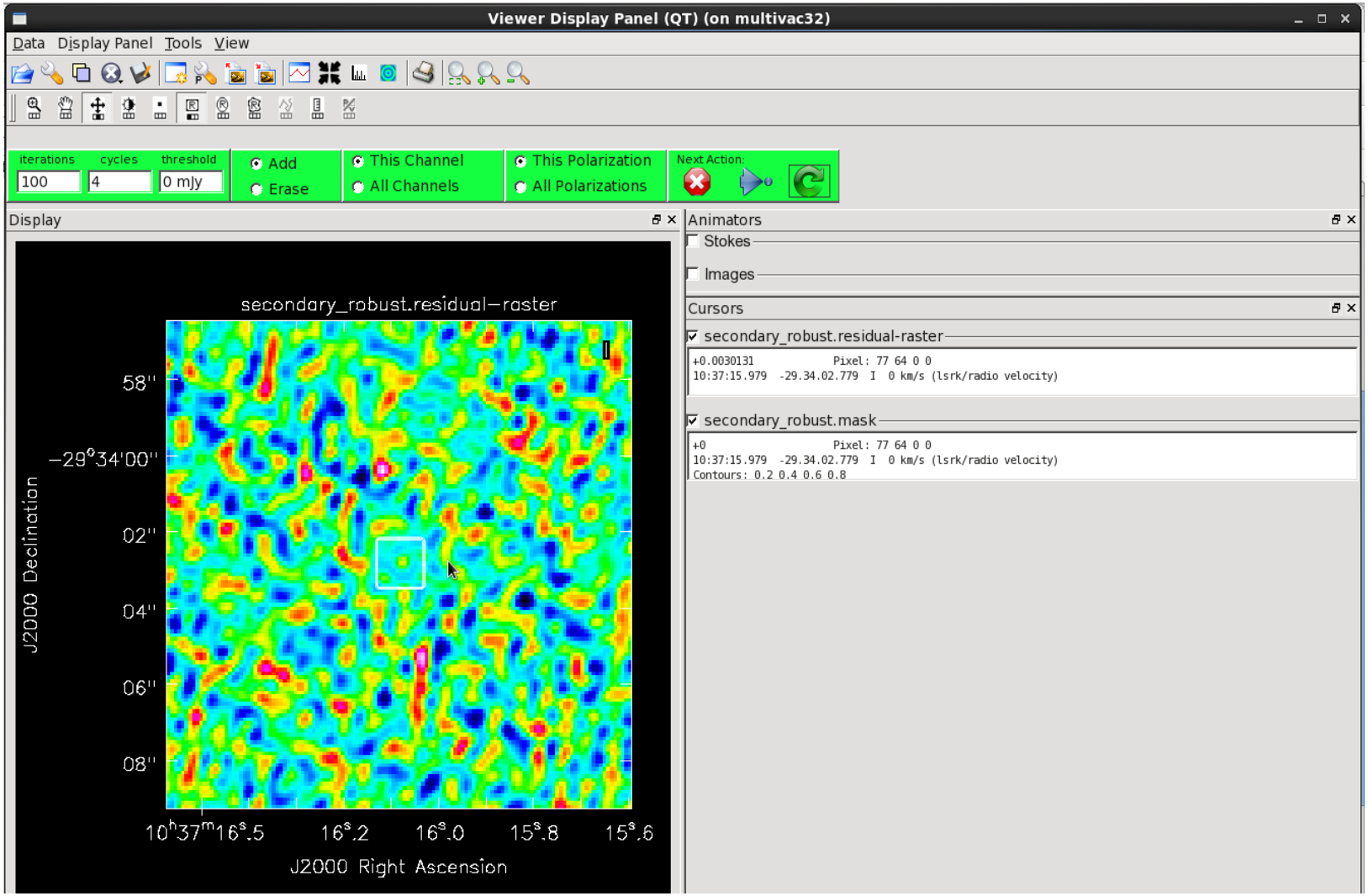


Basic Imaging

```
clean(vis='sis14_twhya_calibrated_flagged.ms', imagename='secondary_robust', field='3', spw='',
      mode='mfs', nterms=1, imsize=[128,128], cell=['0.1arcsec'], weighting='briggs', robust=-1.0,
      threshold='0mJy', interactive=True)
```



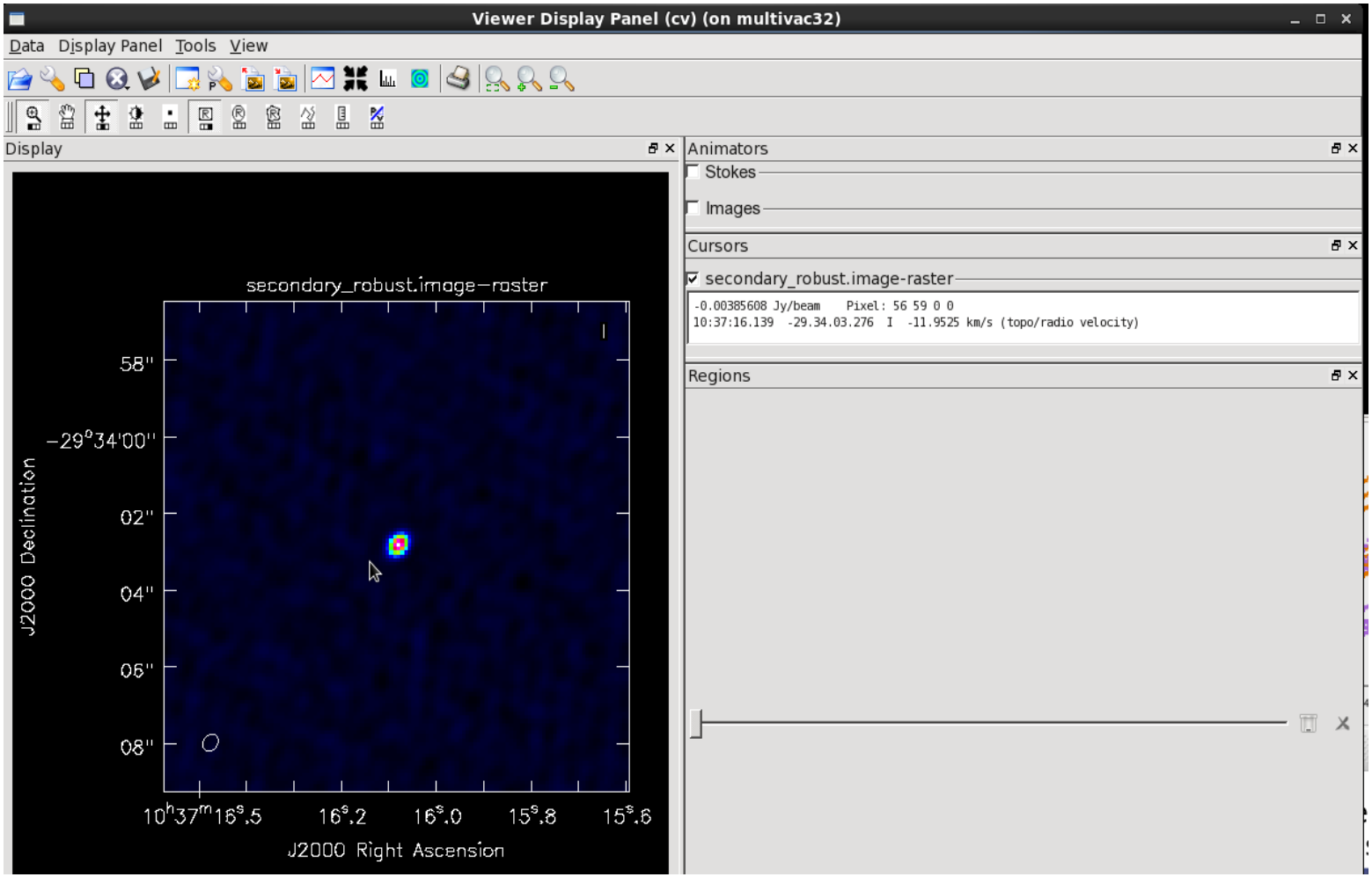
Basic Imaging



Clean Residuals

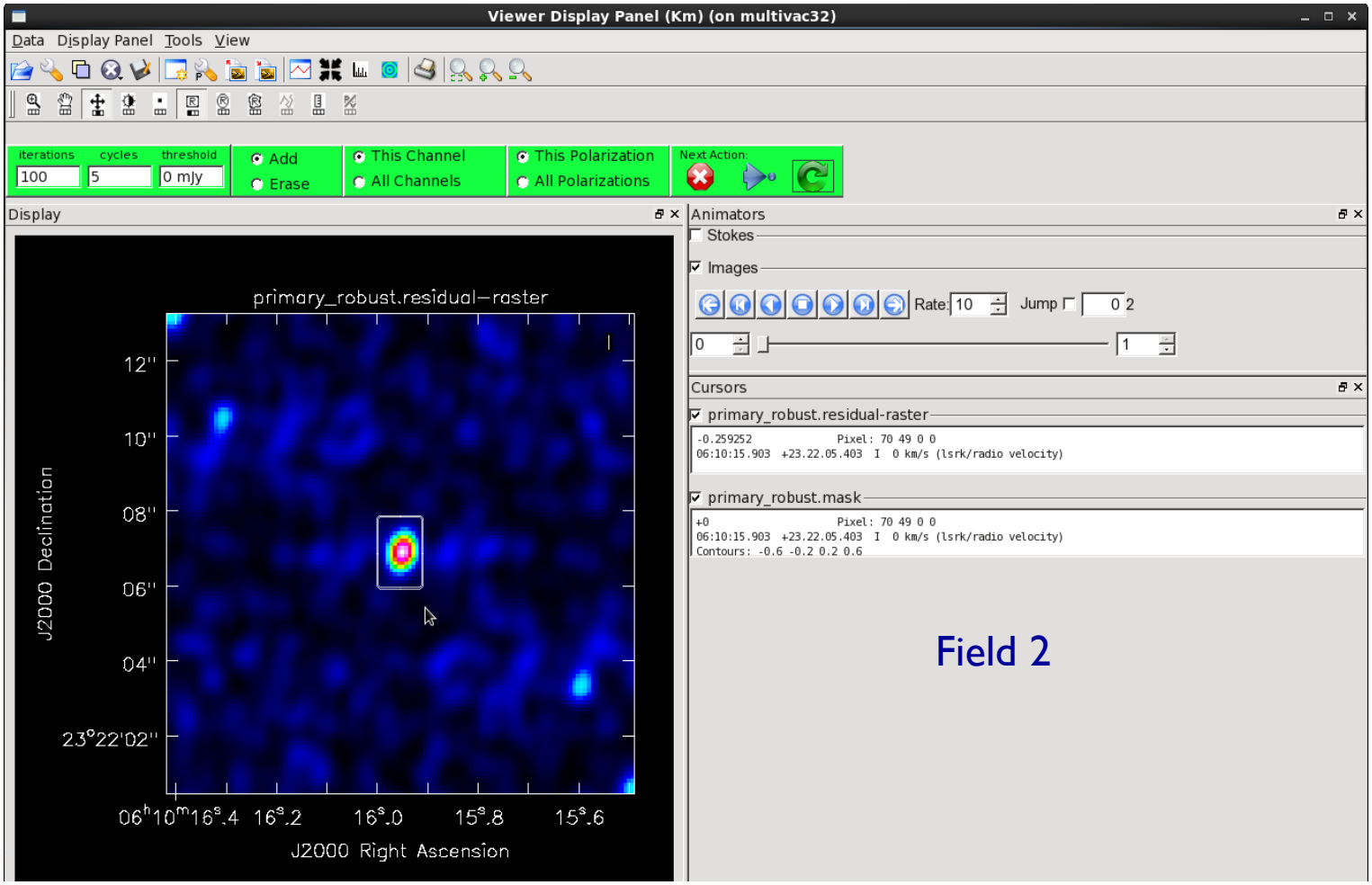
Basic Imaging

```
imview("secondary_robust.image")
```



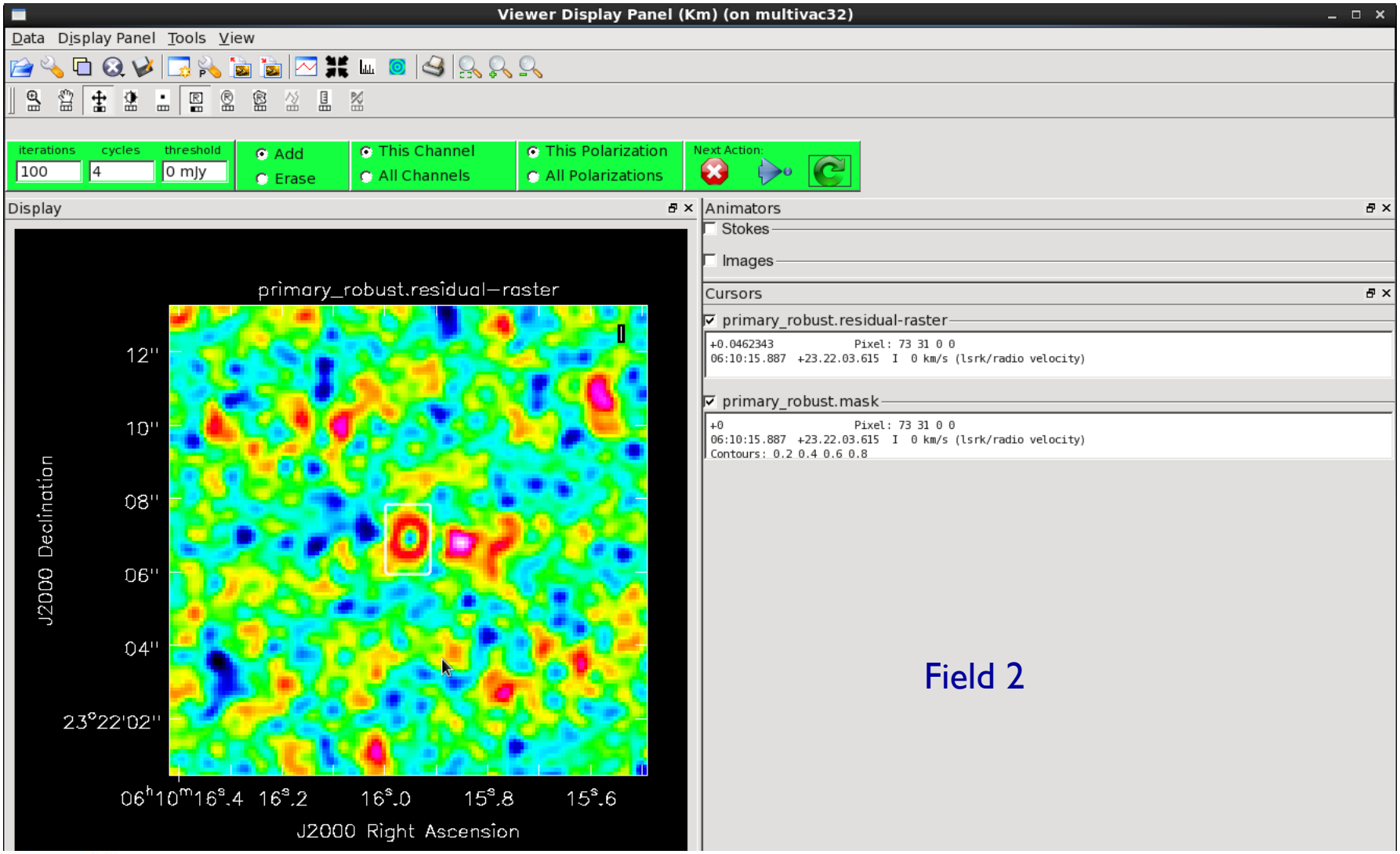
Basic Imaging

```
clean(vis='sis14_twhya_calibrated_flagged.ms', imagename='primary_robust', field='2', spw='',
      mode='mfs', nterms=1, imsize=[128,128], cell=['0.1arcsec'], weighting='natural',
      threshold='0mJy', interactive=True)
```



Field 2

Basic Imaging

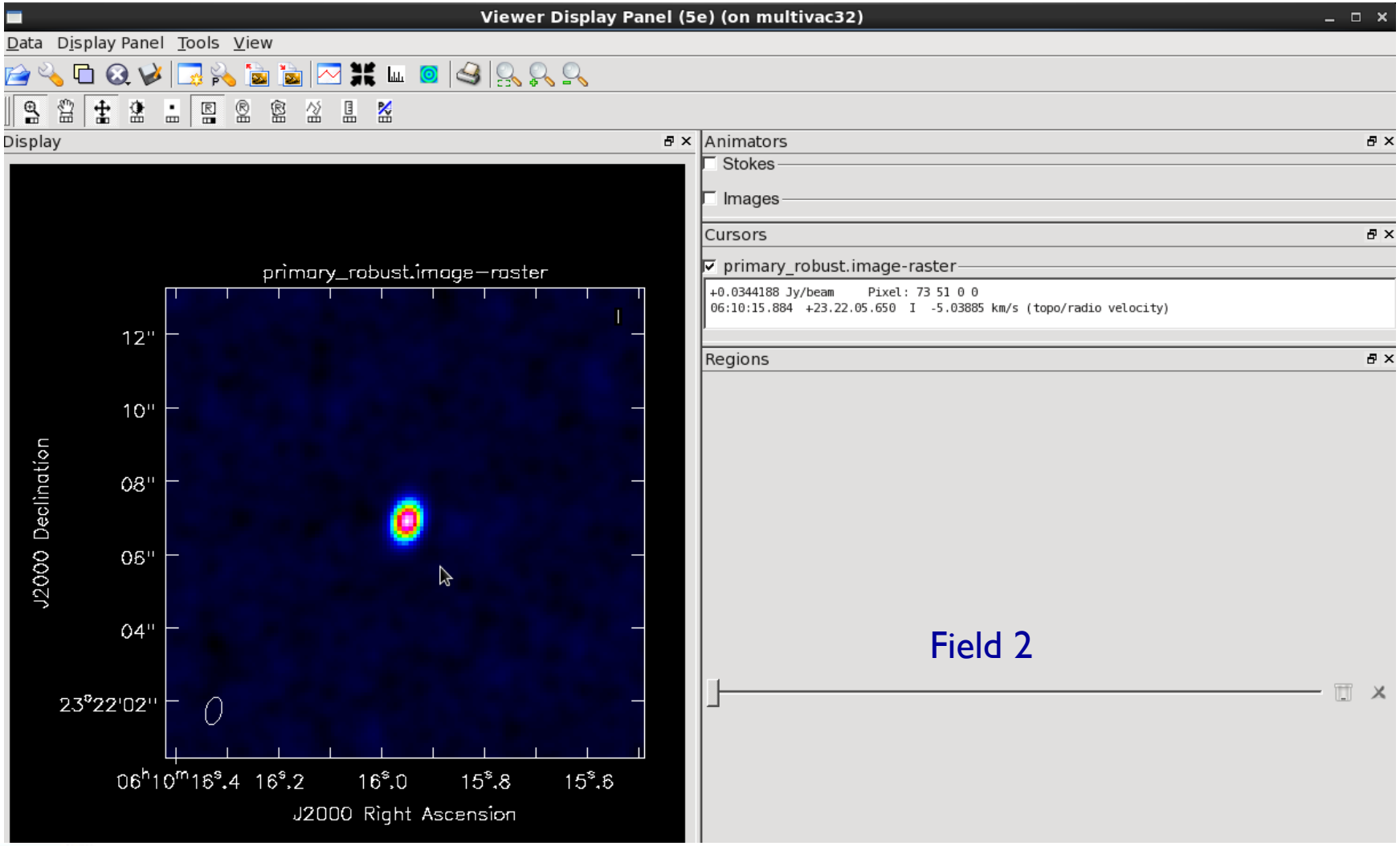


Field 2

Clean Residuals

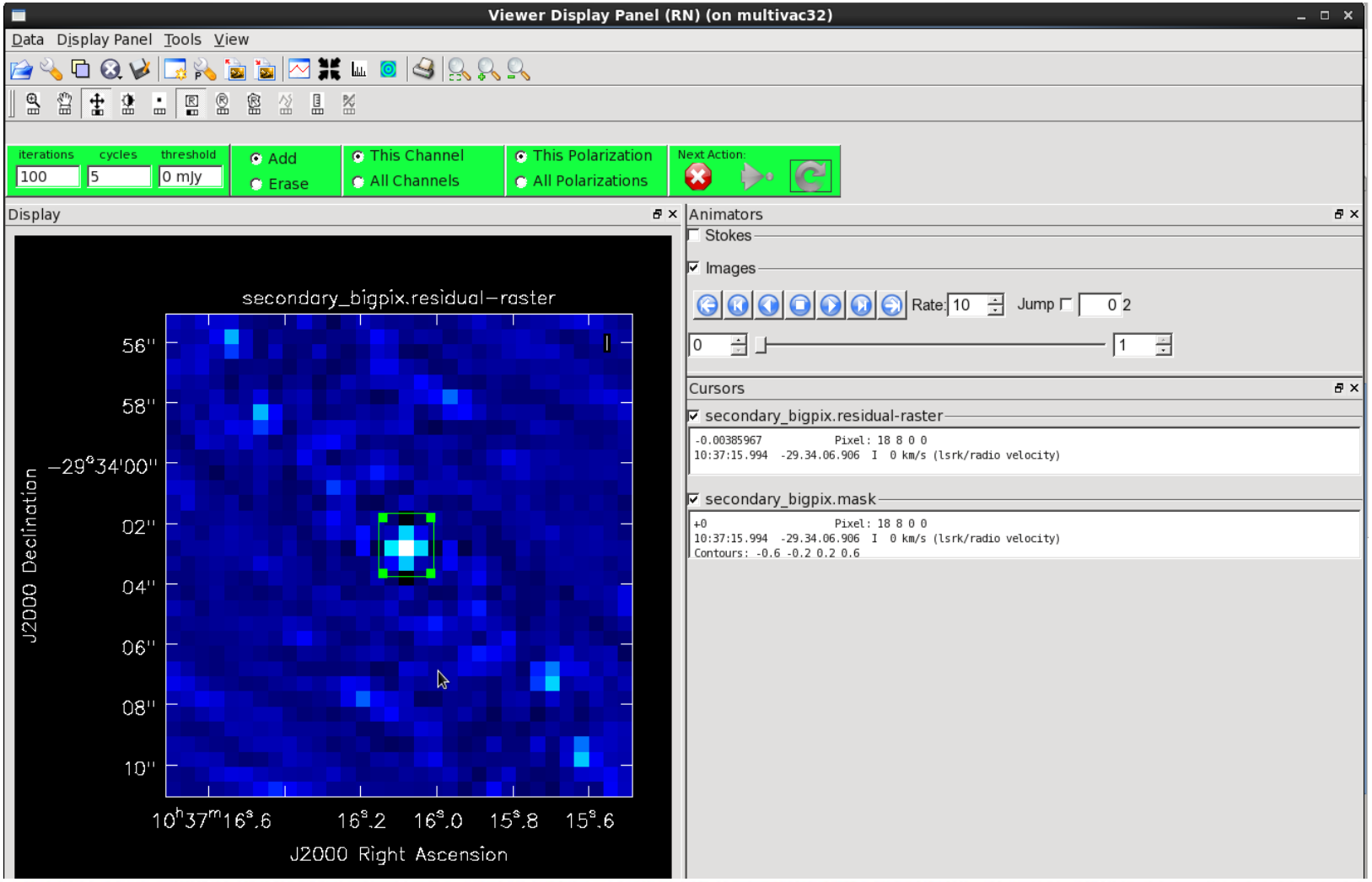
Basic Imaging

```
imview("primary_robust.image")
```

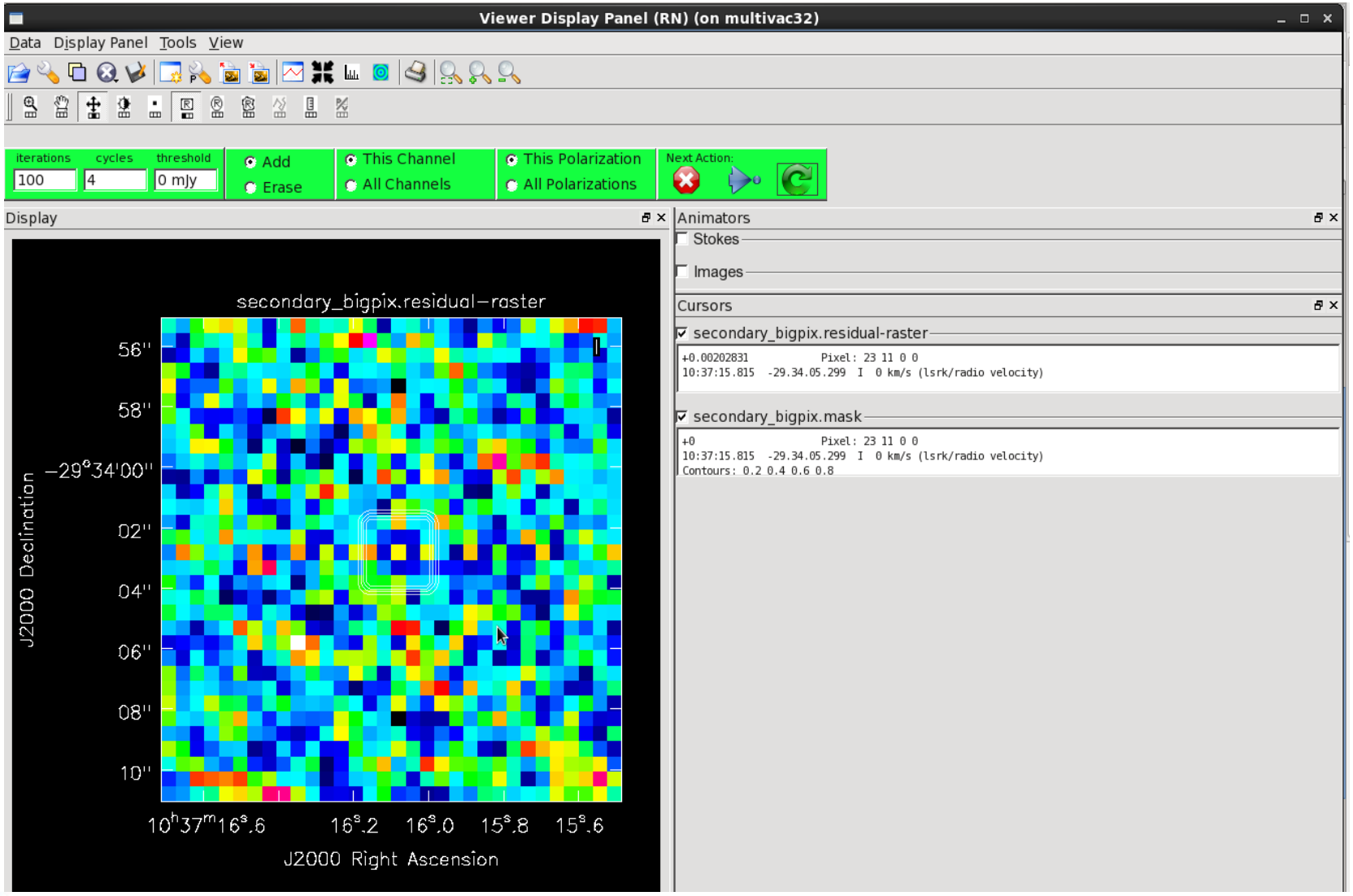


Basic Imaging

```
clean(vis='sis14_twhya_calibrated_flagged.ms', imagename='secondary_bigpix', field='3', spw='',
      mode='mfs', nterms=1, imsize=[32,32], cell=['0.5arcsec'], weighting='natural', threshold='0mJy',
      interactive=True)
```



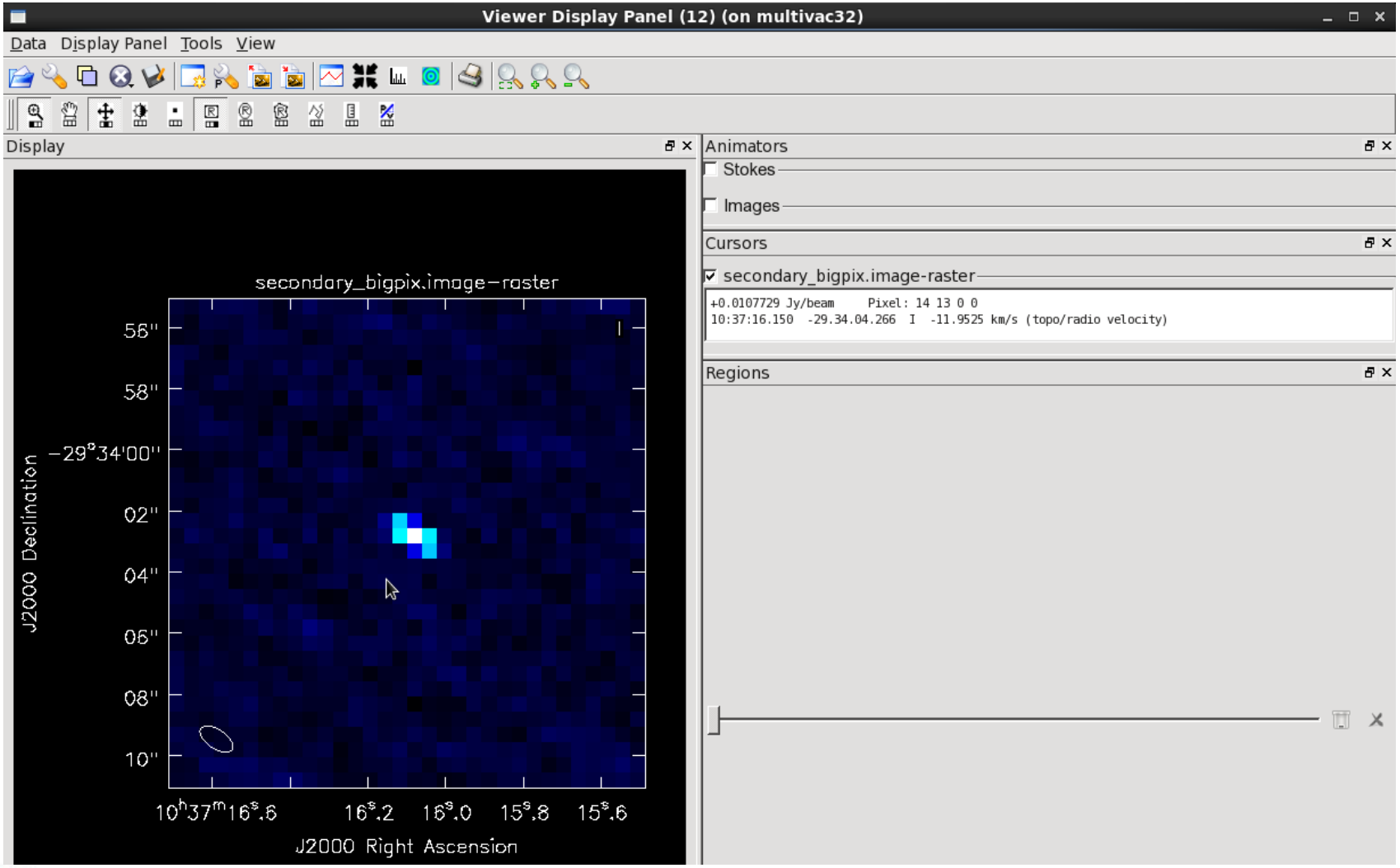
Basic Imaging



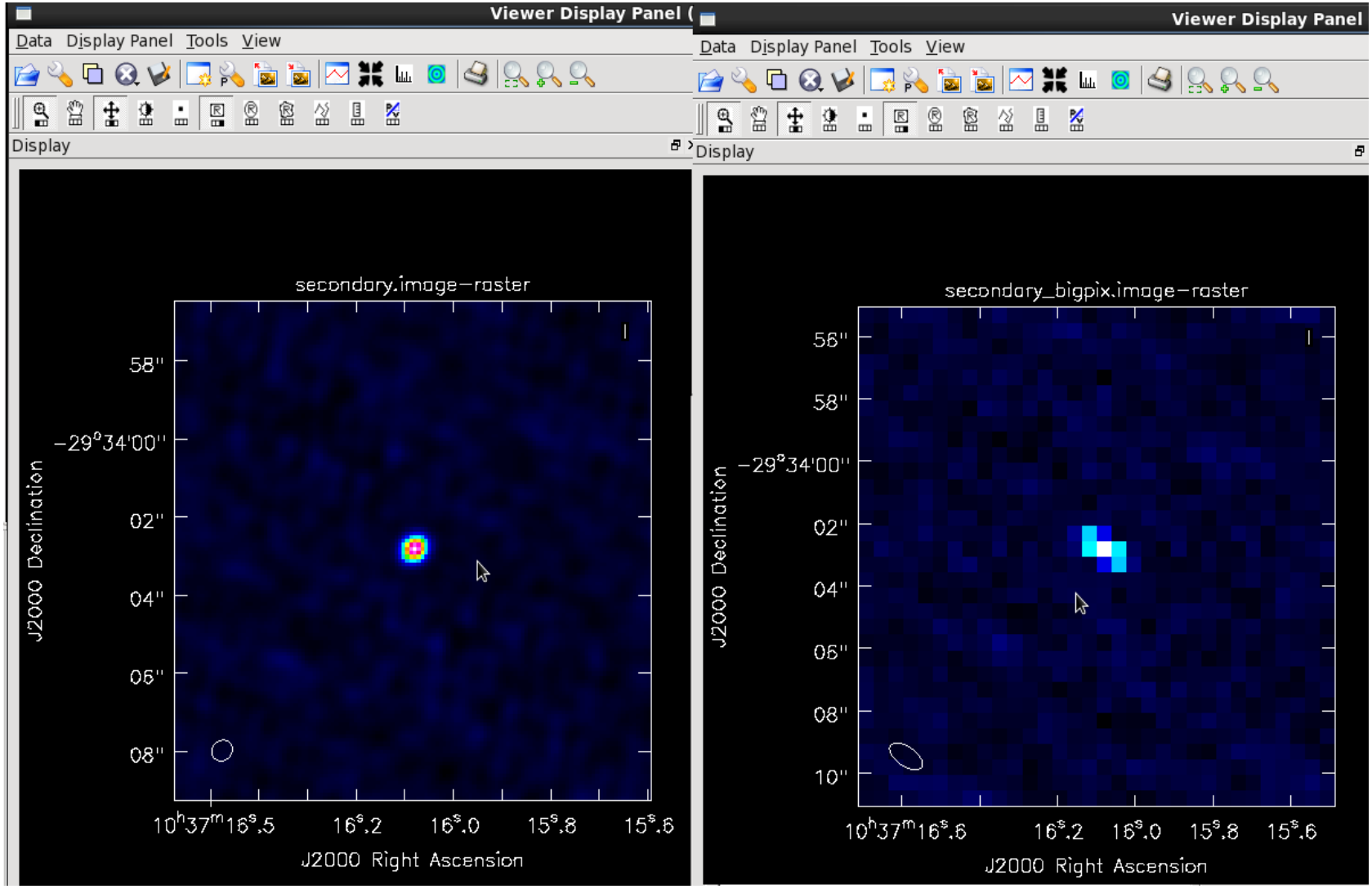
Clean Residuals

Basic Imaging

```
imview("secondary_bigpix.image")
```

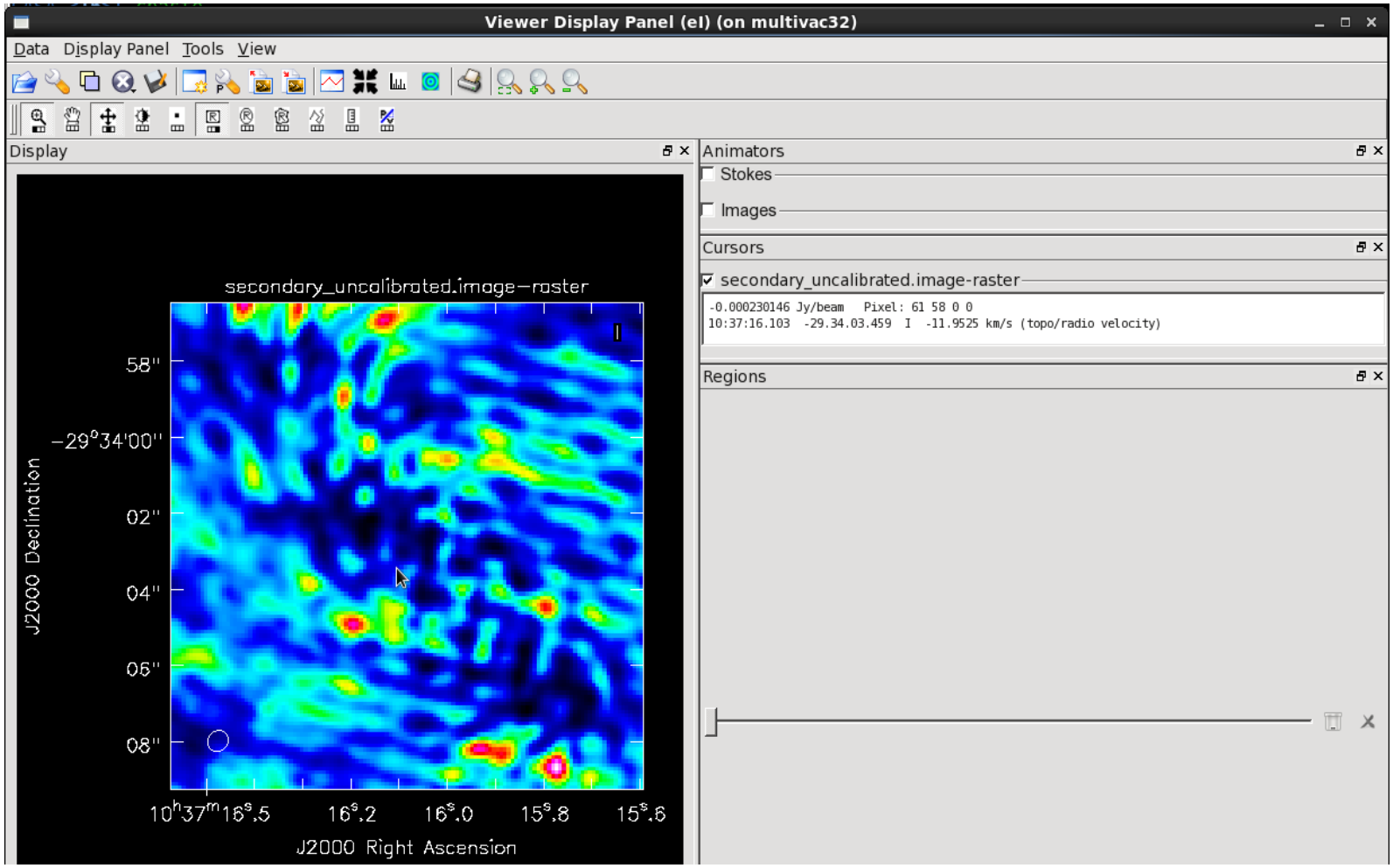


Basic Imaging



Basic Imaging - Aside

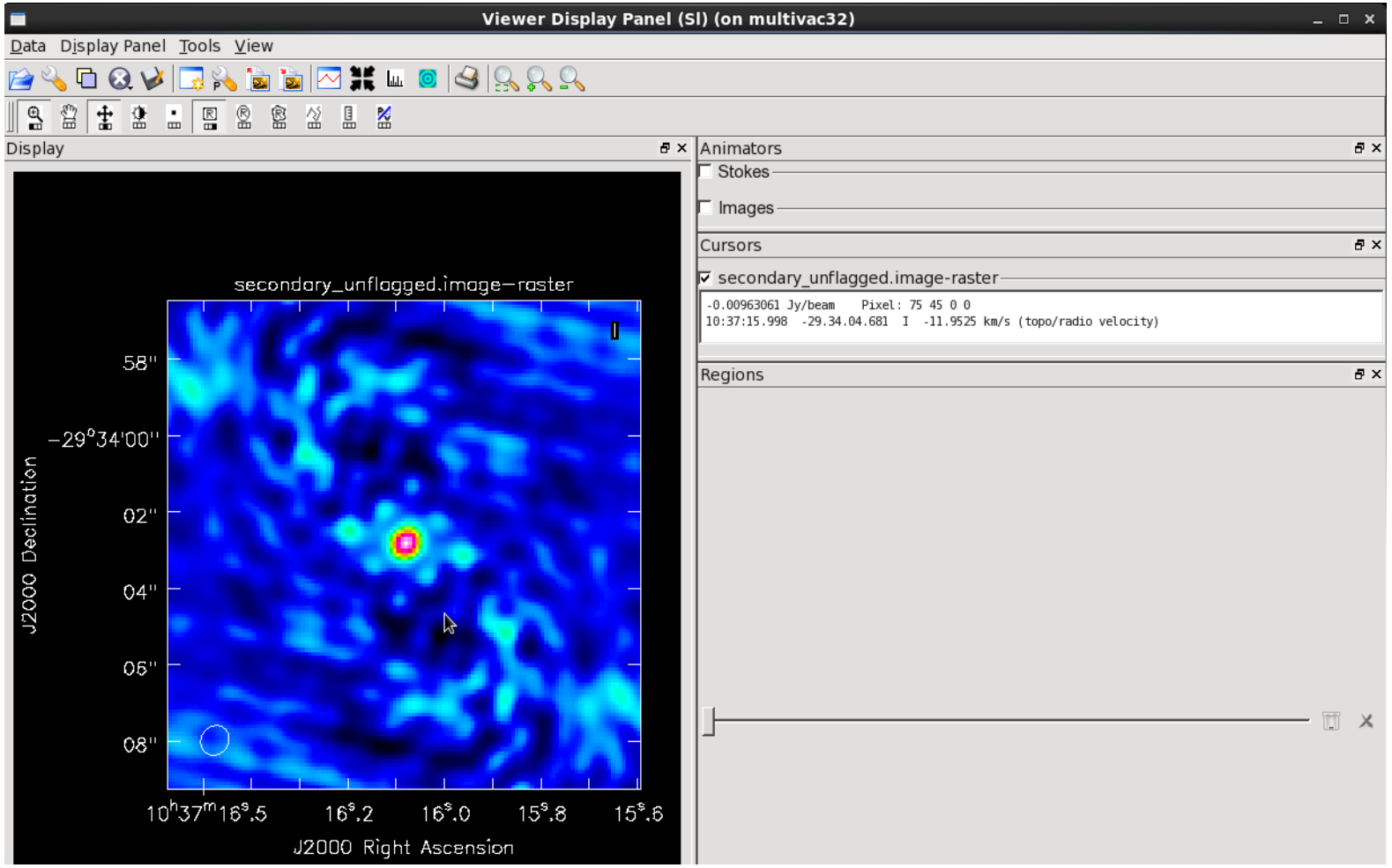
```
clean(vis='sis14_twhya_uncalibrated.ms', imagename='secondary_uncalibrated', field='3', spw='',
      mode='mfs', nterms=1, imsize=[128,128], cell=['0.1arcsec'], weighting='natural',
      threshold='0mJy', interactive=True)
```



Dirty Image

Basic Imaging - Aside

```
clean(vis='sis14_twhya_calibrated.ms', imagename='secondary_unflagged', field='3', spw='',
      mode='mfs', nterms=1, imsize=[128,128], cell=['0.1arcsec'], weighting='natural',
      threshold='0mJy', interactive=True)
```



Dirty Image

Basic Imaging

Split out the science data and smooth using width='10'

```
split(vis='sis14_twhya_calibrated_flagged.ms', field='5', width='10',
      outputvis='twhya_smoothed.ms', datacolumn='data')
```

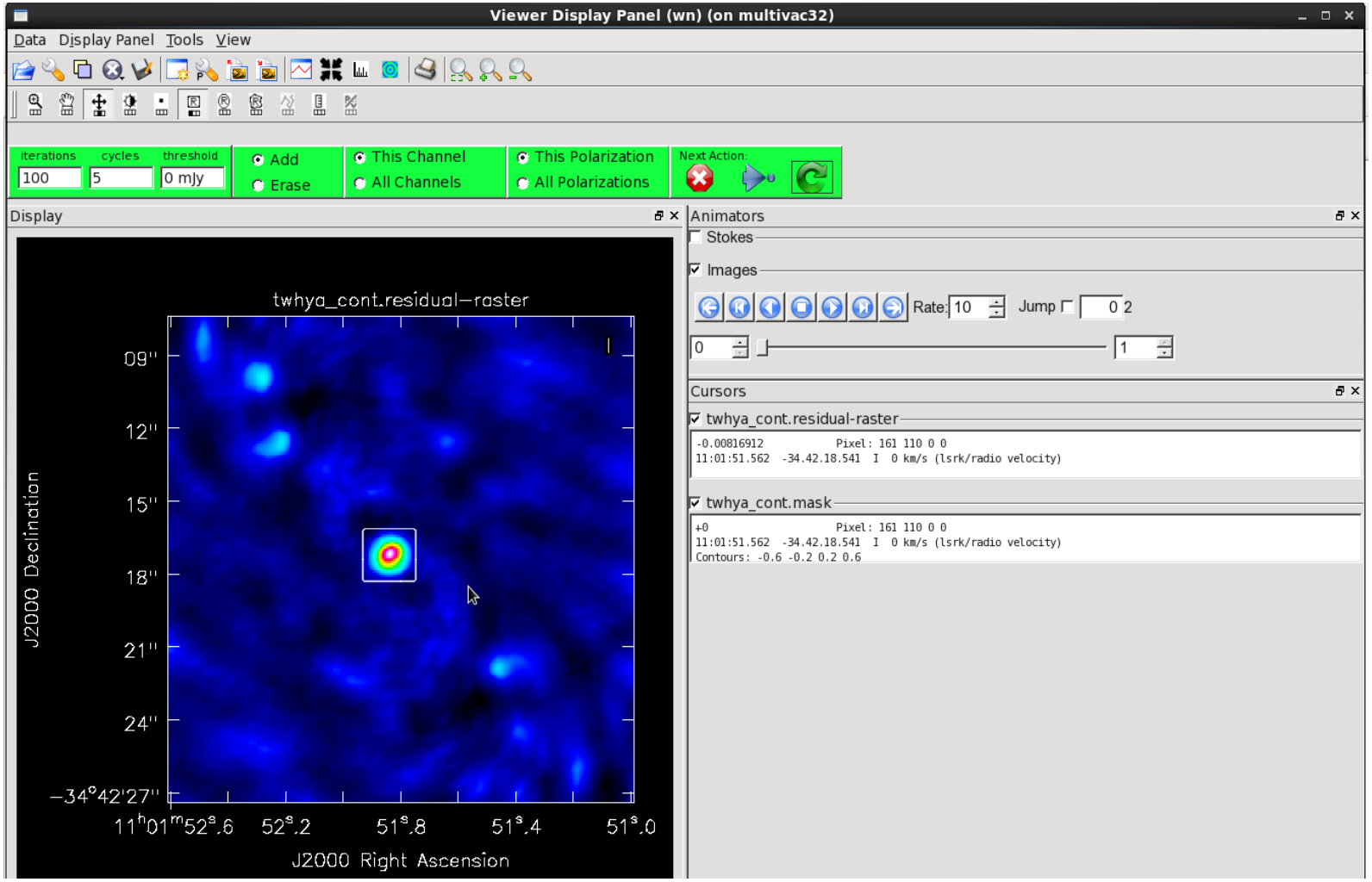
Orient yourself

```
listobs('sis14_twhya_calibrated_flagged.ms')
```

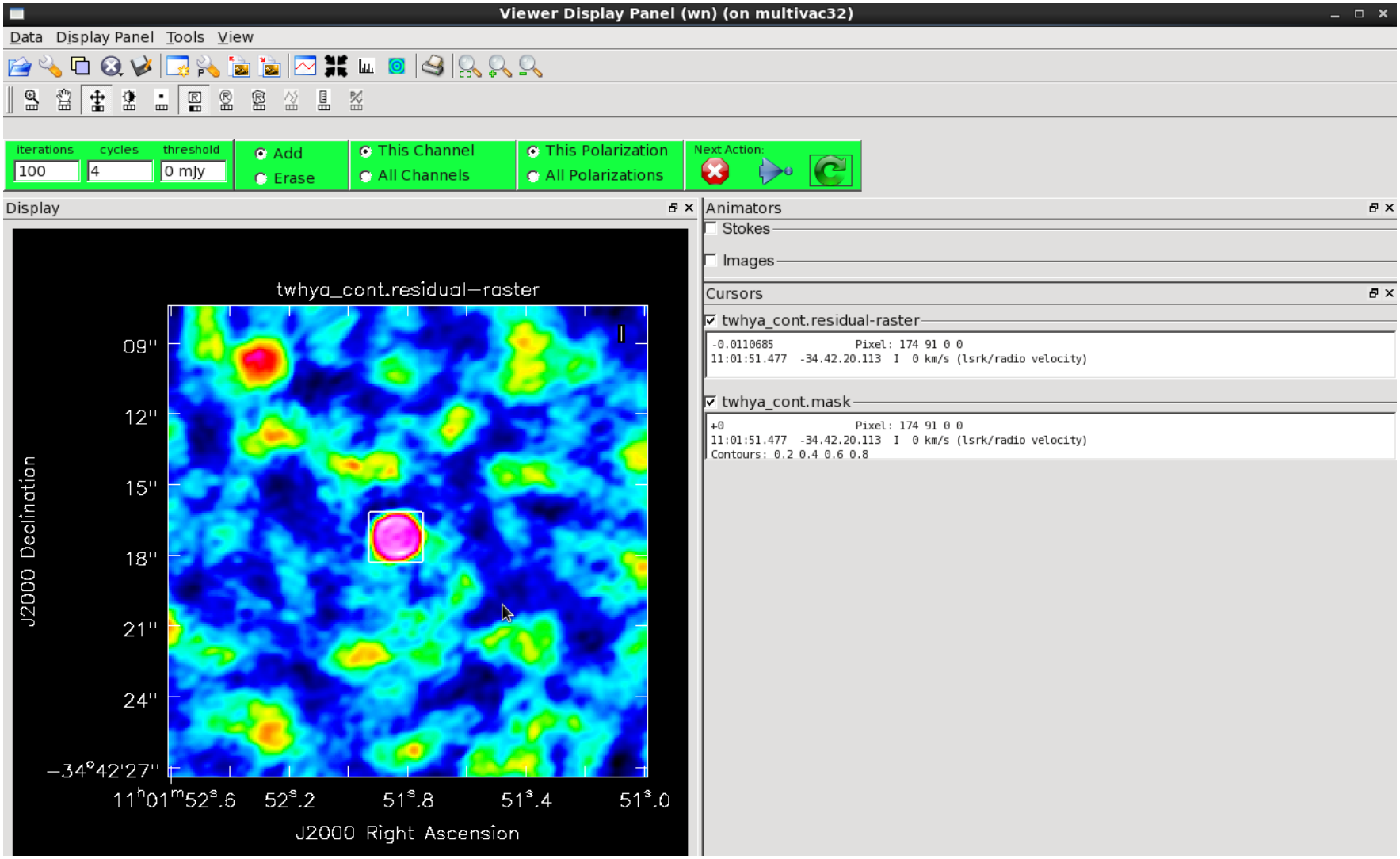
```
                                                  .
                                                  .
                                                  .
INFO listobs         Date        Timerange (UTC)          Scan  FldId FieldName  nRows SpwIds Average Interval(s)    ScanIntent
INFO listobs         19-Nov-2012/07:56:23.5 - 08:02:11.3    12      0 TW Hya      8514  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
INFO listobs                 08:08:09.6 - 08:13:57.3    16      0 TW Hya     10360  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
INFO listobs                 08:19:53.9 - 08:25:41.7    20      0 TW Hya     10321  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
INFO listobs                 08:32:00.5 - 08:37:48.2    24      0 TW Hya     10324  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
INFO listobs                 08:43:45.6 - 08:49:33.4    28      0 TW Hya      9462  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
INFO listobs                 09:05:15.6 - 09:07:31.6    36      0 TW Hya      4180  [0]  [6.05] [OBSERVE_TARGET#ON_SOURCE]
INFO listobs               (nRows = Total number of rows per scan)
INFO listobs         Fields: 1
INFO listobs           ID   Code Name            RA               Decl          Epoch   SrcId      nRows
INFO listobs           0    none TW Hya          11:01:51.796000 -34.42.17.36600 J2000   0          53161
INFO listobs         Spectral Windows:  (1 unique spectral windows and 1 unique polarization setups)
INFO listobs           SpwID  Name                          #Chans   Frame   Ch0(MHz)  ChanWid(kHz)  TotBW(kHz) BBC Num  Corrs
INFO listobs           0      ALMA_RB_07#BB_2#SW-01#FULL_RES    39   TOPO  372535.833     6103.516    234375.0      2  XX  YY
                                                  .
                                                  .
                                                  .
```

Basic Imaging

```
clean(vis='twhya_smoothed.ms', imagename='twhya_cont', field='0', spw='', mode='mfs', nterms=1,
      imsize=[250,250], cell=['0.08arcsec'], weighting='briggs', robust=0.5, threshold='0mJy',
      interactive=True)
```

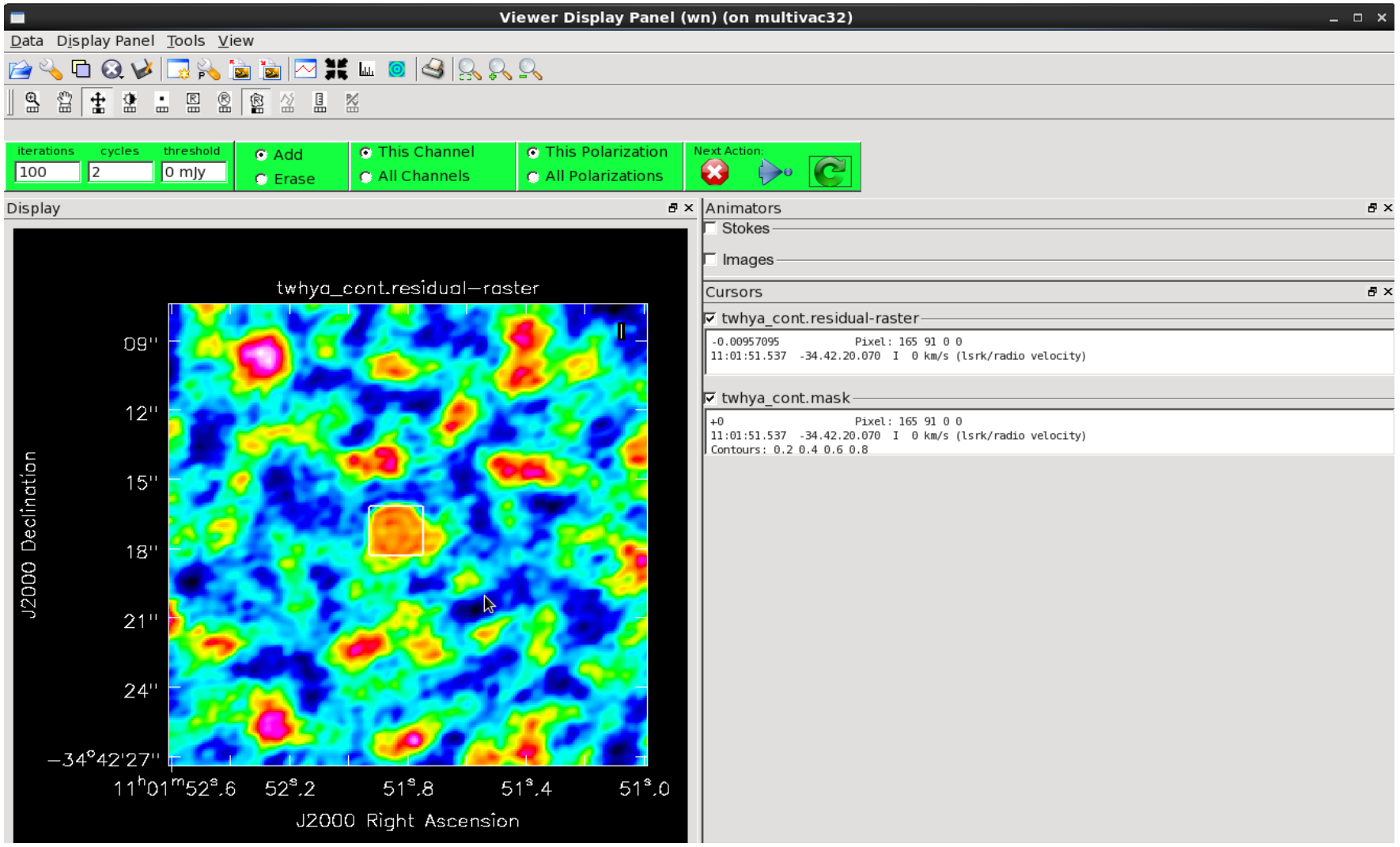


Basic Imaging



One cycle of cleaning

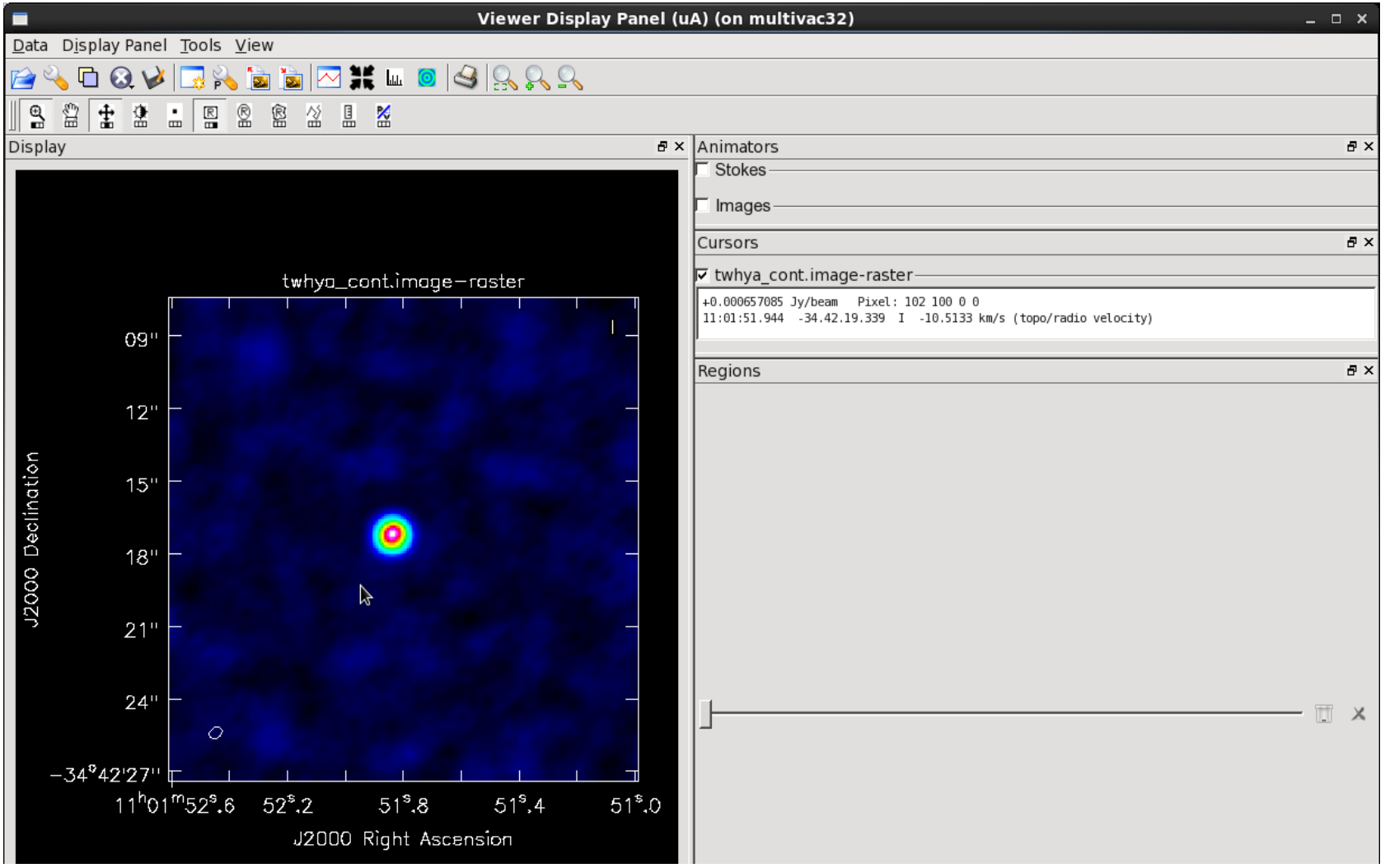
Basic Imaging



Three cycles of cleaning

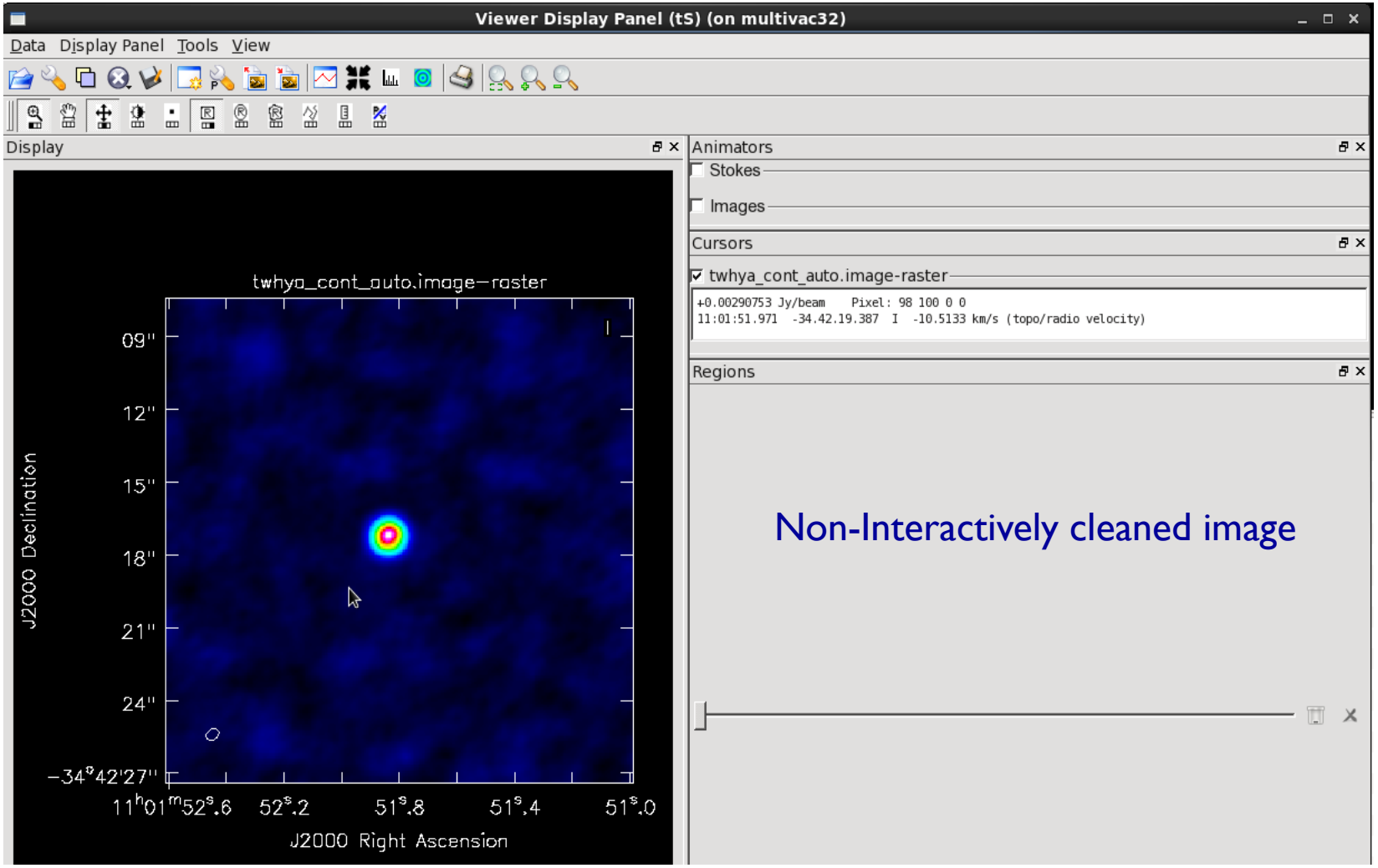
Basic Imaging

```
imview("twhya_cont.image")
```



Basic Imaging

```
clean(vis='twhya_smoothed.ms', imagename='twhya_cont_auto', field='0', spw='', mode='mfs', nterms=1,
      imsize=[250,250], cell=['0.08arcsec'], mask='box [ [ 100pix , 100pix] , [150pix, 150pix ] ]',
      weighting='briggs', robust=0.5, threshold='15mJy', niter=5000, interactive=False)
```



Non-Interactively cleaned image