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

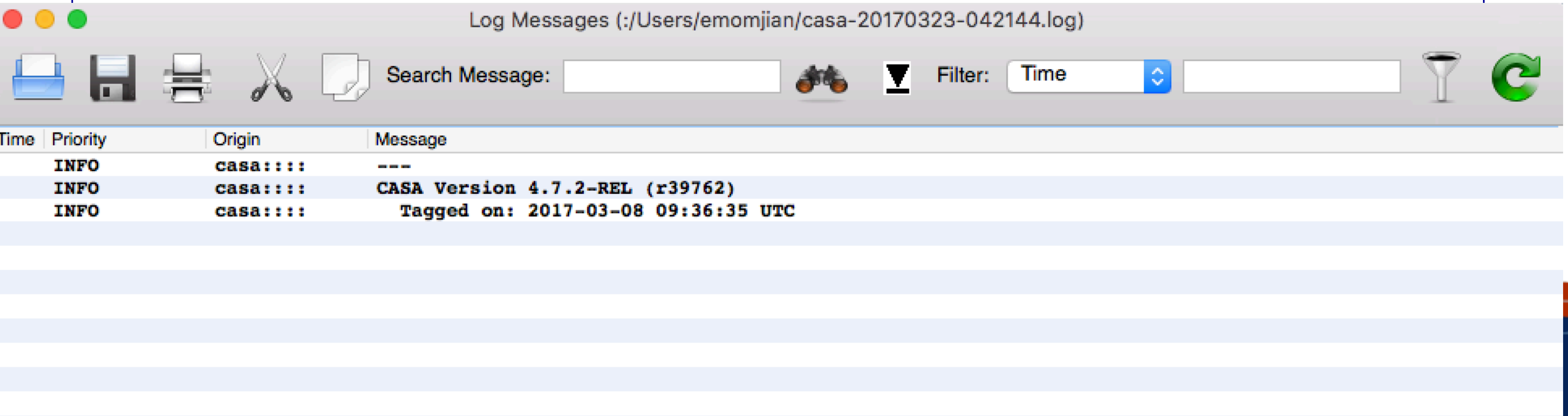
- Documentation is available at http://casa.nrao.edu/ → 'Using CASA'
- Training material is available at http://casaguides.nrao.edu
- For help, use the NRAO help desk at: http://help.nrao.edu

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
CASA Version 4.7.2-REL (r39762)
   Compiled on: Wed 2017/03/08 09:55:37 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-20170323-042148.log
Mode           : backupOutput logging : False
Raw input log  : FalseTimestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (rev#URL:) import complete ***
CASA <2>:
```



```
Time  Priority  Origin     Message
      INFO      casa::::   ---
      INFO      casa::::   CASA Version 4.7.2-REL (r39762)
      INFO      casa::::     Tagged on: 2017-03-08 09:36:35 UTC
```

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 difined 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

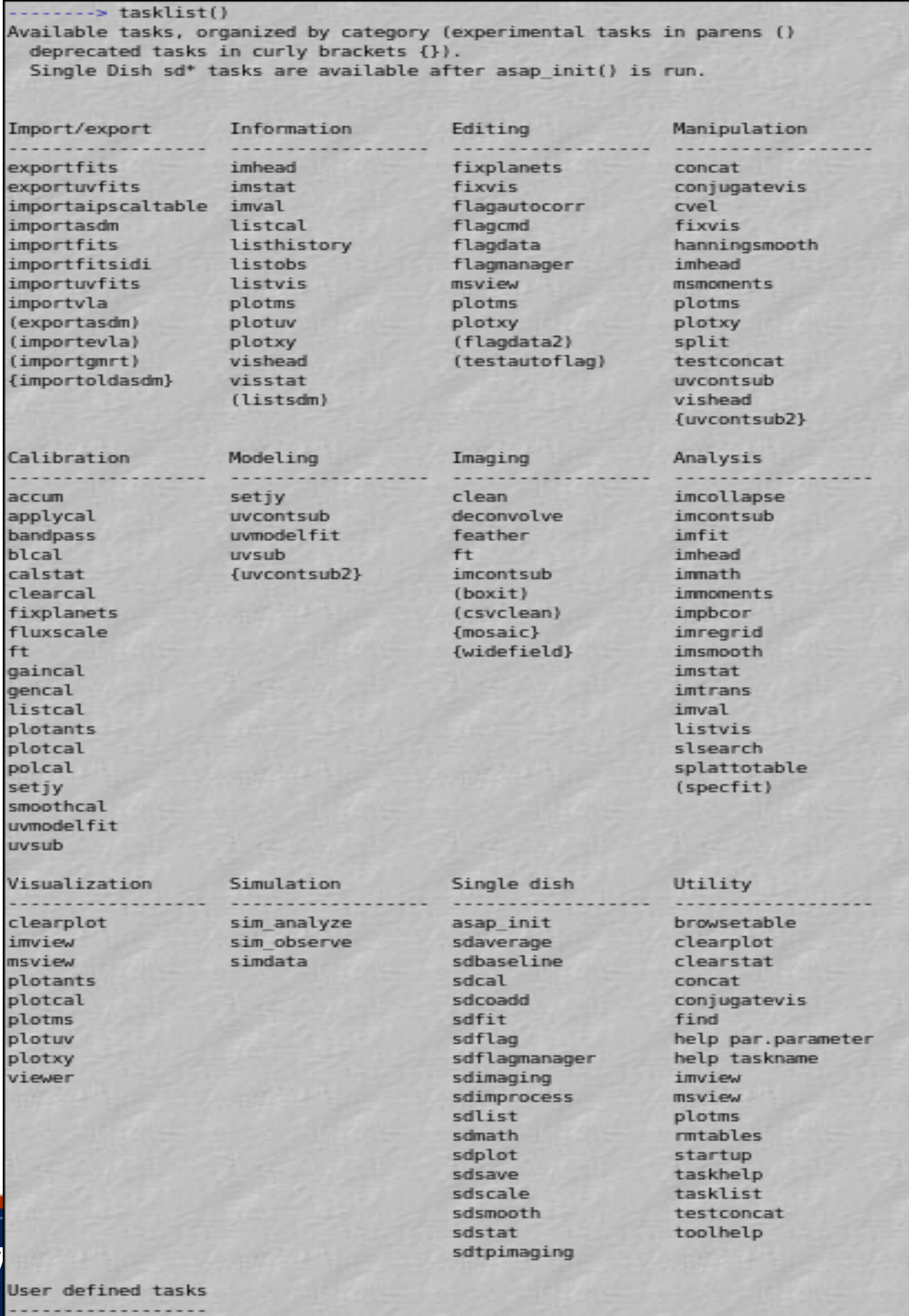
- Shell commands can be run with a leading exclamation mark   !du –hs

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
- (Historic) MIRIAD-CASA and CLIC-CASA dictionaries are available in the CASA cookbook.

Tasks

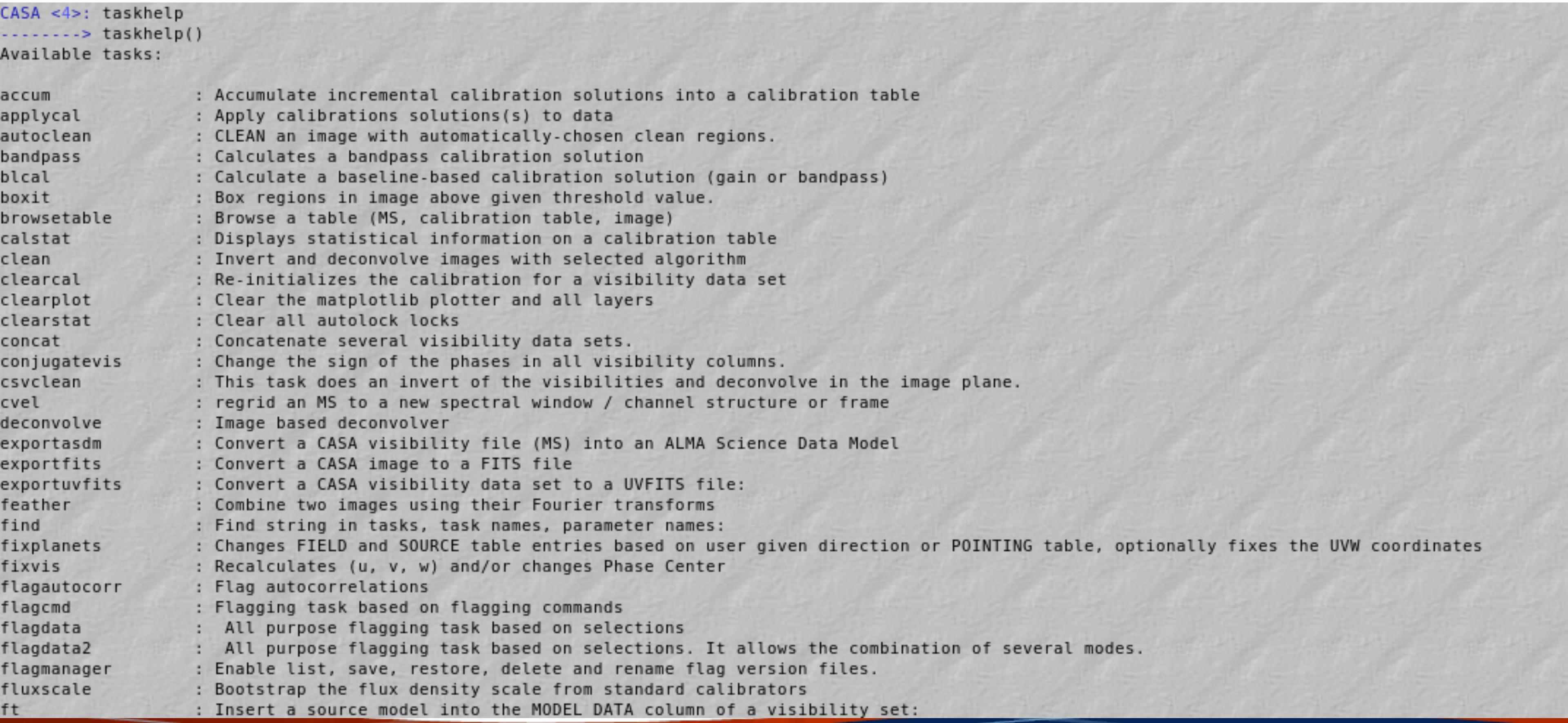
To list the tasks*: tasklist*



```
---------> tasklist()
Available tasks, organized by category (experimental tasks in parens ()
  deprecated tasks in curly brackets {}).
  Single Dish sd* tasks are available after asap_init() is run.


Import/export       Information         Editing             Manipulation
------------------  ------------------  ------------------  ------------------
exportfits          imhead              fixplanets          concat
exportuvfits        imstat              fixvis              conjugatevis
importaipscaltable  imval               flagautocorr        cvel
importasdm          listcal             flagcmd             fixvis
importfits          listhistory         flagdata            hanningsmooth
importfitsidi       listobs             flagmanager         imhead
importuvfits        listvis             msview              msmoments
importvla           plotms              plotms              plotms
(exportasdm)        plotuv              plotxy              plotxy
(importevla)        plotxy              (flagdata2)         split
(importgmrt)        vishead             (testautoflag)      testconcat
{importoldasdm}     visstat                                 uvcontsub
                    (listsdm)                               vishead
                                                            {uvcontsub2}

Calibration         Modeling            Imaging             Analysis
------------------  ------------------  ------------------  ------------------
accum               setjy               clean               imcollapse
applycal            uvcontsub           deconvolve          imcontsub
bandpass            uvmodelfit          feather             imfit
blcal               uvsub               ft                  imhead
calstat             {uvcontsub2}        imcontsub           immath
clearcal                                (boxit)             immoments
fixplanets                              (csvclean)          impbcor
fluxscale                               {mosaic}            imregrid
ft                                      {widefield}         imsmooth
gaincal                                                     imstat
gencal                                                      imtrans
listcal                                                     imval
plotants                                                    listvis
plotcal                                                     slsearch
polcal                                                      splattotable
setjy                                                       (specfit)
smoothcal
uvmodelfit
uvsub

Visualization       Simulation          Single dish         Utility
------------------  ------------------  ------------------  ------------------
clearplot           sim_analyze         asap_init           browsetable
imview              sim_observe         sdaverage           clearplot
msview              simdata             sdbaseline          clearstat
plotants                                sdcal               concat
plotcal                                 sdcoadd             conjugatevis
plotms                                  sdfit               find
plotuv                                  sdflag              help par.parameter
plotxy                                  sdflagmanager       help taskname
viewer                                  sdimaging           imview
                                        sdimprocess         msview
                                        sdlist              plotms
                                        sdmath              rmtables
                                        sdplot              startup
                                        sdsave              taskhelp
                                        sdscale             tasklist
                                        sdsmooth            testconcat
                                        sdstat              toolhelp
                                        sdtpimaging

User defined tasks
------------------
```

Tasks

To see list of tasks with short help:
taskhelp



```
CASA <4>: taskhelp
--------> taskhelp()
Available tasks:

accum              : Accumulate incremental calibration solutions into a calibration table
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 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
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
deconvolve         : Image based deconvolver
exportasdm         : Convert a CASA visibility file (MS) into an ALMA 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 given direction or POINTING table, optionally fixes the UVW coordinates
fixvis             : Recalculates (u, v, w) and/or changes Phase Center
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 combination of several modes.
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 set:
```

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())

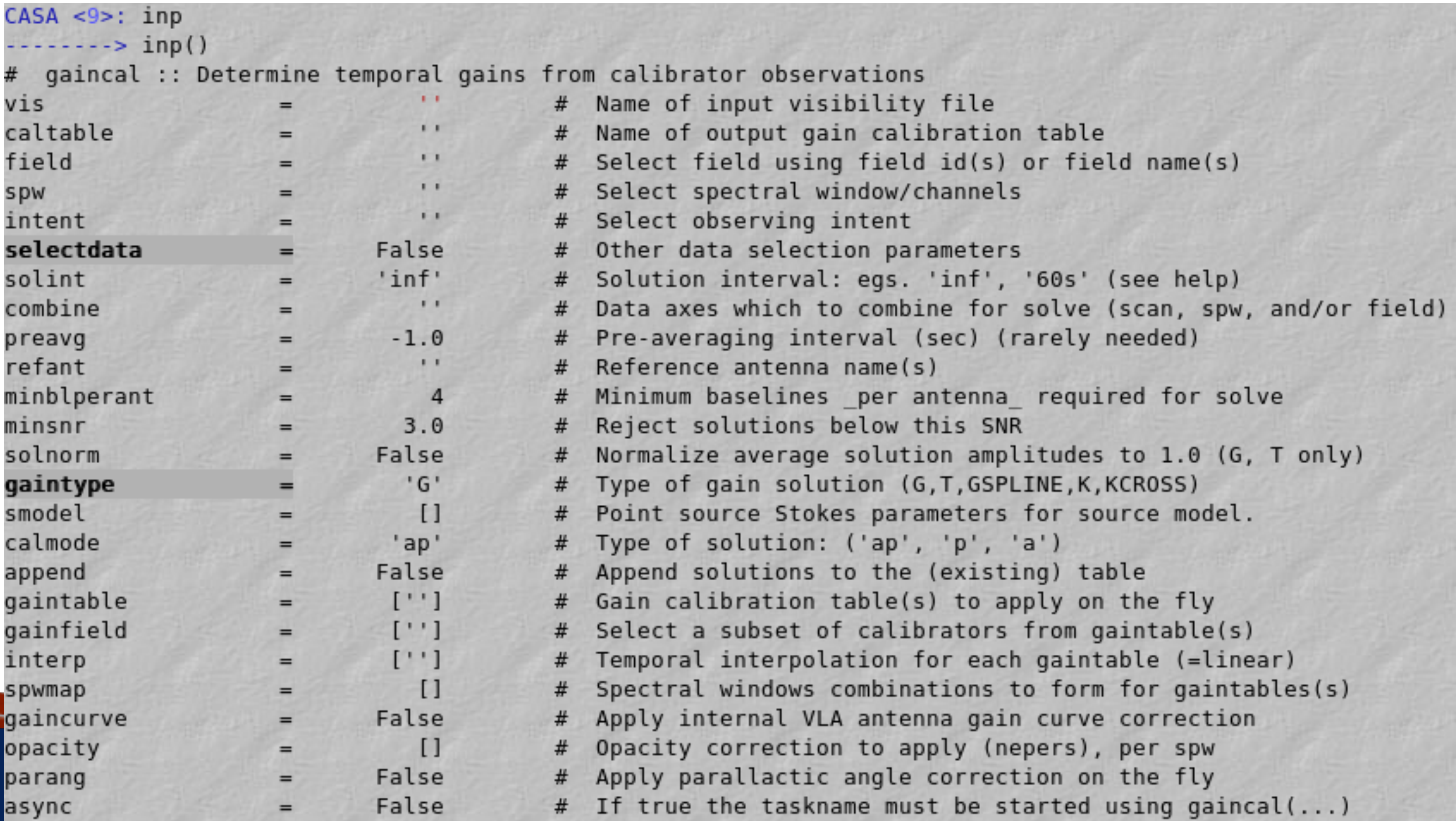
Task Interface

Call a task by

>inp <*taskname*>

if default values are desired, first type

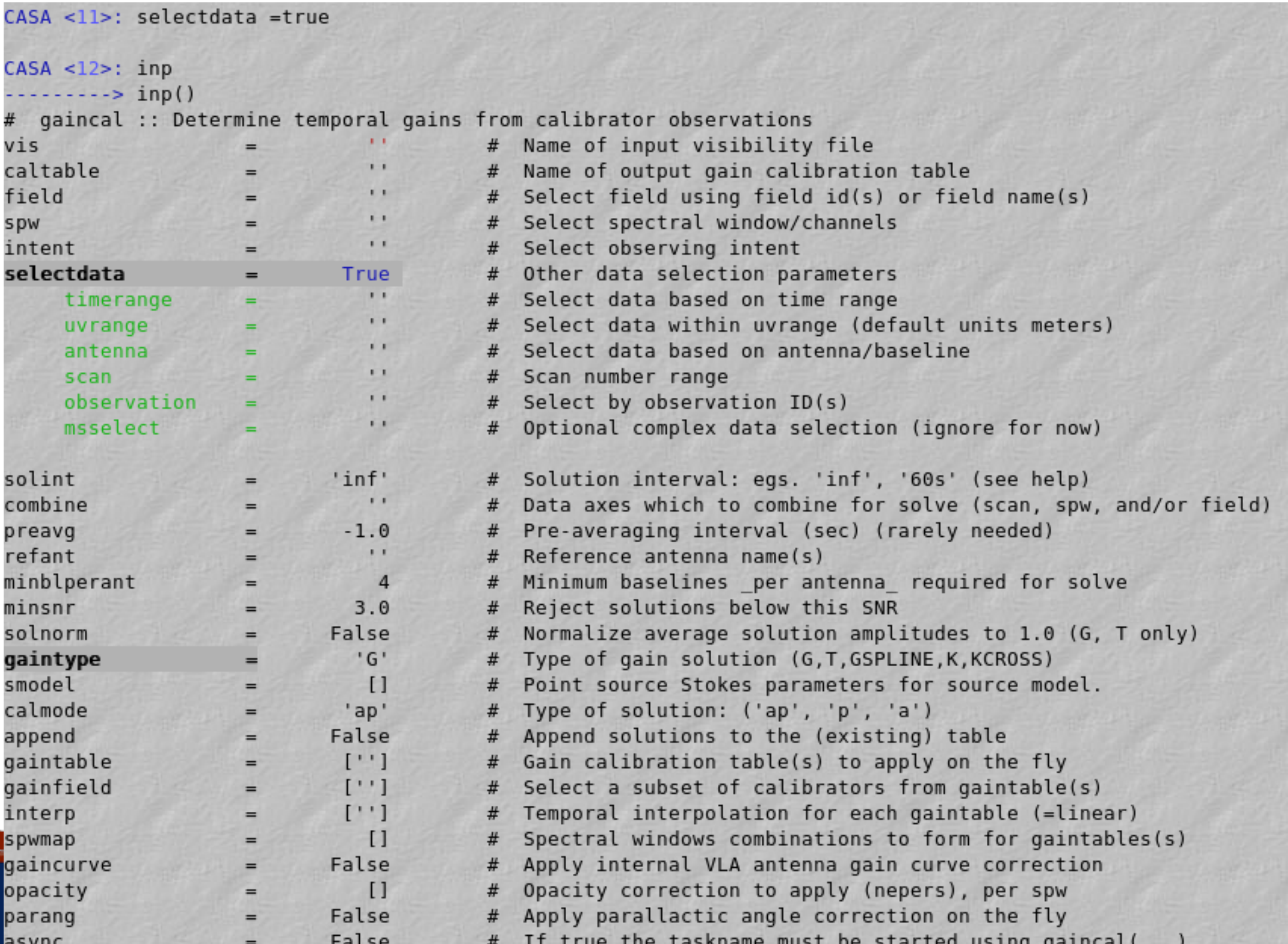
>default <*taskname*>, followed by inp



```
CASA <9>: 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          =       False       #  Other data selection parameters
solint              =       'inf'       #  Solution interval: egs. 'inf', '60s' (see help)
combine             =          ''       #  Data axes which to combine for solve (scan, spw, and/or field)
preavg              =        -1.0       #  Pre-averaging interval (sec) (rarely needed)
refant              =          ''       #  Reference antenna name(s)
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
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)
gaincurve           =       False       #  Apply internal VLA antenna gain curve correction
opacity             =          []       #  Opacity correction to apply (nepers), per spw
parang              =       False       #  Apply parallactic angle correction on the fly
async               =       False       #  If true the taskname must be started using gaincal(...)
```

Task Interface

Some parameters are expandable, e.g., selectdata



```
CASA <11>: selectdata =true

CASA <12>: 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: egs. 'inf', '60s' (see help)
combine             =         ''        #  Data axes which to combine for solve (scan, spw, and/or field)
preavg              =       -1.0        #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''        #  Reference antenna name(s)
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
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)
gaincurve           =      False        #  Apply internal VLA antenna gain curve correction
opacity             =         []        #  Opacity correction to apply (nepers), per spw
parang              =      False        #  Apply parallactic angle correction on the fly
async               =      False        #  If true the taskname must be started using gaincal(...)
```

Task Execution

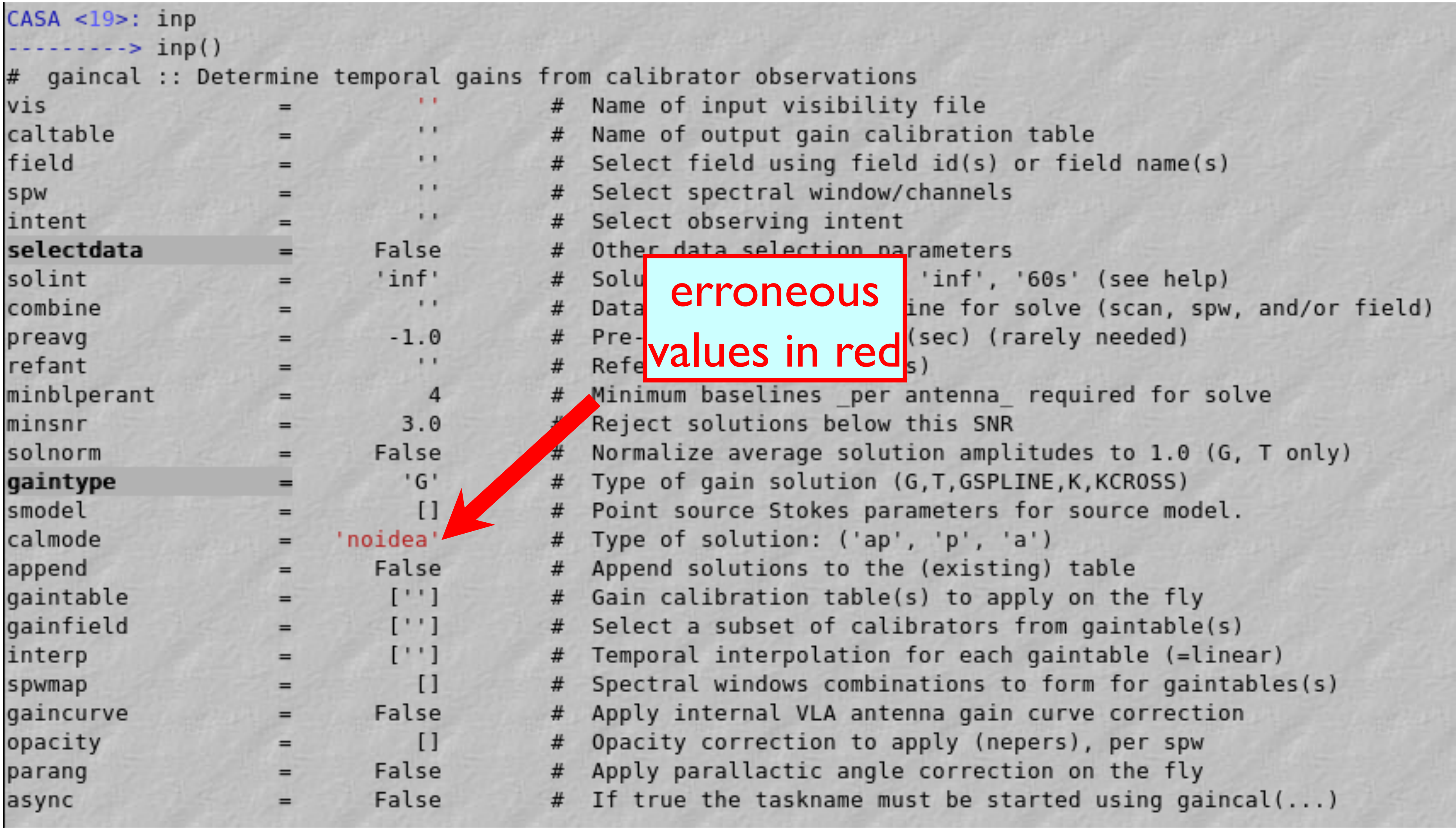
- Two ways to invoke:
  - call from Python as functions with arguments

    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



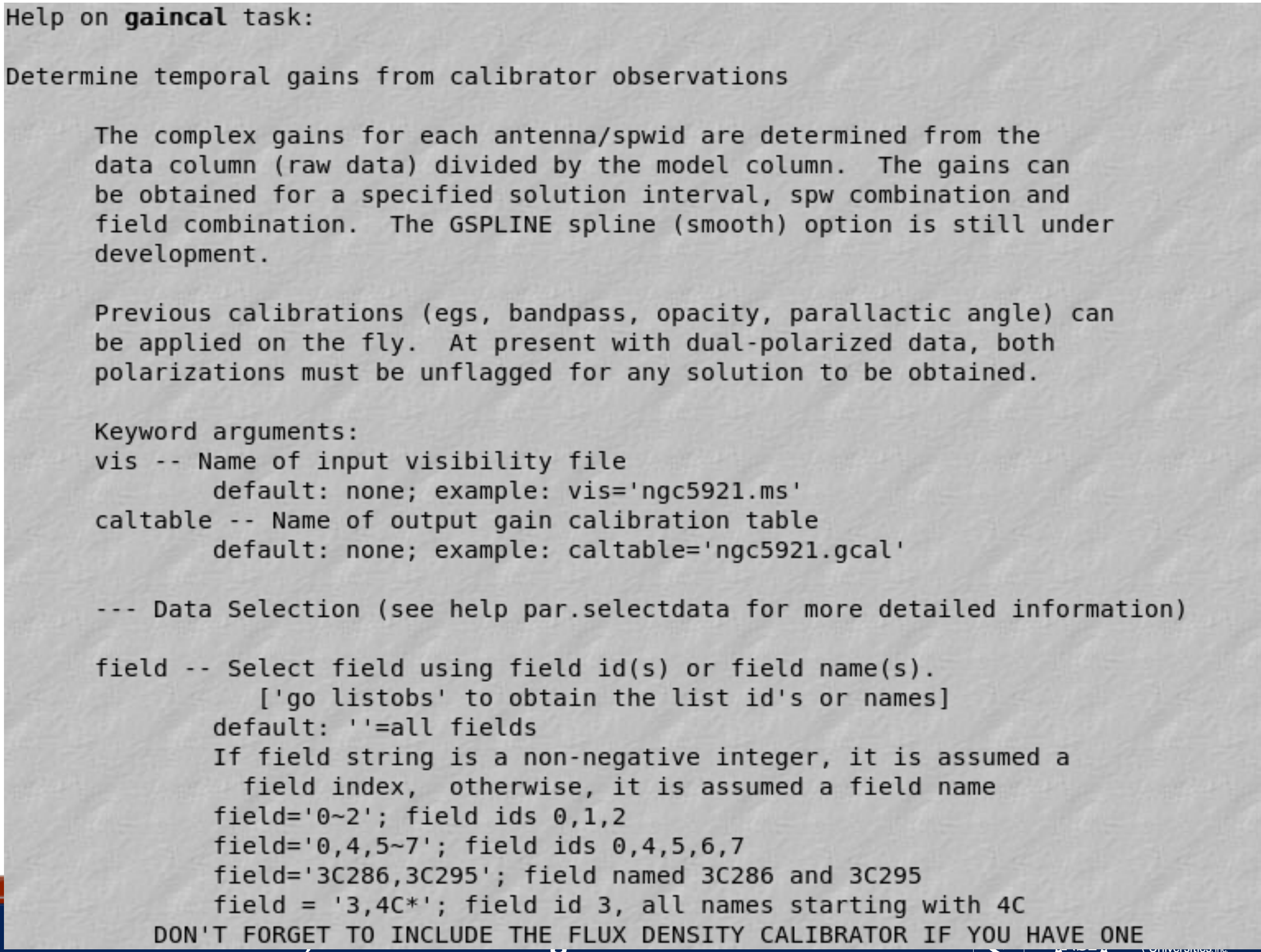
```
CASA <19>: inp
----------> inp()
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''        #  Name of input visibility file
caltable            =         ''        #  Name of output gain calibration table
field               =         ''        #  Select field using field id(s) or field name(s)
spw                 =         ''        #  Select spectral window/channels
intent              =         ''        #  Select observing intent
selectdata          =      False        #  Other data selection parameters
solint              =      'inf'        #  Solu                  'inf', '60s' (see help)
combine             =         ''        #  Data                  ine for solve (scan, spw, and/or field)
preavg              =       -1.0        #  Pre-                  (sec) (rarely needed)
refant              =         ''        #  Refe                  s)
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
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)
gaincurve           =      False        #  Apply internal VLA antenna gain curve correction
opacity             =         []        #  Opacity correction to apply (nepers), per spw
parang              =      False        #  Apply parallactic angle correction on the fly
async               =      False        #  If true the taskname must be started using gaincal(...)
```

erroneous values in red

Help on Tasks

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

>help gaincal



```
Help on gaincal task:

Determine temporal gains from calibrator observations

      The complex gains for each antenna/spwid are determined from the
      data column (raw data) divided by the model column.  The gains can
      be obtained for a specified solution interval, spw combination and
      field combination.  The GSPLINE spline (smooth) option is still under
      development.

      Previous calibrations (egs, bandpass, opacity, parallactic angle) can
      be applied on the fly.  At present with dual-polarized data, both
      polarizations must be unflagged for any solution to be obtained.

      Keyword arguments:
      vis -- Name of input visibility file
              default: none; example: vis='ngc5921.ms'
      caltable -- Name of output gain calibration table
              default: none; example: caltable='ngc5921.gcal'

      --- Data Selection (see help par.selectdata for more detailed information)

      field -- Select field using field id(s) or field name(s).
                 ['go listobs' to obtain the list id's or names]
              default: ''=all fields
              If field string is a non-negative integer, it is assumed a
                field index,  otherwise, it is assumed a field name
              field='0~2'; field ids 0,1,2
              field='0,4,5~7'; field ids 0,4,5,6,7
              field='3C286,3C295'; field named 3C286 and 3C295
              field = '3,4C*'; field id 3, all names starting with 4C
          DON'T FORGET TO INCLUDE THE FLUX DENSITY CALIBRATOR IF YOU HAVE ONE
```

Tools in CASA

- What if there's no task?

→ use CASA tools (tasks are built upon tools)

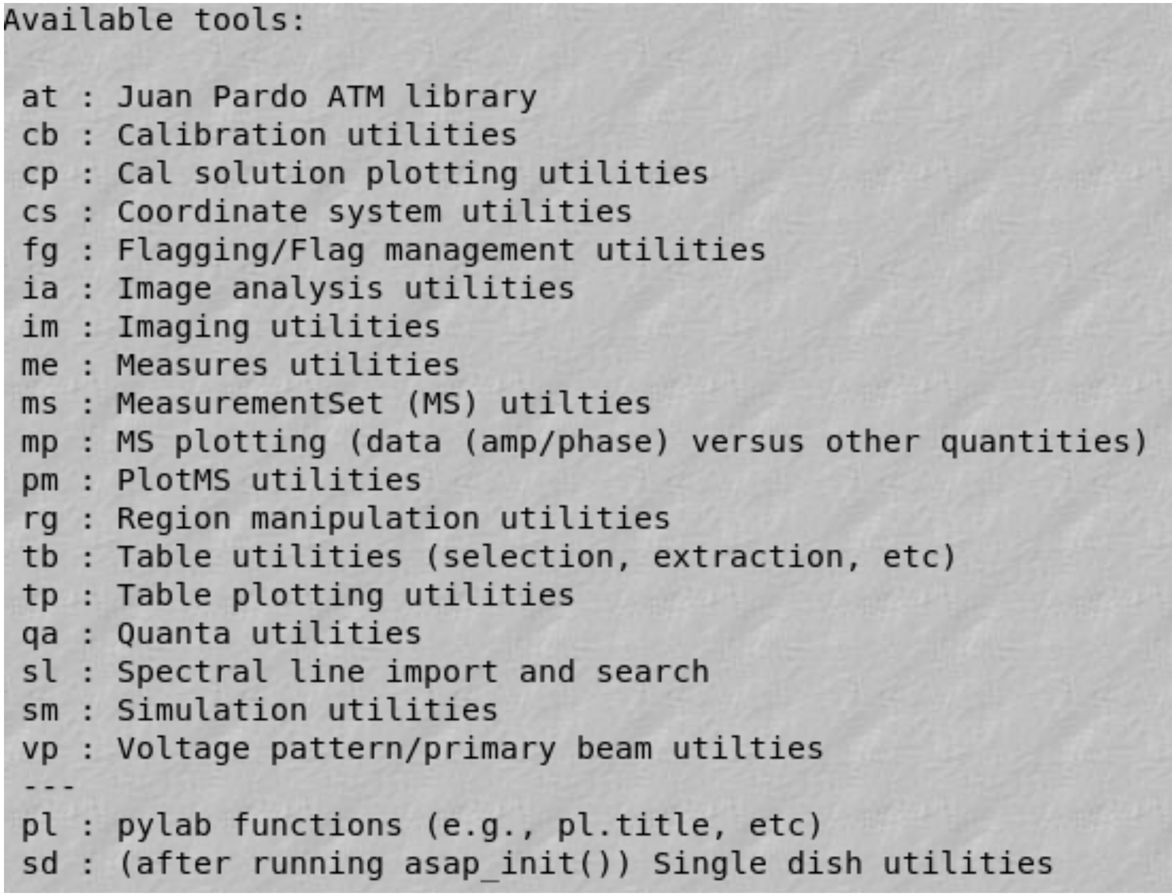
- tools are functions/methods
  - call from casapy as <tool>.<method>()
  - default tool objects are pre-constructed
    - e.g. imager (im) , calibrater (cb), ms (ms) , etc. (see toolhelp)

CASA Tool List

To list the default tools:

>toolhelp

~1000 tools available



```
Available tools:

 at : Juan Pardo ATM library
 cb : Calibration utilities
 cp : Cal solution plotting utilities
 cs : Coordinate system utilities
 fg : Flagging/Flag management utilities
 ia : Image analysis utilities
 im : Imaging utilities
 me : Measures utilities
 ms : MeasurementSet (MS) utilties
 mp : MS plotting (data (amp/phase) versus other quantities)
 pm : PlotMS utilities
 rg : Region manipulation utilities
 tb : Table utilities (selection, extraction, etc)
 tp : Table plotting utilities
 qa : Quanta utilities
 sl : Spectral line import and search
 sm : Simulation utilities
 vp : Voltage pattern/primary beam utilties
 ---
 pl : pylab functions (e.g., pl.title, etc)
 sd : (after running asap_init()) Single dish utilities
```

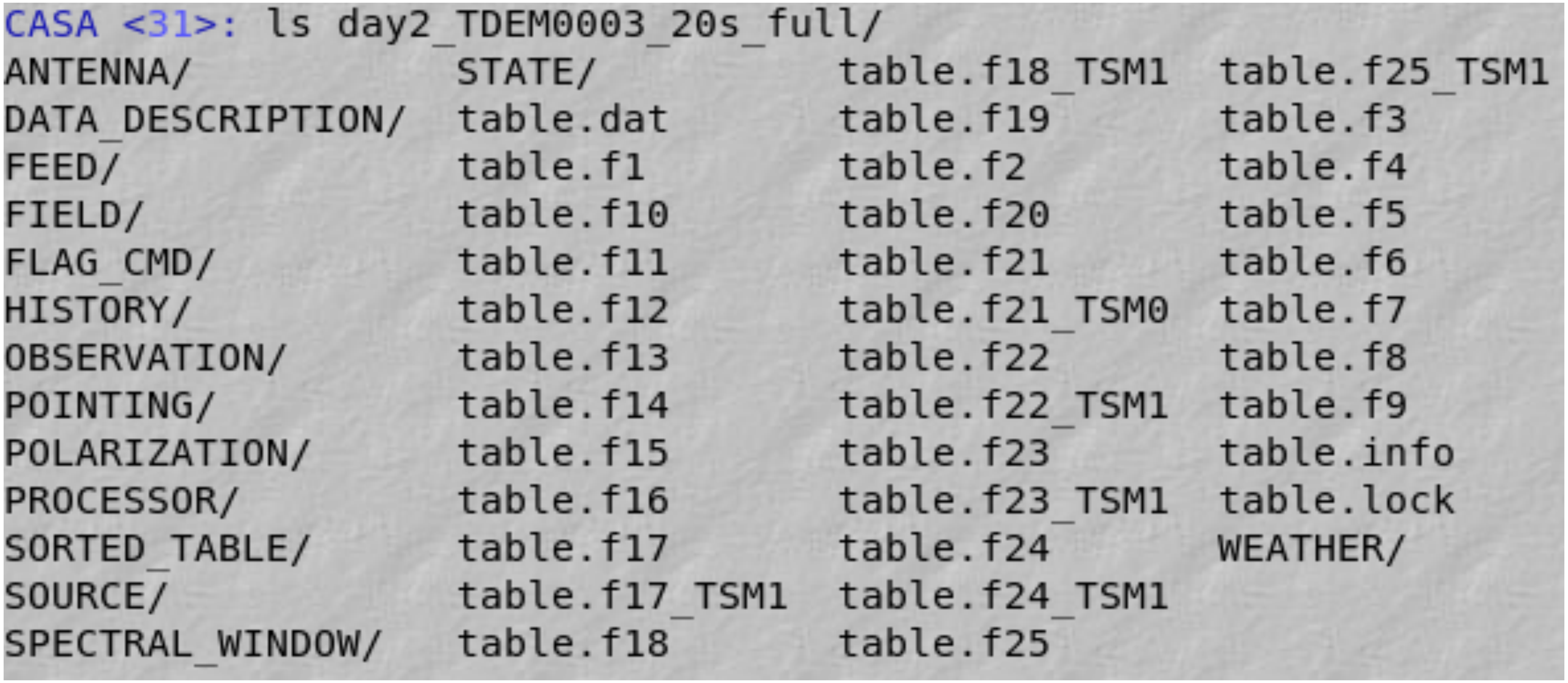
Tools are described in the CASA Toolkit Reference:

http://casa.nrao.edu/docs/CasaRef/CasaRef.html

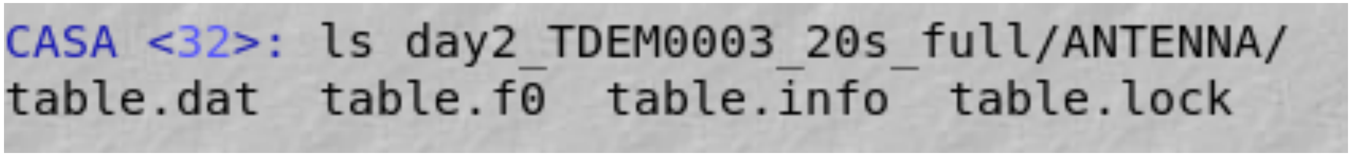
The Measurement Set

- 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

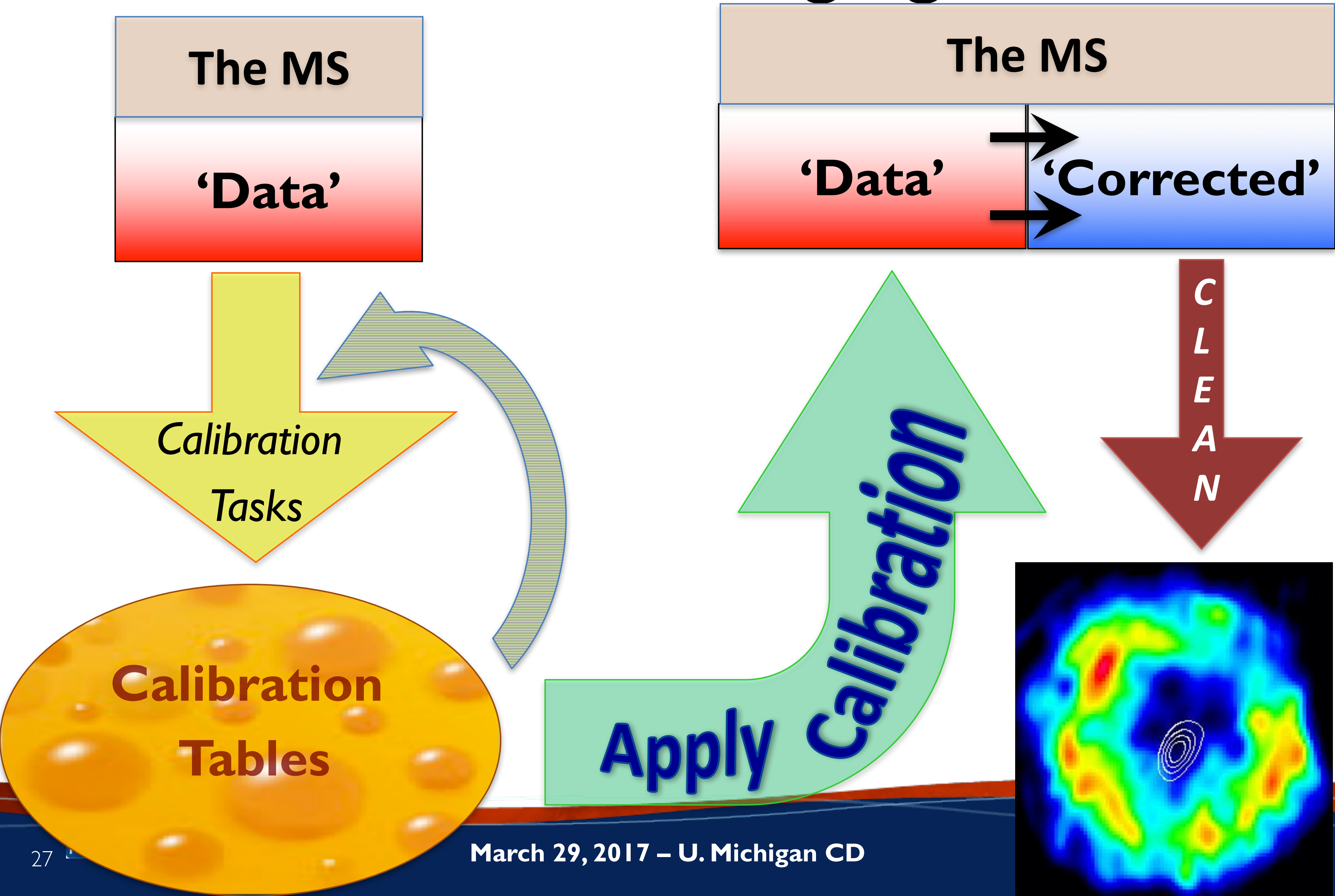
- specify ‘T0~T1’, missing parts of T1 default to T0.
- example: timerange = ‘2007/10/16/01:00:00~06:30:00’
- If year, month, day are not specified → defaults to 1st day in the data set.

The MS structure

‘Data’ column	**‘Corrected’ Column**	**‘Model’ Column** (optional)
Raw Data	Calibrated Data	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.
- The ‘Model’ Column is optional.
  - If not created → MS doubling in size.
  - Models can be “attached” to the MS, and used when needed (replacing the need for the ‘Model’ column).

Calibration & Imaging Flow



The MS

‘Data’

Calibration Tasks

Calibration Tables

Apply Calibration

The MS

‘Data’

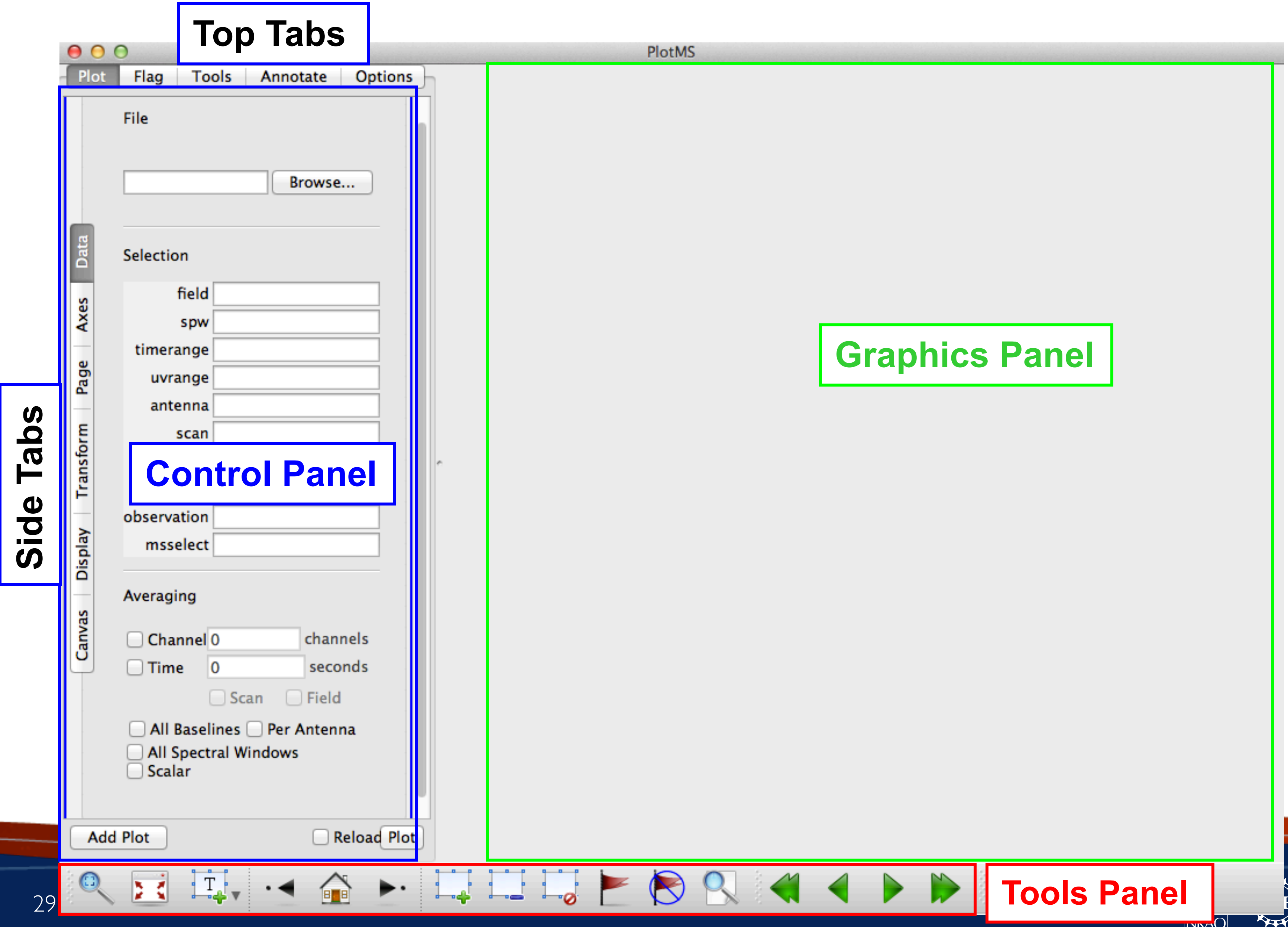
‘Corrected’

CLEAN

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)*

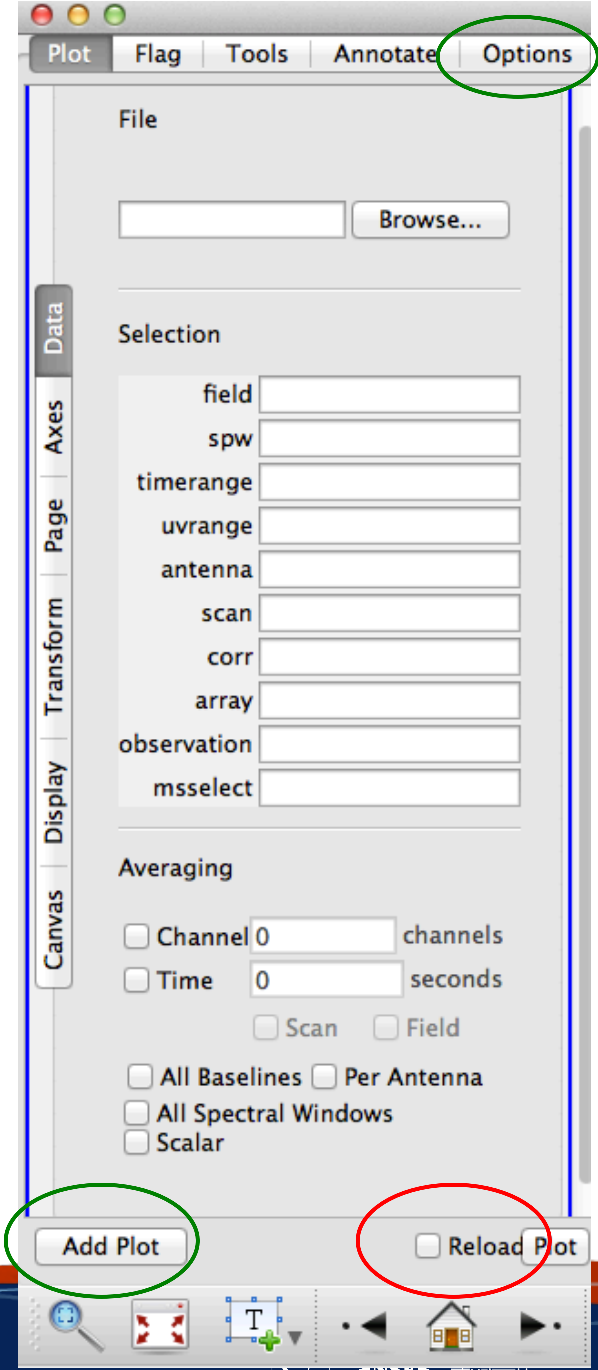


Data Review: *plotms*

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 graphic panel.



Data Review: *plotms*

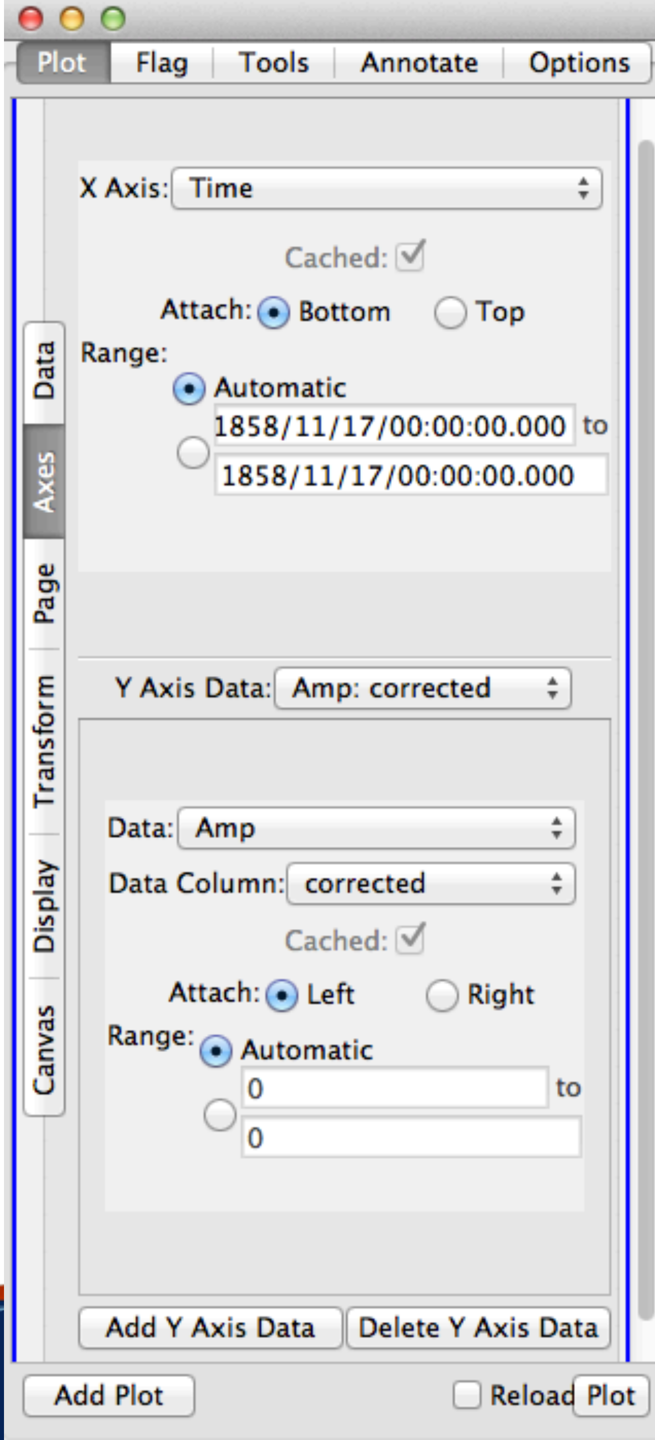
Axes

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' (absoute row Id from the MS)

Visibility values, flags:

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



X Axis: Time
Cached: ☑
Attach: ● Bottom ○ Top
Range:
● Automatic
○ 1858/11/17/00:00:00.000 to 1858/11/17/00:00:00.000

Y Axis Data: Amp: corrected
Data: Amp
Data Column: corrected
Cached: ☑
Attach: ● Left ○ Right
Range:
● Automatic
○ 0 to 0

□ Reload

Data Review: *plotms*

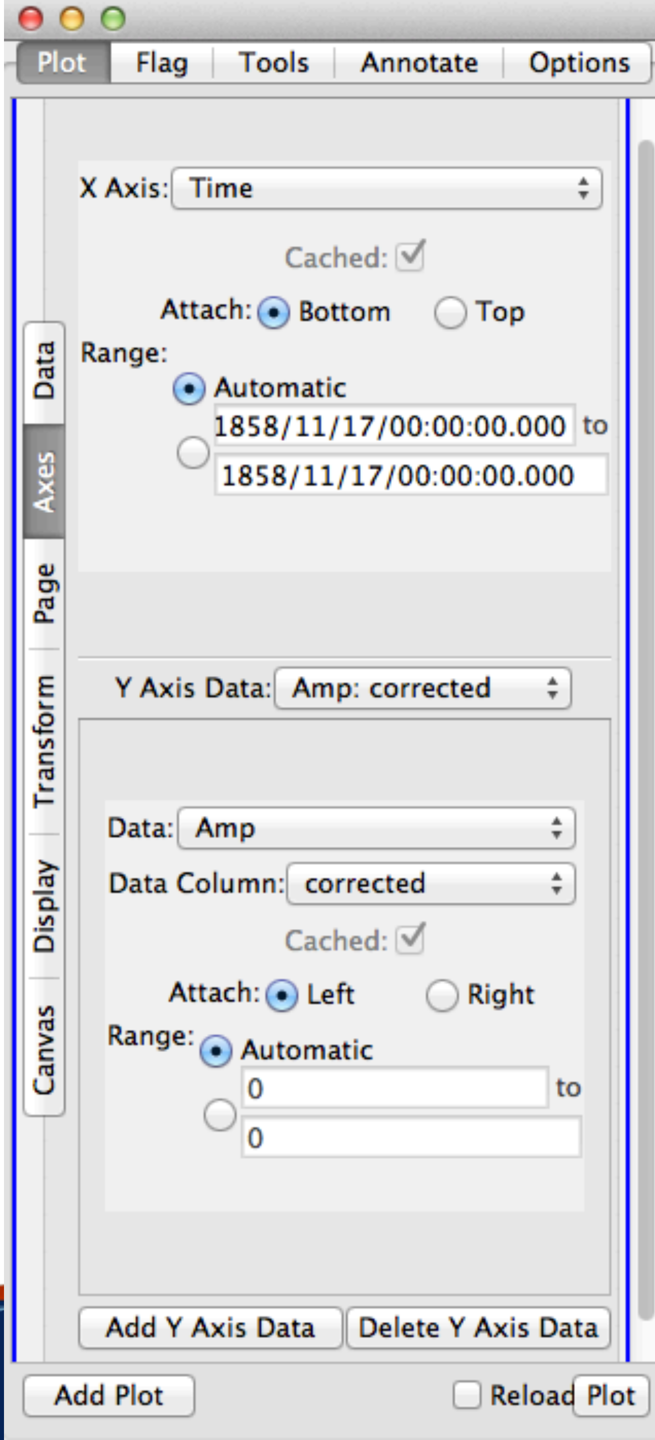
Axes

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 Ids):

- 'ant'='antenna'
- 'ant-azimuth'
- 'ant-elevation'
- 'ant-parang'='ant-parangle'



X Axis: Time

Cached: ☑

Attach: ● Bottom ○ Top

Range:
● Automatic
○ 1858/11/17/00:00:00.000 to 1858/11/17/00:00:00.000

Y Axis Data: Amp: corrected

Data: Amp

Data Column: corrected

Cached: ☑

Attach: ● Left ○ Right

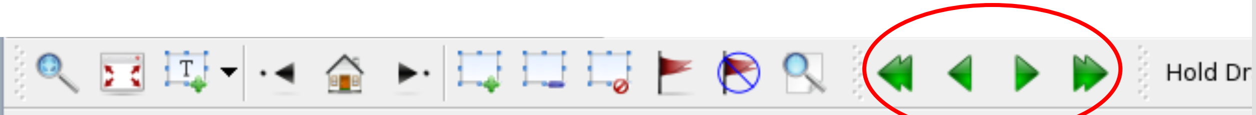
Range: ● Automatic
○ 0 to 0

□ Reload

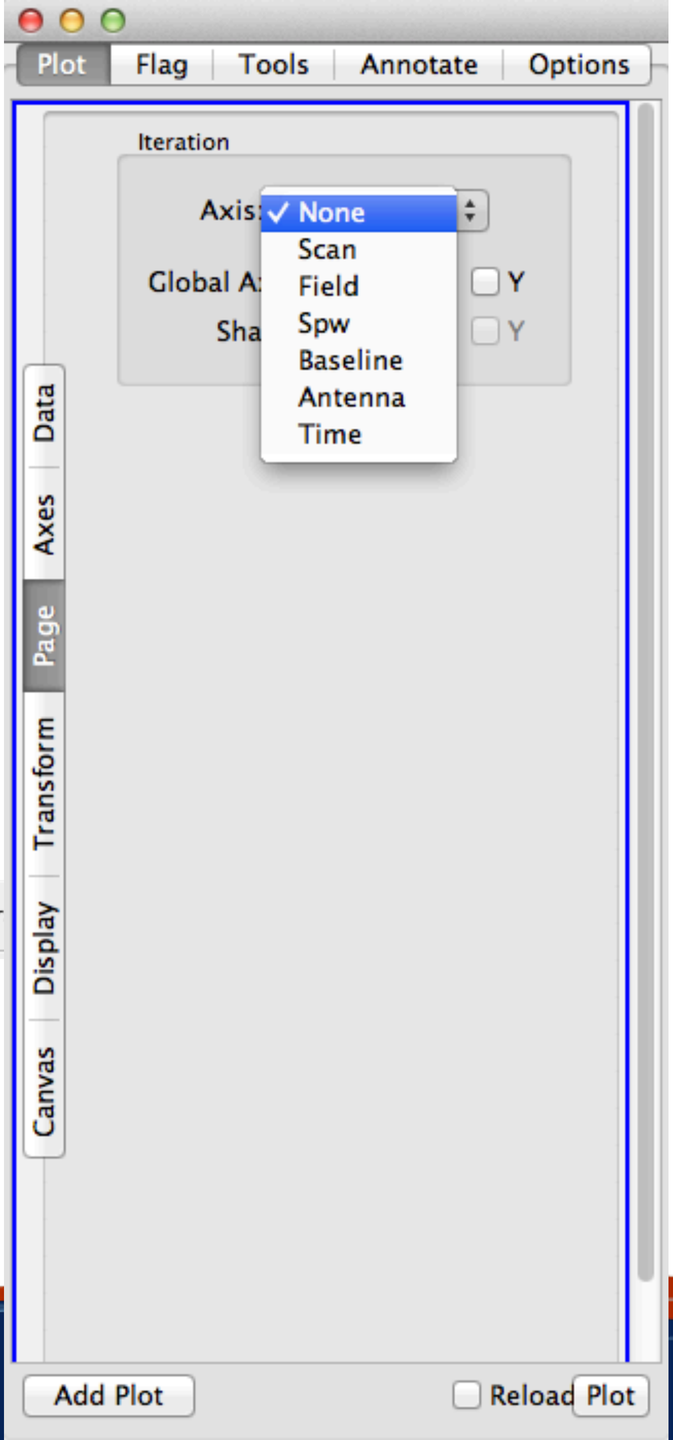
Data Review: *plotms*

Page: to iterate on

Scan
Field
Spw
Baseline
Antenna
Time



Tool panel

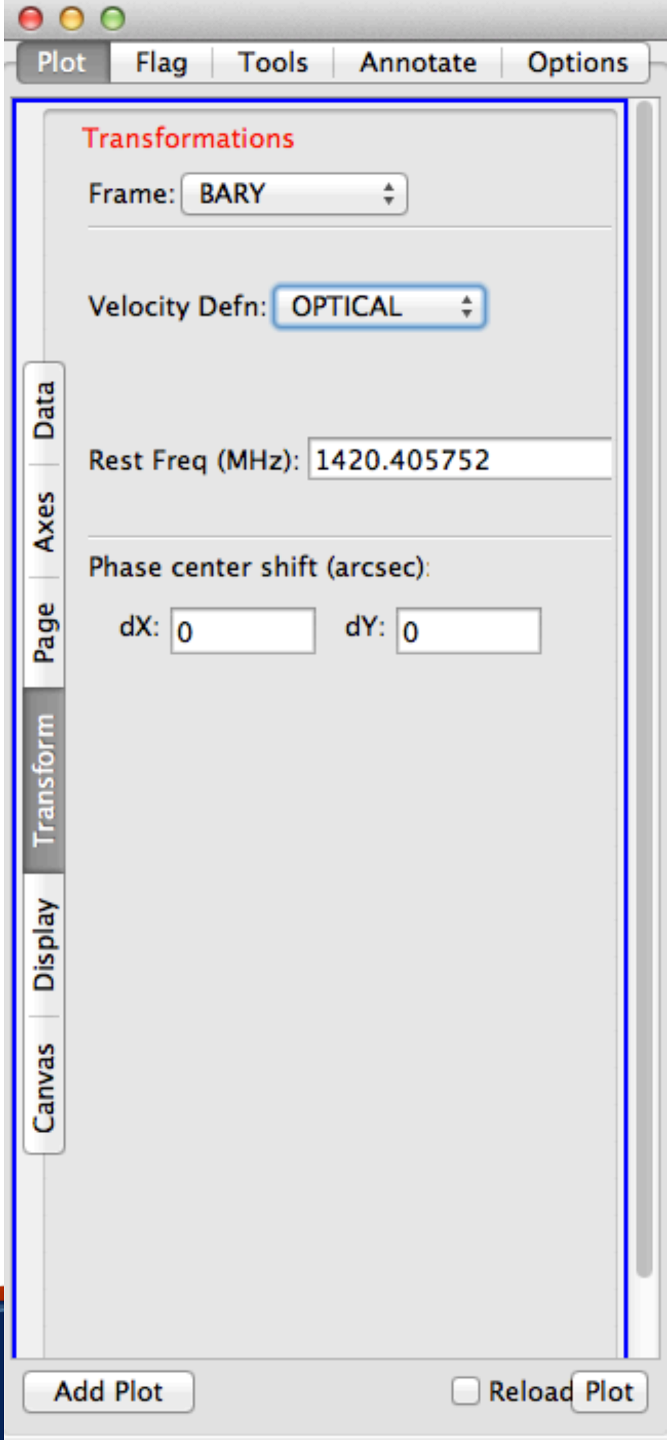


Axis: ✓ None

Data Review: *plotms*

Transformations

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



Transformations

Frame: BARY

Velocity Defn: OPTICAL

Rest Freq (MHz): 1420.405752

Phase center shift (arcsec):

dX: 0 dY: 0

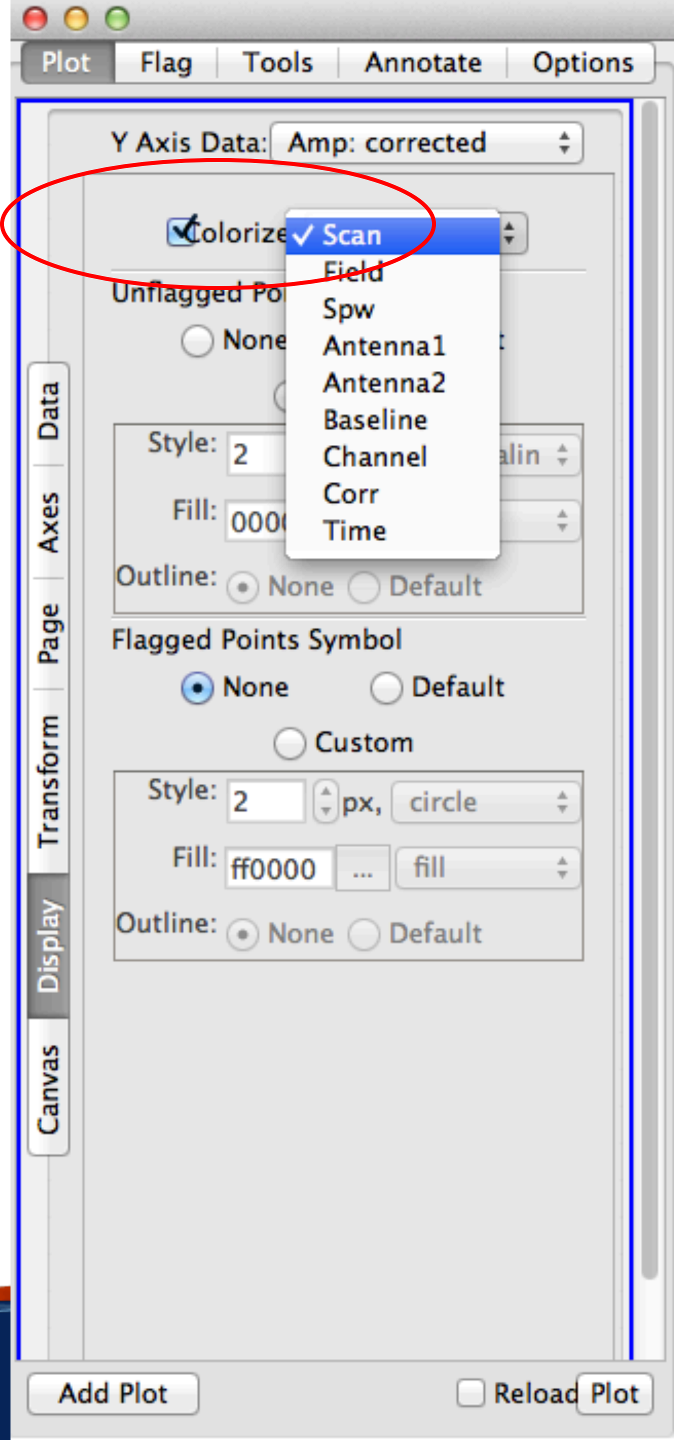
□ Reload

Data Review: *plotms*

Display

Colorize by:

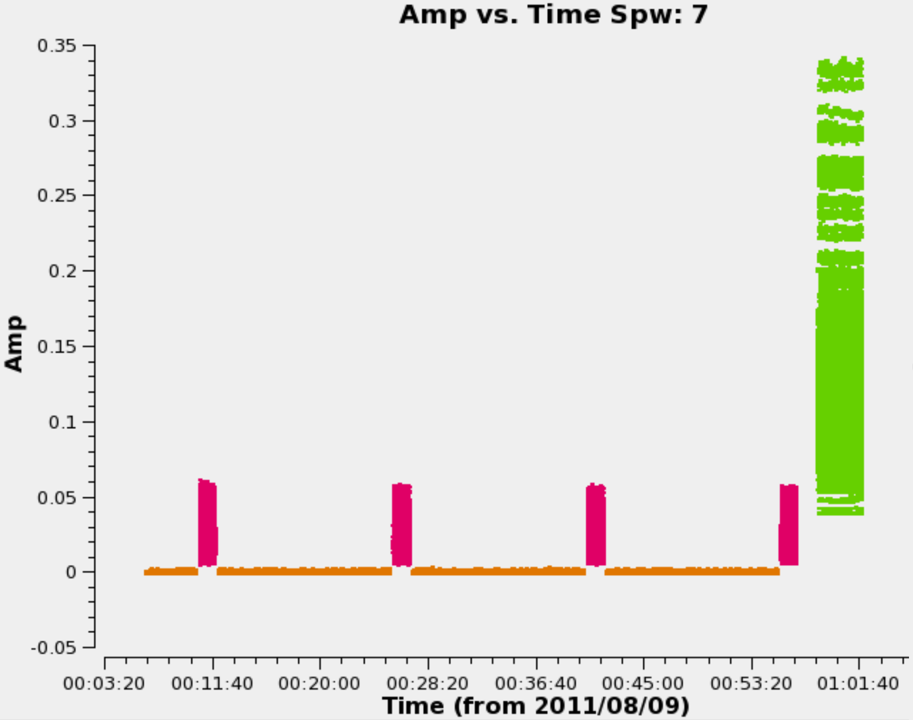
- Scan
- Field
- Spw
- Antenna1
- Antenna2
- Baseline
- Channel
- Correlation
- Time

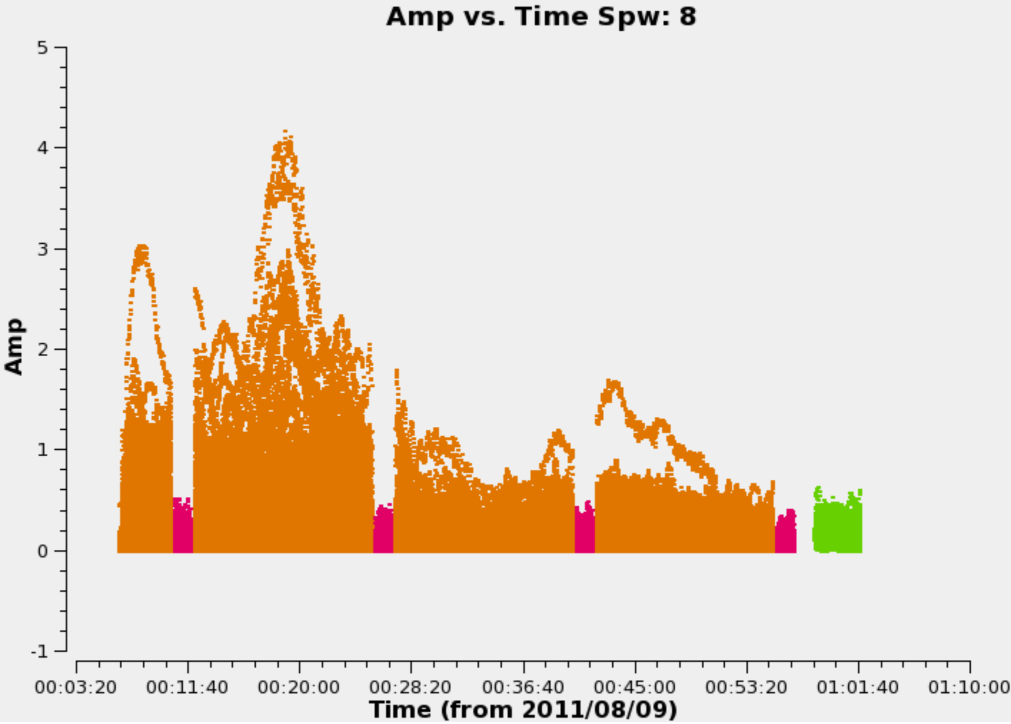


Data Review: *plotms*

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

iter: spw (with all channels averaged)

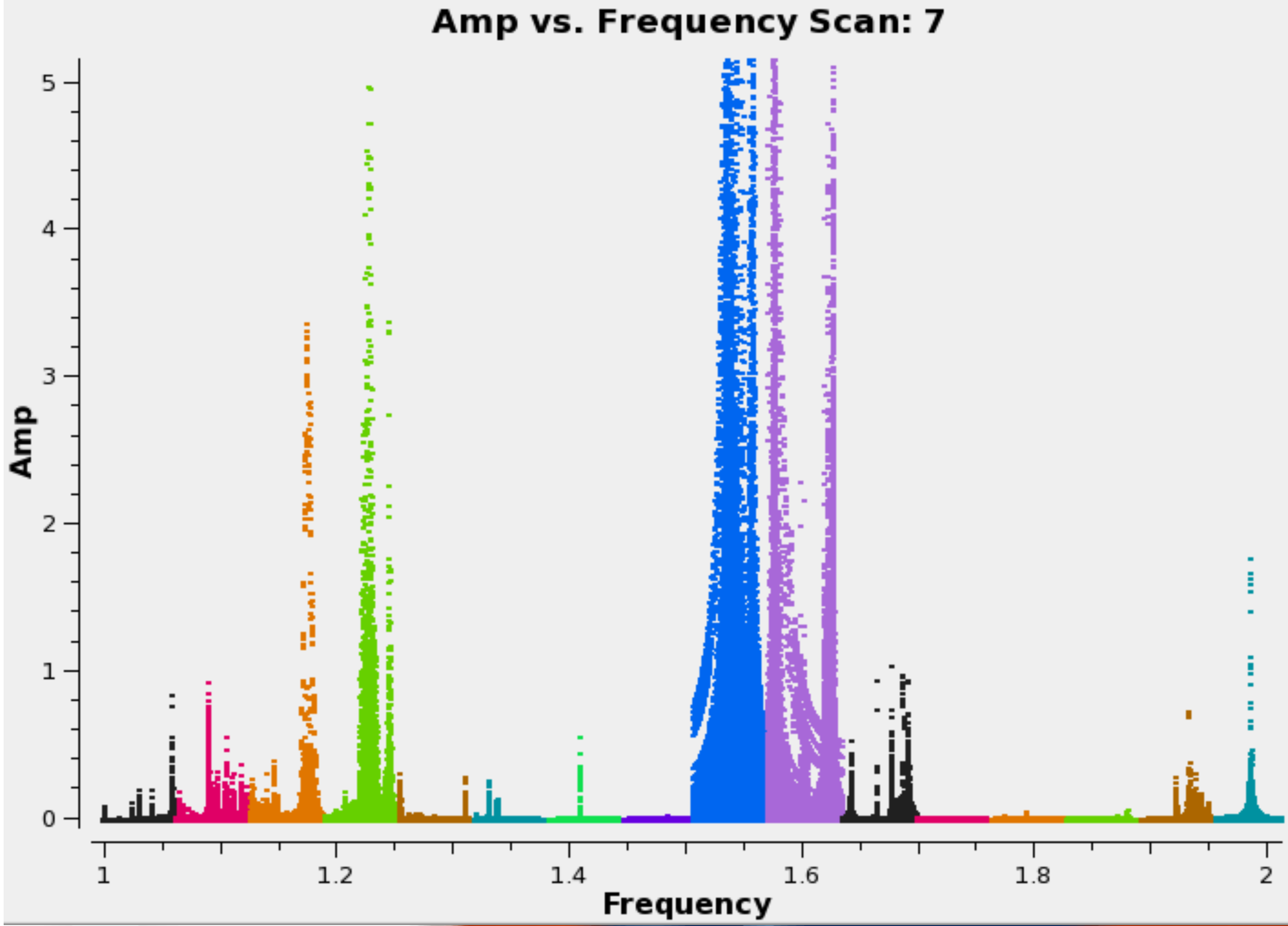




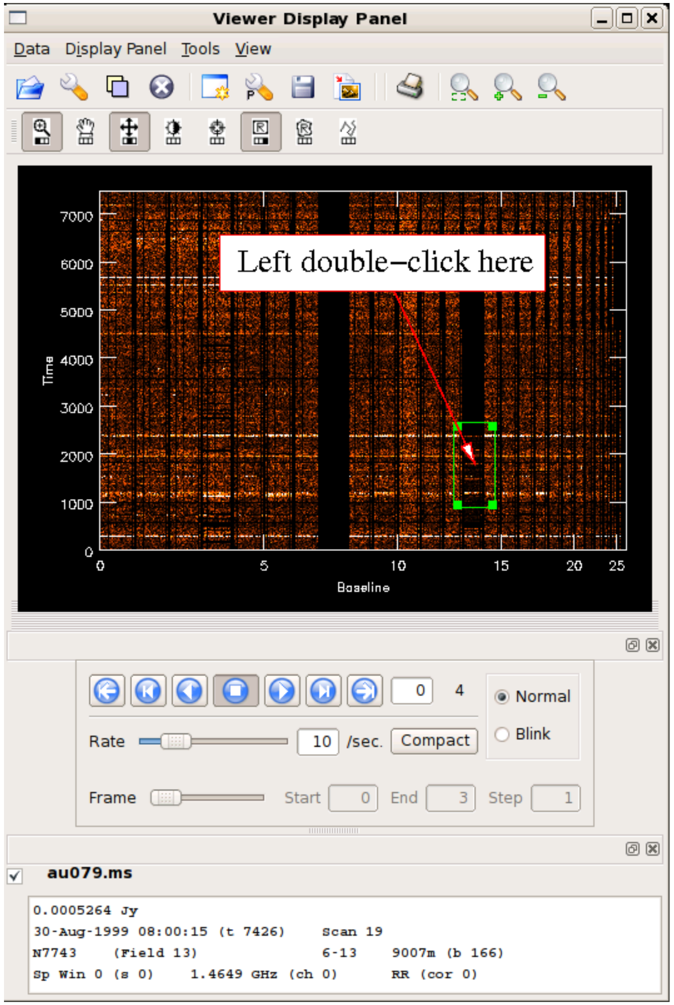
Data Review: *plotms*

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

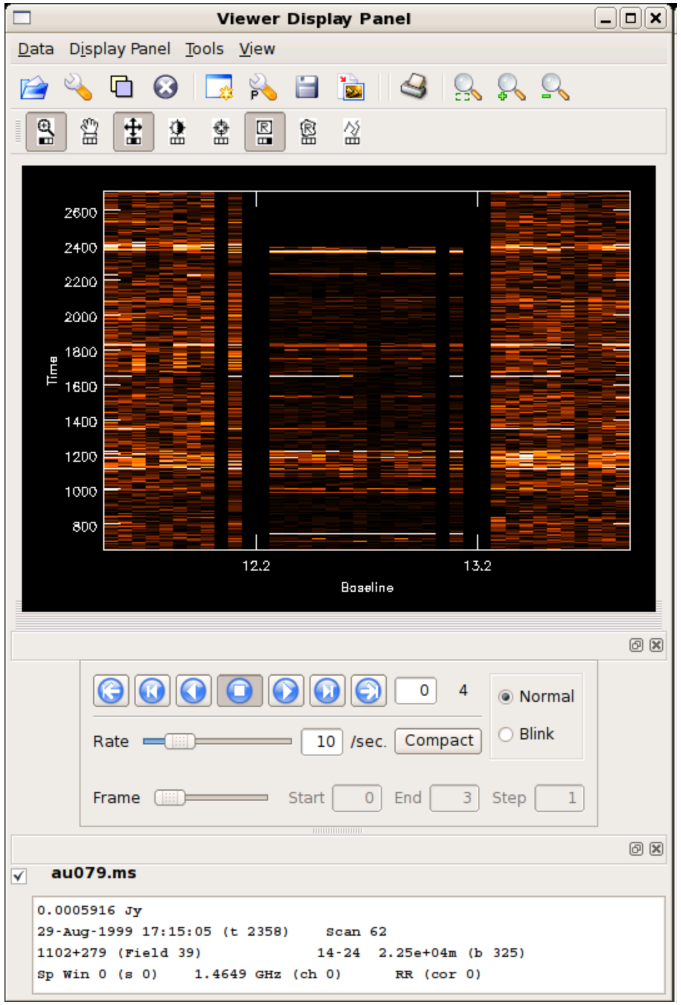
iteration: scan



Data review: *msview*

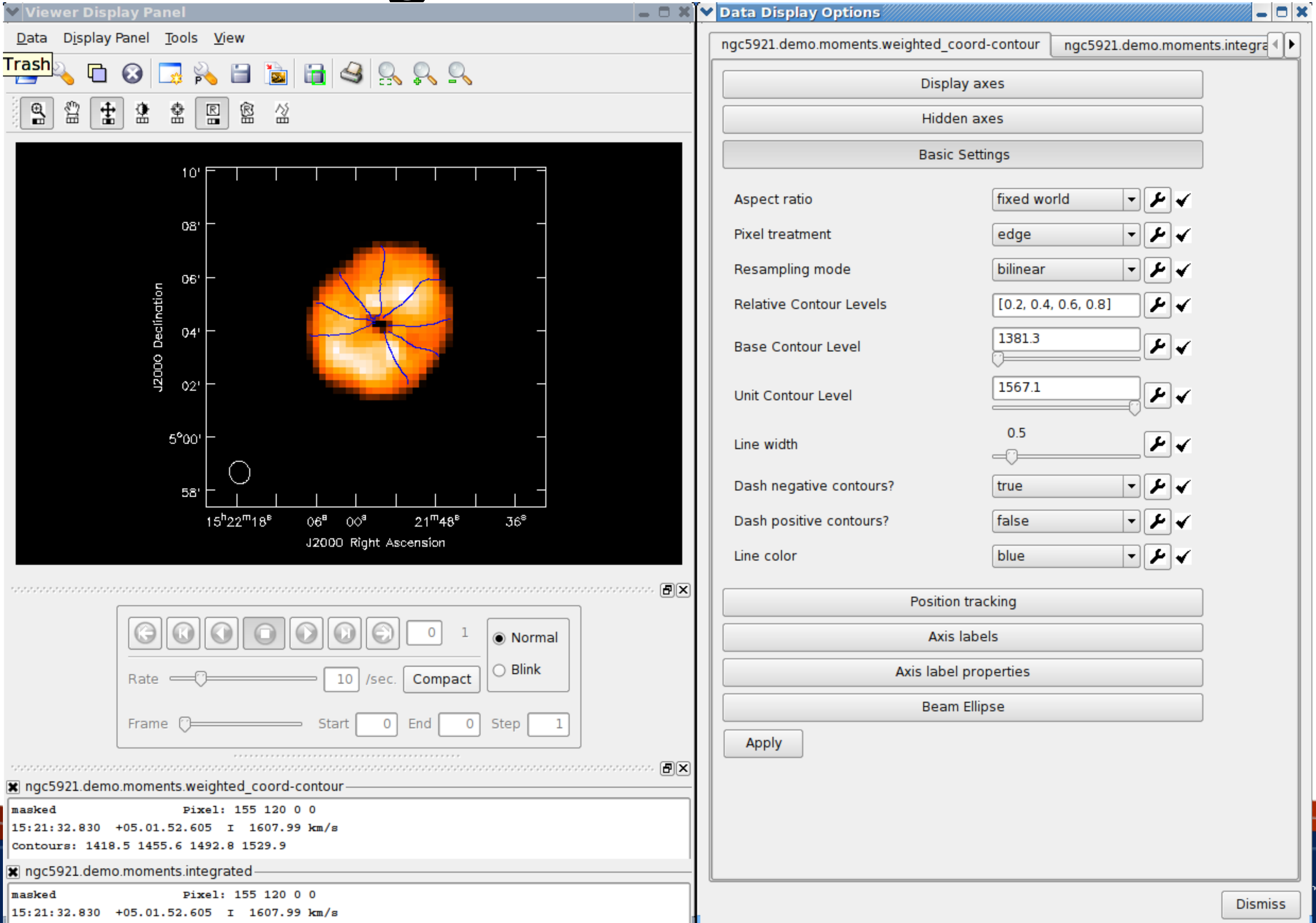


```
0.0005264 Jy
30-Aug-1999 08:00:15 (t 7426)    Scan 19
N7743    (Field 13)              6-13    9007m (b 166)
Sp Win 0 (s 0)    1.4649 GHz (ch 0)      RR (cor 0)
```



```
0.0005916 Jy
29-Aug-1999 17:15:05 (t 2358)    Scan 62
1102+279 (Field 39)              14-24  2.25e+04m (b 325)
Sp Win 0 (s 0)    1.4649 GHz (ch 0)      RR (cor 0)
```

Image Viewer: *viewer*



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

Rate: 10 /sec. Frame: 0 of 1. Start: 0, End: 0, Step: 1. ● Normal ○ Blink

☒ ngc5921.demo.moments.weighted_coord-contour

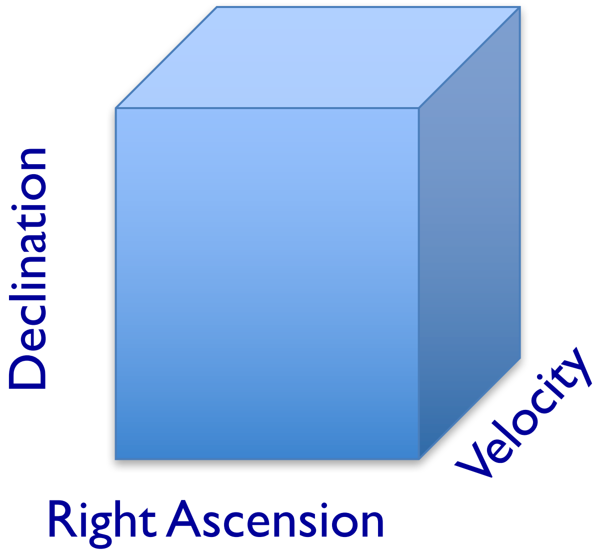
```
masked               Pixel: 155 120 0 0
15:21:32.830  +05.01.52.605  I  1607.99 km/s
Contours: 1418.5 1455.6 1492.8 1529.9
```

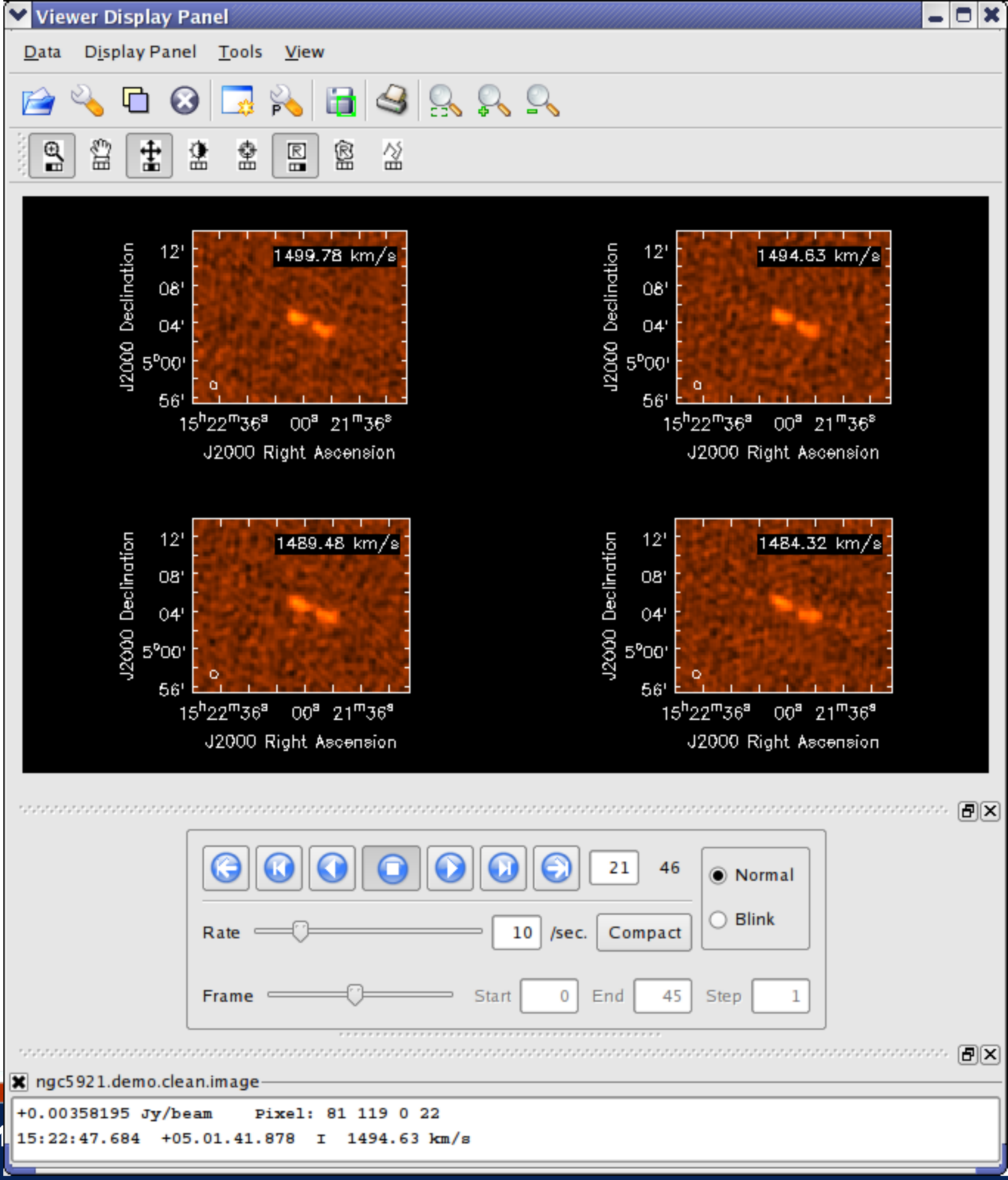
☒ ngc5921.demo.moments.integrated

```
masked               Pixel: 155 120 0 0
15:21:32.830  +05.01.52.605  I  1607.99 km/s
```

Image Viewer

- Displaying cubes
- Movies
- Channel maps





Rate 10 /sec.

21 46

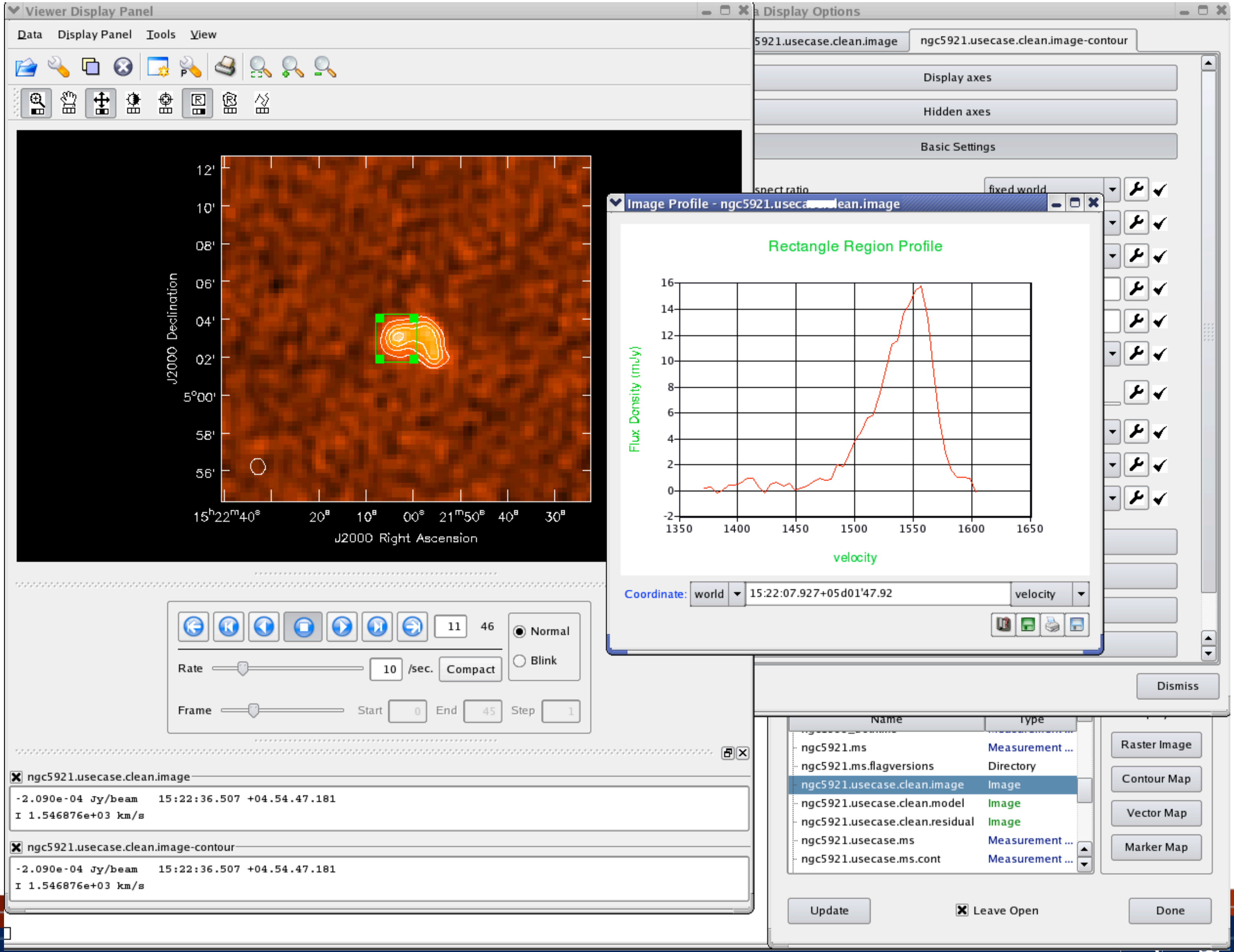
● Normal
○ Blink

Frame Start 0 End 45 Step 1

ngc5921.demo.clean.image

```
+0.00358195 Jy/beam    Pixel: 81 119 0 22
15:22:47.684  +05.01.41.878  I  1494.63 km/s
```

M



Coordinate: world 15:22:07.927+05d01'47.92 velocity

☒ ngc5921.usecase.clean.image

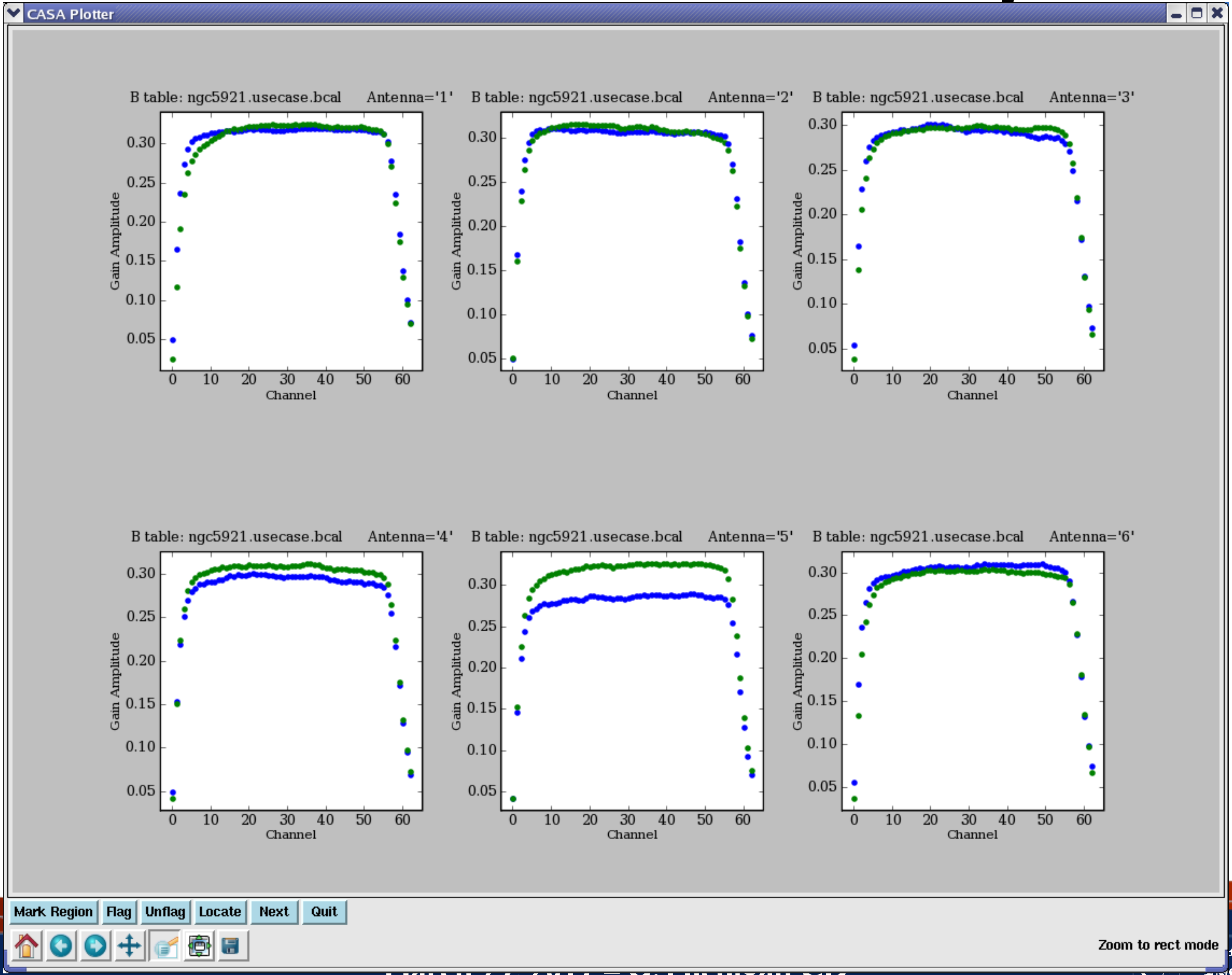
```
-2.090e-04 Jy/beam   15:22:36.507 +04.54.47.181
I 1.546876e+03 km/s
```

☒ ngc5921.usecase.clean.image-contour

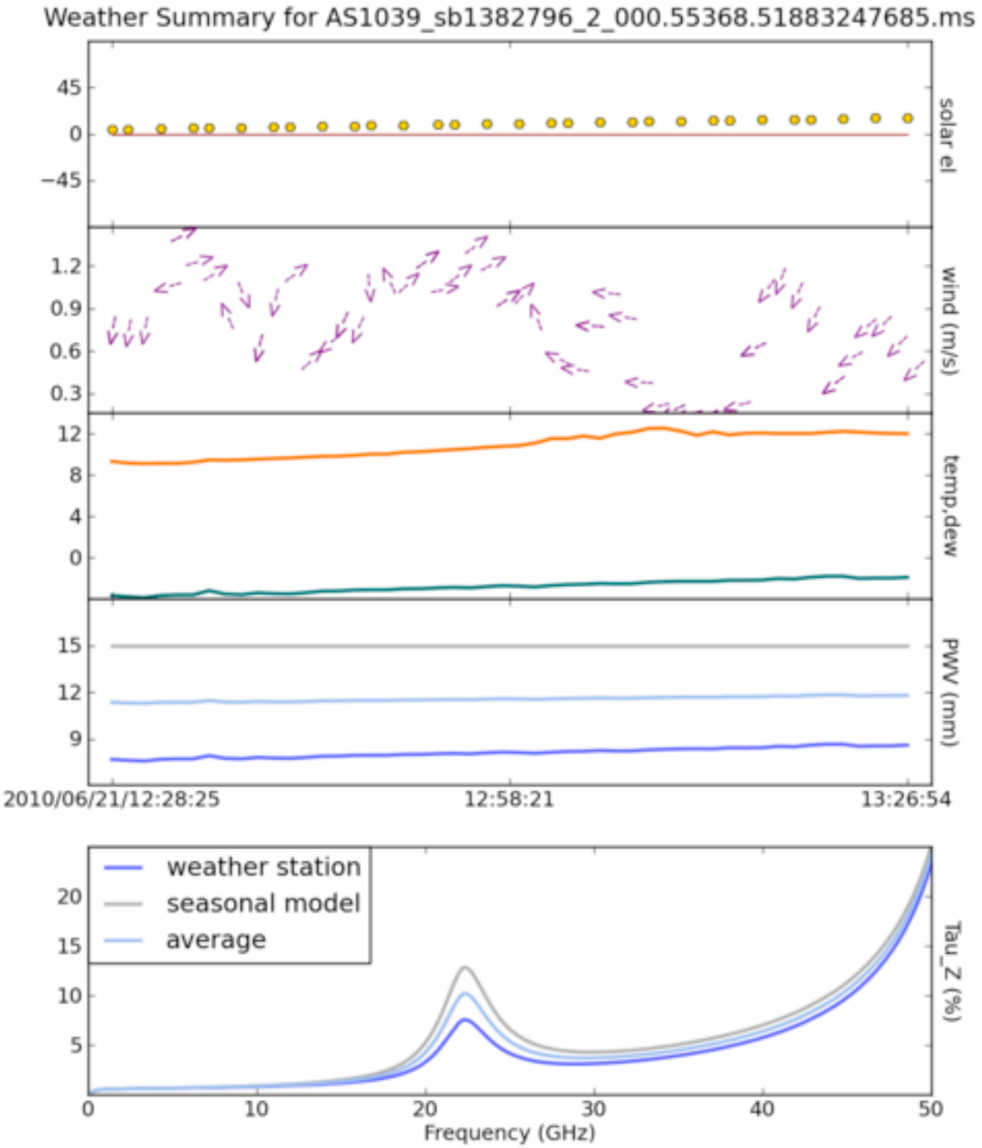
```
-2.090e-04 Jy/beam   15:22:36.507 +04.54.47.181
I 1.546876e+03 km/s
```

Name	Type
ngc5921.ms	Measurement ...
ngc5921.ms.flagversions	Directory
ngc5921.usecase.clean.image	Image
ngc5921.usecase.clean.model	Image
ngc5921.usecase.clean.residual	Image
ngc5921.usecase.ms	Measurement ...
ngc5921.usecase.ms.cont	Measurement ...

Review calibration tables: *plotcal*


CASA Plotter
B table: ngc5921.usecase.bcal Antenna='1'
B table: ngc5921.usecase.bcal Antenna='2'
B table: ngc5921.usecase.bcal Antenna='3'
B table: ngc5921.usecase.bcal Antenna='4'
B table: ngc5921.usecase.bcal Antenna='5'
B table: ngc5921.usecase.bcal Antenna='6'
Gain Amplitude
Channel
0.05
0.10
0.15
0.20
0.25
0.30
0
10
20
30
40
50
60
Mark Region
Flag
Unflag
Locate
Next
Quit
Zoom to rect mode

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

NRAO

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.