NRAO

CASA Intro

National Radio Astronomy Observatory

Juergen Ott (NRAO)

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)

Bryan Butler (CASA VLA subsystem scientist)

Jeff Kern (CASA manager)

CASA
Common Astronomy Software Applications

NRAO ESO 国立天文台 NAOJ CSIRO ALMA ASTRON

CASA (Common Astronomy Software Applications)

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

NRAO CASA Common Astronomy Software Applications

Outline

- CASA startup
- CASA basic python interface
- Tasks and tools
- The Measurement Set
- Data selection syntax
- Calibration
- Imaging
- Visualization tools
- Image analysis
- Build your own task!
- User support/Documentation

CASA (Common Astronomy Software Applications)

Current version: 4.0.1

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

For download go to the CASA homepage:

casa.nrao.edu

We have versions for 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, or if you want to check for a bugfix.

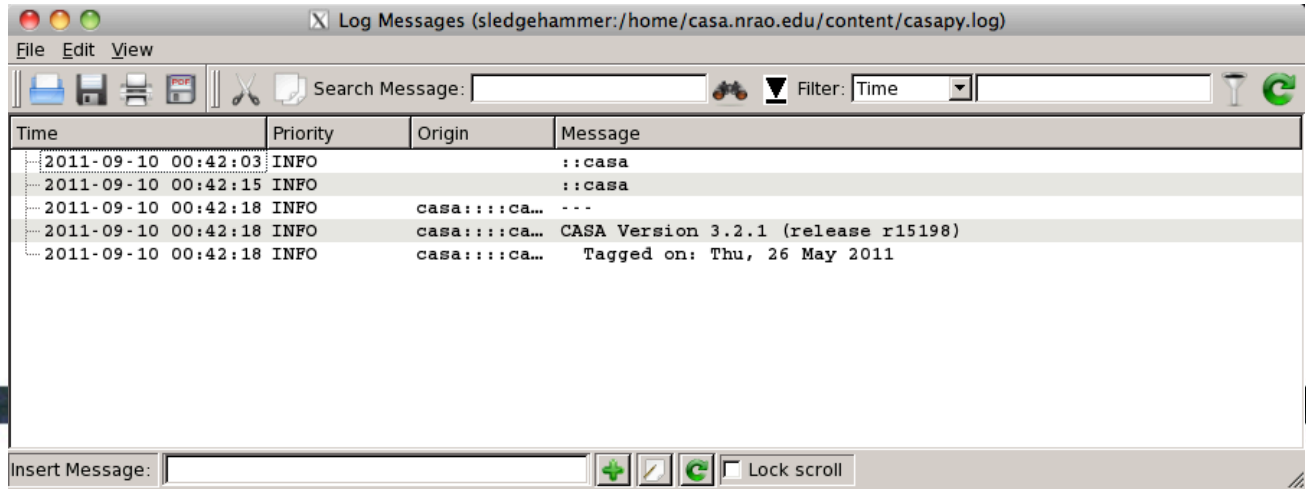
CASA Startup

$ casapy (or simply "casa")

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>:



NRAO CASA Common Astronomy Software Applications

CASA Interactive Interface

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

Basic Python tips

- to run a python ".py" script:

  execfile('<scriptname>')

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

Some python specialties:

- indentation matters!
  - indentation in python is for loops, conditions etc.
  - be careful when doing cut-and-paste to python
  - cut a few (4-6) lines at a time
- python counts from 0 to n-1!
- variables are global when using task interface
- tasknames are objects (not variables)

Tasks and tools in CASA

- Tasks - high-level functionality
  - function call or parameter handling interface
  - these are what you should use in tutorials
- Tools - complete functionality
  - **tool.method()** calls, they are internally used by tasks or can be used on their own
  - sometimes shown in tutorial scripts
- 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**

(some key shell commands like "ls" work without the exclamation mark)

Find the right Task

To see list of tasks organized by type:

tasklist

Default

New Info Customize Bookmarks Close

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

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


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



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

NRAO CASA Common Astronomy Software Applications

Find the right Task

To see list of tasks with short help:

taskhelp

NRAO CASA Common Astronomy Software Applications

Task Interface

examine task parameters with **inp** :

Default

New Info Customize Bookmarks Close

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

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

CASA <13>:
```

Default | Default | Default | Default | Default

NRAO

CASA Common Astron Software Applicat

Task Interface

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

  <param> = <value>

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

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

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

Expandable Parameters

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

xterm

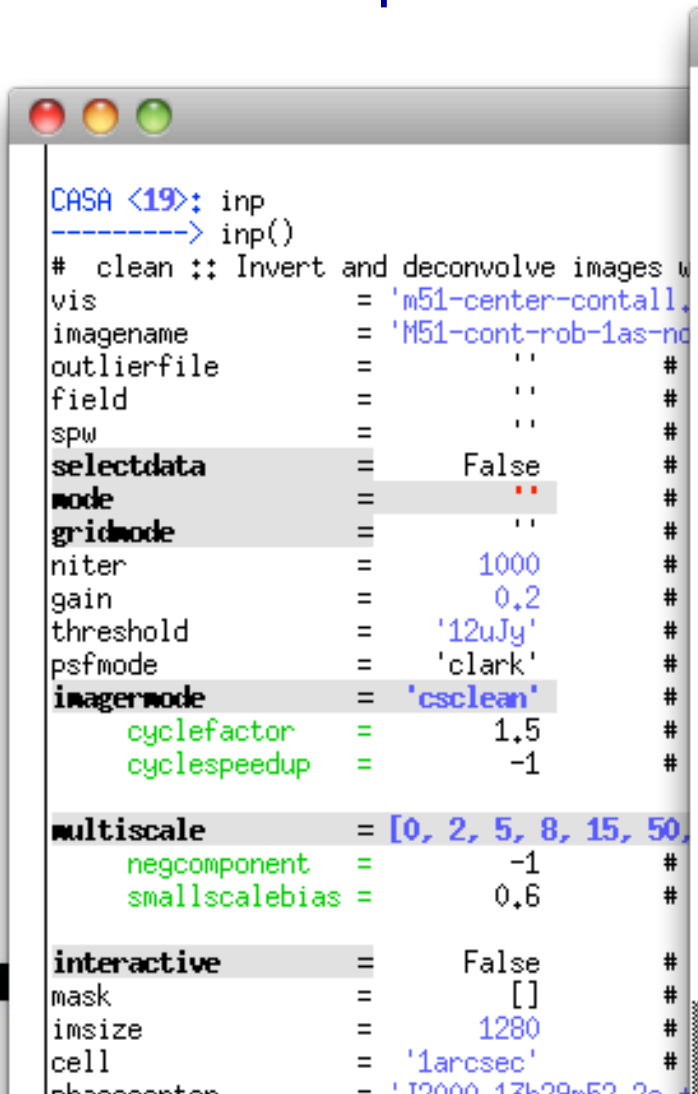
```
CASA <19>: inp
---------> inp()
#  clean :: Invert and deconvolve images with selected algorithm
vis                 = 'm51-center-contall.ms' #  Name of input visibility file
imagename           = 'M51-cont-rob-1as-noninteractive-mult-phasecenter-nterm2' #  Pre-name of output images
outlierfile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

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

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

Expandable Parameters

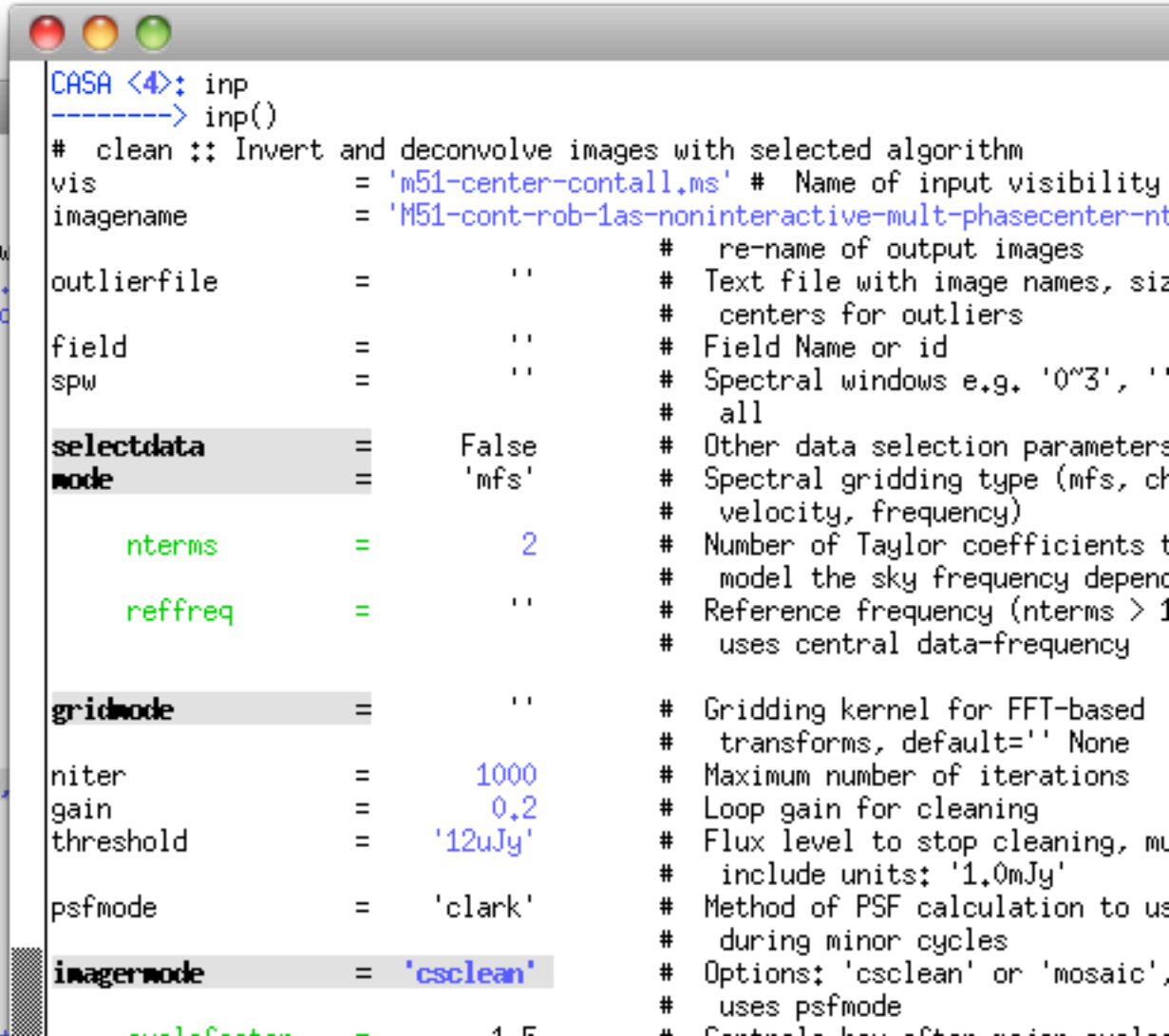
- Boldface parameters have subparameters that unfold when main parameter is set



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

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

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

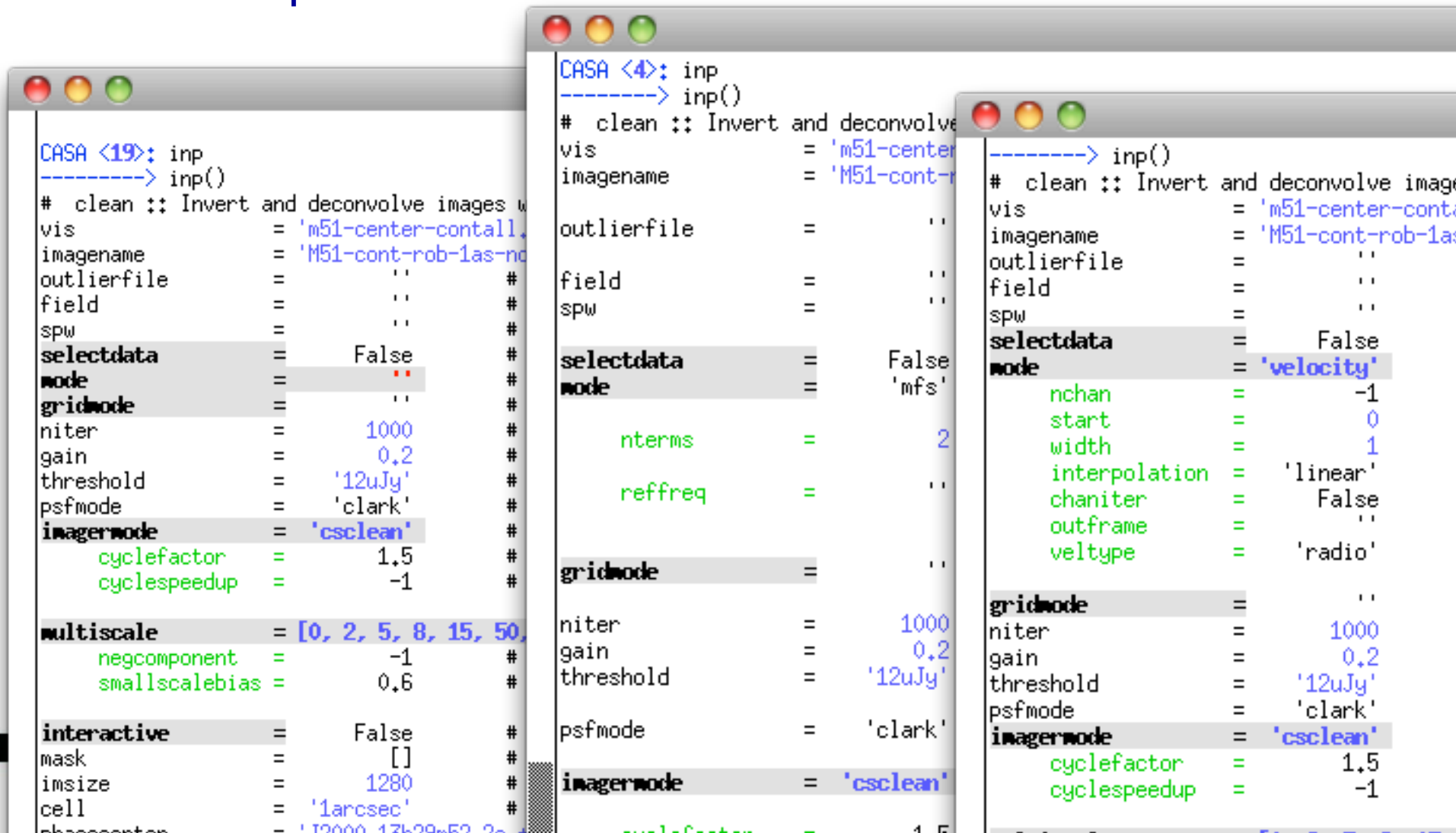


```
CASA <4>: inp
--------> inp()
#  clean :: Invert and deconvolve images with selected algorithm
vis                 = 'm51-center-contall.ms' #  Name of input visibility
imagename           = 'M51-cont-rob-1as-noninteractive-mult-phasecenter-nt
                                              #   re-name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

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

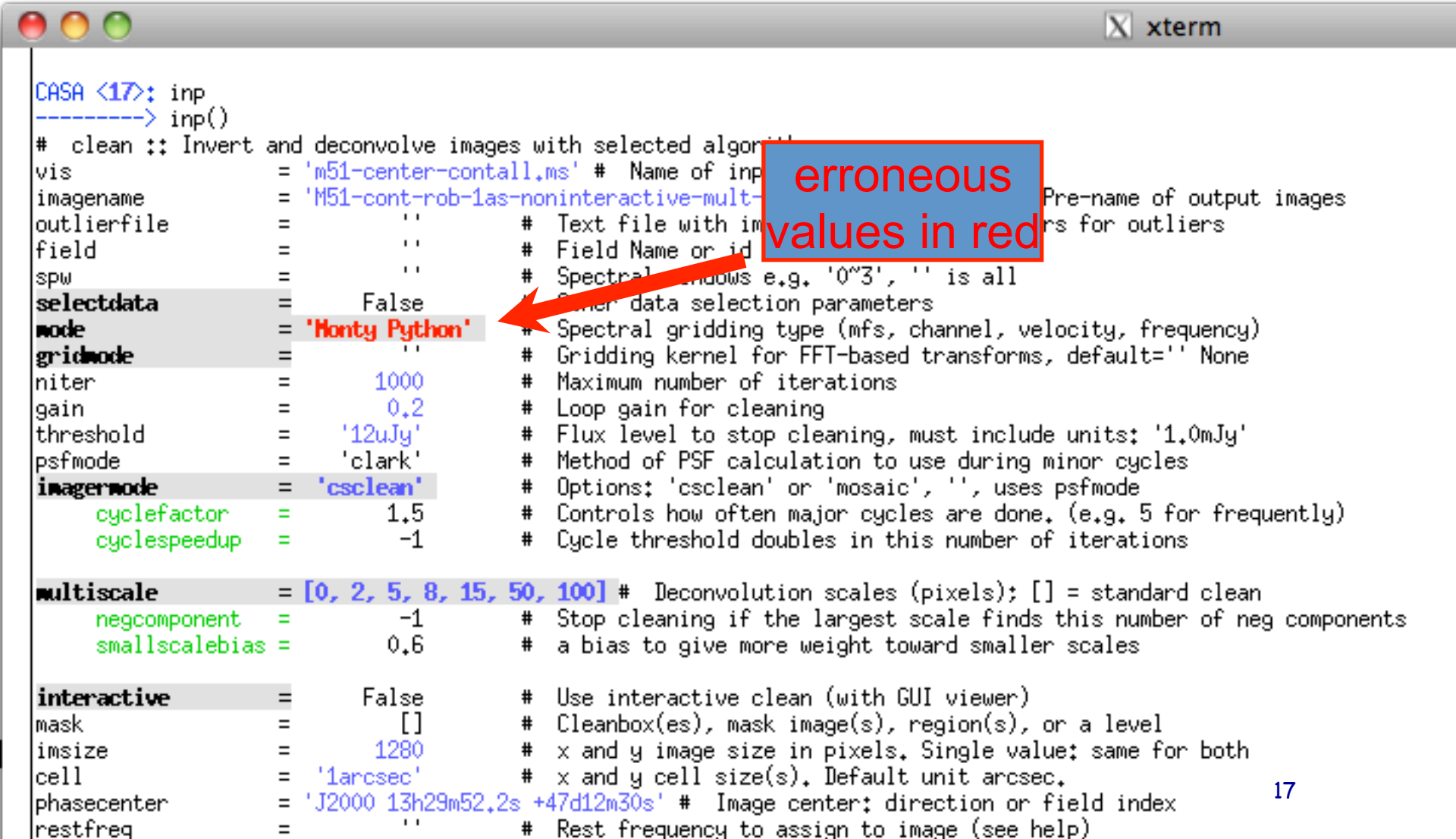
Expandable Parameters

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



Parameter Checking

sanity checks of parameters in inp :



Help on Tasks

In-line help:

help clean (or **pdoc clean**)

xterm

```
Help on clean task:

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

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


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

Task Execution

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

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

  e.g.

  clean(vis='input.ms', imagename='galaxy',selectvis=T, robust=0.5, imsize=[200,200])

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

Tools in CASA

- What if there's no task?

→ *use CASA tools!*

CASA tools are the building blocks for our tasks, so they contain all functionality albeit less bundled tool objects are,

e.g. imager (im) , calibrater (cb), ms (ms), image (ia), etc. (see **toolhelp**)

- Every tool has a bunch of methods, they are what you will use like: **functions.methods**

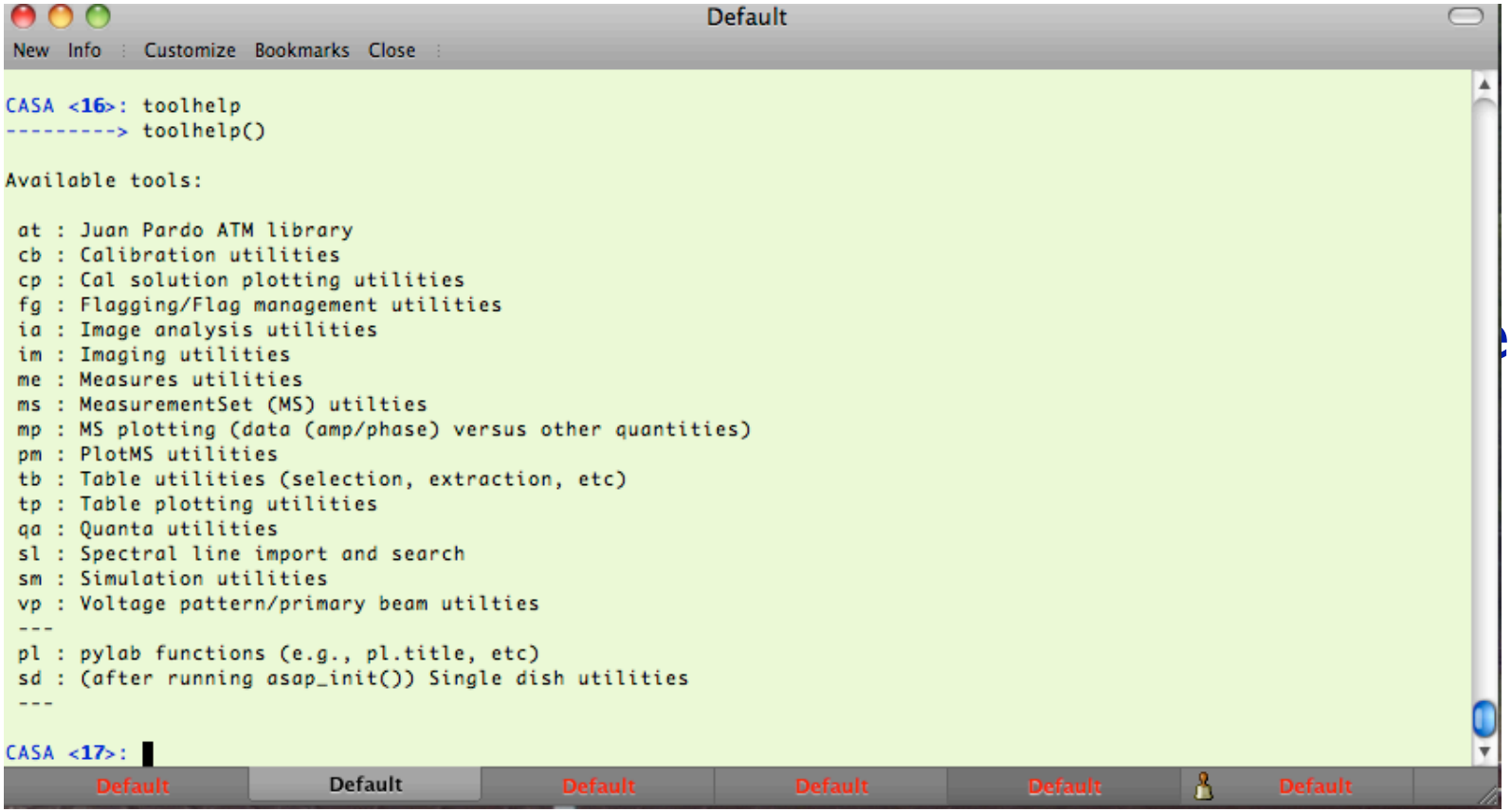
  call from casapy as **<tool>.<method>()**

  e.g. **ia.open('image.im')**

- Typically, one has to open close a dataset explicitly, if it is not closed, it may block other tasks from executing (table locks) and clutter the memory

CASA Tool List

list of default tools from **toolhelp** :



CASA Tool List

Execute tool methods…

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 p
 pm : Plot
 tb : Tabl
 tp : Tabl
 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
 ---

CASA <17>:
```

```
ia.open('rotated.image')
ia.rebin(region=mybox,outfile='slice.image',bin=[1,33,1,1],dropdeg=True)
ia.close('rotated.image')
```

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 and use this resource!

~1000 tool methods available!

Tool methods described in the CASA Toolkit Reference:

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

CASA data are Tables in Directories

- The *Measurement Set* contains your *visibilities*.
  Images are in the *CASA image format*.
  Calibration information is stored in *Calibration tables*.

- ALL of these are *directories* which contain the necessary information

- So copy them via **cp –r** (or **!cp –r** within CASA),

- you may need to **tar** them for data transfer.

- Delete tables within casa via **rmtables('universe.ms')**

Outside CASA **'rm –rf'** , or in python **os.system('rm –rf universe.ms')** may also work. But those methods may leave traces in the cache.

The Measurement Set

The Measurement Set (MS)

- Contains the visibilities in the MAIN table in table.* files
- also contains sub-tables
  e.g. FIELD, SOURCE, ANTENNA, WEATHER etc.
  sub-tables are sub-directories
- The **tb** tools can manipulate the tables directly
- Definition of the MS (and other formats) can be found on *casa.nrao.edu* → Using CASA → Other Documentation

Example MS

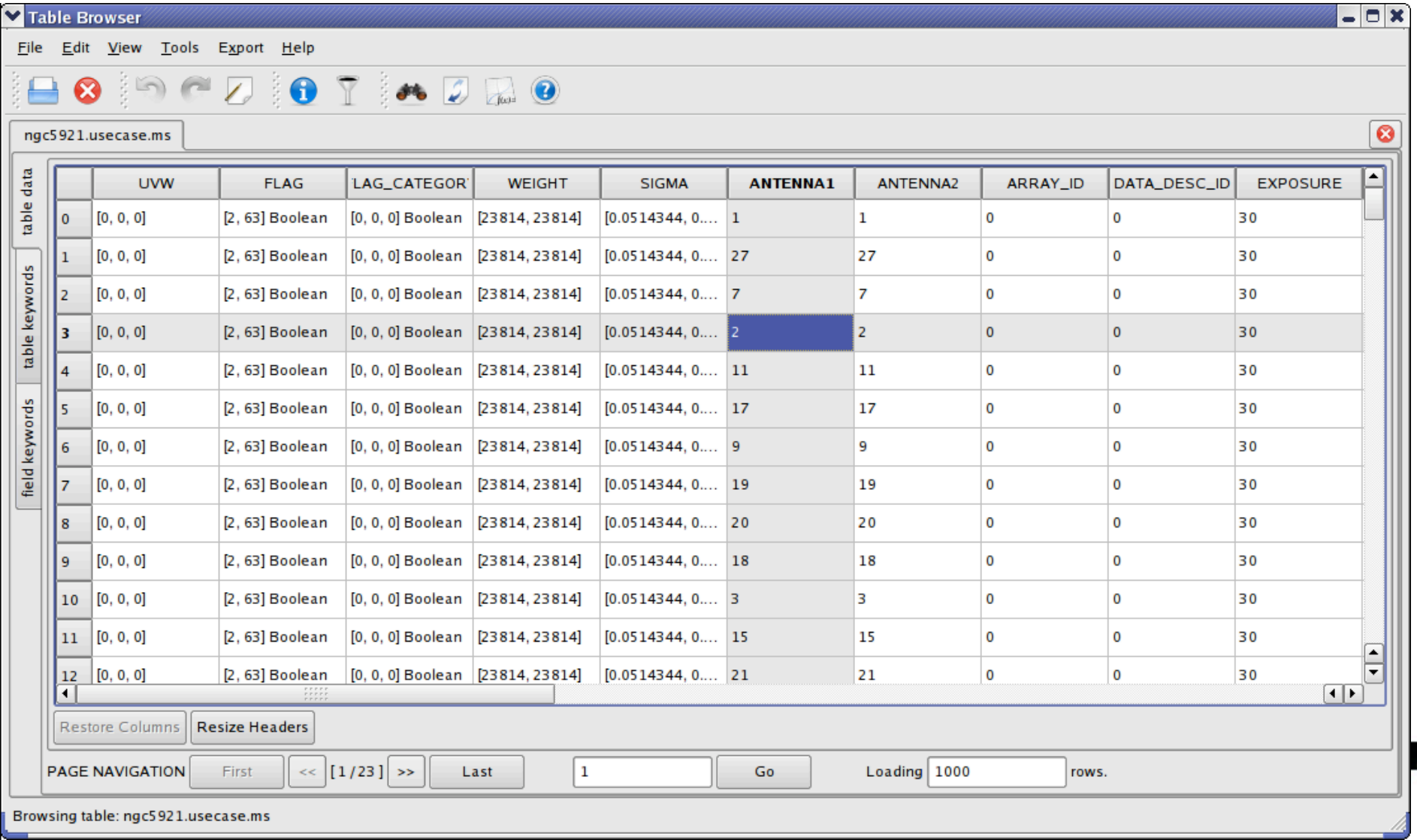
Example: `ls ngc5921.usecase.ms`

```
smyers@olorin ~/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@olorin ~/CASA/Test $ ls ngc5921.usecase.ms/FIELD
table.dat       table.f0        table.f0i       table.info      table.lock
```

MAIN Table Contents

Inspect with task **browsetable** (application **casabrowser**)



	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

MS Data Selection Syntax

- Frequently one likes to select a subset of visibilities to perform an action, e.g., based on antennas, baselines, frequencies, time, polarization etc. The standard CASA selection syntax is the following:

- **field** (*spatial*) 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** (*spectral*) 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

- **timerange** (*temporal*) - 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'

- **antenna** - string with antenna name or ID
  - first check for name, then ID (beware VLA name 1-27, ID 0-26)
  - example: antenna = '1~5,11' ; antenna = 'ea*', '!va'
  - Baselines: 'ea01&ea10'
  - Antenna pad names are supported after '@', e.g. 'ea12@N01' only selects antenna ea12 when it was occupying the N01 antenna pad

NRAO CASA Common Astronomy Software Applications

Selection Syntax

- **scan** – the scan numbers (an execution sequence)
  e.g. can='3~14'

- **correlation** – polarization products
  e.g. correlation='LL,RR,RL'

- **observation** – an observation id (when mutiple observation runs are merged together)

- **uvrange** – select on uvranges
  e.g. uvrange='30m~600m'

NRAO CASA Common Astronomy Software Applications

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)
```

Calibration

- Data structure: *2 data columns*
- **DATA** column (raw data)
- **CORRECTED_DATA** (calibrated data)
- A **MODEL** is constructed or provided
- **Calibration tables** are used to transform DATA to the MODEL on calibrators, then transferred to the source data
- Sets of calibration tables applied **incrementally** (apply all previous calibration tables before solving/application)
- Applycal *creates and overwrites* **CORRECTED_DATA** (can **split** to **DATA**)

(a MODEL is usually attached as an image, but it can be reproduced as a third column, setting "usescratch=T" keyword in *setjy* and *clean*)

Calibration continued

- Solvers (e.g. **bandpass**, **gaincal**, **polcal**, **blcal**)
- Based on data x 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
- Some caltables are created from other data: opacity, gain-elevation, Tsys, antenna positions, etc. (task **gencal**)

➔ See the calibration talk

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 (n)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

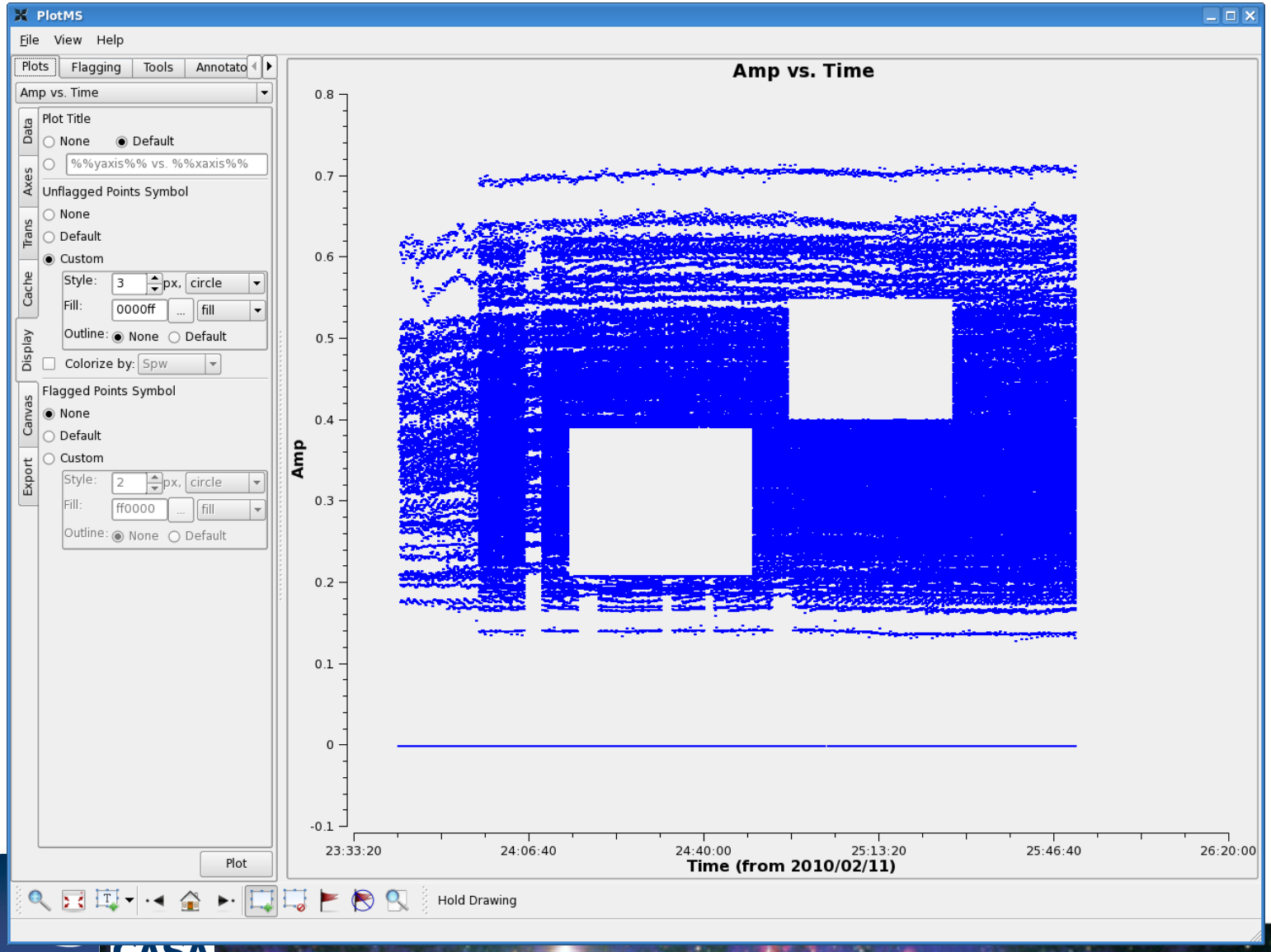

PlotMS
File View Help
Plots Flagging Tools Annotato
Amp vs. Time
Data Axes Trans Cache Display Canvas Export
Plot Title
None Default
%%yaxis%% vs. %%xaxis%%
Unflagged Points Symbol
None
Default
Custom
Style: 3 px, circle
Fill: 0000ff ... fill
Outline: None Default
Colorize by: Spw
Flagged Points Symbol
None
Default
Custom
Style: 2 px, circle
Fill: ff0000 ... fill
Outline: None Default
Plot
Amp vs. Time
Amp
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
23:33:20
24:06:40
24:40:00
25:13:20
25:46:40
26:20:00
Time (from 2010/02/11)
Hold Drawing

Image Viewer

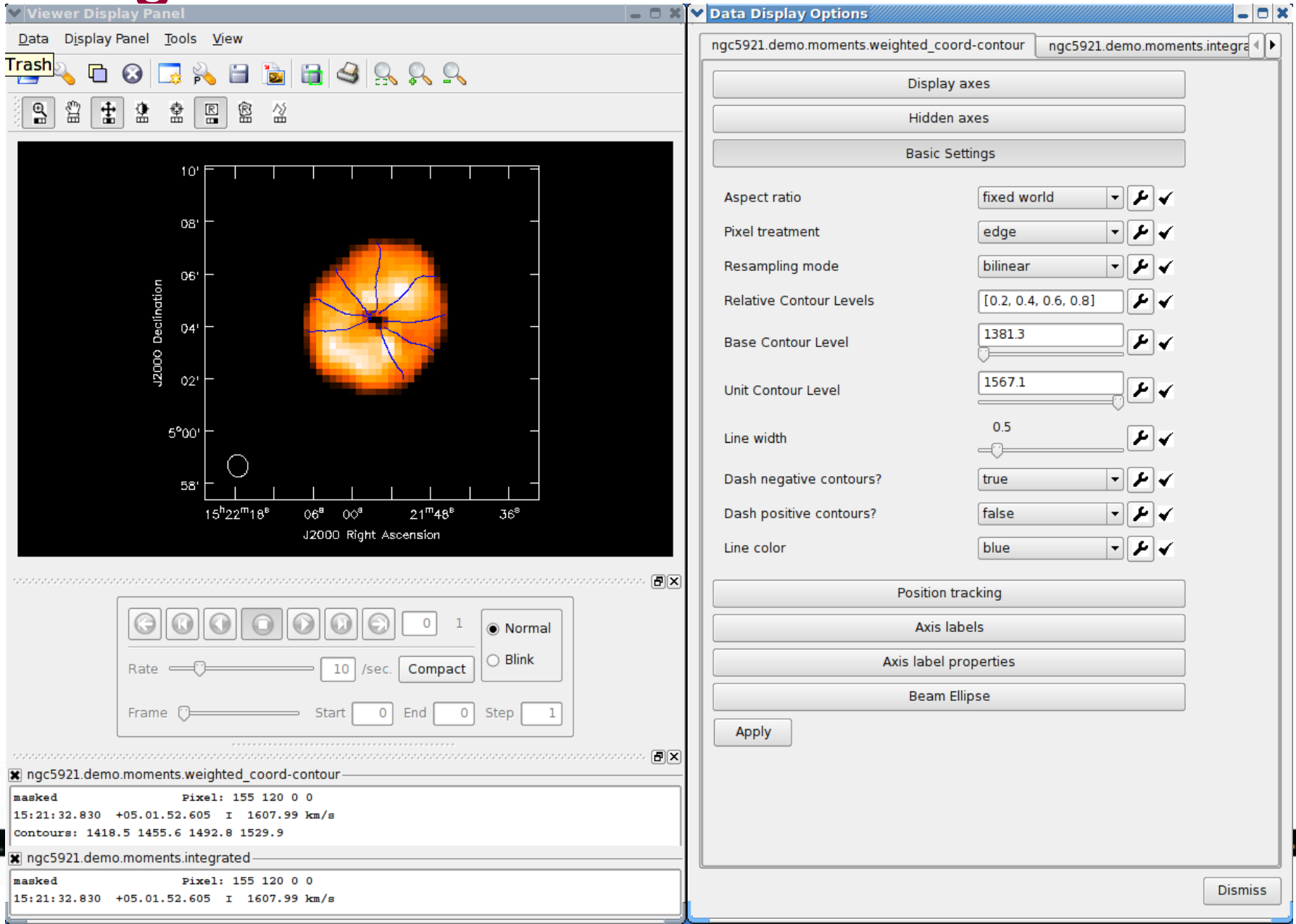
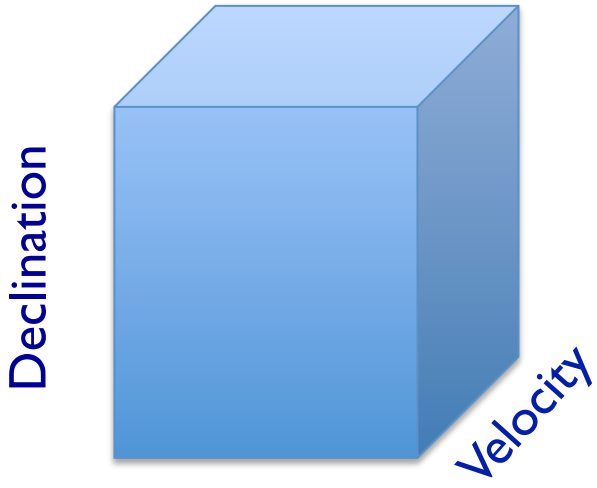
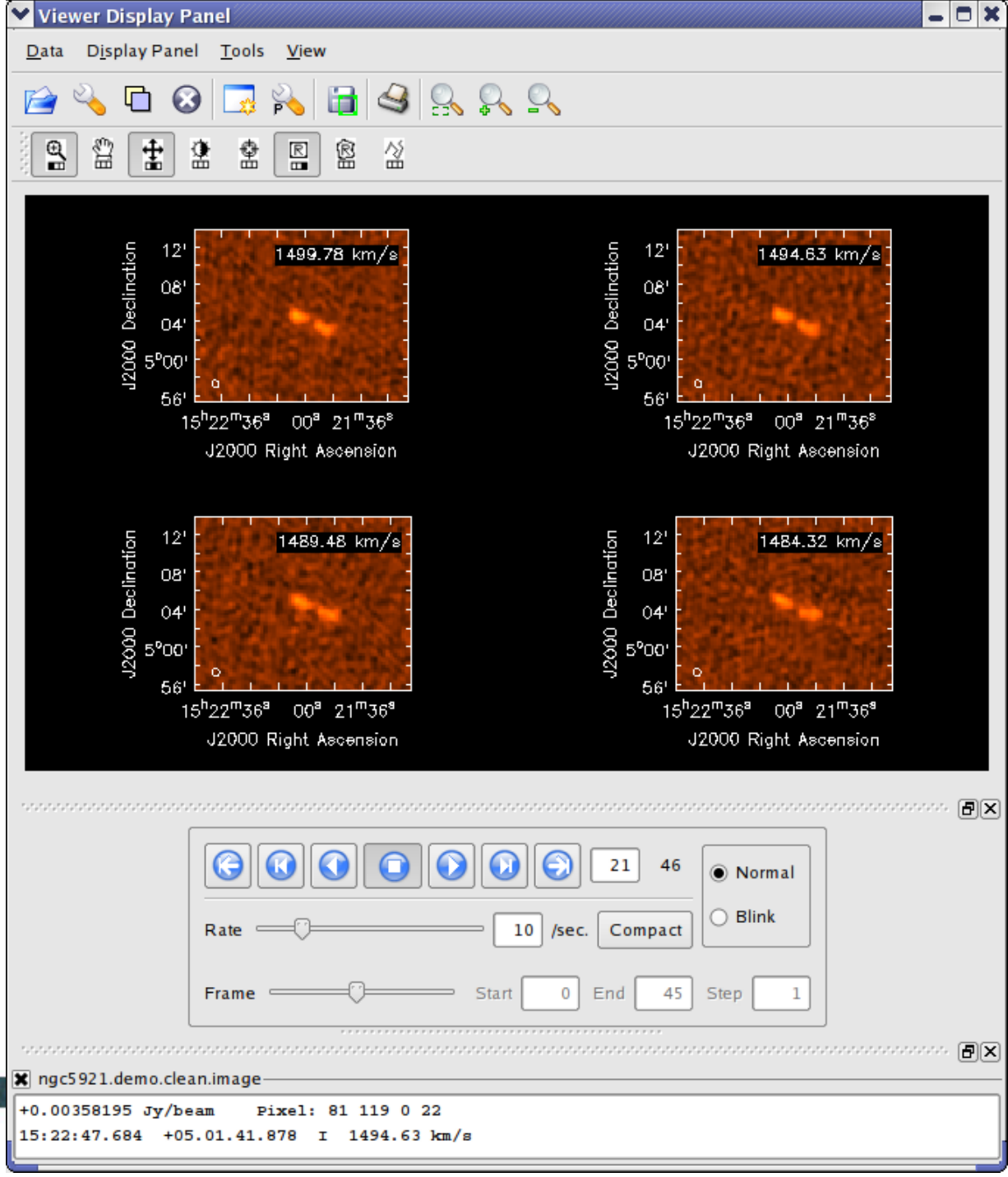


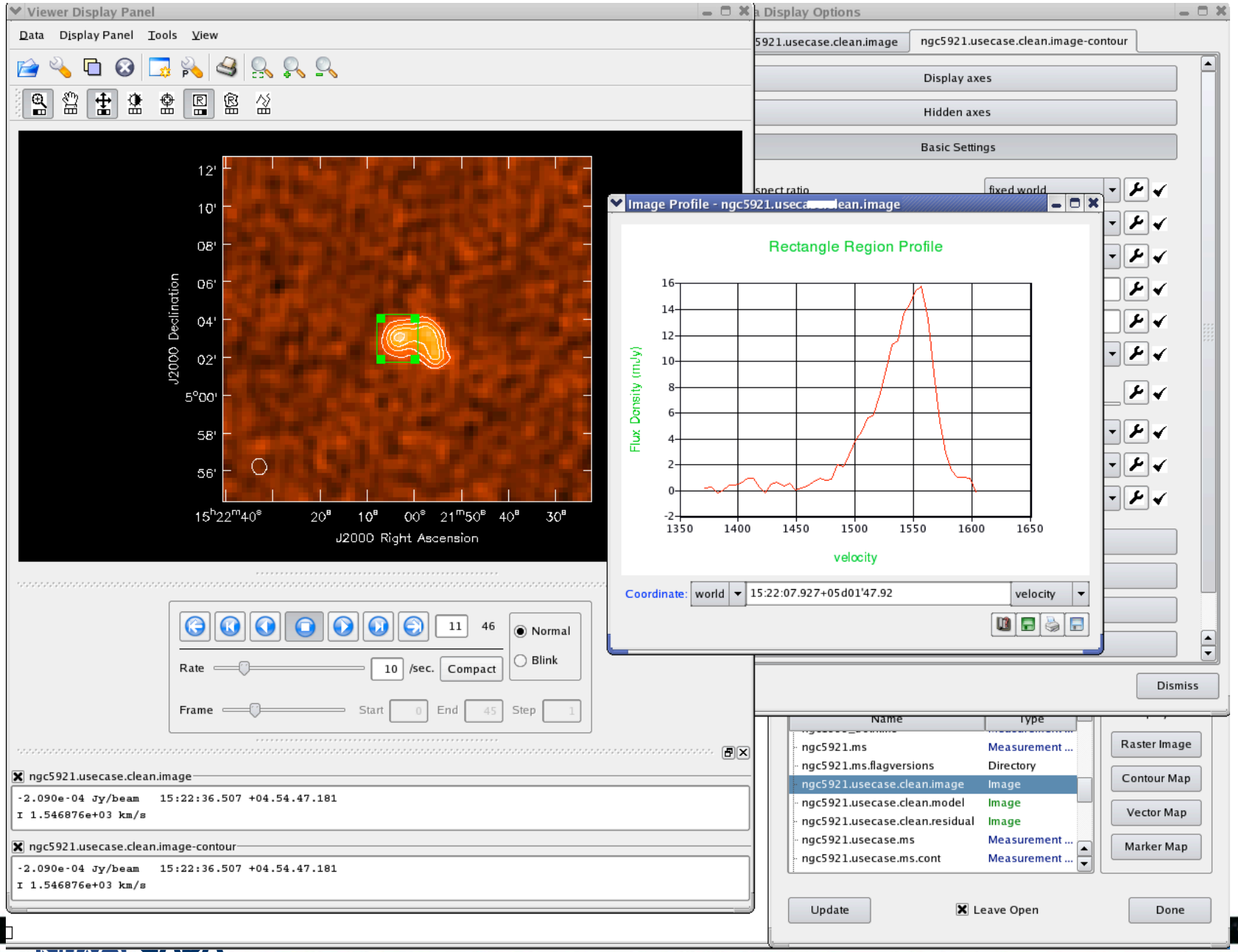
Image Viewer

- Displaying cubes
- Movies
- Channel maps





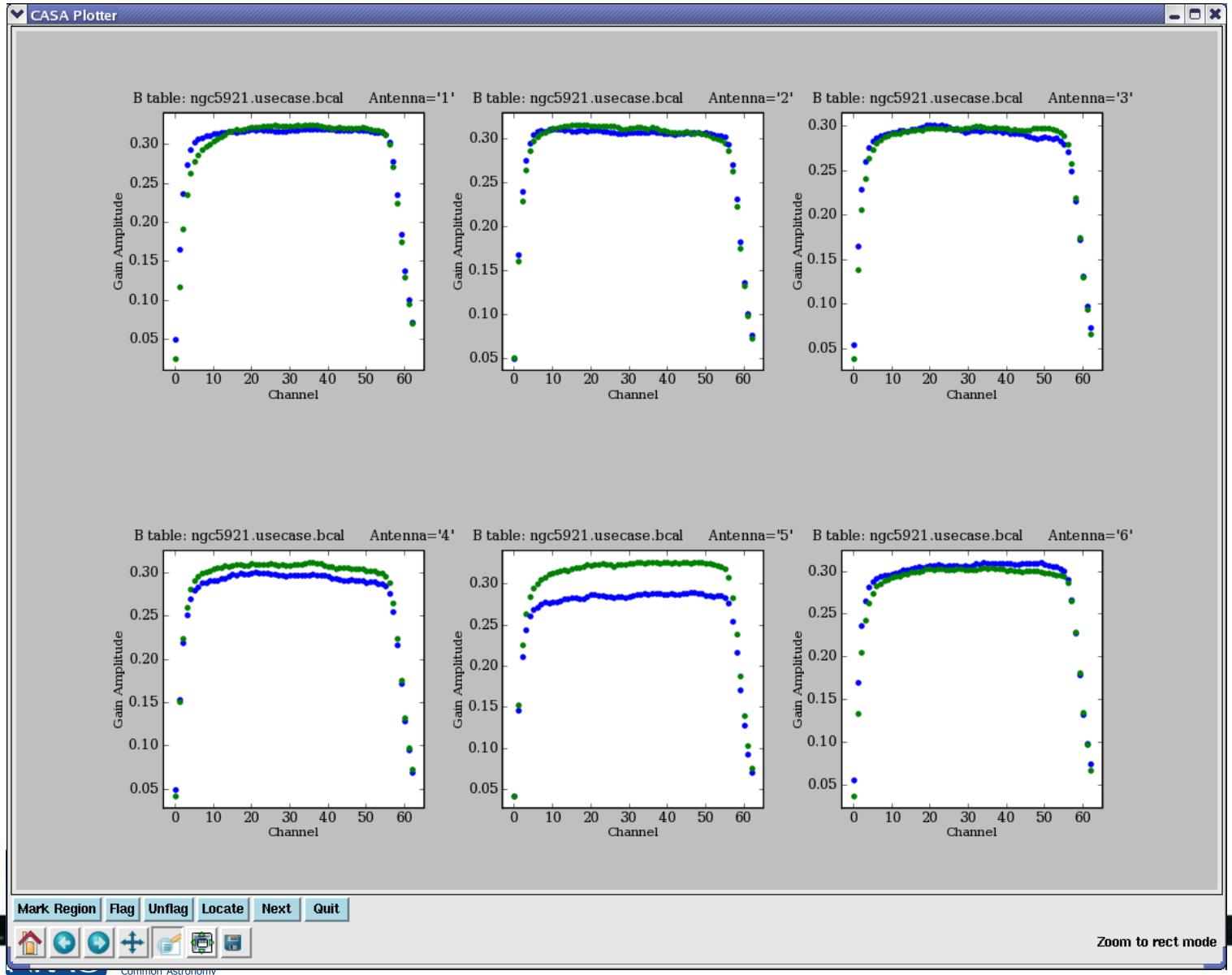
NRAO CASA Common Astronomy Software Applications


Viewer Display Panel
Data
Display Panel
Tools
View
J2000 Declination
J2000 Right Ascension
Image Profile - ngc5921.usecase.clean.image
Rectangle Region Profile
Flux Density (mJy)
velocity
Coordinate:
world
15:22:07.927+05d01'47.92
velocity
Display Options
ngc5921.usecase.clean.image
ngc5921.usecase.clean.image-contour
Display axes
Hidden axes
Basic Settings
Normal
Blink
Rate
/sec.
Compact
Frame
Start
End
Step
Dismiss
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 ...
Raster Image
Contour Map
Vector Map
Marker Map
Update
Leave Open
Done
-2.090e-04 Jy/beam 15:22:36.507 +04.54.47.181
I 1.546876e+03 km/s

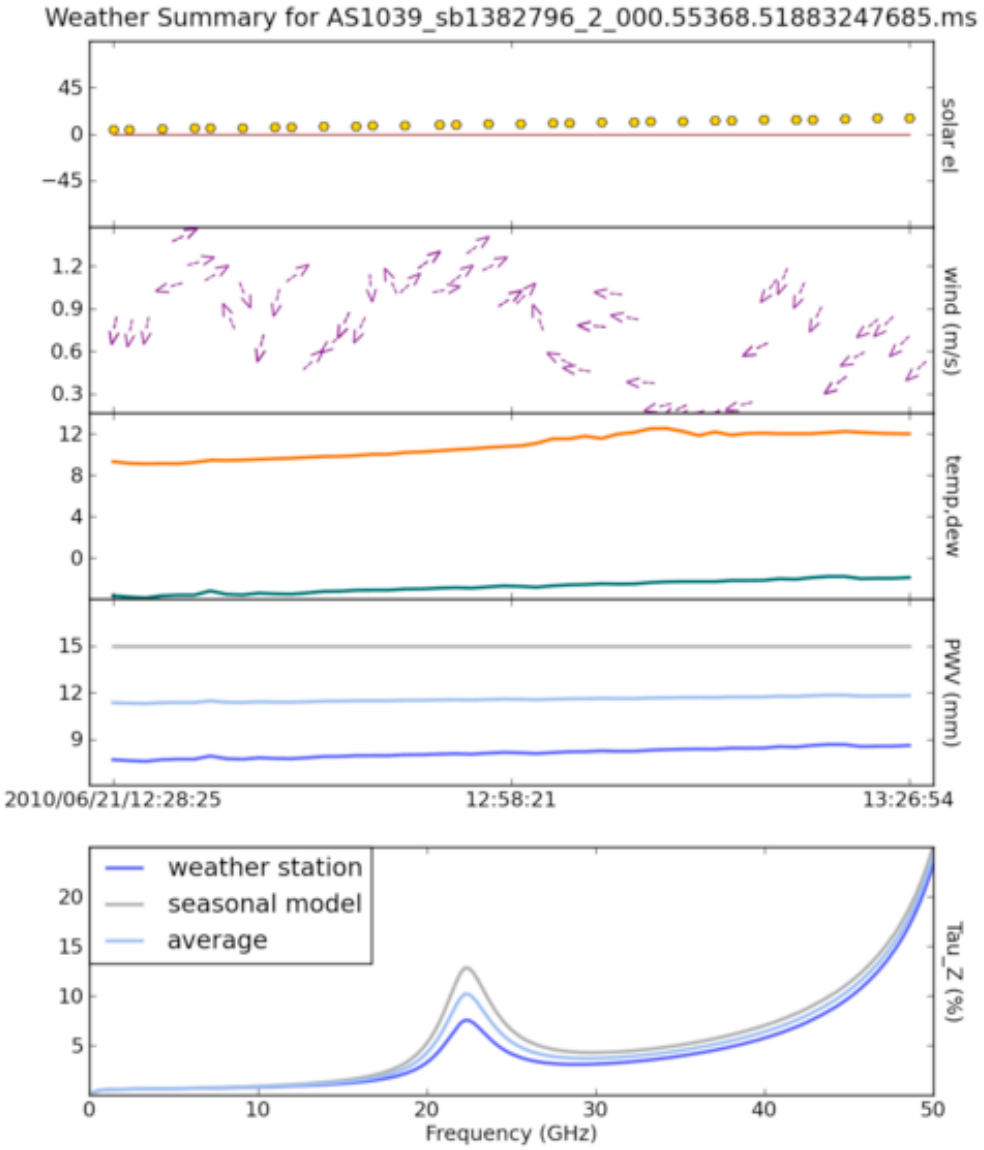
MSView


Data Display Options
n4826_16apr.ms
Advanced
MS and Visibility Selection
Display Axes
X Axis
Baseline
Y Axis
Time
Animation Axis
Spectral Window
Channel
33
Polarization
0
Flagging Options
Basic Settings
600
2000
3000
4000
5000
6000
8000
Baseline
2
6
Normal
Blink
10
/sec.
Compact

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
Mark Region
Flag
Unflag
Locate
Next
Quit
Zoom to rect mode

Plot Anything - matplotlib



NRAO CASA Common Astronomy Software Applications

Image analysis

- **Immoments**: create moment maps of spectral cubes
- **specfit**: to fit 1-dimensional Gaussians and/or polynomial models to an image or image region.
- **imfit** : fit one or more spatial elliptical Gaussian components to sources
- **immath**: use to do maths with images, or create spectral index maps
- Also **imstat**, **imval, imcollapse, …**
- Don't forget the power of Python plus toolkit
- Contributed scripts can be used (and submitted by you).
- NRAO Contributed scripts are currently available on http://casaguides.nrao.edu/
- We encourage to submit your own scripts to the NRAO forums: *science.nrao.edu/forums*

NRAO CASA Common Astronomy Software Applications

Buildmytasks

- Write your own task!
- **task_plotWX.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 overlaping
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**

Getting User Support

- CASA Home: http://casa.nrao.edu
  - Reference Manual & Cookbook, online task/toolhelp, download, example scripts

- CASAguides.nrao.edu
  - For data reduction tutorials, tips, tricks, …

- "Helpdesk" at help.nrao.edu (for ALMA: help.almascience.org)
  - 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

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/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:
  http://casaguides.nrao.edu
- Python:
  http://python.org/doc (e.g., see Tutorial for novices)
- IPython:
  http://ipython.org
- matplotlib:
  http://matplotlib.sourceforge.net/