



Introduction to CASA and Data Structure

Emmanuel Momjian

CASA



- CASA is the offline data reduction package for ALMA and the (Jansky) VLA
 - data from other telescopes usually work, too, but not primary goal of CASA).
- Import/export data, edit, calibrate, image, analyze.
- Also supports single dish (based on ATNF's ASAP).
- CASA has many tasks and a LOT of tools.

CASA



- Easy to write scripts and tasks.
- Lots of documentation, reduction tutorials, helpdesk.
- CASA has some of the most sophisticated algorithms implemented (multi-scale clean, Taylor term expansion for wide bands, W-term projection, OTF mosaicing, etc.)
- It has an active Algorithm Research Group.

CASA

- Web site: <http://casa.nrao.edu/>
- Available for both Linux and Mac OS.
 - Make sure to subscribe to the CASA mailing list for announcements of new releases, workshops, etc... (casa-announce), or for critical bugs and code updates (casa-users) at:
<http://casa.nrao.edu/> → Getting Help → Mailing lists

CASA

- Documentation is available at <http://casa.nrao.edu/> → 'Documentation'
- Training material is available at <http://casaguides.nrao.edu>
- For help, use the NRAO help desk at: <http://help.nrao.edu>
choose the 'CASA Data Reduction' Department

Outline

- CASA startup
- CASA basic python interface
- Tasks and tools
- The Measurement Set
- Data selection syntax
- Visualization tools
- Make your own task!

CASA Startup

> casa

==>

=====

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

=====

IPython 5.3.0 -- An enhanced Interactive Python.
CASA 5.1.2-4 -- Common Astronomy Software Applications

--> CrashReporter initialized.
Enter doc('start') for help getting started with CASA...
Using matplotlib backend: TkAgg

Log Messages (:/Users/emomjian/casa-20180522-004411.log)

Search Message: Filter: Time

Time	Priority	Origin	Message
2018-05-22 00:44:19	INFO	::casa	
2018-05-22 00:44:19	INFO	::casa	CASA Version 5.1.2-4

Insert Message: Lock scroll

CASA Interface

- Uses IPython for its command line interface:
 - Filesystem navigation, shell access
 - Namespace completion (<TAB>)
 - Session logging
 - **ipython.log** – ipython command history
 - **casapy.log** – casa messages
 - Numbered input/output with command history, full searching

Python Pointers

- to run a .py script:

`execfile('<scriptname>')`

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

- indentation matters!
 - 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*
- Task names are objects (not variables)

Tasks and tools in CASA

- **Tasks** - high-level functionality, well defined purpose
 - function call or parameter handling interface
 - these are what you should use in tutorial
- **Tools** - complete functionality
 - **tool.method** calls, used by tasks
 - sometimes shown in tutorial scripts
- Shell commands can be run with a leading exclamation mark **!du -hs**

CASA

- All CASA tasks can be listed by *tasklist*.
- The tasks are grouped as:
 - Import/export
 - Information
 - Editing
 - Manipulation
 - Calibration
 - Modeling
 - Imaging
 - Analysis
 - Visualization
 - Simulation
 - Single dish
 - Utility
- AIPS – CASA dictionary is available at
<https://safe.nrao.edu/wiki/bin/view/Software/CASA-AIPSDictionary>

Tasks

To list the tasks: *tasklist*

Import/export	Information	Editing	Manipulation
exportasdm exportfits exportuvfits importasdm importatca importfits importfitsidi importmiriad importuvfits importvla (importevla) (importgmt)	imhead imreframe imstat imval listcal listfits listhistory listobs listpartition listvis plotms plotuv vishead visstat visstat2 visstatold (asdmsummary) (listsdm) (makemask)	fixplanets fixvis flagcmd flagdata flagmanager msview plotms	concat conjugatevis cvel fixvis hanningsmooth imhead mstransform oldhanningsmooth oldsplit partition plotms split testconcat uvcontsub virtualconcat vishead (cvel2) (statwt) (uvcontsub3)
Calibration	Modeling	Imaging	Analysis
accum applycal bandpass blcal calstat clearcal delmod fixplanets fluxscale ft gaincal genical initweights listcal plotants plotbandpass plotcal polcal predictcomp rerefant setjy smoothcal uvmodelfit uvsub wvrgcal	predictcomp setjy uvcontsub uvmodelfit uvsub (uvcontsub3)	clean deconvolve feather ft imcontsub (boxit) (cvclean) (tclean) (tclean2) (widebandpbcor) {mosaic} {widefield}	imcollapse imcontsub imdev imfit imhead imhistory immath imoments impcor imprv inrebin inreframe inregrid insmooth imstat imsubimage imtrans imval listvis rmfit slsearch specflux specsmooth splattotable (specfit) (spxfit)
Visualization	Simulation	Single dish	Utility
clearplot imview msview plotants plotbandpass plotcal plotms plotprofilemap plotuv viewer (plotweather)	simanalyze simobserve (simalma)	importasap sdbaseline sdcal sdffit sdfixscan sdimaging sdsmooth (sdgaincal)	browseable caltabconvert clearplot clearstat concat conjugatevis find help par.parameter help taskname imview msview plotms rmtables startup taskhelp tasklist testconcat toolhelp virtualconcat

To see list of tasks with short help: *taskhelp*

```
[CASA <2>: taskhelp
→ taskhelp()
Available tasks:

accum          : Accumulate incremental calibration solutions into a calibration table
applycal      : Apply calibrations solutions(s) to data
asdmsummary   : Summarized description of an ASDM dataset.
autoclean     : CLEAN an image with automatically-chosen clean regions.
bandpass      : Calculates a bandpass calibration solution
blcal         : Calculate a baseline-based calibration solution (gain or bandpass)
boxit         : Box regions in image above given threshold value.
browsetable   : Browse a table (MS, calibration table, image)
calstat       : Displays statistical information on a calibration table
caltabconvert : Convert old-style caltables into new-style caltables.
clean         : Invert and deconvolve images with selected algorithm
clearcal      : Re-initializes the calibration for a visibility data set
clearplot     : Clear the matplotlib plotter and all layers
clearstat     : Clear all autolock locks
concat        : Concatenate several visibility data sets.
conjugatevis  : Change the sign of the phases in all visibility columns.
csvclean      : This task does an invert of the visibilities and deconvolve in the image plane.
cvel          : regrid an MS to a new spectral window / channel structure or frame
cvel2         : Regrid an MS or MMS to a new spectral window, channel structure or frame
deconvolve    : Image based deconvolver
delnod        : Deletes model representations in the MS
exportasdm    : Convert a CASA visibility file (MS) into an ALMA or EVLA Science Data Model
exportfits    : Convert a CASA image to a FITS file
exportuvfits  : Convert a CASA visibility data set to a UVFITS file:
feather       : Combine two images using their Fourier transforms
find          : Find string in tasks, task names, parameter names:
fixplanets    : Changes FIELD and SOURCE table entries based on user-provided direction or POINTING table, optionally fixes the UJV coordin
fixvis        : Recalculates (u, v, w) and/or changes Phase Center
flagcmd       : Flagging task based on batches of flag-commands
flagdata      : All-purpose flagging task based on data-selections and flagging modes/algorithms.
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 a visibility set:
gaincal       : Determine temporal gains from calibrator observations
gencal        : Specify Calibration Values of Various Types
hanningsmooth : Hanning smooth frequency channel data to remove Gibbs ringing
incollapse    : Collapse image along one axis, aggregating pixel values along that axis.
incontsub     : Estimates and subtracts continuum emission from an image cube
indev         : Create an image that can represent the statistical deviations of the input image.
infit         : Fit one or more elliptical Gaussian components on an image region(s)
inhead        : List, get and put image header parameters
inhistory     : Retrieve and modify image history
imath         : Perform math operations on images
imoments      : Compute moments from an image
impcor        : Construct a primary beam corrected image from an image and a primary beam pattern.
importasap    : Convert ASAP Scantable data into a CASA visibility file (MS)
importasdm    : Convert an ALMA Science Data Model observation into a CASA visibility file (MS)
importatca    : Import ATCA RPFITS file(s) to a measurement set
importevla    : Convert an Science Data Model observation into a CASA Measurement Set
importfits    : Convert an image FITS file into a CASA image
importfitsidi : Convert a FITS-IDI file to a CASA visibility data set
importgmr     : Convert a UVFITS file to a CASA visibility data set
importmiriad  : Convert a Miriad visibility file into a CASA MeasurementSet
importnro     : Convert NOSTAR data into a CASA visibility file (MS)
importuvfits  : Convert a UVFITS file to a CASA visibility data set
importvla     : Import VLA archive file(s) to a measurement set
imv           : Construct a position-velocity image by choosing two points in the direction plane.
inrebin       : Rebin an image by the specified integer factors
inreframe     : Change the frame in which the image reports its spectral values
inregrid      : regrid an image onto a template image
insmooth      : Smooth an image or portion of an image
instat        : Displays statistical information from an image or image region
insubimage    : Create a (sub)image from a region of the image
intrans       : Reorder image axes
imval         : Get the data value(s) and/or mask value in an image.
imview        : View an image
initweights   : Initializes weight information in the MS
listcal       : List antenna gain solutions
listfits      : List the HDU and typical data rows of a fits file:
listhistory   : List the processing history of a dataset:
listobs       : List the summary of a data set in the logger or in a file
listpartition : List the summary of a multi-MS data set in the logger or in a file
listsdm       : Lists observation information present in an SDM directory.
listvis       : List measurement set visibilities.
makemask      : Makes and manipulates image masks
mosaic        : Create a multi-field deconvolved image with selected algorithm
mstransform   : Split the MS, combine/separate/regrid spws and do channel and time averaging
msuvbin       : grid the visibility data onto a defined uniform grid (in the form of an ms); multiple MS's can be done onto the same grid
```

Task Interface

- parameters are set as global Python variables

(set) <param> = <value>

(e.g. , vis = 'ngc5921.demo.ms')

- using inp, default, saveinputs, tget, tput
- execute

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

Task Interface

Call a task by

>inp <taskname>

if default values are desired, first type

>default <taskname>, followed by inp

```
[CASA <3>: inp gaincal
-----> inp(gaincal)
# gaincal :: Determine temporal gains from calibrator observations
vis                =      ''      # Name of input visibility file
caltable           =      ''      # Name of output gain calibration table
field              =      ''      # Select field using field id(s) or field name(s)
spw                =      ''      # Select spectral window/channels
intent             =      ''      # Select observing intent
selectdata         =      True     # Other data selection parameters
    timerange       =      ''      # Select data based on time range
    uvrange         =      ''      # Select data within uvrange (default units meters)
    antenna         =      ''      # Select data based on antenna/baseline
    scan            =      ''      # Scan number range
    observation      =      ''      # Select by observation ID(s)
    msselect        =      ''      # Optional complex data selection (ignore for now)

solint             =      'inf'    # Solution interval: egs. 'inf', '60s' (see help)
combine           =      ''      # Data axes which to combine for solve (obs, scan, spw, and/or field)
preavg            =      -1.0     # Pre-averaging interval (sec) (rarely needed)
refant            =      ''      # Reference antenna name(s)
refantmode        =      'flex'   # Reference antenna mode
minblperant       =      4        # Minimum baselines _per antenna_ required for solve
minsnr            =      3.0      # Reject solutions below this SNR
solnorm           =      False    # Normalize average solution amplitudes to 1.0 (G, T only)
gaintype          =      'G'      # Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel            =      []       # Point source Stokes parameters for source model.
calmode           =      'ap'     # Type of solution: ('ap', 'p', 'a')
append            =      False    # Append solutions to the (existing) table
docallib          =      False    # Use callib or traditional cal apply parameters
    gaintable       =      []     # Gain calibration table(s) to apply on the fly
    gainfield       =      []     # Select a subset of calibrators from gaintable(s)
    interp          =      []     # Temporal interpolation for each gaintable (=linear)
    spwmap          =      []     # Spectral windows combinations to form for gaintables(s)

parang            =      False    # Apply parallactic angle correction on the fly
```

Task Execution

- Two ways to invoke:
 - call from Python as functions with arguments
`taskname(arg1=val1, arg2=val2, ...)`, like
`clean(vis= 'input.ms' ,
 imagename= 'galaxy' ,selectvis=T, robust=0.5,
 imsize=[200,200])`
unspecified parameters will be defaulted
 - use standard tasking interface.

Parameter Checking

```
[CASA <5>: inp
→ inp()
# gaincal :: Determine temporal gains from calibrator observations
vis = '' # Name of input visibility file
caltable = '' # Name of output gain calibration table
field = '' # Select field using field id(s) or field name(s)
spw = '' # Select spectral window/channels
intent = '' # Select observing intent
selectdata = True # Other data selection parameters
    timerange = '' # Select data based on time range
    uvrange = '' # Select data within uvrange (default units meters)
    antenna = '' # Select data based on antenna/baseline
    scan = '' # Scan number range
    observation = '' # Select by observation ID(s)
    msselect = '' # Optional complex data selection (ignore for now)

solint = 'inf' # Solution interval (seconds) (default 60s' (see help)
combine = '' # Data axis combination (solve, solve+uv, solve+uv+spw, solve+uv+spw+field)
preavg = -1.0 # Pre-averaging factor (1.0 = no pre-averaging)
refant = '' # Reference antenna (needed)
refantmode = 'flex' # Reference antenna mode
minblperant = 4 # Minimum baselines _per antenna_ required for solve
minsnr = 3.0 # Reject solutions below this SNR
solnorm = False # Normalize average solution amplitudes to 1.0 (G, T only)
gaintype = 'G' # Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel = '' # Point source Stokes parameters for source model.
calmode = 'noidea' # Type of solution: ('ap', 'p', 'a')
append = False # Append solutions to the (existing) table
docallib = False # Use callib or traditional cal apply parameters
    gaintable = [] # Gain calibration table(s) to apply on the fly
    gainfield = [] # Select a subset of calibrators from gaintable(s)
    interp = [] # Temporal interpolation for each gaintable (=linear)
    spwmap = [] # Spectral windows combinations to form for gaintables(s)

parang = False # Apply parallactic angle correction on the fly
```

erroneous
values in red

Help on Tasks

In-line help for all tasks (help <taskname>)

>help gaincal

```
| Methods defined here:
|
| __call__(self, vis=None, caltable=None, field=None, spw=None, intent=None, selectdata=None
None, refant=None, refantmode=None, minblperant=None, minsnr=None, solnorm=None, gaintype=None
table=None, gainfield=None, interp=None, spwmap=None, parang=None)
|     Determine temporal gains from calibrator observations
|
|     Detailed Description:
|
|     The complex gains for each antenna/spwid are determined from the
|     data column (raw data), divided by the model column, for the
|     specified fields. The gains can be obtained for a
|     specified solution interval for each spectral window, or by a spline
|     fit to all spectral windows simultaneously.
|
|     Previous calibrations (egs. bandpass) should be applied on the fly.
|
|     Arguments :
|         vis:      Name of input visibility file
|                   Default Value:
|
|         caltable:  Name of output gain calibration table
|                   Default Value:
|
|         field:     Select field using field id(s) or field name(s)
|                   Default Value:
|
|         spw:       Select spectral window/channels
|                   Default Value:
|
|         intent:    Select observing intent
|                   Default Value:
|
|         selectdata: Other data selection parameters
|                   Default Value: True
|
|         timerange: Select data based on time range
|                   Default Value:
|
|         uvrange:   Select data within uvrange (default units meters)
|                   Default Value:
```

Tools in CASA

☞ What if there's no task?

→ use CASA tools (tasks are built upon tools)

☞ tools are functions/methods

☞ call from casapy as `<tool>.<method>()`

☞ default tool objects are pre-constructed

☞ e.g. imager (im) , calibrator (cb), ms (ms) , etc.
(see toolhelp)

CASA Tool List

To list the default tools:

>toolhelp

~1000 tools available

```
[CASA <7>: toolhelp
→ toolhelp()

Available tools:

af : Agent flagger utilities
at : Juan Pardo ATM library
ca : Calibration analysis utilities
cb : Calibration utilities
cl : Component list utilities
cp : Cal solution plotting utilities
cs : Coordinate system utilities
cu : Class utilities
dc : Deconvolver utilities
fi : Fitting utilities
fn : Functional utilities
ia : Image analysis utilities
im : Imaging utilities
lm: linear mosaic
me : Measures utilities
ms : MeasurementSet (MS) utilities
msmd : MS metadata accessors
mt : MS transformer utilities
qa : Quanta utilities
pm : PlotMS utilities
po : Imagepol utilities
rg : Region manipulation utilities
sdms : MeasurementSet (MS) utilities for Single-Dish
sl : Spectral line import and search
sm : Simulation utilities
tb : Table utilities (selection, extraction, etc)
tp : Table plotting utilities
vp : Voltage pattern/primary beam utilities
—
pl : pylab functions (e.g., pl.title, etc)
—
```

The Measurement Set

- The MS is a directory on disk, it consists of a MAIN table and sub-tables.
 - The MAIN table contains the visibility data. It consists of the `table.*` files.
 - The sub-tables (e.g. FIELD, SOURCE, ANTENNA, etc.) contain auxiliary and secondary information.
 - The sub-tables are sub-directories.
- To copy: must use `cp -rf` to get contents
- Best to remove MS with `rmtables('filename')`

Example MS

```
CASA <31>: ls day2_TDEM0003_20s_full/
ANTENNA/          STATE/          table.f18_TSM1  table.f25_TSM1
DATA_DESCRIPTION/ table.dat       table.f19       table.f3
FEED/             table.f1       table.f2       table.f4
FIELD/            table.f10      table.f20      table.f5
FLAG_CMD/         table.f11      table.f21      table.f6
HISTORY/          table.f12      table.f21_TSM0 table.f7
OBSERVATION/      table.f13      table.f22      table.f8
POINTING/         table.f14      table.f22_TSM1 table.f9
POLARIZATION/     table.f15      table.f23      table.info
PROCESSOR/        table.f16      table.f23_TSM1 table.lock
SORTED_TABLE/     table.f17      table.f24      WEATHER/
SOURCE/           table.f17_TSM1 table.f24_TSM1
SPECTRAL_WINDOW/  table.f18      table.f25
```

```
CASA <32>: ls day2_TDEM0003_20s_full/ANTENNA/
table.dat  table.f0  table.info  table.lock
```


Data Selection Syntax

- field - string with source name or field ID
 - can use '*' as wildcard, first checks for name, then ID
 - example: field = '1331+305' ; field = '3C*' ; field = '0,1,4~5'
- spw - string with spectral window ID plus channels
 - use ':' as separator of spw from optional channelization
 - example: spw = '0~2' ; spw = '1:10~30'

Selection Syntax

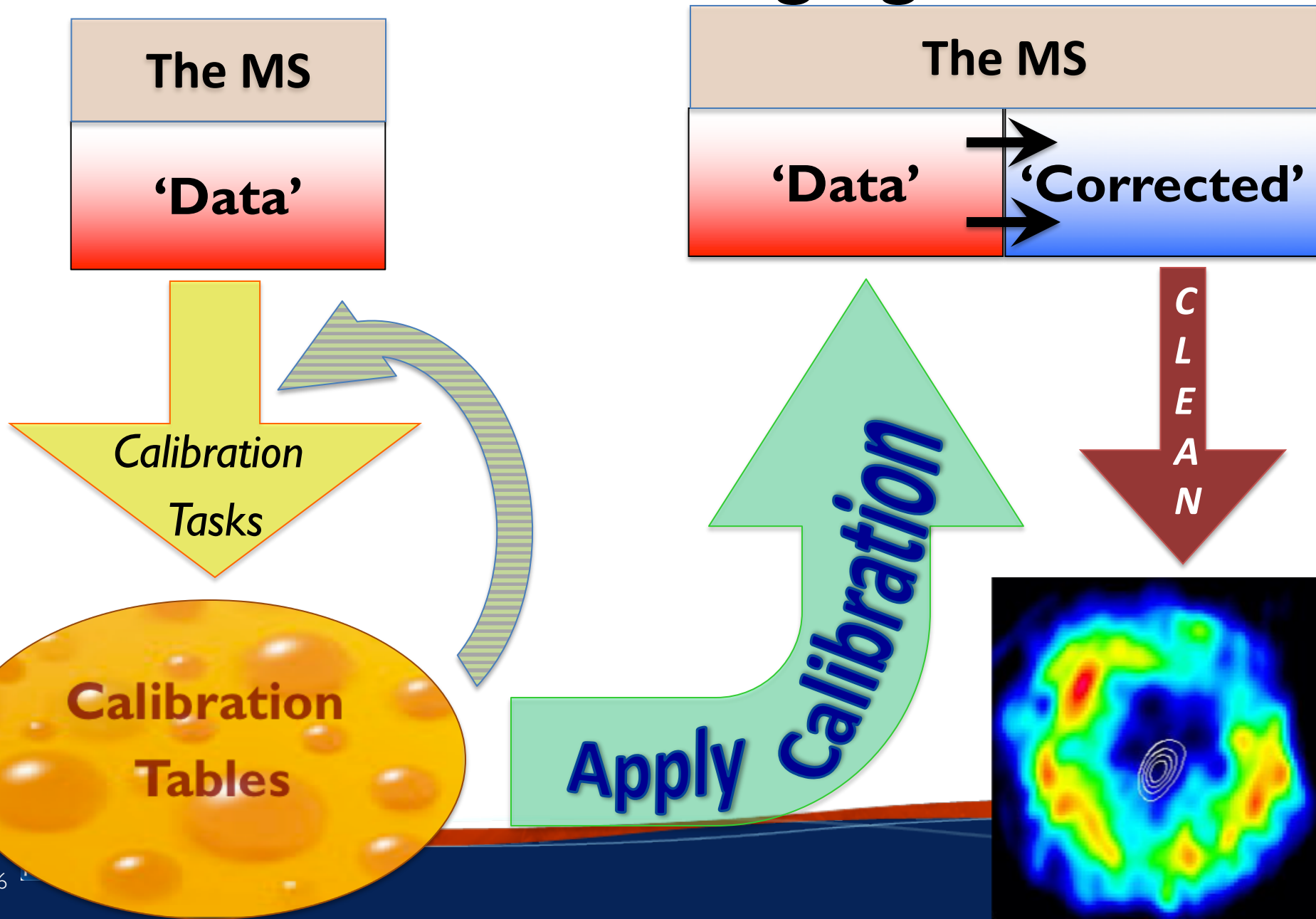
- antenna - string with antenna name or ID
 - first check for name, then pad name, then ID
 - example: antenna = '1~5,11' ; antenna = 'ea*', '!ea01'
 - For a baseline, use: antenna = 'ea01&ea10'
- timerange - string with date/time range
 - specify 'T0~T1', missing parts of T1 default to T0.
 - example: timerange = '2007/10/16/01:00:00~06:30:00'
 - If year, month, day are not specified → defaults to 1st day in the data set.

The MS structure

'Data' column Raw Data	'Corrected' Column Calibrated Data	'Model' Column FT of source model
----------------------------------	--	---

- When you load your data from the archive, your MS will only have the 'Data' column.
- The other two columns can be created by various means.
- The creation of the other two columns → MS tripling in size.

Calibration & Imaging Flow



Visualization Tools

- Visibilities: plotms, msview
- Images: viewer, imview
- Calibration tables: plotcal (or plotms)
- Any table values: browsetable
- Single dish: sdplot
- Plot anything: use python's matplotlib

Data Review: *plotms* (unix command line *casaplotms*)

Top Tabs

Side Tabs

Control Panel

Graphics Panel

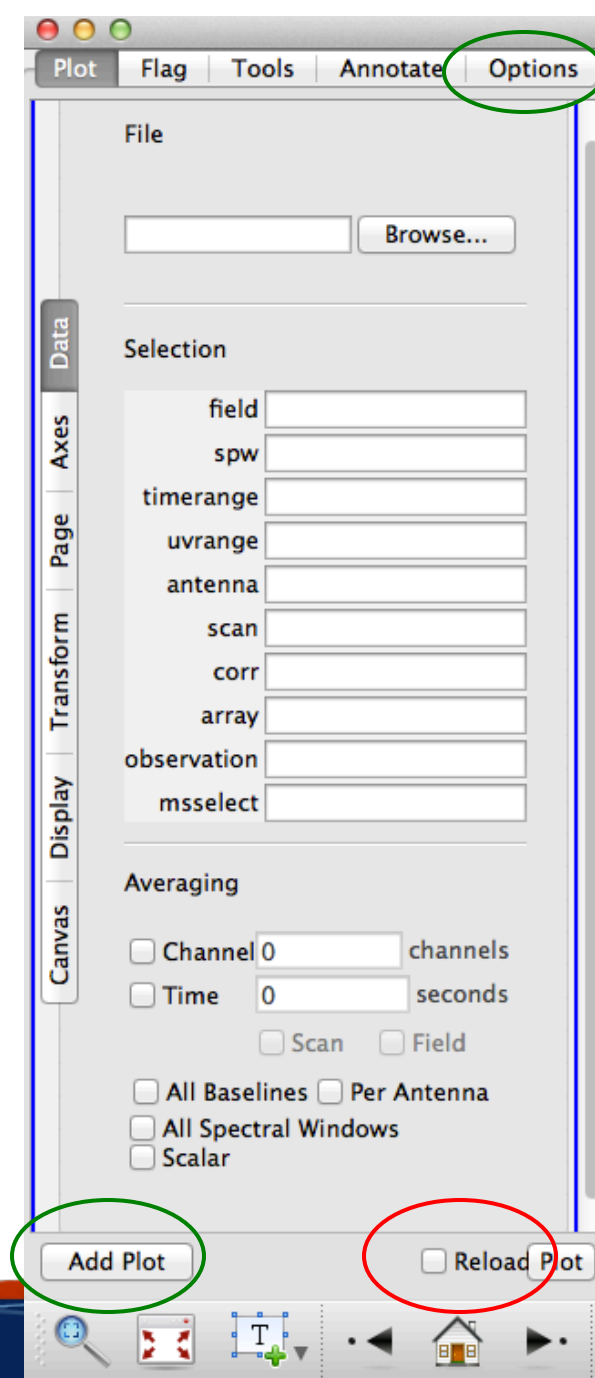
Tools Panel

Data Review: *plotms*

Control Panel: Data

Check the 'Reload' box if the MS has been modified through another task.

Use the 'Options' to divide the screen into multiple panels, and 'Add plot' to be able make plots of multiple data sets (or one data set but using different axes) onto the graphics panel.



Data Review: *plotms*

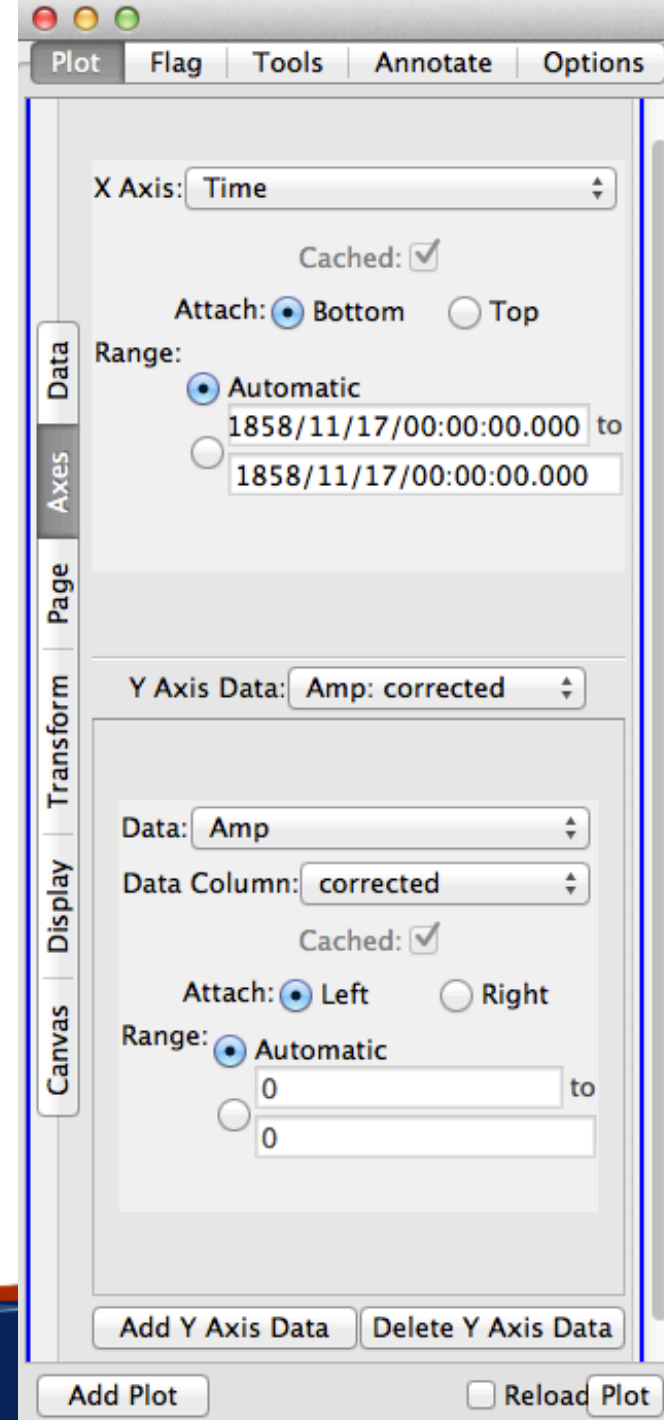
MS Ids and other meta info:

'scan' (number)
'field' (index)
'time',
'interval'='timeint'='timeinterval'='time_interval'
'spw' (index)
'chan'='channel' (index)
'freq'='frequency' (GHz)
'vel'='velocity' (km/s)
'corr'='correlation' (index)
'ant1'='antenna1' (index)
'ant2'='antenna2' (index)
'baseline' (a baseline index)
'row' (absolute row Id from the MS)

Visibility values, flags:

'amp'='amplitude'
'phase' (deg)
'real'
'imag'='imaginary'
'wt'='weight'
'flag'
'flagrow'

Axes



Data Review: *plotms*

Observational geometry:

'uvdist' (meters)

'uvwave'='uvdistl'='uvdist_l' (wavelengths, per channel)

'u' (meters)

'v' (meters)

'w' (meters)

'azimuth' (at array reference; degrees)

'elevation' (at array reference; degrees)

'hourang'='hourangle' (at array reference; hours)

'parang'='parangle'='parallacticangle' (at array reference; degrees)

Antenna-based (only works vs. data lds):

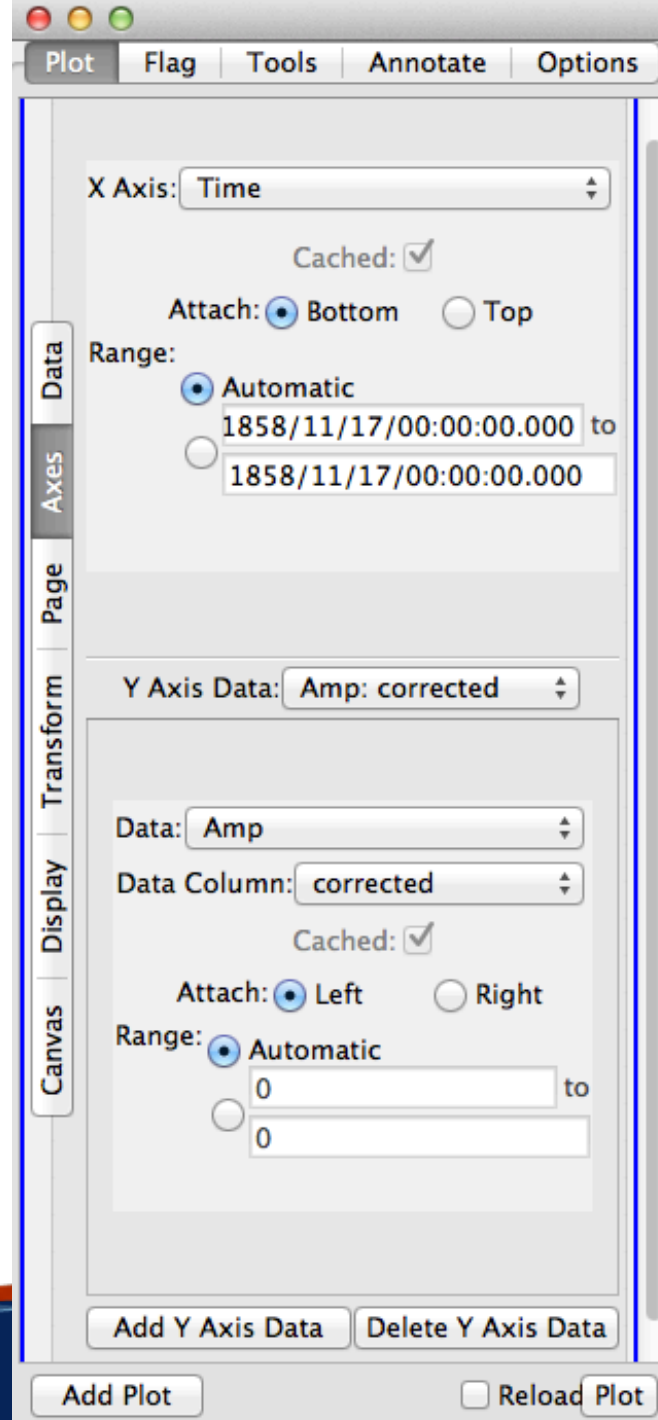
'ant'='antenna'

'ant-azimuth'

'ant-elevation'

'ant-parang'='ant-parangle'

Axes



Data Review: *plotms*

Page: to iterate on

Scan

Field

Spw

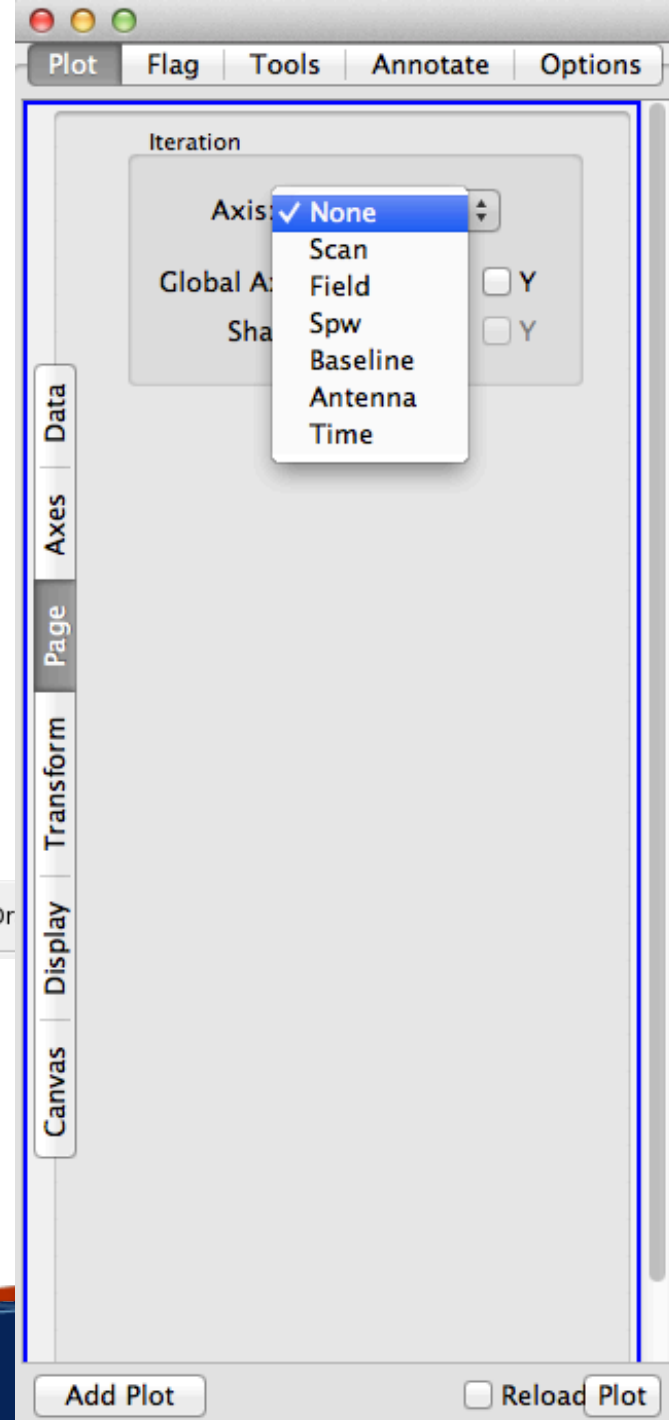
Baseline

Antenna

Time



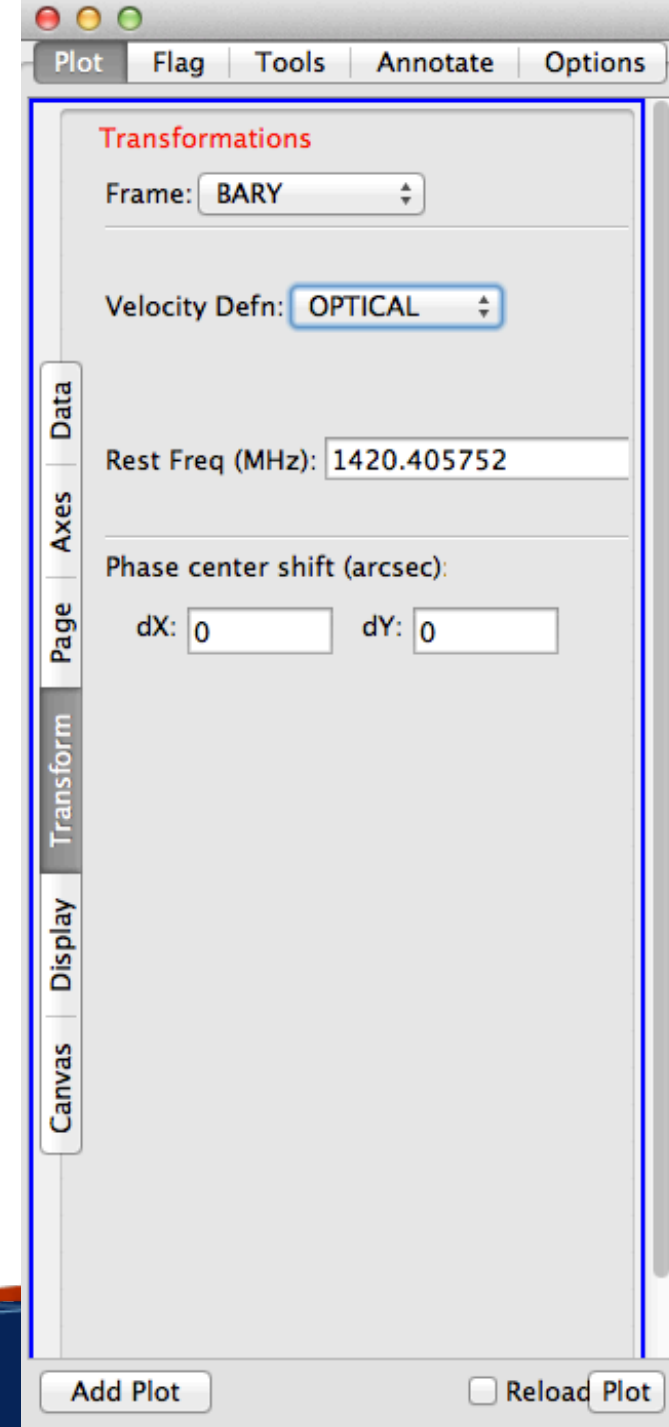
Tool panel



Data Review: *plotms*

Transformations

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



The image shows a screenshot of the 'Transformations' panel in the 'plotms' software interface. The panel is titled 'Transformations' in red text. It contains several input fields and a vertical sidebar on the left. The sidebar has buttons for 'Canvas', 'Display', 'Transform' (which is highlighted), 'Page', 'Axes', and 'Data'. The main area of the panel has the following settings:

- Frame:** A dropdown menu set to 'BARY'.
- Velocity Defn:** A dropdown menu set to 'OPTICAL'.
- Rest Freq (MHz):** A text input field containing '1420.405752'.
- Phase center shift (arcsec):** Two text input fields, 'dX' and 'dY', both containing '0'.

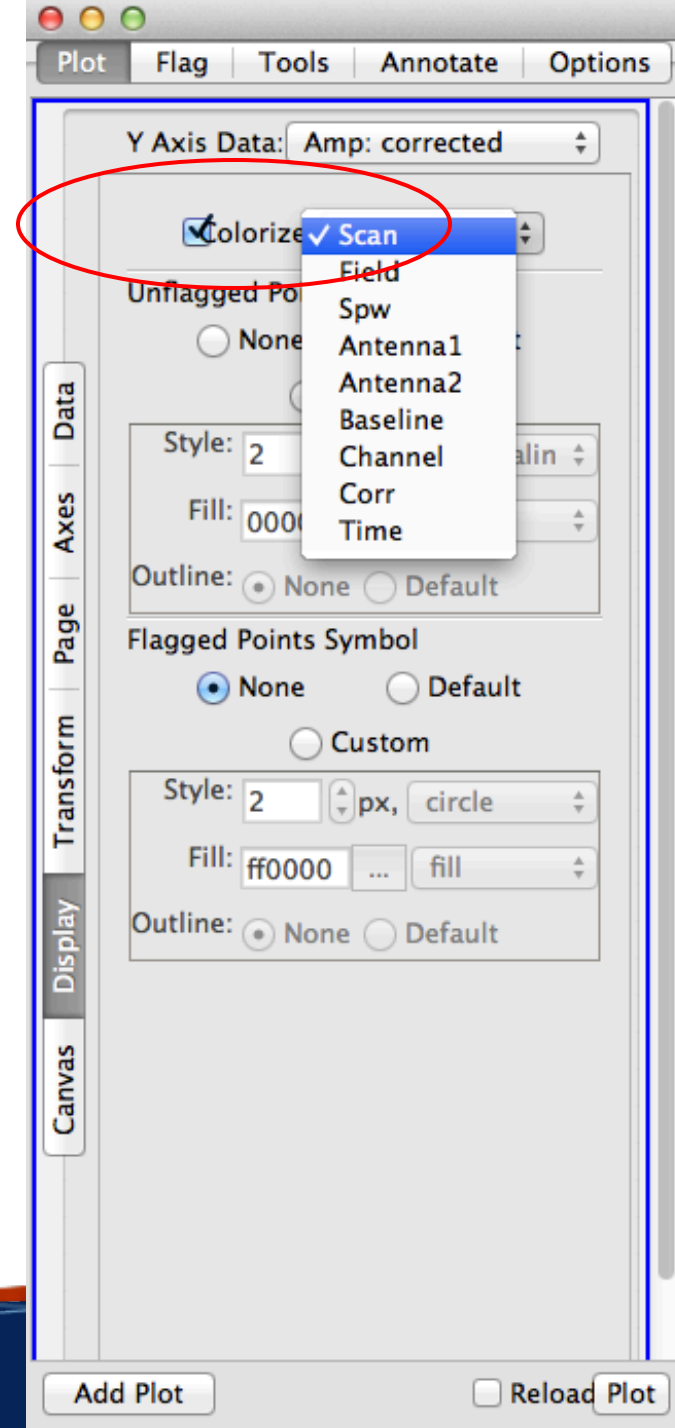
At the bottom of the panel, there are two buttons: 'Add Plot' and 'Reload Plot' (which is disabled).

Data Review: *plotms*

Display

Colorize by:

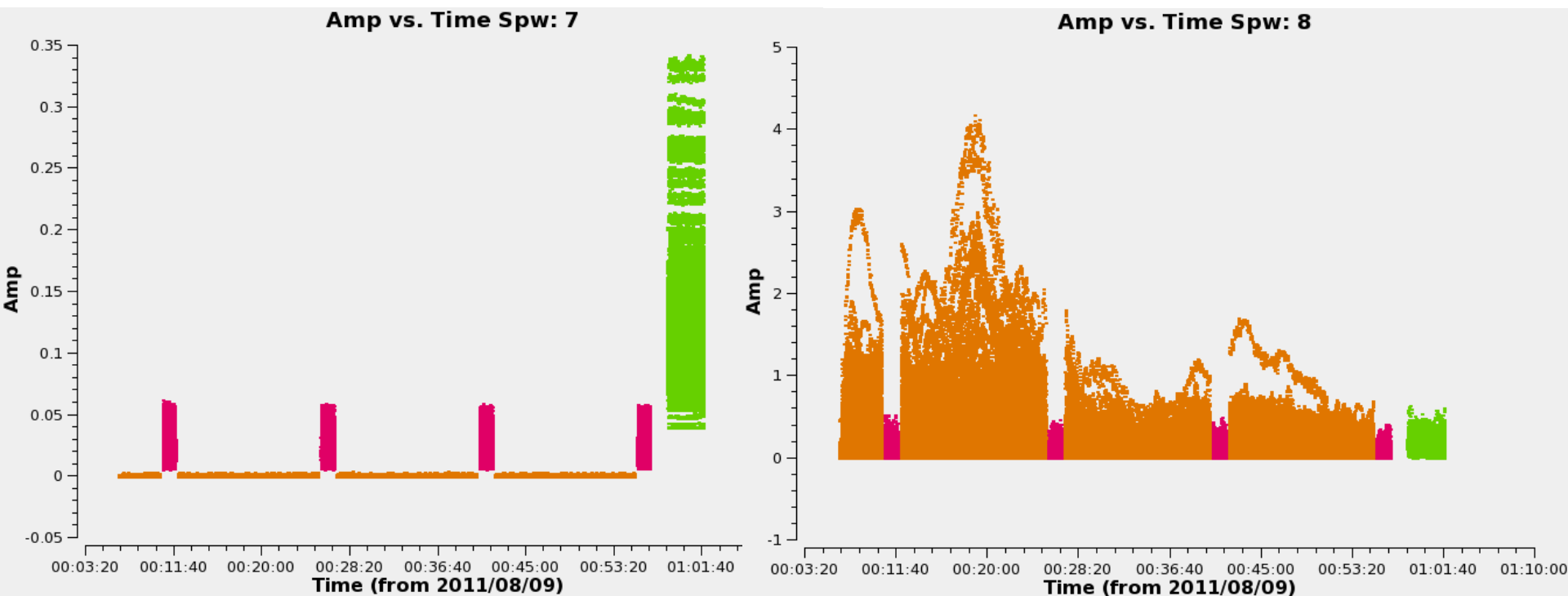
- Scan
- Field
- Spw
- Antenna1
- Antenna2
- Baseline
- Channel
- Correlation
- Time



Data Review: *plotms*

Example: x-axis: time, y-axis: amp

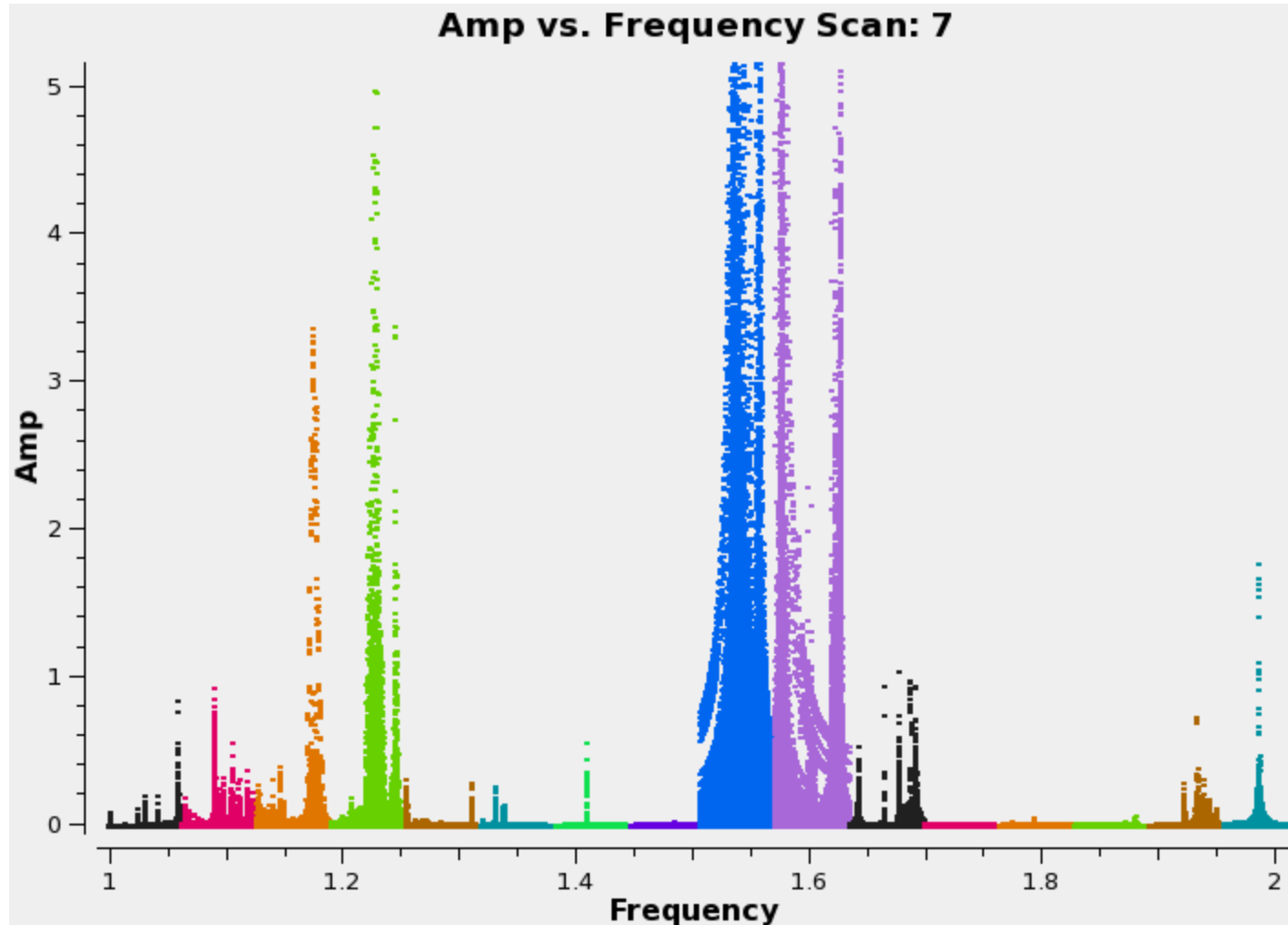
iter: spw (with all channels averaged)



Data Review: *plotms*

Example: x-axis: frequency, y-axis: amp

iteration: scan



Data review: *msview*

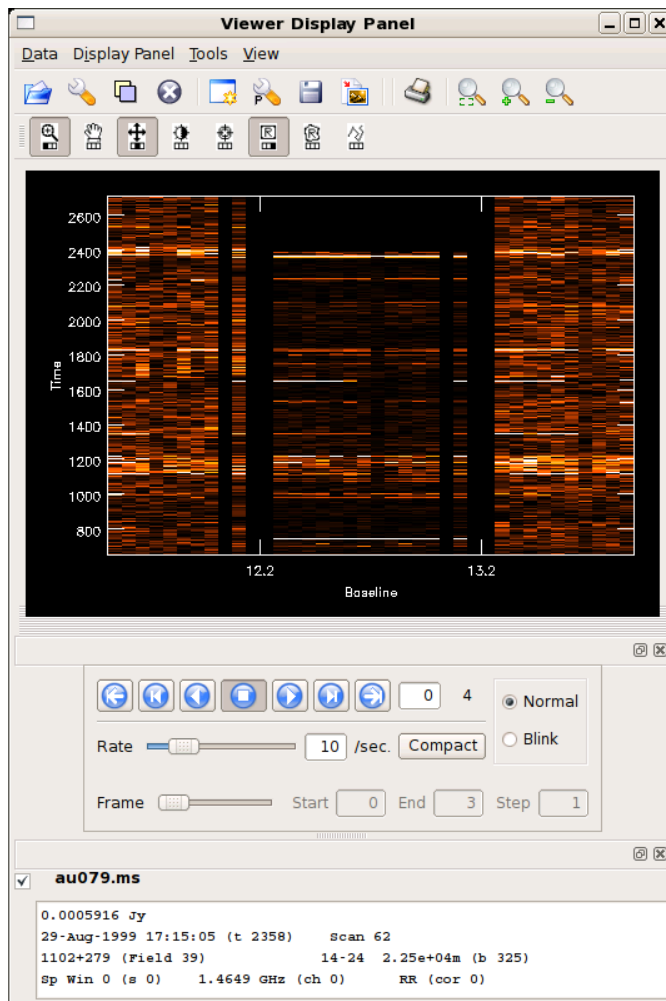
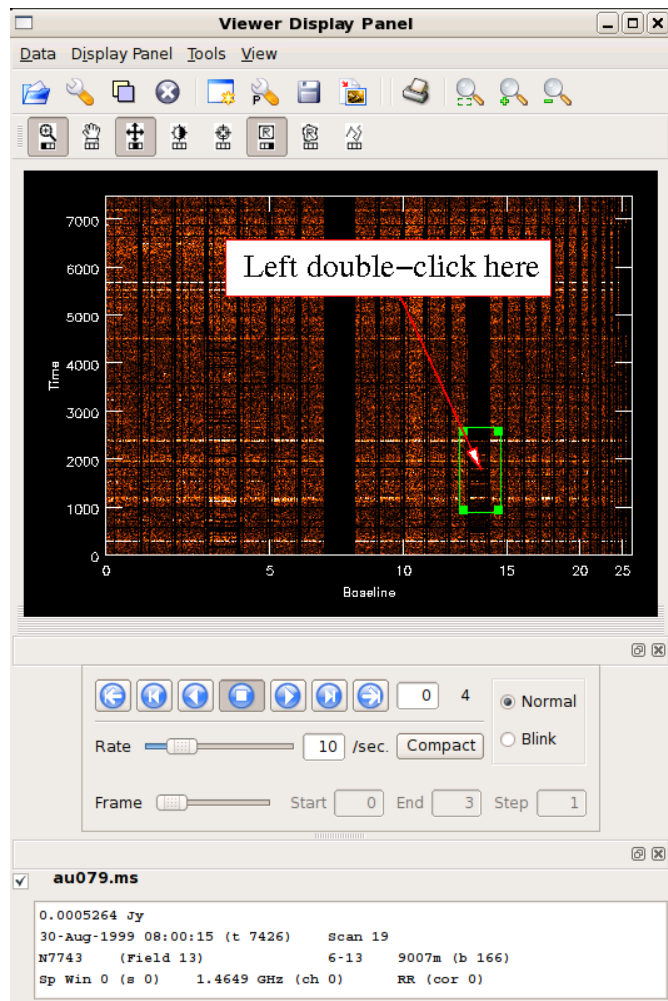


Image Viewer: *viewer*

The screenshot displays the 'Image Viewer' application window, which is divided into several panels. The main panel shows a J2000 astronomical image of a bright, orange, circular object (likely a galaxy or nebula) with overlaid blue contour lines. The axes are labeled 'J2000 Declination' (y-axis, ranging from 58' to 10') and 'J2000 Right Ascension' (x-axis, ranging from 15h 22m 18s to 36s). Below the main image is a control panel with various icons for navigation and zooming, a 'Rate' slider set to 10 /sec, and a 'Frame' slider set to 0. To the right of the main image is a 'Data Display Options' panel with various settings for the display, including 'Aspect ratio' (fixed world), 'Pixel treatment' (edge), 'Resampling mode' (bilinear), 'Relative Contour Levels' (0.2, 0.4, 0.6, 0.8), 'Base Contour Level' (1381.3), 'Unit Contour Level' (1567.1), 'Line width' (0.5), 'Dash negative contours?' (true), 'Dash positive contours?' (false), and 'Line color' (blue). At the bottom of the window, there are two tabs for data display: 'ngc5921.demo.moments.weighted_coord-contour' and 'ngc5921.demo.moments.integrated'. The first tab is active, showing a masked image with a pixel value of 155 120 0 0 and coordinates 15:21:32.830 +05:01:52.605, with a velocity of 1607.99 km/s. The second tab is also visible, showing a masked image with a pixel value of 155 120 0 0 and coordinates 15:21:32.830 +05:01:52.605, with a velocity of 1607.99 km/s.

Viewer Display Panel

Data Display Panel Tools View

Trash

J2000 Declination

J2000 Right Ascension

Rate 10 /sec. Compact

Frame Start 0 End 0 Step 1

ngc5921.demo.moments.weighted_coord-contour

masked Pixel: 155 120 0 0

15:21:32.830 +05:01:52.605 1607.99 km/s

Contours: 1418.5 1455.6 1492.8 1529.9

ngc5921.demo.moments.integrated

masked Pixel: 155 120 0 0

15:21:32.830 +05:01:52.605 1607.99 km/s

Data Display Options

ngc5921.demo.moments.weighted_coord-contour ngc5921.demo.moments.integrated

Display axes

Hidden axes

Basic Settings

Aspect ratio fixed world

Pixel treatment edge

Resampling mode bilinear

Relative Contour Levels [0.2, 0.4, 0.6, 0.8]

Base Contour Level 1381.3

Unit Contour Level 1567.1

Line width 0.5

Dash negative contours? true

Dash positive contours? false

Line color blue

Position tracking

Axis labels

Axis label properties

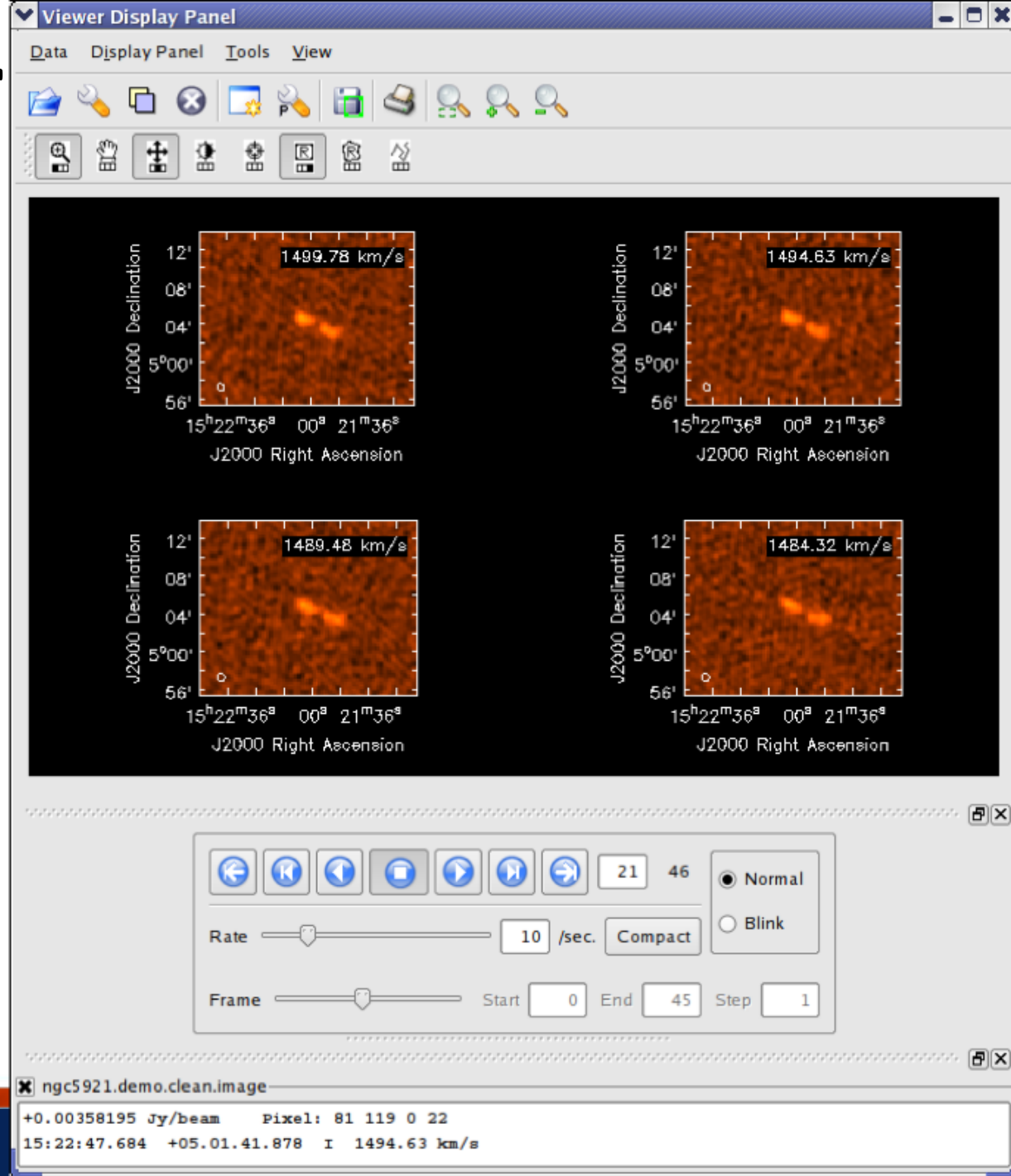
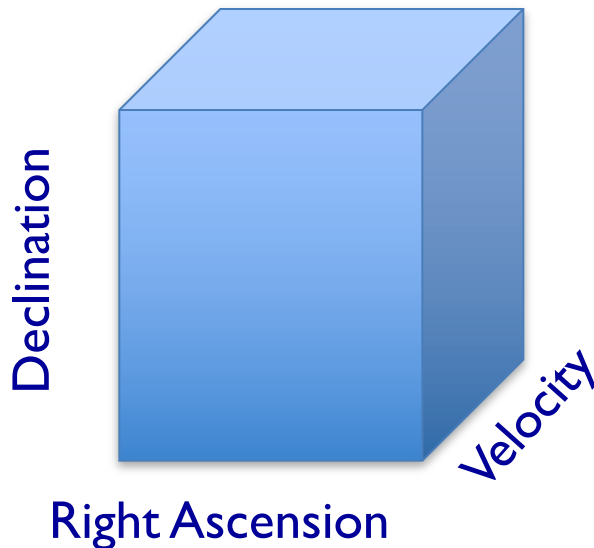
Beam Ellipse

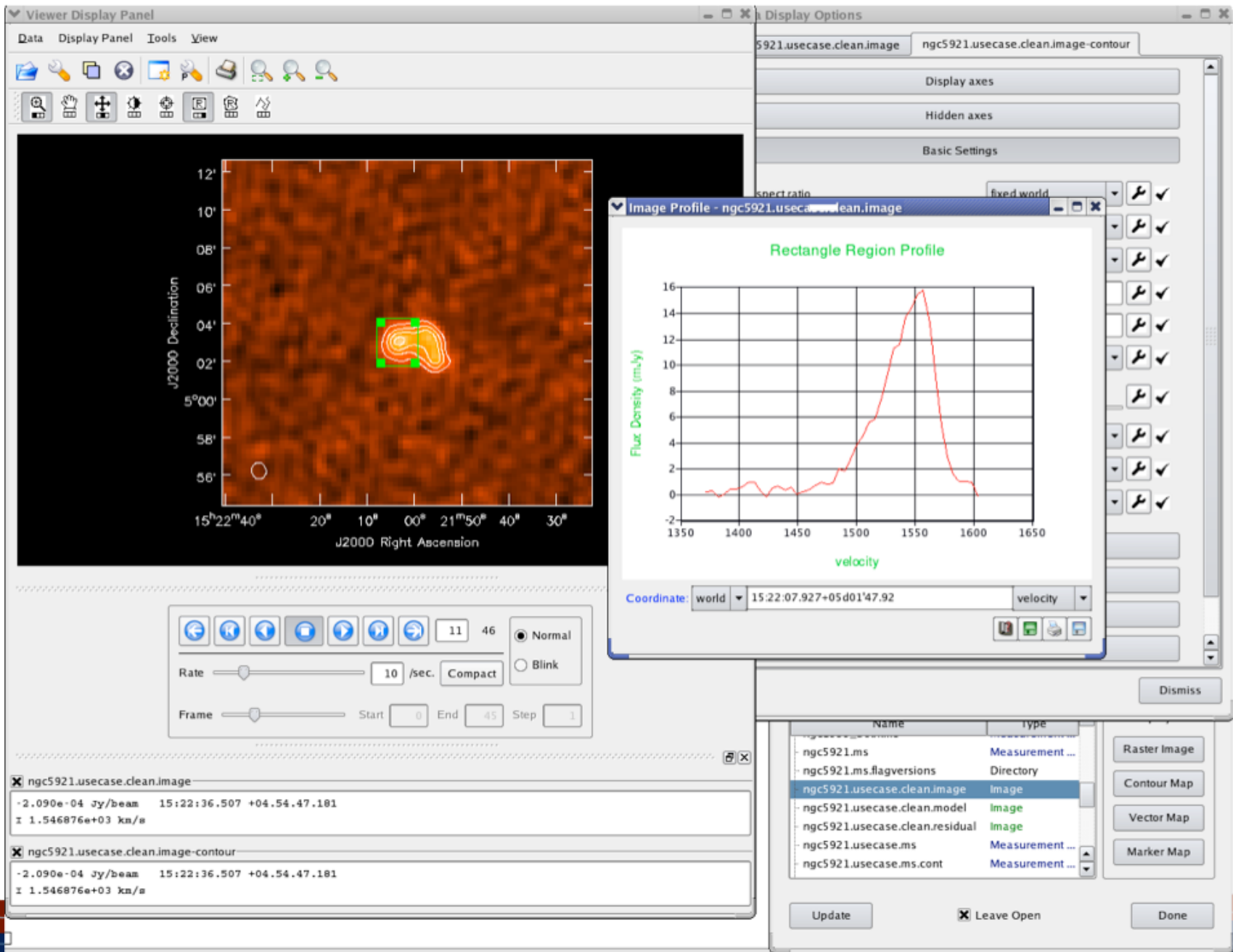
Apply

Dismiss

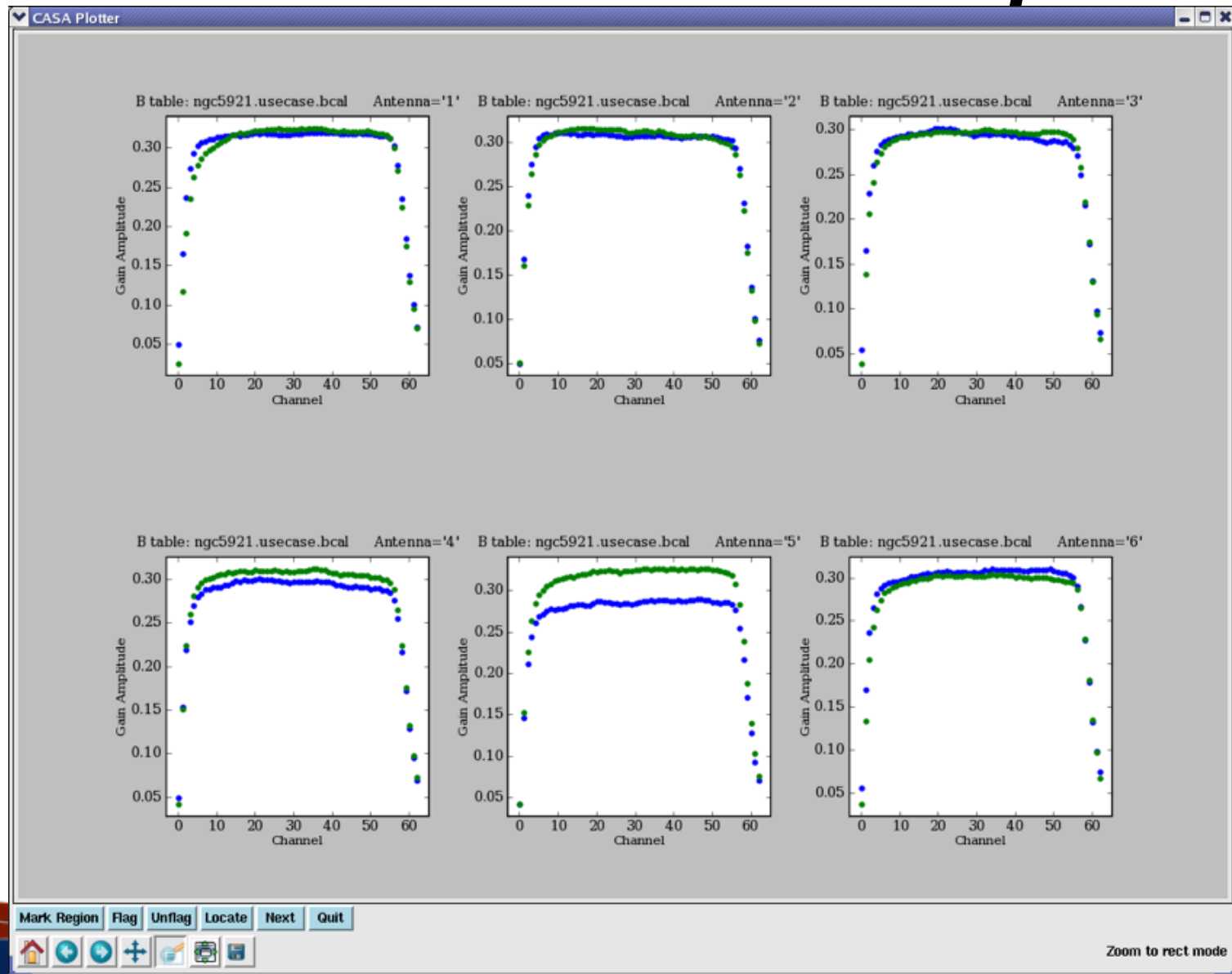
Image Viewer

- Displaying cubes
- Movies
- Channel maps

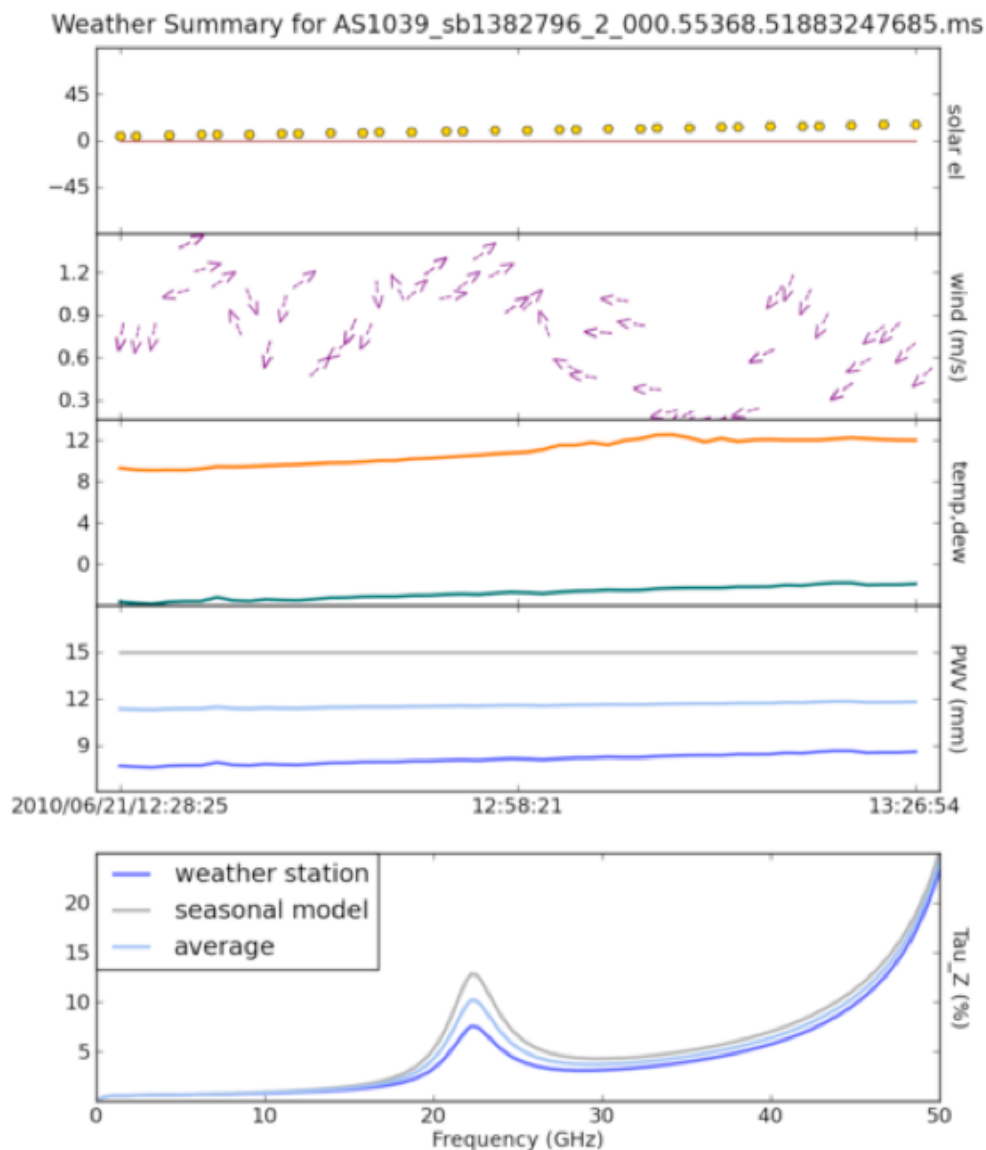




Review calibration tables: *plotcal*



Anything - matplotlib



Buildmytasks

- Using Python, you can write your own scripts!
- Such scripts can be converted to tasks.
- If you wish, you can share them with the community (e.g., through NRAO).
- Contributed scripts are currently available at:
<https://casaguides.nrao.edu/index.php/UST2>



www.nrao.edu
science.nrao.edu
public.nrao.edu

*The National Radio Astronomy Observatory is a facility of the National Science Foundation
operated under cooperative agreement by Associated Universities, Inc.*