

# NRAO



## **CASA Intro**

Juergen Ott (NRAO)

## **National Radio Astronomy Observatory**



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



# NRAO



## CASA Intro

Juergen Ott (NRAO)

## National Radio Astronomy Observatory



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



# Introduction to CASA

Juergen Ott (CASA project scientist)  
Crystal Brogan (CASA ALMA subsystem scientist)  
Steven Myers (CASA EVLA subsystem scientist)  
Jeff Kern (CASA manager)





# CASA (Common Astronomy Software Applications)

- CASA is the offline data reduction package for ALMA and the EVLA (data from other telescopes usually work, too, but not primary goal of CASA)
- C++ bound to Python (plus some Qt or other apps)
- Import/export data, edit, calibrate, image, 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
- CASA has some of the most sophisticated algorithms implemented (multi-scale clean, Taylor term expansion for wide bandwidths, VV-term projection, OTF mosaicing, etc.)
- We have a active Algorithm Research Group, so more goodness to come

# Outline

- CASA startup
- CASA basic python interface
- Tasks and tools
- The Measurement Set
- Data selection syntax
- Visualization tools
- Data analysis
- User support/Documentation

# CASA (Common Astronomy Software Applications)

 Current version: 3.3.0 (release r16856 built 2 Nov 2011)

New releases about every 6 months (May and November).

**For download: [casa.nrao.edu](http://casa.nrao.edu) Linux, Mac OS X**

In addition to the full release, we regularly create “stable” versions of CASA. They are markers on the way to the next release with more functionality but likely contain unfinished developments, less tested code, and no up-to-date documentation. Download if you are brave enough.

# CASA Startup

**\$ casapy**

CASA Version 3.2.1 (r15198)

Compiled on: Fri 2011/05/27 02:52:18 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

Single Dish sd\* tasks are available after asap\_init() is run

---

Activating auto-logging. Current session state plus future input saved.

Filename     : ipython.log

Mode        : backup

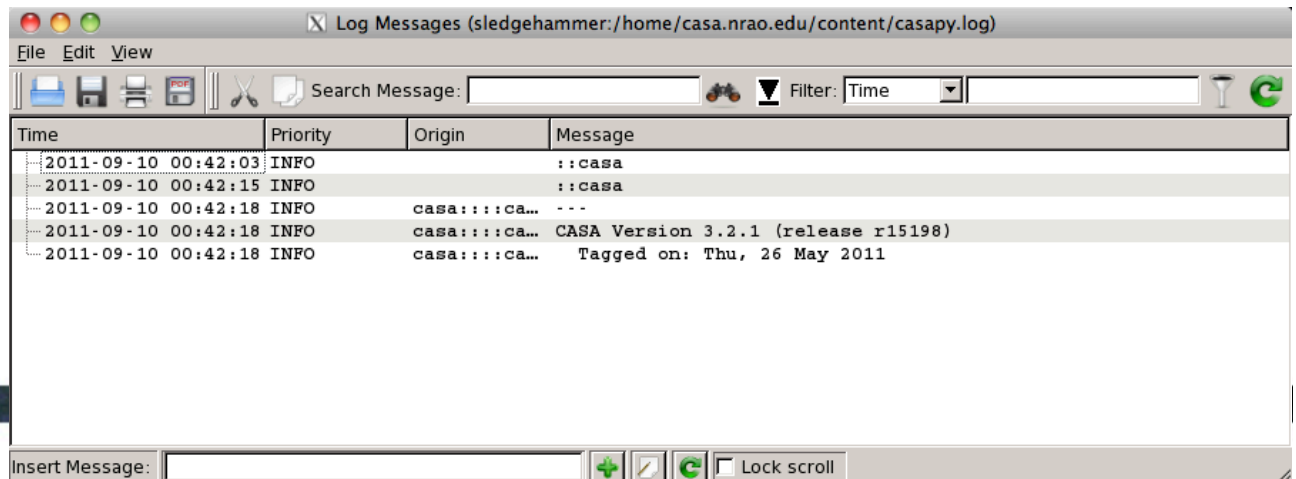
Output logging : False

Raw input log : False

Timestamping : False

State       : active

CASA <2>:



Time	Priority	Origin	Message
2011-09-10 00:42:03	INFO	::casa	::casa
2011-09-10 00:42:15	INFO	::casa	::casa
2011-09-10 00:42:18	INFO	casa:::ca...	---
2011-09-10 00:42:18	INFO	casa:::ca...	CASA Version 3.2.1 (release r15198)
2011-09-10 00:42:18	INFO	casa:::ca...	Tagged on: Thu, 26 May 2011



# CASA Interactive Interface

- IPython ([ipython.org](http://ipython.org))
- Features:
  - shell access
  - auto-parenthesis (autocall)
  - command history
  - session logging
    - [ipython.log](#) – ipython command history
    - [casapy.log](#) – casa logger messages
  - numbered input/output
  - history/searching



# Basic Python tips

- to run a .py script:

`execfile('<scriptname>')`

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

- 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 tutorial
- **Tools** - complete functionality
  - `tool.method()` calls, used by tasks
  - sometimes shown in tutorial scripts
- **Applications** – some tasks/tools invoke standalone apps
  - e.g. `casaviewer`, `casaplotms`, `casabrowser`, `asdm2MS`
- Shell commands can be run with a leading exclamation mark    `!du -hs`

# Key Tasks

To see list of tasks organized by type:

**tasklist**

```

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
-----
imcontsub          simdata        browsetable       (after running asap_init())
imhead            (simdata2)    casalogger
imfit             clearplot
immath            clearstat
immoments         csvclean
imregrid          filecatalog
imsmooth          find
imstat            help par.parameter
imval             help task
(specfit)         rmtables
                  startup
                  taskhelp
                  tasklist
                  toolhelp

sdaverage
sdbaseline
sdcal
sdcoadd
sdffit
sdflag
sdimaging
sdimprocess
sdlist
sdmath
sdplot
sdsave
sdscale
sdsmooth
sdstat
sdtpimaging
(sdsim)
(msmoments)

User defined tasks
-----

CASA <3>:

```

# Key Tasks

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 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:
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 combination of se
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:
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 that axis.
imcontsub      : Subtracts specified continuum channels from a spectral line data set
imfit          : Fit one or more elliptical Gaussian components on an image region(s)
imhead         : List, get and put image header parameters
immath         : Perform math operations on images
immoments      : Compute moments from an image

Default      Default      Default      Default      Default
```

# Task Interface

examine task parameters with inp :

```

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>:

```



# Task Interface

- standard tasking interface
- parameter manipulation commands
  - `inp`, `default`, `saveinputs`, `tget`, `tput`
- use parameters set as global Python variables

`<param> = <value>`

(e.g. `vis = 'ngc592l.demo.ms'` )

- execute
  - `<taskname>` or `go` ( e.g. `clean()` )
- return values (except when using “go”)
  - some tasks return Python dictionaries, e.g.  
`myval=imval()`

# 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 (globals not used)
  - use standard tasking interface  
use global variables for task parameters
  - see Chapter 1.3 in Cookbook

# 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.
phasecenter        = 'J2000 13h29m52.2s +47d12m30s' # 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       = 'briggs'               # Weighting of uv (natural, uniform, briggs, ...)
```

# Expandable Parameters

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

```
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-las-no' # re-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 = '' # Number of Taylor coefficients to
# model the sky frequency dependence
niter = 1000 # Reference frequency (nterms > 1), ''
# uses central data-frequency
gain = 0.2 # Gridding kernel for FFT-based
# transforms, default='' None
threshold = '12uJy' # Maximum number of iterations
# Loop gain for cleaning
psfmode = 'clark' # Flux level to stop cleaning, must
# include units: '1.0mJy'
imagermode = 'csclean' # Method of PSF calculation to use
# during minor cycles
# Options: 'csclean' or 'mosaic', '',
# uses psfmode
cyclefactor = 1.5 # Controls how often major cycles are
# done (e.g. 5 for frequently)
cyclespeedup = -1 #
#
multiscale = [0, 2, 5, 8, 15, 50]
negcomponent = -1 #
smallscalebias = 0.6 #
#
interactive = False
mask = [] #
imsize = 1280 #
cell = '1arcsec' #
phasecenter = 'J2000 13h29m52.2s' #
restfreq = '' #
stokes = 'I' #
weighting = 'briggs'
```

```
CASA <4>: inp
-----> inp()
# clean :: Invert and deconvolve images with selected algorithm
vis = 'm51-center-contall.ms' # Name of input visibility file
imagename = 'M51-cont-rob-las-noninteractive-mult-phasecenter-nterm2' #
# re-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 = 2 # 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 = 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
# Options: 'csclean' or 'mosaic', '',
# uses psfmode
imagermode = 'csclean' # Controls how often major cycles are
# done (e.g. 5 for frequently)
cyclefactor = 1.5
```

# Expandable Parameters

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

The image shows three overlapping screenshots of the CASA (Common Astronomy Software Applications) command-line interface, demonstrating how boldface parameters expand into their subparameters.

**Left Screenshot (CASA <19>):** Shows the initial state where boldface parameters like **selectdata**, **inagermode**, and **multiscale** are set, but their subparameters are not yet expanded.

```
CASA <19>: inp
-----> inp()
# clean :: Invert and deconvolve images with
vis = 'm51-center-contall.ms'
imagermode = 'M51-cont-rob-1as-nonin
outlierfile = ''
field = ''
spw = ''
selectdata = False
mode = ''
gridmode = ''
niter = 1000
gain = 0.2
threshold = '12uJy'
psfmode = 'clark'
inagermode = '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'
phasecenter = 'J2000 13h29m52.2s +
restfreq = ''
stokes = 'I'
weighting = 'briggs'
```

**Middle Screenshot (CASA <4>):** Shows the expansion of **selectdata** and **inagermode** into their subparameters.

```
CASA <4>: inp
-----> inp()
# clean :: Invert and deconvolve images with
vis = 'm51-center-contall.ms'
imagermode = 'M51-cont-rob-1as-nonin
outlierfile = ''
field = ''
spw = ''
selectdata = False
mode = 'mfs'
  nterms = 2
  reffreq = ''
gridmode = ''
niter = 1000
gain = 0.2
threshold = '12uJy'
psfmode = 'clark'
inagermode = 'csclean'
  cyclefactor = 1.5
  cyclespeedup = -1
```

**Right Screenshot:** Shows the expansion of **multiscale** into its subparameters.

```
-----> inp()
# clean :: Invert and deconvolve images with
vis = 'm51-center-contall.ms'
imagermode = 'M51-cont-rob-1as-nonin
outlierfile = ''
field = ''
spw = ''
selectdata = False
mode = 'velocity'
  nchan = -1
  start = 0
  width = 1
  interpolation = 'linear'
  chaniter = False
  outframe = ''
  veltype = 'radio'
gridmode = ''
niter = 1000
gain = 0.2
threshold = '12uJy'
psfmode = 'clark'
inagermode = 'csclean'
  cyclefactor = 1.5
  cyclespeedup = -1
multiscale = [0, 2, 5, 8, 15, 50, 10]
  negcomponent = -1
  smallscalebias = 0.6
```



# Parameter Checking

sanity checks of parameters in `inp` :

```
CASA <17>: inp
--> inp()
# clean :: Invert and deconvolve images with selected algorithm
vis          = 'm51-center-contall.ms' # Name of input visibility file
imagename    = 'M51-cont-rob-las-noninteractive-multiscale' # Pre-name of output images
outlierfile  = '' # Text file with image names for outliers
field        = '' # Field Name or id
spw          = '' # Spectral windows e.g. '0~3', '' is all
selectdata   = False # Select 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

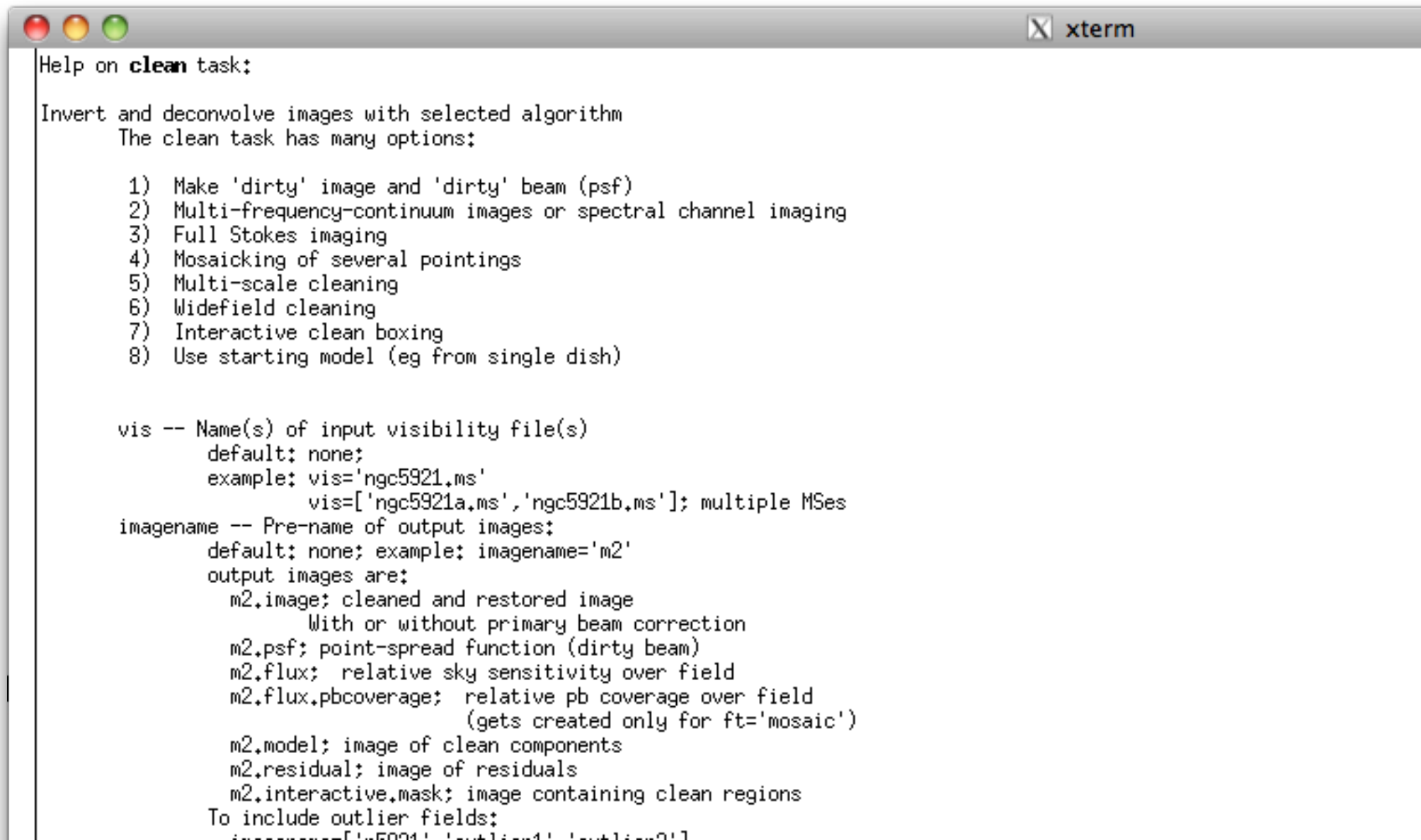
interactive  = False # Use interactive clean (with GUI viewer)
mask         = [] # Cleanbox(es), mask image(s), region(s), or a level
imsize       = 1280 # x and y image size in pixels. Single value: same for both
cell         = '1arcsec' # x and y cell size(s). Default unit arcsec.
phasecenter  = 'J2000 13h29m52.2s +47d12m30s' # Image center: direction or field index
restfreq     = '' # Rest frequency to assign to image (see help)
stokes       = 'I' # Stokes params to image (eg I,IV,IQ,IQUV)
weighting    = 'briggs' # Weighting of uv (natural, uniform, briggs, ...)
  robust      = 0.5 # Briggs robustness parameter
```

erroneous values in red

# Help on Tasks

 In-line help:

>help clean OR >pdoc clean



```
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
image -- Pre-name of output images:
        default: none; example: image='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
          m2.residual; image of residuals
          m2.interactive.mask; image containing clean regions
To include outlier fields:
image=[ 'ngc5921', 'outlier1', 'outlier2']
```

# Tools in CASA

💡 What if there's no task?

→ use CASA tools! (tasks are built upon tools)

💡 CASA Toolkit underneath tasks

💡 core AIPS++ code (mostly in C++)

💡 tools are functions.methods

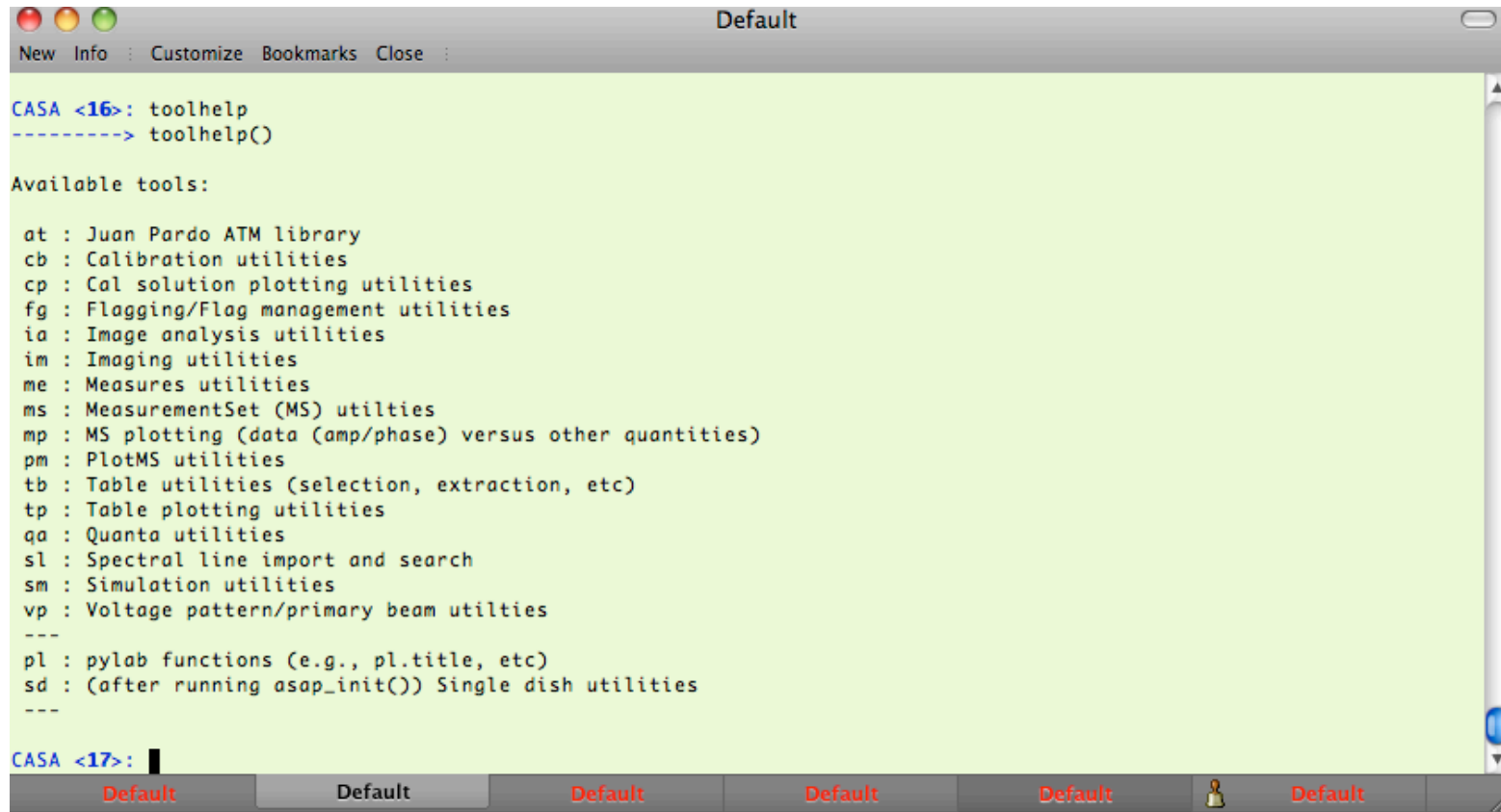
💡 call from casapy as `<tool>.<method>()`

💡 default tool objects are pre-constructed

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

# CASA Tool List

 list of default tools from [toolhelp](#) :



```
CASA <16>: toolhelp
-----> toolhelp()

Available tools:

at : Juan Pardo ATM library
cb : Calibration utilities
cp : Cal solution plotting utilities
fg : Flagging/Flag management utilities
ia : Image analysis utilities
im : Imaging utilities
me : Measures utilities
ms : MeasurementSet (MS) utilities
mp : MS plotting (data (amp/phase) versus other quantities)
pm : PlotMS 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 utilities
---
pl : pylab functions (e.g., pl.title, etc)
sd : (after running asap_init()) Single dish utilities
---
```

CASA <17>:

# CASA Tool List

 list of default tools from [toolhelp](#) :



```
Default
New Info : Customize Bookmarks Close :

CASA <16>: toolhelp
-----> toolhelp()

Available tools:

at : Juan Pardo ATM library
cb : Calibration utilities
cp : Cal solution plotting utilities
fg : Flagging/Flag management utilities
ia : Image analysis utilities
im : Imaging utilities
me : Meas
ms : Meas
mp : MS
pm : Plot
tb : Tabl
tp : Tabl
qa : Quanta utilities
sl : Spectral line import and search
sm : Simulation utilities
vp : Voltage pattern/primary beam utilities
---
pl : pylab functions (e.g., pl.title, etc)
sd : (after running asap_init()) Single dish utilities
---

CASA <17>:

Default Default Default Default Default Default
```



# CASA Tool List

- ☞ There's a good chance that your
- ☞ problem can be solved on the
- ☞ tool level, don't be afraid!
- ☞ ~1000 tool methods available!
- ☞ Tool methods described in the CASA
- ☞ Toolkit Reference:
  - ☞ <http://casa.nrao.edu/docs/CasaRef/CasaRef.html>

[What imager produces:](#)  
[What imager does not do:](#)  
[What improvement to imager are in the works:](#)  
[Advanced use of imager:](#)  
[Overview of imager tool functions:](#)

## 2.4.1 imager - Tool

[imager.imager - Function](#)  
[imager.advise - Function](#)  
[imager.approximatepsf - Function](#)  
[imager.boxmask - Function](#)  
[imager.calcuvw - Function](#)  
[imager.clean - Function](#)  
[imager.clipimage - Function](#)  
[imager.clipvis - Function](#)  
[imager.close - Function](#)  
[imager.defineimage - Function](#)  
[imager.done - Function](#)  
[imager.drawmask - Function](#)  
[imager.exprmask - Function](#)  
[imager.feather - Function](#)  
[imager.filter - Function](#)  
[imager.fitpsf - Function](#)  
[imager.fixvis - Function](#)  
[imager.ft - Function](#)  
[imager.linear mosaic - Function](#)  
[imager.make - Function](#)  
[imager.makeimage - Function](#)  
[imager.makemodelfromsd - Function](#)  
[imager.mask - Function](#)  
[imager.mem - Function](#)  
[imager.nnls - Function](#)  
[imager.open - Function](#)  
[imager.pb - Function](#)  
[imager.plotsummary - Function](#)  
[imager.plotuv - Function](#)  
[imager.plotvis - Function](#)  
[imager.plotweights - Function](#)  
[imager.regionmask - Function](#)  
[imager.regiontoimagemask - Function](#)  
[imager.residual - Function](#)  
[imager.restore - Function](#)  
[imager.sensitivity - Function](#)

# The Measurement Set

- ④ The MS is a directory on disk
  - ④ the MAIN table in `table.*` files
  - ④ also contains sub-tables
    - ④ e.g. FIELD, SOURCE, ANTENNA, etc.
  - ④ sub-tables are sub-directories
  - ④ to copy must `cp -rf` to get contents (tarball to transfer)
  - ④ Best to remove ms with `rmtables('filename')`
    - ④ Or `rm -rf`
- ④ **WARNING:** renaming a MS can break cal-table dependencies
  - ④ (cal-tables will be standalone in CASA 3.4)

# Example MS

🔗 Example: `ls ngc5921.usecase.ms`

```
smyers@colorin ~/CASA/Test $ ls ngc5921.usecase.ms
```

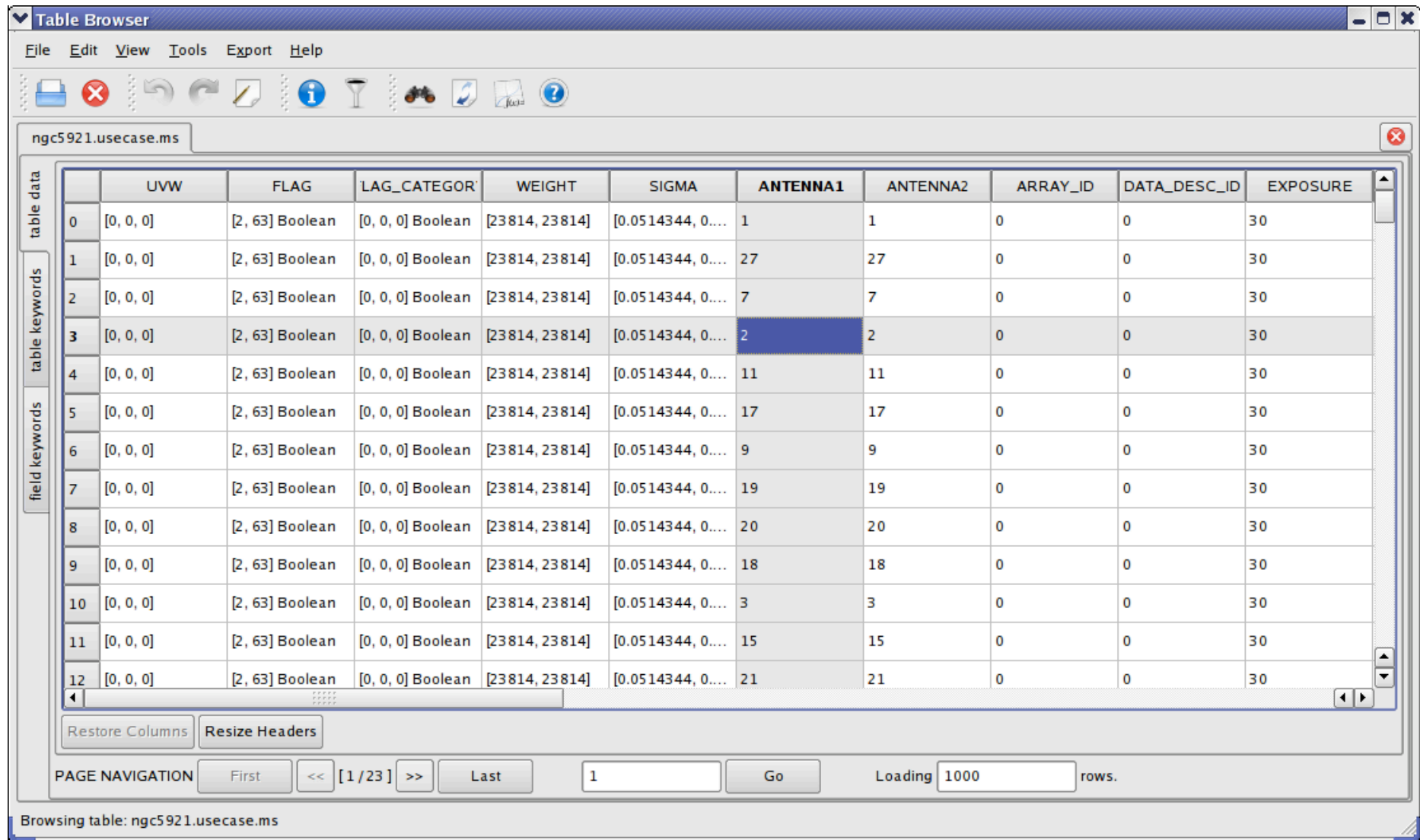
ANTENNA	POLARIZATION	table.f1	table.f3_TSM1	table.f8
DATA_DESCRIPTION	PROCESSOR	table.f10	table.f4	table.f8_TSM1
FEED	SORTED_TABLE	table.f10_TSM1	table.f5	table.f9
FIELD	SOURCE	table.f11	table.f5_TSM1	table.f9_TSM1
FLAG_CMD	SPECTRAL_WINDOW	table.f11_TSM1	table.f6	table.info
HISTORY	STATE	table.f2	table.f6_TSM0	table.lock
OBSERVATION	table.dat	table.f2_TSM1	table.f7	
POINTING	table.f0	table.f3	table.f7_TSM1	

```
smyers@colorin ~/CASA/Test $ ls ngc5921.usecase.ms/FIELD
```

```
table.dat    table.f0    _    table.f0i    table.info    table.lock
```

# MAIN Table Contents

Example using task browsetable: (application casabrowser)



The screenshot shows the 'Table Browser' application window. The title bar reads 'Table Browser'. The menu bar includes 'File', 'Edit', 'View', 'Tools', 'Export', and 'Help'. Below the menu bar is a toolbar with various icons. The main window displays a table titled 'ngc5921.usecase.ms'. The table has 11 columns: 'UVW', 'FLAG', 'LAG\_CATEGOR', 'WEIGHT', 'SIGMA', 'ANTENNA1', 'ANTENNA2', 'ARRAY\_ID', 'DATA\_DESC\_ID', and 'EXPOSURE'. The table contains 13 rows of data. The row with 'ANTENNA1' value 2 is highlighted. On the left side of the table, there are three vertical tabs: 'table data', 'table keywords', and 'field keywords', with 'table data' selected. At the bottom of the window, there is a 'PAGE NAVIGATION' section with buttons for 'First', '<<', '[ 1 / 23 ]', '>>', and 'Last'. There is also a 'Loading' section with a text box containing '1000' and the text 'rows.'. The status bar at the bottom left says 'Browsing table: ngc5921.usecase.ms'.

	UVW	FLAG	LAG_CATEGOR	WEIGHT	SIGMA	ANTENNA1	ANTENNA2	ARRAY_ID	DATA_DESC_ID	EXPOSURE
0	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	1	1	0	0	30
1	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	27	27	0	0	30
2	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	7	7	0	0	30
3	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	2	2	0	0	30
4	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	11	11	0	0	30
5	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	17	17	0	0	30
6	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	9	9	0	0	30
7	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	19	19	0	0	30
8	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	20	20	0	0	30
9	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	18	18	0	0	30
10	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	3	3	0	0	30
11	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	15	15	0	0	30
12	[0, 0, 0]	[2, 63] Boolean	[0, 0, 0] Boolean	[23814, 23814]	[0.0514344, 0....	21	21	0	0	30

# Data Selection Example

🕒 standard selection parameters

🕒 e.g. for task gaincal:

```
CASA <14>: inp
-----> inp()
# gaincal :: Determine temporal gains from calibrator observations:

vis                = 'ngc5921.ms'      # Name of input visibility file
caltable           = 'ngc5921.gcal'    # Name of output calibration table
field              = '0,1'             # field names or index of calibrators ''==>all
spw                = '0:2~56'         # spectral window:channels: ''==>all
selectdata        = True              # Other data selection parameters
  timerange         = ''               # time range: ''==>all
  uvrange           = ''               # uv range''=all
  antenna           = ''               # antenna/baselines: ''==>all
  scan              = ''               # scan numbers
  msselect          = ''               # Optional data selection (Specialized. but see help)
```



# Data Selection Syntax

- see Chapter 2.5 of Cookbook
  - 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
    - use '^' as separator of channels from step/width
    - example: spw = '0~2' ; spw = '1:10~30;50~65' ; spw = '2~5:5~54^5'

# Selection Syntax

- see Chapter 2.5 of Cookbook
  - antenna - string with antenna name or ID
    - first check for name, then ID (beware VLA name I-27, ID 0-26)
    - example: antenna = '1~5,11' ; antenna = 'ea\*', '!va'
    - Baselines: 'ea01&ea10'
  - timerange - string with date/time range
    - specify 'T0~T1' , missing parts of T1 default to T0, can give 'T0+dT'
    - example: timerange = '2007/10/16/01:00:00~06:30:00'

# Calibration

- Data structure: 3 columns (data + 2 scratch columns):
- **DATA** column (raw data)
- **MODEL** (Fourier transform of source model onto data)
- **CORRECTED\_DATA** (calibrated data)
- Columns created for calibration, this may take some time
- Sets of calibration tables applied **incrementally** (apply all previous calibration tables before solving/application)
- Applycal creates and overwrites **CORRECTED\_DATA** (can split to **DATA**)
- Refactoring underway to work without scratch columns

# Calibration continued

- Solvers (e.g. bandpass, gaincal, polcal, blcal)
- Based on data  $\times$  calibration - model
- Uses Hamaker-Bregman-Sault Measurement Equation formalism (using Jones and Mueller matrices)
- Generate calibration tables by type, e.g. bandpass (B), gain (G,T), delay (K), pol leakage (D), pol angle (X), place into equation
- Some types have channel dependencies (Df,Xf) or polynomial (BPOLY) or spline (GSPLINE) representations
- Currently, caltable headers point to MS for some information; so keep the MS and the caltable in the same directory (will change in CASA 3.4)

# Imaging

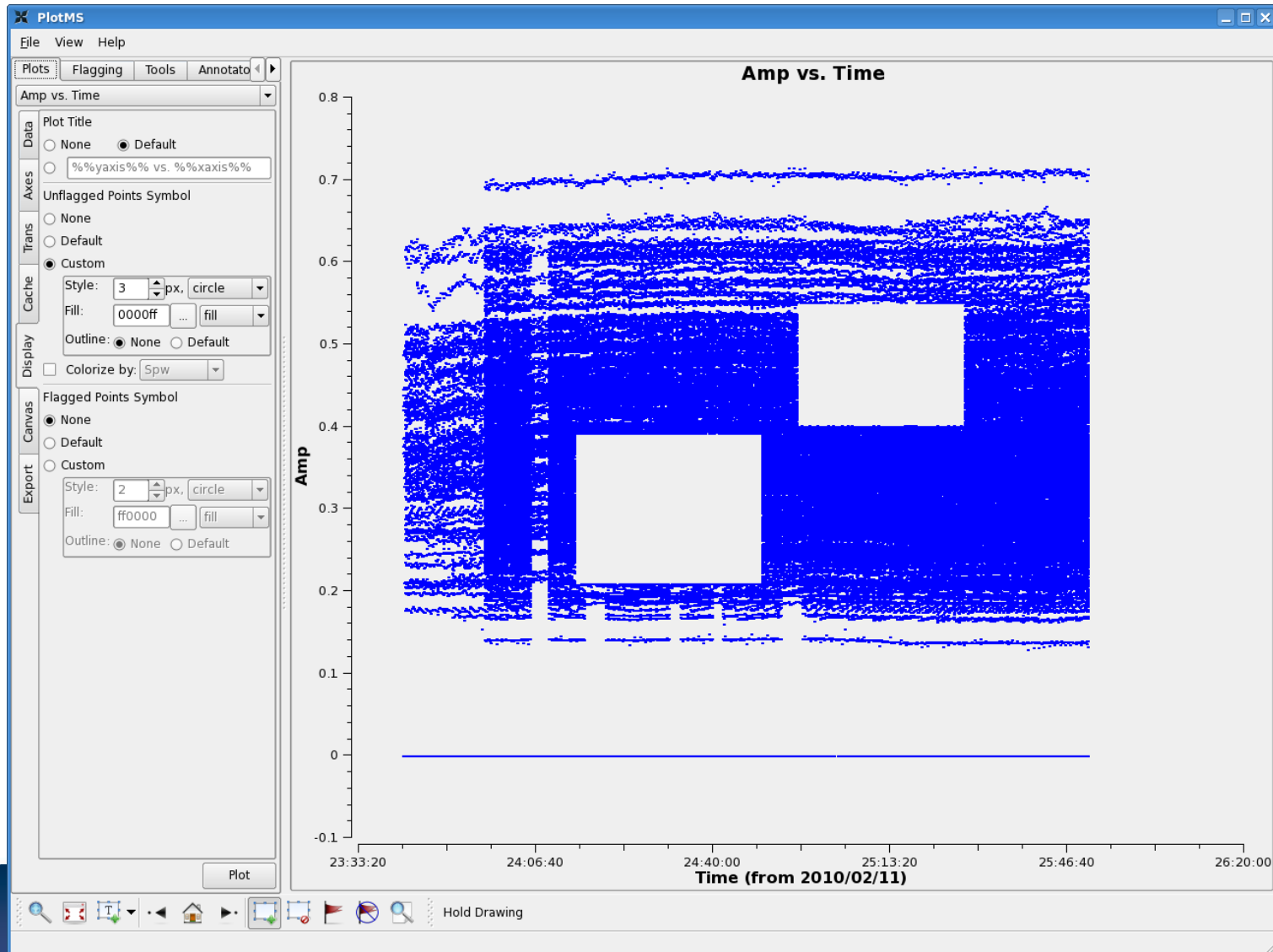
- FFT and deconvolution using **clean** task
- Grid data onto uv-plane, transform to residual image, find model components (minor cycles), transform back to data and subtract to form residual data (major cycles), repeat [Cotton-Schwab clean]
- Control of algorithms used (e.g. csclean, mosaic), automatic mapping to output cube planes (mfs, channel, velocity, frequency)
- Multi-frequency synthesis (mfs) for continuum, including higher order Taylor terms (intensity,  $\alpha$ ,...)
- Mosaicing using convolutional gridding to single uv-plane, plus uv-faceting

# Visualization Tools

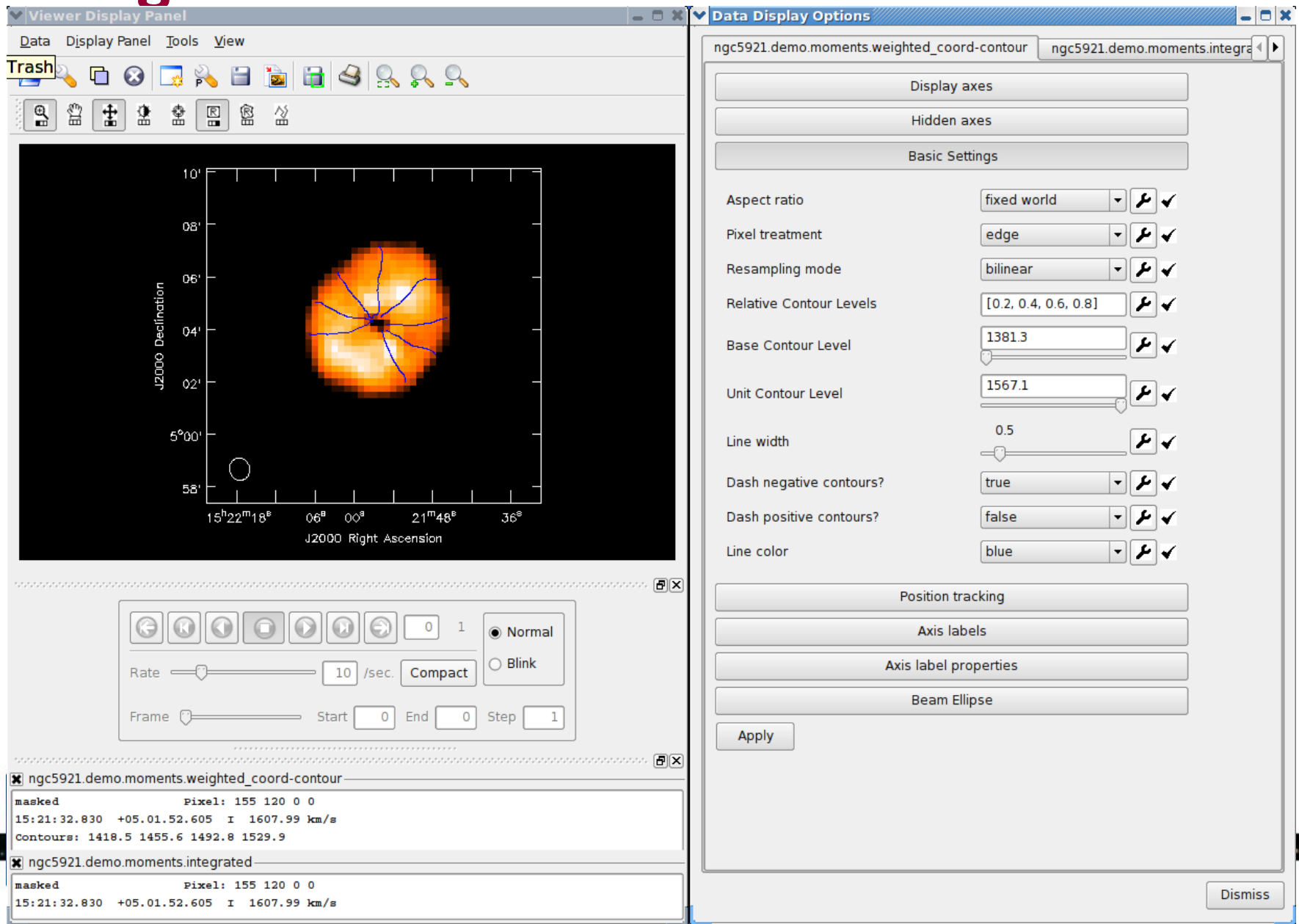
- Data needs to be displayed to understand it!
  - Can be a challenge for large datasets
- Visibilities: `plotms`, `msview`
- Images: `viewer`, `imview`
- Calibration tables: `plotcal` (soon `plotms`)
- Any table values: `browsetable`
- Single dish: `sdplot`
  
- Plot anything: use Python's `matplotlib`



# PlotMS

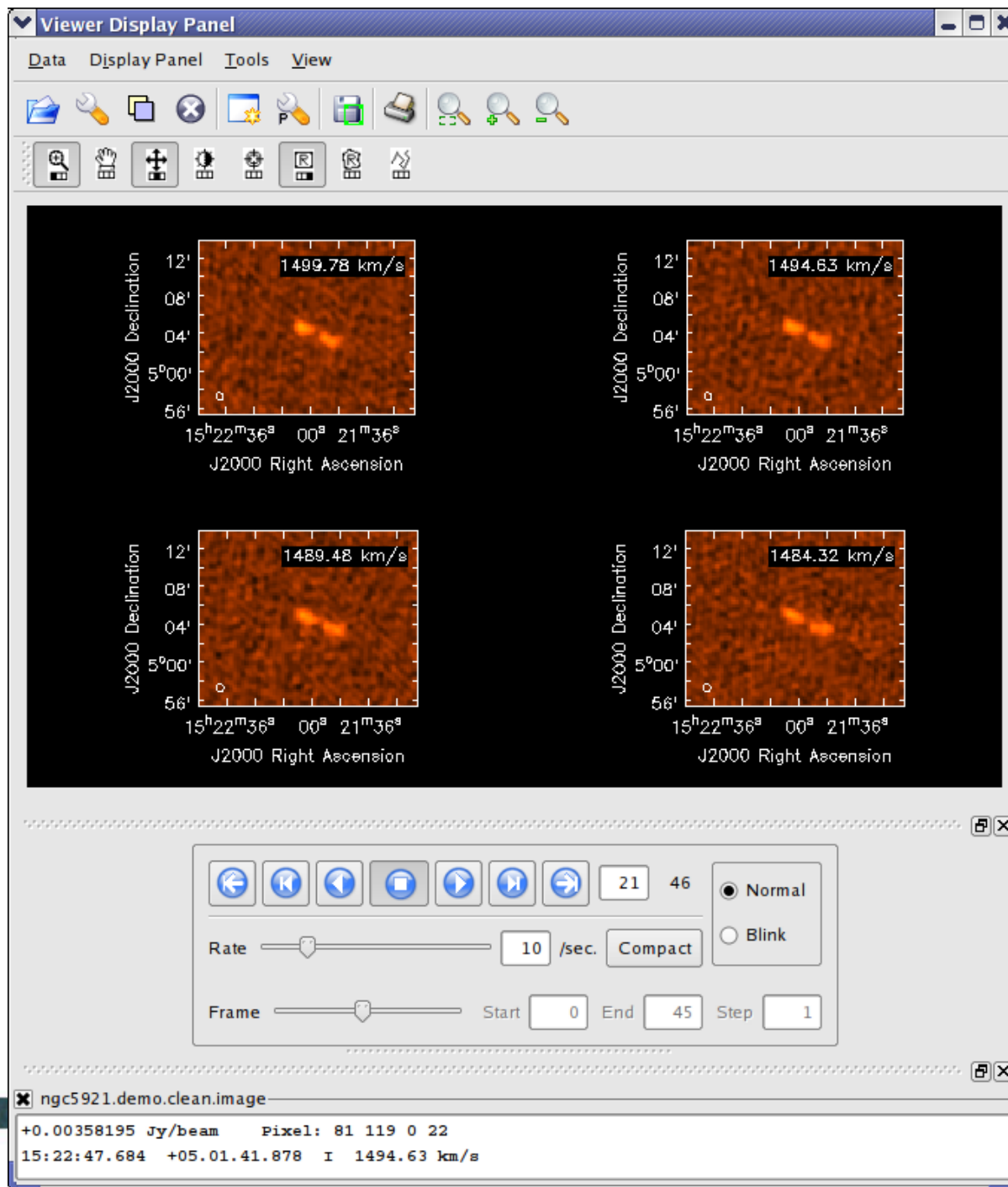
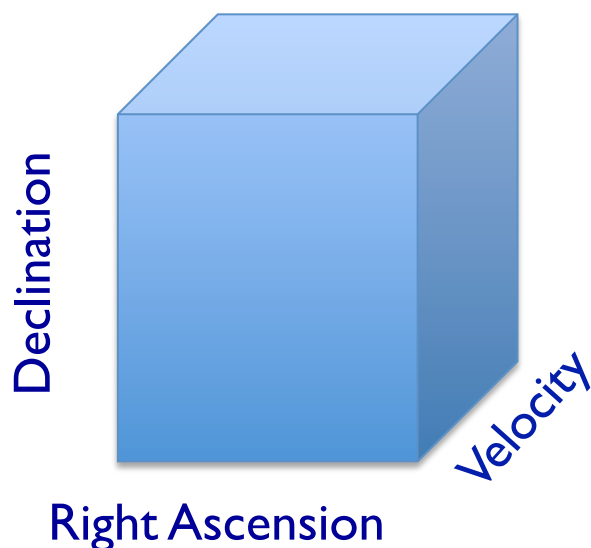


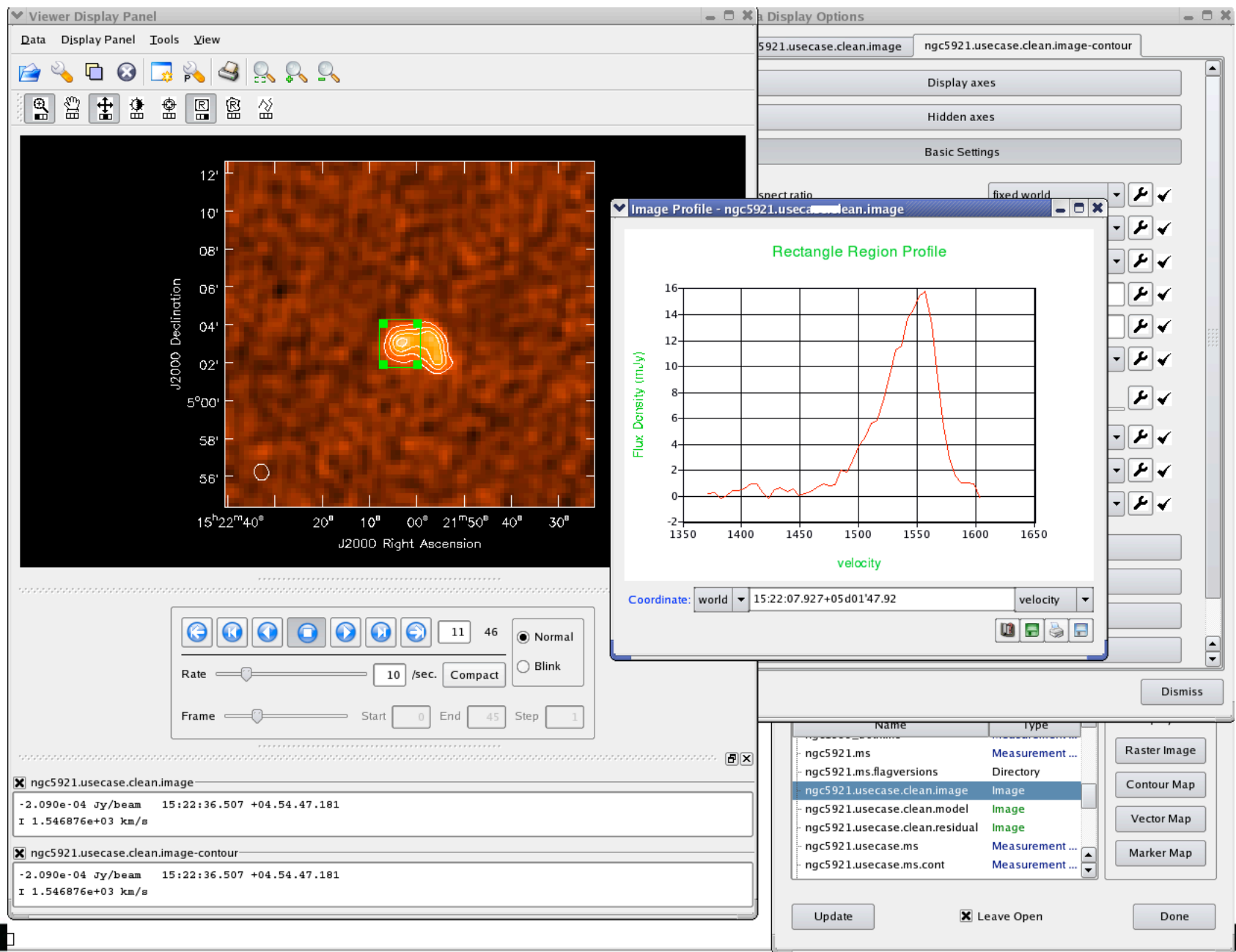
# Image Viewer



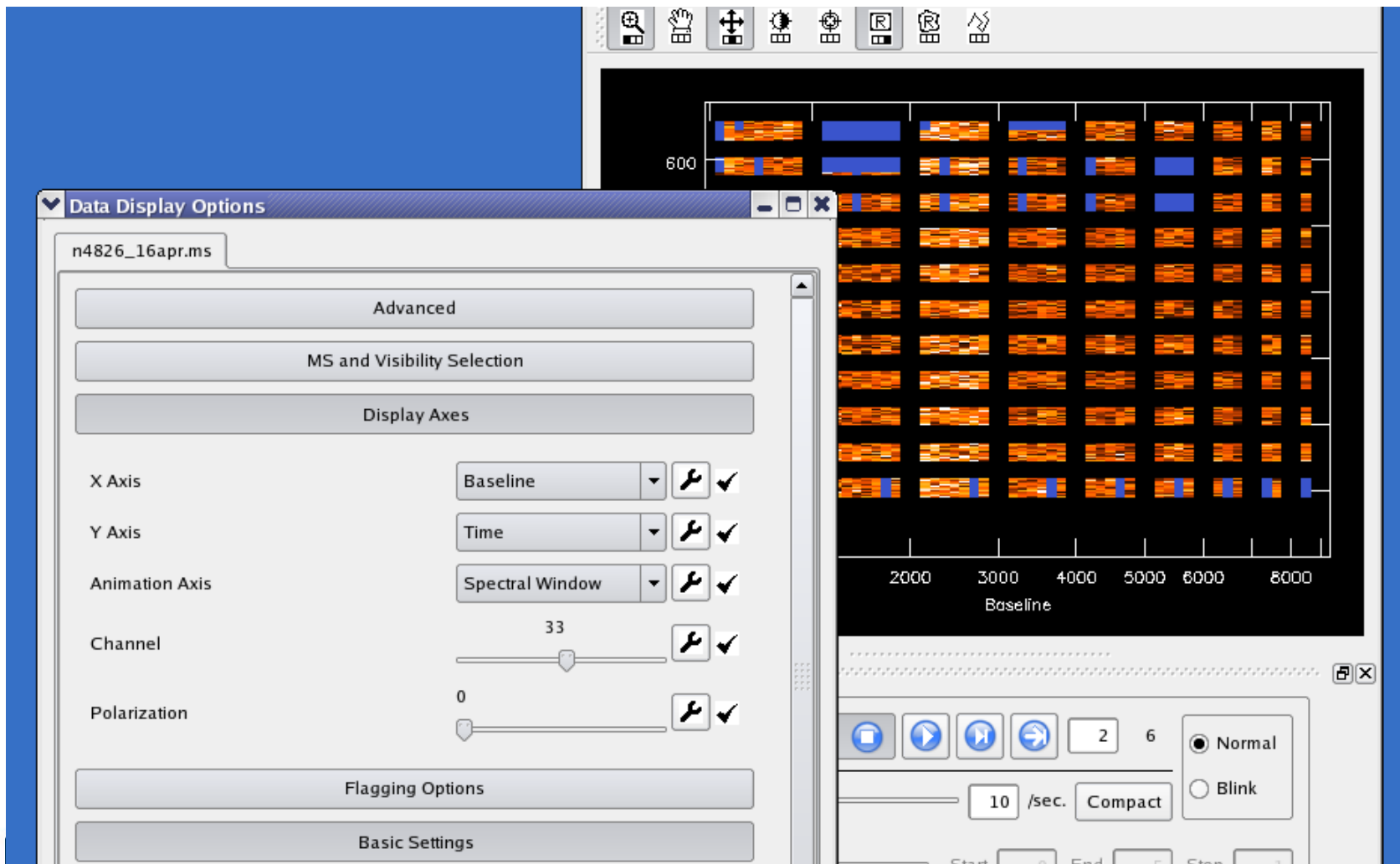
# Image Viewer

- Displaying cubes
- Movies
- Channel maps

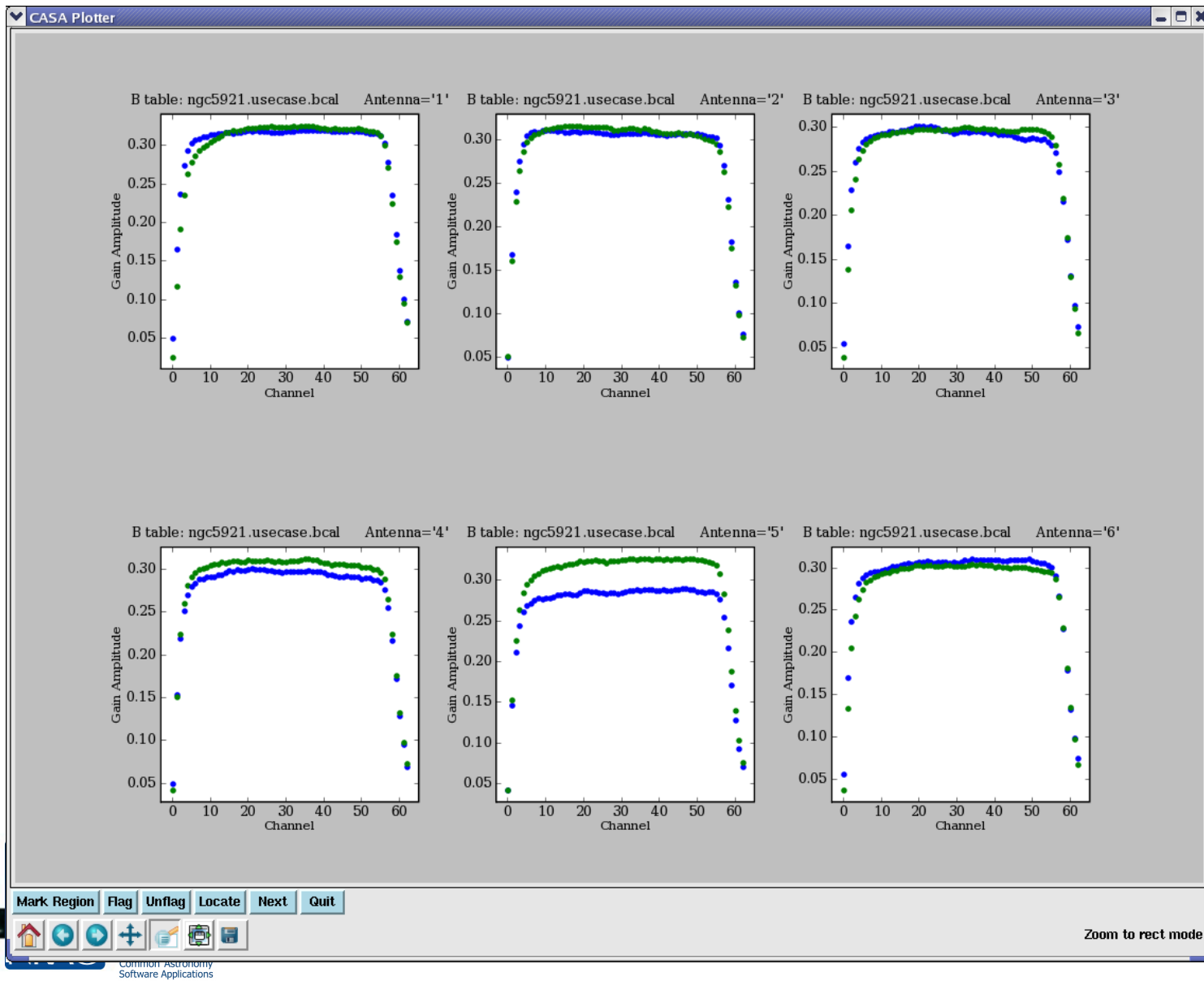




# MSView

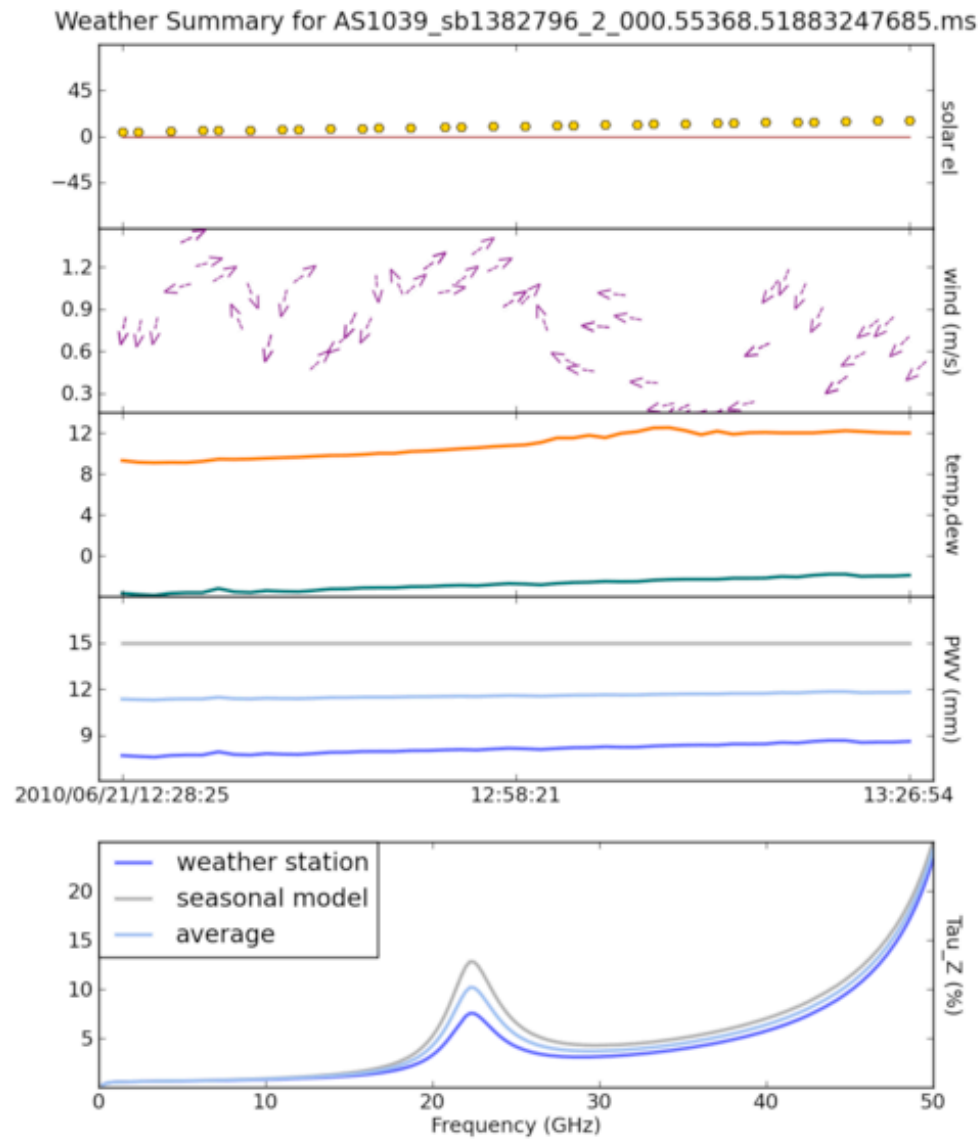


# Plotcal





# Plot Anything - matplotlib



# Image analysis

- **specfit**: to fit 1-dimensional Gaussians and/or polynomial models to an image or image region.
- **imfit** : fit one or more elliptical Gaussian components on an image region(s).
- Also immath, imstat, imval
- Don't forget the power of Python plus toolkit
- Contributed scripts can be used (and submitted by you).
- Contributed scripts are currently available at:  
<http://casaguides.nrao.edu/> → Data Reduction Guides  
→ EVLA Guides → Contributed Scripts

# Buildmytasks

- Write your own task!
- `task_plotWX.py` for the python code
- `plotWX.xml` for the interface and inline help
- Then build the task, best within CASA:
- CASA <22>: `!buildmytasks`
- This will create a few files like `*cli*`, `*pyc`, `mytasks.py`

# Buildmytasks

- Write your own task!
- `task_plotVWX.py` for the python code

```
import casac
from tasks import *
from taskinit import *
import pylab as pl
from math import pi,floor
#from matplotlib import rc

#rc('text', usetex=True)

#####
## hides the extreme Y-axis ticks, helps stack plots close together without labels overlapping
def jm_clip_Yticks():
    xa=pl.gca()
    nlabels=0
    for label in xa.yaxis.get_ticklabels():
        nlabels+=1
    thislabel=0
    if nlabels>3:
```

# Buildmytasks

- Write your own task!
- `task_plotWX.py` for the python code
- `plotWX.xml` for the interface and inline help

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" ?>
<casaxml xmlns="http://casa.nrao.edu/schema/psetTypes.html"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://casa.nrao.edu/schema/casa.xsd
file:///opt/casa/code/xmlcasa/xml/casa.xsd>

<task type="function" name="plotWX">
<shortdescription>Plot elements of the weather table for a given MS</shortdescription>

<input>

<param type="string" name="vis" kind="ms" mustexist="true">
<description>MS name</description>
<value></value>
</param>

<param type="double" name="seasonal_weight">
<description>weight of the seasonal model</description>
<value>0.5</value>
</param>
```

# Buildmytasks

- Write your own task!
- `task_plotWX.py` for the python code
- `plotWX.xml` for the interface and inline help
- Then build the task, best within CASA:
- CASA <22>: `!buildmytasks`
- This will create a few files like `*cli*`, `*pyc`, `mytasks.py`
- Finally run
- CASA<23>: `execfile('mytasks.py')`
- CASA <24>: `inp plotWX`
- `-----> inp(plotWX)`
- `# plotWX :: Plot elements of the weather table for a given MS`
- `vis = 'day2_TDEM0003_10s_norx' # MS name`
- `seasonal_weight = 0.5 # weight of the seasonal model`
- `doPlot = True # set this to True to create a plot`
- `async = False # If true the taskname must be started using plotWX(...)`



# Major Changes for CASA 3.4









- MODEL column will be optional
- new calibration table structure
- Parallelization of spectral line and continuum cleaning
- Full support of the new region file format in the viewer
- Reworked flagging

See Jeff Kern's talk

# Getting User Support

- CASA Home: <http://casa.nrao.edu>
  - Reference Manual & Cookbook, online task/toolhelp, download, example scripts
- [CASAguides.nrao.edu](http://CASAguides.nrao.edu)
  - For data reduction tutorials, tips, tricks, ...
- “Helpdesk” at [help.nrao.edu](http://help.nrao.edu) (for ALMA: [alma-help.nrao.edu](http://alma-help.nrao.edu))
  - Submit questions, suggestions, bugs (needs my.nrao.edu registration)
- CASA mailing lists: [casa-announce](#), [casa-users](#)
- CASA topic in NRAO Science Forum: [science.nrao.edu/forums](http://science.nrao.edu/forums)

# CASA Documentation

- Homepage: <http://casa.nrao.edu> → Using CASA
- CASA Reference Manual & Cookbook:  
 [http://casa.nrao.edu/Doc/Cookbook/casa\\_cookbook.pdf](http://casa.nrao.edu/Doc/Cookbook/casa_cookbook.pdf)  
 <http://casa.nrao.edu/docs/UserMan/UserMan.html>
- CASA Task Reference (same as inline help):  
 <http://casa.nrao.edu/docs/TaskRef/TaskRef.html>
- CASA Toolkit Manual:  
 <http://casa.nrao.edu/docs/casaref/CasaRef.html>
- CASAguides Wiki:  
 <http://casaguides.nrao.edu>
- Python:  
 <http://python.org/doc> (e.g., see Tutorial for novices)
- IPython:  
 <http://ipython.org>
- matplotlib:  
 <http://matplotlib.sourceforge.net/>



Large but  
detailed!