Introduction to CASA

Simulations and Imaging

Erin Cox
Author: Meredith MacGregor

Atacama Large Millimeter/submillimeter Array
Karl G. Jansky Very Large Array

NSF · Associated Universities, Inc. · NRAO

What Is CASA?

CASA, the Common Astronomy Software Applications package, is being developed with the primary goal of supporting the data post-processing needs of the next generation of radio astronomical telescopes such as ALMA and VLA.

For you: CASA is your go-to tool for simulations, data analysis, and imaging

Important note: CASA has an iPython interface

CASA

Common Astronomy Software Applications

NRAO

Where Do You Get CASA?

If you haven't downloaded in advance, you will need it to participate in the afternoon hands-on time!

https://casa.nrao.edu/casa_obtaining.shtml

Download most recent version 5.4.0 (pipeline)

Some Helpful Resources

CASA has lots of online guides to help get you started:

http://casaguides.nrao.edu/index.php?title=Main_Page

The full reference/cookbook is available here:

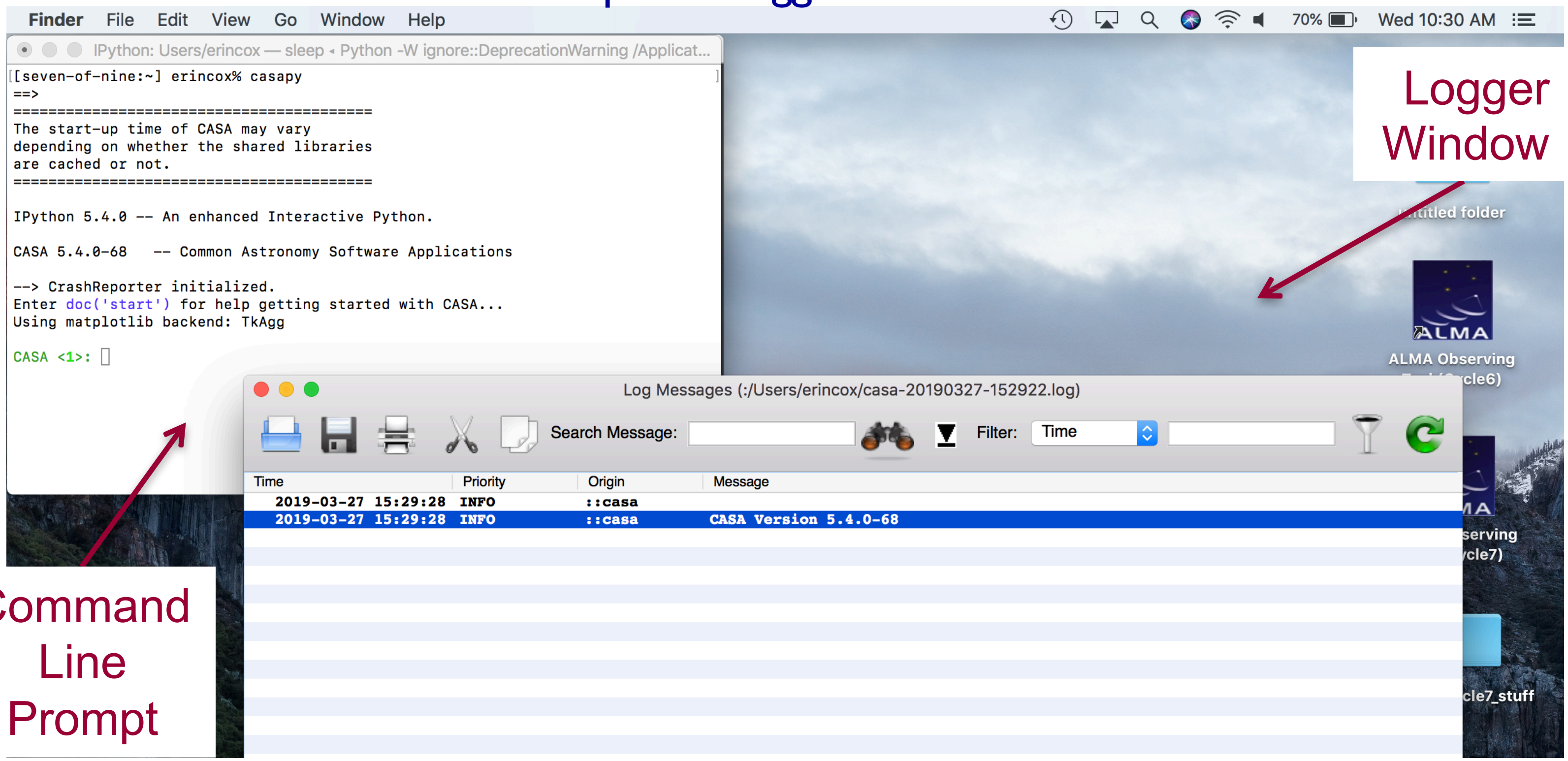
http://casa.nrao.edu/Doc/Cookbook/casa_cookbook.pdf

ALMA-specific tutorials are available here:

http://casaguides.nrao.edu/index.php?title=ALMAguides

Starting CASA

When CASA first opens, you will see a Python command line prompt and a separate logger window



Pay attention to the logger window! Most tasks write important information to this window. All logger messages are also saved into a file labeled ‘casa##.log’

NRAO

Calling CASA Tasks

If you want to know what a task's parameters are, type:

tget <taskname>

```
CASA <2>: tget clean
--------> tget(clean)
Restored parameters from file clean.last

CASA <3>:
```

Then, type:

inp

This will bring up a list of all possible parameters for that task. It will also retrieve any previously used parameter values. If you want to restore the default parameters, instead use:

default('<taskname>')

To run a task type:

go

You can always get help on a task by typing:

help <taskname>

NRAO

Simulating Observations

Running a simulation can help convince the TAC that your proposed observations are feasible.

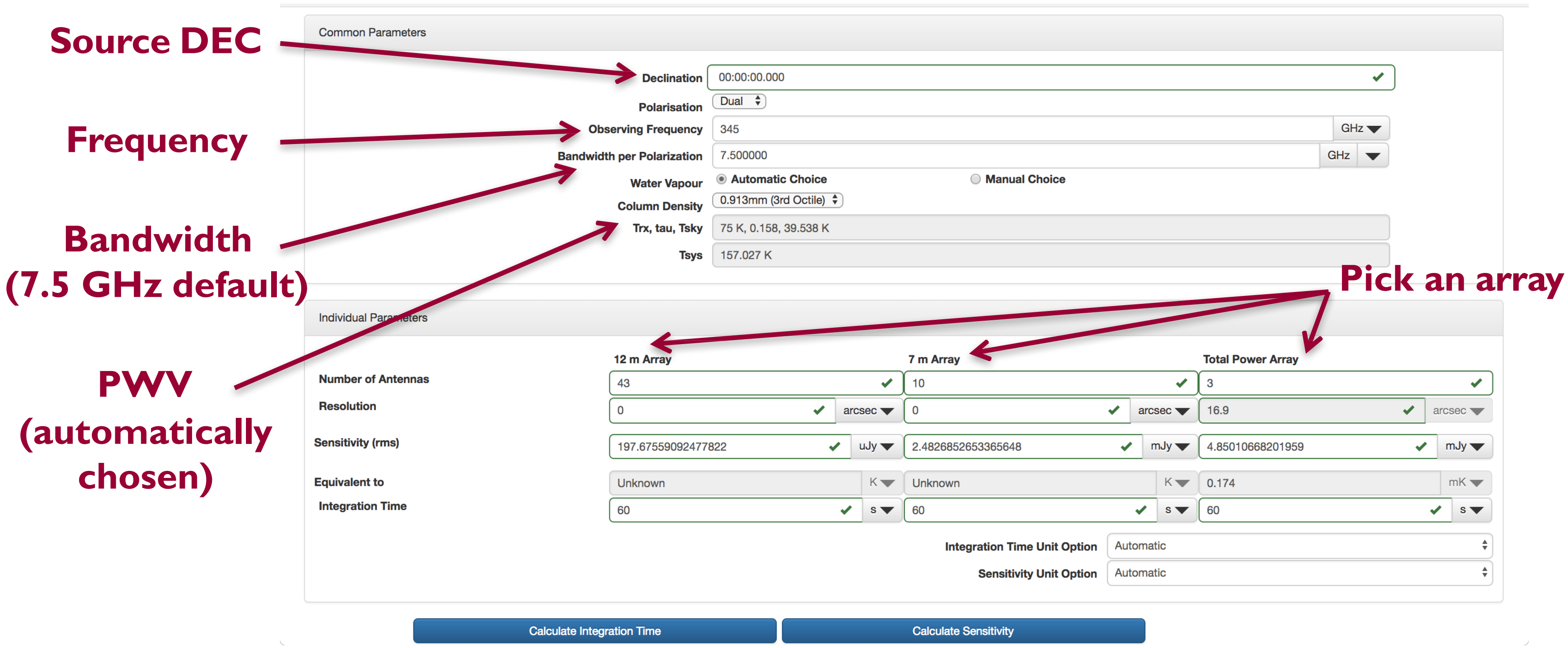
Steps for simulating observations:

1. Use the ALMA sensitivity calculator to determine the necessary observing time for your science goals
2. Generate simulated visibilities using the 'simobserve' task in CASA (takes FITS input)
3. Image, analyze, and evaluate the resulting visibilities

Repeat for different antenna configurations, observing times, etc.

Simulating Observations

Sensitivity calculator available here: https://almascience.eso.org/proposing/sensitivity-calculator



Either enter a sensitivity (rms) and calculate integration time or enter an integration time and calculate sensitivity

Simulating Observations

All of the 'simobserve' parameters in CASA

'simobserve' takes FITS images as inputs and generates simulated visibilities for given antenna configurations, observing time, and PWV

A CASA guide on simulating observations is available here: https://casaguides.nrao.edu/index.php/Simulating_Observations_in_CASA_5.4

```
#  simobserve :: visibility simulation task
project             =  'hd10647'       #  root prefix for output file names
skymodel            = 'hd10647_model.fits' #  model image to observe
     inbright       =         ''       #  scale surface brightness of brightest pixel
                                       #   e.g. "1.2Jy/pixel"
     indirection    =         ''       #  set new direction e.g. "J2000 19h00m00
                                       #   -40d00m00"
     incell         =         ''       #  set new cell/pixel size e.g. "0.1arcsec"
     incenter       =         ''       #  set new frequency of center channel e.g.
                                       #   "89GHz" (required even for 2D model)
     inwidth        =         ''       #  set new channel width e.g. "10MHz" (required
                                       #   even for 2D model)

complist            =         ''       #  componentlist to observe
setpointings        =       True
     integration    =      '10s'       #  integration (sampling) time
     direction      =         ''       #  "J2000 19h00m00 -40d00m00" or "" to center on
                                       #   model
     mapsize        =   ['', '']       #  angular size of map or "" to cover model
     maptype        =     'ALMA'       #  hexagonal, square (raster), ALMA, etc
     pointingspacing =        ''       #  spacing in between pointings or "0.25PB" or ""
                                       #   for ALMA default INT=lambda/D/sqrt(3),
                                       #   SD=lambda/D/3

obsmode             =      'int'       #  observation mode to simulate
                                       #   [int(interferometer)|sd(singledish)|""(none)]
     antennalist    = 'alma.out10.cfg' #  interferometer antenna position file
     refdate        = '2014/05/21'     #  date of observation - not critical unless
                                       #   concatting simulations
     hourangle      =  'transit'       #  hour angle of observation center e.g.
                                       #   "-3:00:00", "5h", "-4.5" (a number without
                                       #   units will be interpreted as hours), or
                                       #   "transit"
     totaltime      =    '7200s'       #  total time of observation or number of
                                       #   repetitions
     caldirection   =         ''       #  pt source calibrator [experimental]
     calflux        =      '1Jy'

outframe            =     'LSRK'       #  spectral frame of MS to create
thermalnoise        = 'tsys-atm'       #  add thermal noise: [tsys-atm|tsys-manual|""]
     user_pwv       =        0.5       #  Precipitable Water Vapor in mm
     t_ground       =      269.0       #  ambient temperature
     seed           =      11111       #  random number seed

leakage             =        0.0       #  cross polarization (interferometer only)
graphics            =     'both'       #  display graphics at each stage to
                                       #   [screen|file|both|none]

verbose             =      False
overwrite           =       True       #  overwrite files starting with $project
```

Simulating Observations

General Parameters

```
#  simobserve :: visibility simulation task
project            =  'hd10647'         #  root prefix for output file names
skymodel           = 'hd10647_model.fits' #  model image to observe
     inbright      =          ''        #  scale surface brightness of brightest pixel
                                        #   e.g. "1.2Jy/pixel"
     indirection   =          ''        #  set new direction e.g. "J2000 19h00m00
                                        #   -40d00m00"
     incell        =          ''        #  set new cell/pixel size e.g. "0.1arcsec"
     incenter      =          ''        #  set new frequency of center channel e.g.
                                        #   "89GHz" (required even for 2D model)
     inwidth       =          ''        #  set new channel width e.g. "10MHz" (required
                                        #   even for 2D model)

complist           =          ''        #  componentlist to observe
setpointings       =        True
     integration   =       '10s'        #  integration (sampling) time
     direction     =          ''        #  "J2000 19h00m00 -40d00m00" or "" to center on
                                        #   model
     mapsize       =    ['', '']        #  angular size of map or "" to cover model
     maptype       =      'ALMA'        #  hexagonal, square (raster), ALMA, etc
     pointingspacing =          ''      #  spacing in between pointings or "0.25PB" or ""
```

project : name of folder for simulation output

skymodel: input FITS image for simulation

incenter: center frequency for observations

inwidth: channel width (set to 7.5 GHz for continuum)

setpointings: sets up pointings for simulation, can also set to 'False' and provide a list of pointings (useful for setting up custom mosaics)

A Note on Pointings

Sometimes it can be helpful to define the pointings for your simulation using a pointing file:

Pointing RA

```
mmacgreg cfa0 7> more ptgfile_mosaic_new.txt
J2000   22h57m39.0462   -29d37m20.0530
J2000   22h57m39.6394   -29d37m40.0392
J2000   22h57m38.4529   -29d37m00.0668
J2000   22h57m38.3648   -29d37m12.6263
J2000   22h57m39.1343   -29d37m07.4872
J2000   22h57m38.9581   -29d37m30.9918
J2000   22h57m39.7276   -29d37m27.4797
mmacgreg cfa0 8>
```

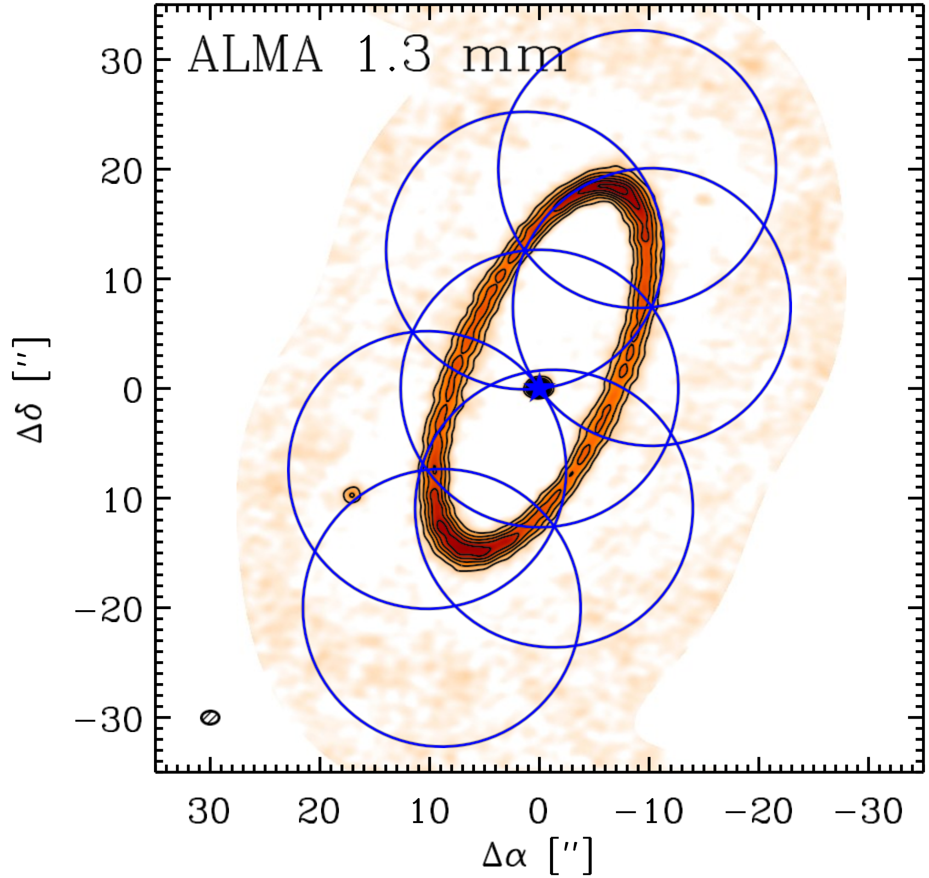
Pointing DEC

This is a must if you want to design a custom mosaic

A Note on Mosaics

Mosaics combine multiple pointings into a single image

If your target does not fit within the primary beam of the telescope, this may be the only way to image it



Simulating Observations

Observing Parameters

```
obsmode            =       'int'        #  observation mode to simulate [int(int
                                        #   erferometer)|sd(singledish)|""(none)
                                        #   ]
     antennalist   = 'alma.out10.cfg'   #  interferometer antenna position file
     refdate       = '2014/05/21'       #  date of observation - not critical
                                        #   unless concatting simulations
     hourangle     =  'transit'         #  hour angle of observation center e.g.
                                        #   "-3:00:00", "5h", "-4.5" (a number
                                        #   without units will be interpreted as
                                        #   hours), or "transit"
     totaltime     =    '7200s'         #  total time of observation or number
                                        #   of repetitions
     caldirection  =          ''        #  pt source calibrator [experimental]
     calflux       =      '1Jy'
```

antennalist : sample configuration file, full list of configuration files available here:

https://almascience.nrao.edu/tools/casa-simulator

refdate: date for simulation (not critical to change)

totaltime: total time for simulated observations

NRAO

Simulating Observations

Noise Parameters

```
thermalnoise        = 'tsys-atm'        #  add thermal noise: [tsys-atm|tsys-
                                        #   manual|""]
     user_pwv       =        0.5        #  Precipitable Water Vapor in mm
     t_ground       =      269.0        #  ambient temperature
     seed           =      11111        #  random number seed

leakage             =        0.0        #  cross polarization (interferometer
                                        #   only)
graphics            =     'both'        #  display graphics at each stage to
                                        #   [screen|file|both|none]
verbose             =      False
overwrite           =       True        #  overwrite files starting with
                                        #   $project
```

thermalnoise : thermal noise model to use, 'tsys-atm' uses a model for the ALMA site, 'tsys-manual' allows the user to specify the zenith sky brightness and opacity manually

user_pwv: precipitable water vapor in mm (usually take from sensitivity calculator)

NRAO

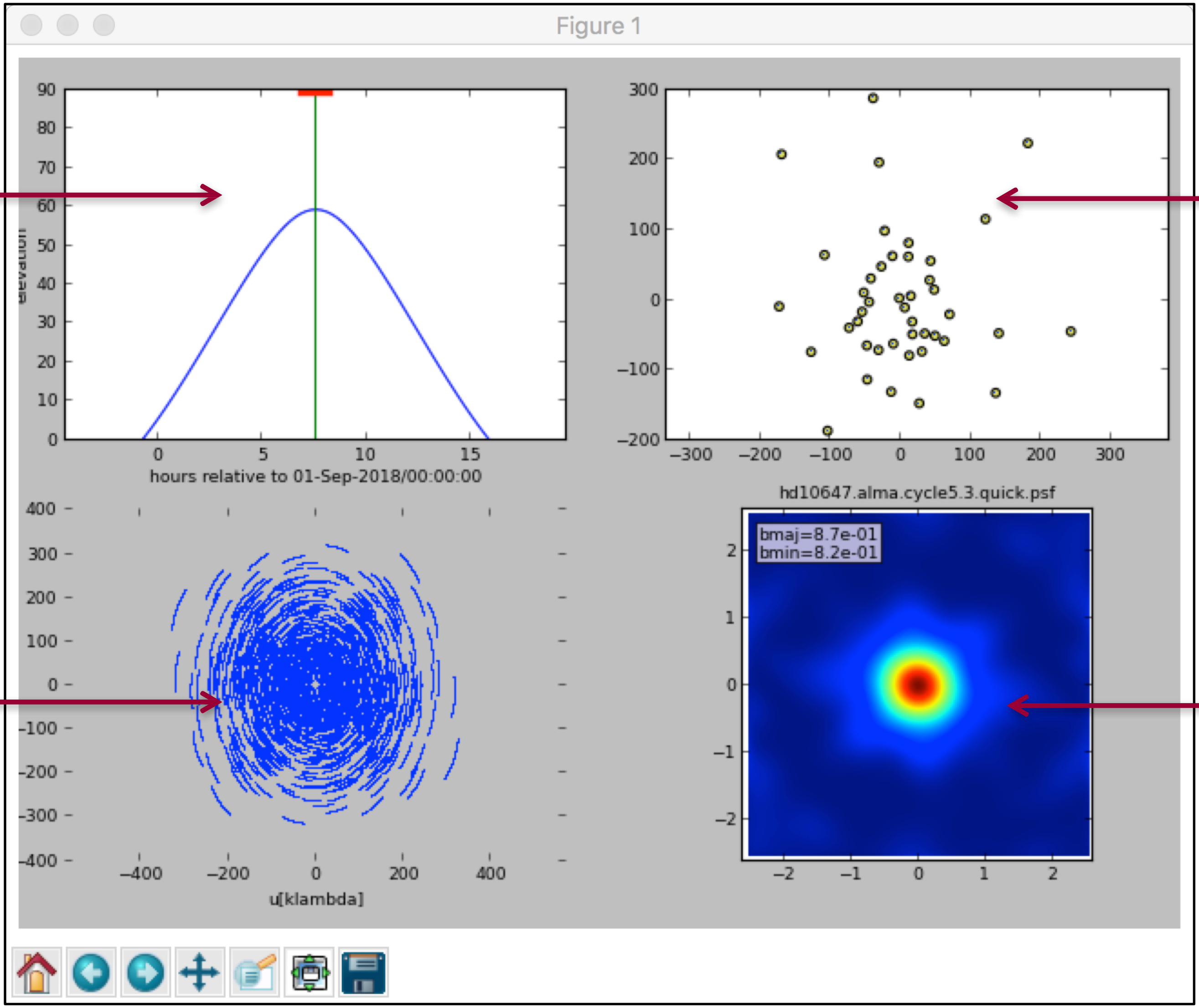
Running 'simobserve'

Elevation vs. LST for observations (in red)

Antenna positions

Observation UV coverage

Synthesized beam



Output from ‘simobserve’

```
CASA <34>: ls
hd10647.alma.cycle5.3.ms/              hd10647.alma.cycle5.3.simobserve.last
hd10647.alma.cycle5.3.noisy.ms/        hd10647.alma.cycle5.3.skymodel/
hd10647.alma.cycle5.3.observe.png      hd10647.alma.cycle5.3.skymodel.flat/
hd10647.alma.cycle5.3.ptg.txt          hd10647.alma.cycle5.3.skymodel.png
hd10647.alma.cycle5.3.quick.psf/
```

What do you do next?

Use ‘tclean’ to image the resulting visibilities –*OR*–
Use ‘simanalyze’ in CASA, which creates images using ‘clean’

Another approach to simulations is to use the ‘simalma’ task:
https://casaguides.nrao.edu/index.php/Simalma
But, ‘simobserve’ is more generalized and provides more capability (e.g. multiple configurations, pointings, etc.)

NRAO

Imaging in CASA

Imaging capabilities of'clean' have been refactored and improved in 'tclean' in the current version of CASA

The ALMA pipeline now uses 'tclean' instead of 'clean' for imaging

Major syntax changes are summarized here: https://casaguides.nrao.edu/index.php/TCLEAN_and_ALMA

All of the 'tclean' parameters in CASA

```
#  tclean :: Radio Interferometric Image Reconstruction
vis                 =          ''        #  Name of input visibility file(s)
selectdata          =       False        #  Enable data selection parameters
datacolumn          = 'corrected'        #  Data column to image(data,corrected)
imagename           =          ''        #  Pre-name of output images
imsize              =       [100]        #  Number of pixels
cell                = ['1arcsec']        #  Cell size
phasecenter         =          ''        #  Phase center of the image
stokes              =         'I'        #  Stokes Planes to make
projection          =       'SIN'        #  Coordinate projection (SIN, HPX)
startmodel          =          ''        #  Name of starting model image
specmode            =       'mfs'        #  Spectral definition mode
                                         #   (mfs,cube,cubedata)
     reffreq        =          ''        #  Reference frequency

gridder             =  'standard'        #  Gridding options (standard, wproject,
                                         #   widefield, mosaic, awproject)
     vptable        =          ''        #  Name of Voltage Pattern table
     pblimit        =         0.2        #  >PB gain level at which to cut off
                                         #   normalizations

deconvolver         =    'hogbom'        #  Minor cycle algorithm (hogbom,clark,m
                                         #   ultiscale,mtmfs,mem,clarkstokes)
restoration         =        True        #  Do restoration steps (or not)
     restoringbeam  =          []        #  Restoring beam shape to use. Default
                                         #   is the PSF main lobe
     pbcor          =       False        #  Apply PB correction on the output
                                         #   restored image

outlierfile         =          ''        #  Name of outlier-field image
                                         #   definitions
weighting           =   'natural'        #  Weighting scheme
                                         #   (natural,uniform,briggs)
     uvtaper        =          []        #  uv-taper on outer baselines in uv-
                                         #   plane

niter               =           0        #  Maximum number of iterations
usemask             =      'user'        #  Type of mask(s) for deconvolution
                                         #   (user, pb, auto-thresh, auto-
                                         #   thresh2, or auto-multithresh)
     mask           =          ''        #  Mask (a list of image name(s) or
                                         #   region file(s) or region string(s) )
     pbmask         =         0.0        #  primary beam mask

restart             =        True        #  True : Re-use existing images. False
                                         #   : Increment imagename
savemodel           =      'none'        #  Options to save model visibilities
                                         #   (none, virtual, modelcolumn)
calcres             =        True        #  Calculate initial residual image
calcpsf             =        True        #  Calculate PSF
parallel            =       False        #  Run major cycles in parallel
```

NRAO

Imaging in CASA

General Parameters

```
#  tclean :: Radio Interferometric Image Reconstruction
vis                 =         ''        #  Name of input visibility file(s)
selectdata          =      False        #  Enable data selection parameters
datacolumn          = 'corrected'       #  Data column to image(data,corrected)
imagename           =         ''        #  Pre-name of output images
imsize              =      [100]        #  Number of pixels
cell                = ['1arcsec']       #  Cell size
phasecenter         =         ''        #  Phase center of the image
stokes              =        'I'        #  Stokes Planes to make
projection          =      'SIN'        #  Coordinate projection (SIN, HPX)
startmodel          =         ''        #  Name of starting model image
```

vis : the name of the visibility file we will give you

selectdata: allows you to select a chunk of data to image

imagename: what you want your image to be called

imsize : size of image in pixels (cover your primary beam!)

cell: size of each pixel in arcsec (~5-8 pixels across)

Imaging in CASA

Key Clean Parameters

```
specmode        =      'mfs'        #  Spectral definition mode
                                    #   (mfs,cube,cubedata)
     reffreq    =         ''        #  Reference frequency

gridder         = 'standard'        #  Gridding options (standard, wproject,
                                    #   widefield, mosaic, awproject)
     vptable    =         ''        #  Name of Voltage Pattern table
     pblimit    =        0.2        #  >PB gain level at which to cut off
                                    #   normalizations

deconvolver     =   'hogbom'        #  Minor cycle algorithm (hogbom,clark,m
                                    #   ultiscale,mtmfs,mem,clarkstokes)
```

specmode: use 'mfs' for continuum images and 'channel/velocity/frequency' for spectral line imaging'

*For line imaging, you will also need to set the dimensions of the cube, rest frequency, velocity frame, and Doppler definition

gridder: 'standard' and 'mosaic' most common for ALMA

deconvolver: allows for different deconvolution options (hogbom, clark, mtmfs, multiscale, clarkstokes)

A Note on Continuum Subtraction

Usually, we want to subtract continuum emission prior to imaging line data

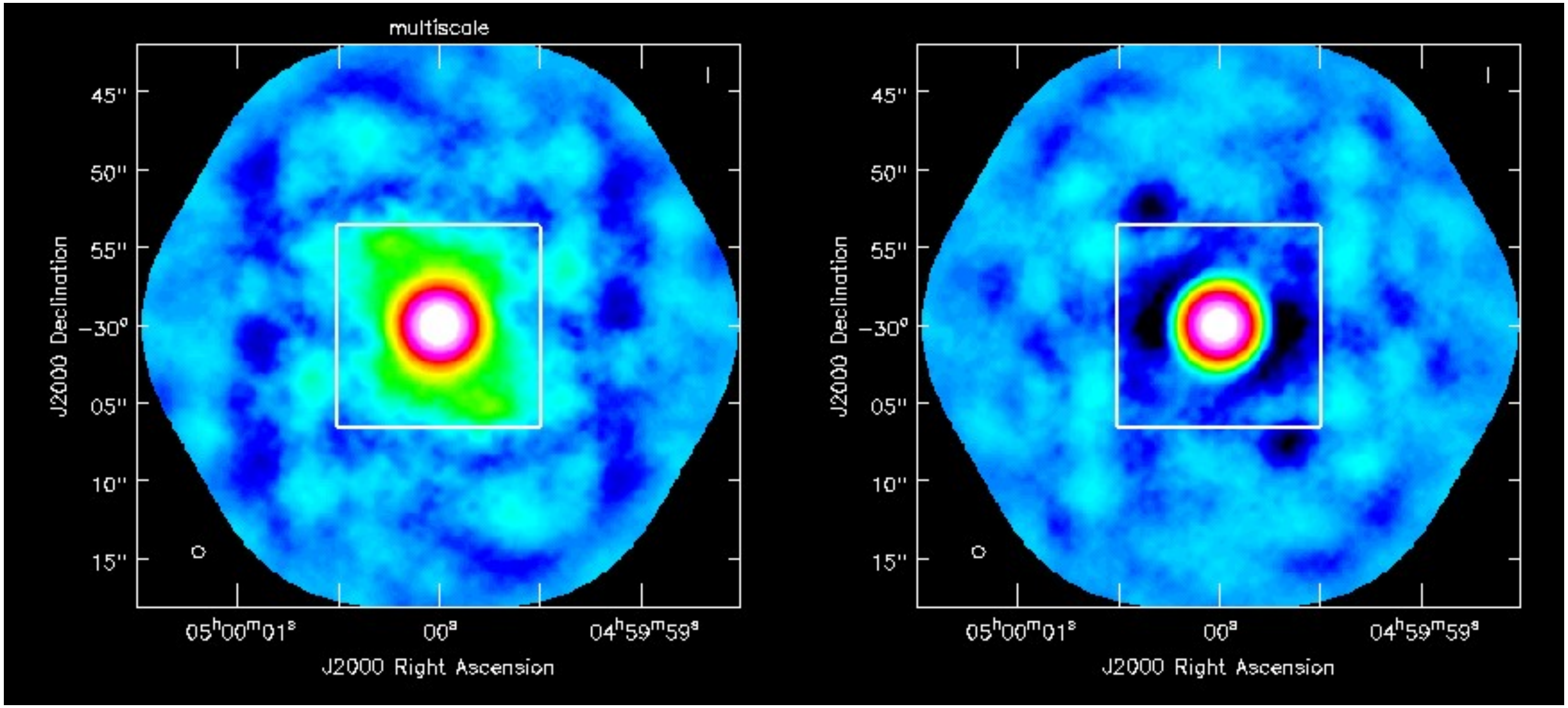
Use uvcontsub to do the subtraction in uv plane

```
CASA <11>: inp
---------> inp()
#  uvcontsub :: Continuum fitting and subtraction in the uv plane
vis                 = 'ngc3256_co.ms'   #  Name of input MS.  Output goes to vis + ".contsub"
field               =         ''        #  Select field(s) using id(s) or name(s)
fitspw              = '0:20~53;71~120'  #  Spectral window:channel selection for fitting the continuum
combine             =         ''        #  Data axes to combine for the continuum estimation (none, or spw and/or scan)
solint              =      'int'        #  Continuum fit timescale (int recommended!)
fitorder            =          0        #  Polynomial order for the fits
spw                 =         ''        #  Spectral window selection for output
want_cont           =      False        #  Create vis + ".cont" to hold the continuum estimate.
async               =      False        #  If true the taskname must be started using uvcontsub(...)
```

Can identify line-free channels by looking at the data with ‘plotms’ (more on this task later)

A Note on Multi-Scale

multi-scale "classic" scale



Uses extended clean components to better match emission scales unlike hogbom or clark, which use delta functions

Suggested scale parameter choice: (1) point source,(2) the size of the synthesized beam, and (3) 3-5 times the synthesized beam

Imaging in CASA

Weighting and Stopping

```
                                        #   definitions
weighting          =   'natural'        #  Weighting scheme
                                        #   (natural,uniform,briggs)
     uvtaper       =          []        #  uv-taper on outer baselines in uv-
                                        #   plane

niter              =         100        #  Maximum number of iterations
     gain          =         0.1        #  Loop gain
     threshold     =         0.0        #  Stopping threshold
     cycleniter    =          -1        #  Maximum number of minor-cycle
                                        #   iterations
     cyclefactor   =         1.0        #  Scaling on PSF sidelobe level to
                                        #   compute the minor-cycle stopping
                                        #   threshold.
     minpsffraction =       0.05        #  PSF fraction that marks the max depth
                                        #   of cleaning in the minor cycle
     maxpsffraction =        0.8        #  PSF fraction that marks the minimum
                                        #   depth of cleaning in the minor cycle
     interactive   =        True        #  Modify masks and parameters at
                                        #   runtime
```

weighting: natural, uniform or robust

uvtaper: apply Gaussian uv taper to visibilities

niter: number of iterations you want clean to do

threshold: flux level you want clean to stop at (~few times the noise)

interactive: run clean interactively

NRAO

A Note On Weighting

By weighting, you are multiplying your uv distribution, S(u,v), by a weighting function, W(u,v), and changing your dirty beam shape.

	Natural	Robust/Uniform	Taper
Resolution	Medium	**Higher**	Lower
Sidelobes	Higher	Lower	Depends
Point Source Sensitivity	**Maximum**	Lower	Lower
Extended Source Sensitivity	Medium	Lower	**Higher**

There are trade-offs with all weighting schemes. Make sure to start conservative and adjust to get the best image to achieve your particular science goals!

New! ‘tclean’ can be restarted

restart = True

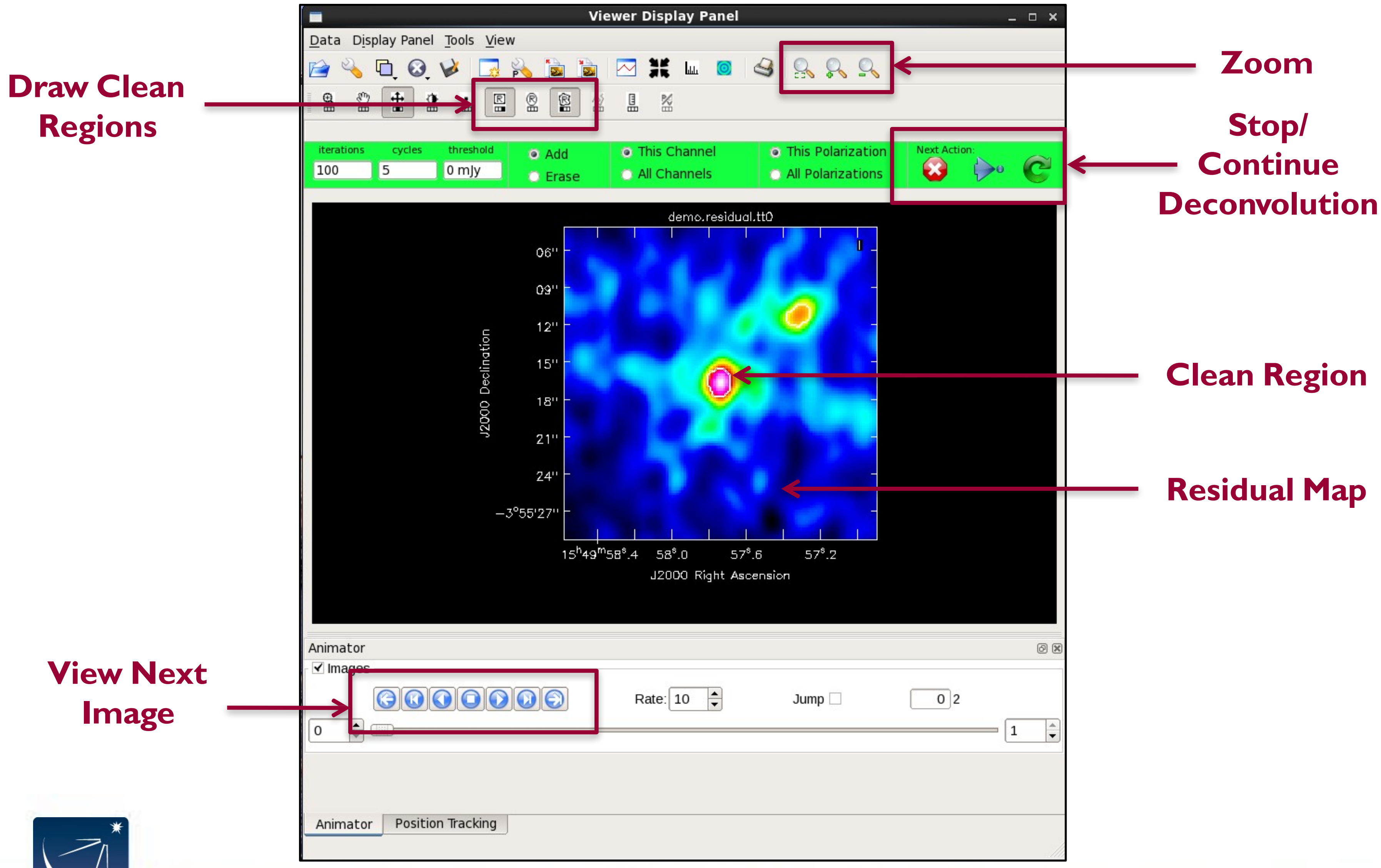
If ‘tclean’ is started again with same image name, it will try to continue deconvolution from where it left off. Make sure this is what you want. If not, give a new name or remove existing files with rmtables(‘my_image.*’).

restart=False

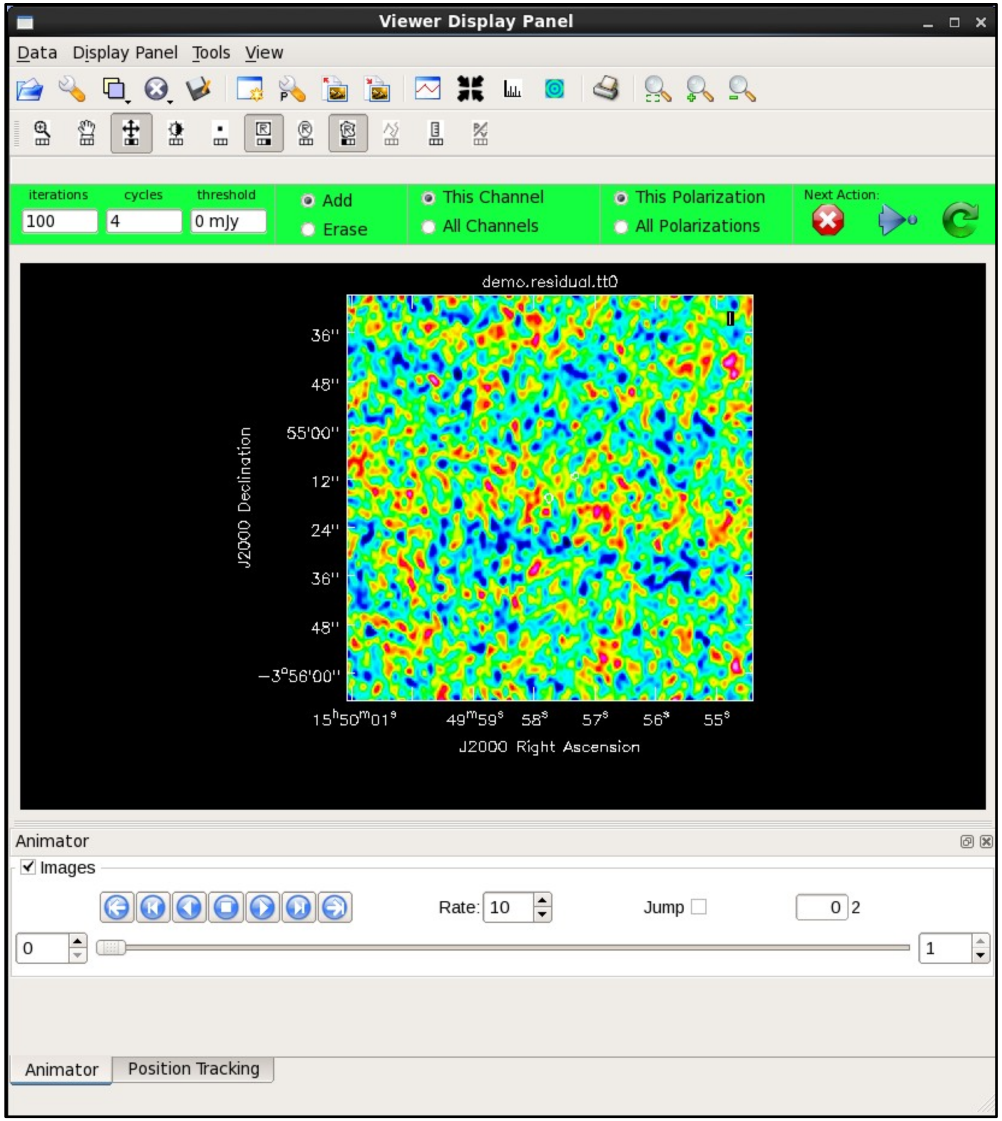
If ‘tclean’ is started again with same image name, it will increment the image name and then start the clean process from the beginning.

***Note: Try NOT to kill using CTRL+C as it could corrupt your measurement set**

Running 'tclean'



Running ‘tclean’



Residual map will look progressively more like noise

When you are happy that you’ve cleaned all significant flux, stop the deconvolution

Viewing Your Final Image

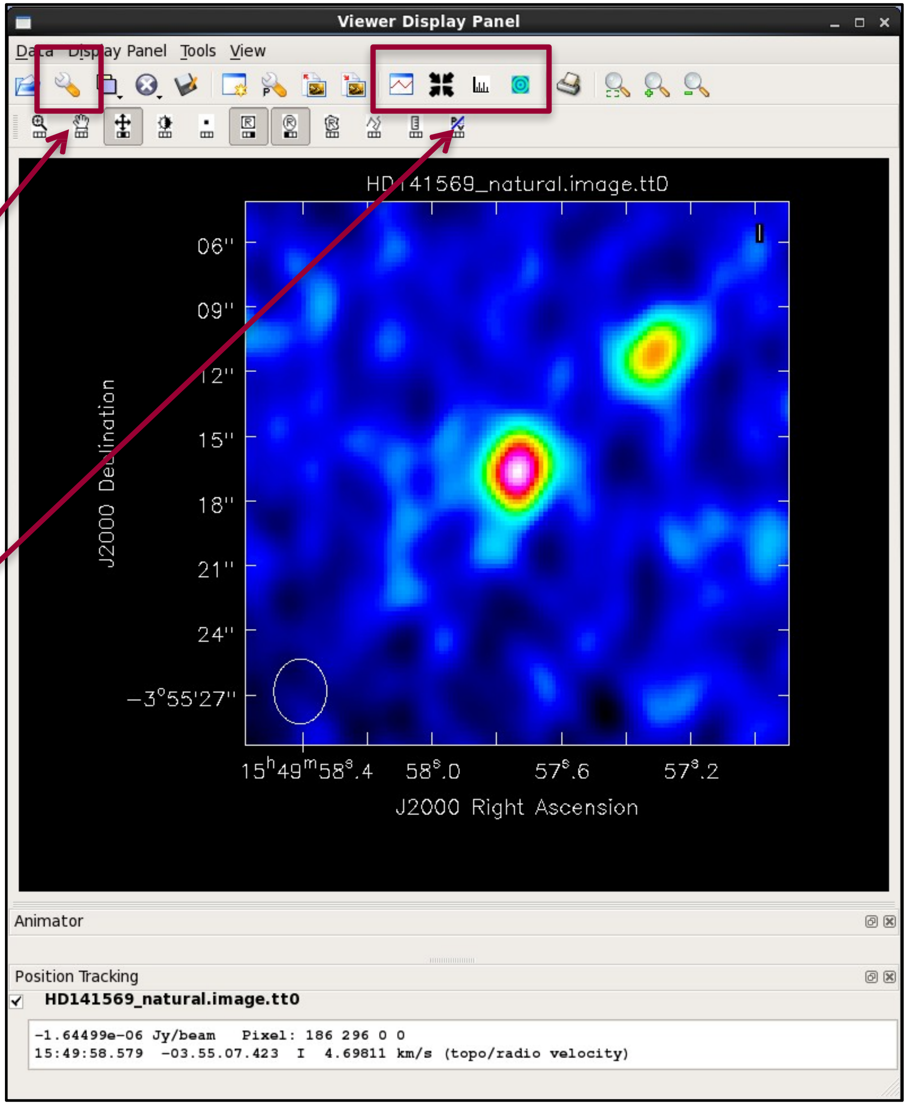
To view your deconvolved image type

viewer('<imagename>')

to open the CASA viewer window

Change image display (color scale, axes labels, etc.)

Make some plots (2D spectra, etc.)



Export Your Final Image

In order to work with your image outside of CASA, you'll need to export it as a FITS file using the **'exportfits'** task in CASA:

```
#  exportfits :: Convert a CASA image to a FITS file
imagename           = 'HD141569_natural.image.tt0' #  Name of input CASA image
fitsimage           =         ''        #  Name of output image FITS file
velocity            =      False        #  Use velocity (rather than frequency) as spectral axis
optical             =      False        #  Use the optical (rather than radio) velocity convention
bitpix              =        -32        #  Bits per pixel
minpix              =          0        #  Minimum pixel value (if minpix > maxpix, value is
                                        #   automatically determined)
maxpix              =         -1        #  Maximum pixel value (if minpix > maxpix, value is
                                        #   automatically determined)
overwrite           =       True        #  Overwrite pre-existing imagename
dropstokes          =      False        #  Drop the Stokes axis?
stokeslast          =       True        #  Put Stokes axis last in header?
history             =       True        #  Write history to the FITS image?
async               =      False        #  If true the taskname must be started using
                                        #   exportfits(...)
```

You can also export your CASA measurement set (the original visibilities) as a FITS file using the **'exportuvfits'** task:

```
#  exportuvfits :: Convert a CASA visibility data set to a UVFITS file:
vis                 = 'field3.ms'       #  Name of input visibility file
fitsfile            =         ''        #  Name of output UV FITS file
datacolumn          = 'corrected'       #  Visibility file data column
field               =         ''        #  Select field using field id(s) or field name(s)
spw                 =         ''        #  Select spectral window/channels
antenna             =         ''        #  Select data based on antenna/baseline
timerange           =         ''        #  Select data based on time range
avgchan             =          1        #  Channel averaging width (value > 1 indicates averaging)
writesyscal         =      False        #  Write GC and TY tables, (Not yet available)
multisource         =       True        #  Write in multi-source format
combinespw          =       True        #  Export the spectral windows as IFs
     padwithflags   =       True        #  Fill in missing data with flags to fit IFs

writestation        =       True        #  Write station name instead of antenna name
async               =      False        #  If true the taskname must be started using
                                        #   exportuvfits(...)
```

Other Useful Tasks

The task **'imstat'** gives you basic statistics on your image (rms noise, peak brightness, etc.):

```
CASA <33>: imstat('HD141569_natural.image.tt0')
  Out[33]:
{'blc': array([0, 0, 0, 0], dtype=int32),
 'blcf': '15:50:01.071, -03.56.06.622, I, 3.2999e+10Hz',
 'flux': array([ 0.00022282]),
 'max': array([  8.40296707e-05]),
 'maxpos': array([250, 250,   0,   0], dtype=int32),
 'maxposf': '15:49:57.730, -03.55.16.622, I, 3.2999e+10Hz',
 'mean': array([  1.85794922e-07]),
 'medabsdevmed': array([  3.51349240e-06]),
 'median': array([  1.20529819e-07]),
 'min': array([ -2.01637122e-05]),
 'minpos': array([270, 194,   0,   0], dtype=int32),
 'minposf': '15:49:57.463, -03.55.27.822, I, 3.2999e+10Hz',
 'npts': array([ 250000.]),
 'quartile': array([  7.02839679e-06]),
 'rms': array([  5.65317532e-06]),
 'sigma': array([  5.65013279e-06]),
 'sum': array([ 0.04644873]),
 'sumsq': array([  7.98959816e-06]),
 'trc': array([499, 499,   0,   0], dtype=int32),
 'trcf': '15:49:54.402, -03.54.26.822, I, 3.2999e+10Hz'}
```

Other Useful Tasks

The task **'immoments'** lets you compute moments for a line image:

```
#  immoments :: Compute moments from an image
imagename           =         ''        #  Name of the input image
moments             =        [0]        #  List of moments you would like to compute
axis                = 'spectral'        #  The momement axis: ra, dec, lat, long,
                                        #   spectral, or stokes
region              =         ''        #  Region selection. Default is to use the full
                                        #   image.
box                 =         ''        #  Rectangular region(s) to select in direction
                                        #   plane. Default is to use the entire direction
                                        #   plane.
chans               =         ''        #  Channels to use. Default is to use all
                                        #   channels.
stokes              =        'I'        #  Stokes planes to use. Default is to use all
                                        #   Stokes planes.
mask                =         []        #  Mask to use. Default is none.
     stretch        =      False        #  Stretch the mask if necessary and possible?

includepix          =         -1        #  Range of pixel values to include
excludepix          =         -1        #  Range of pixel values to exclude
outfile             =         ''        #  Output image file name (or root for multiple
                                        #   moments)
```

moments=0 integrated value of the spectrum
moments=1 intensity weighted coordinate ('velocity fields')
moments=2 intensity weighted dispersion of the coordinate ('velocity dispersion')

Other Useful Tasks

The task **'plotms'** gives you a powerful plotting tool to examine your visibility set. NRAO even has a handy help page on it:

http://casa.nrao.edu/stable/docs/UserMan/UserMansu117.html

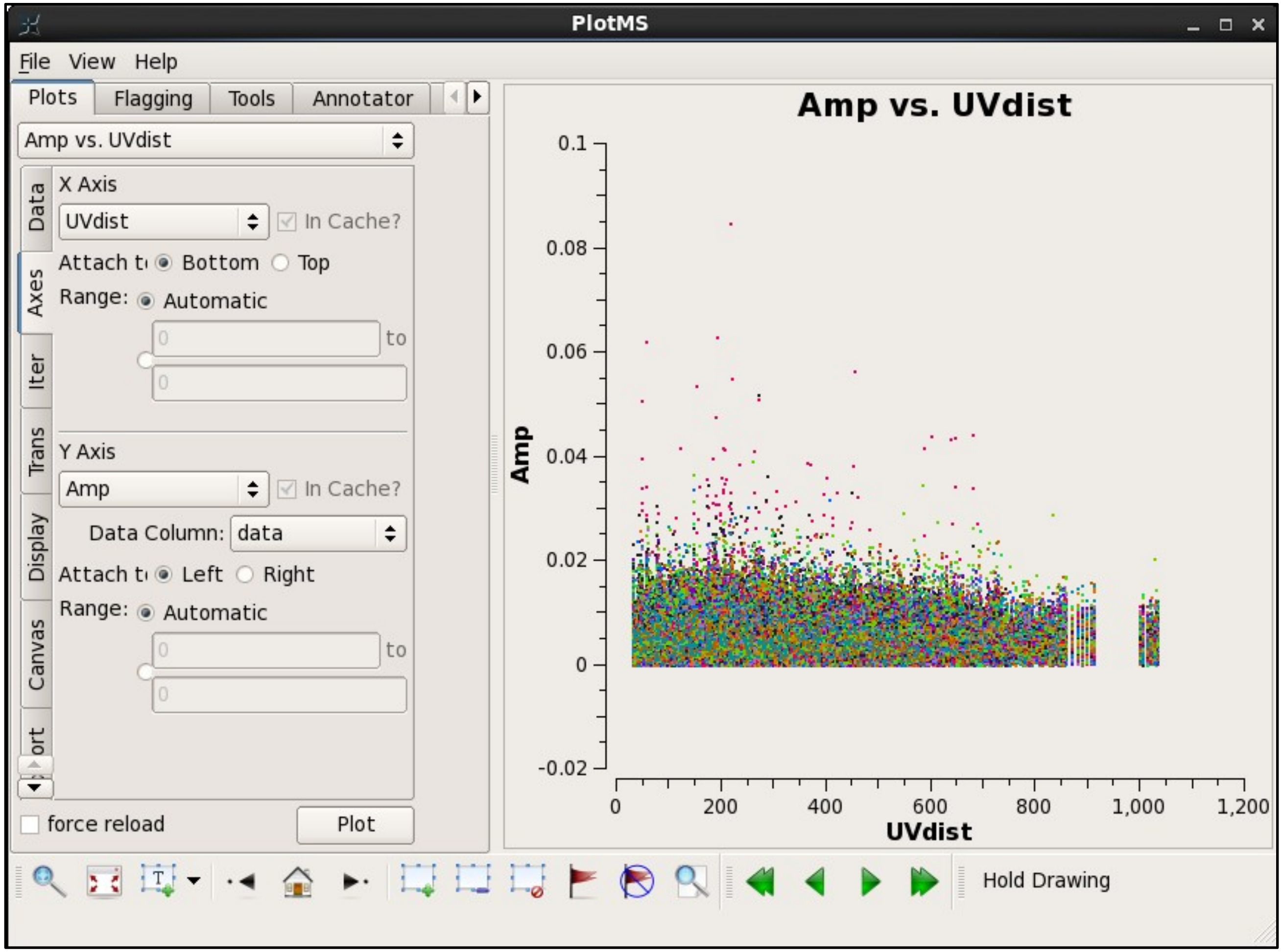
```
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 = 'field3.ms'       #  input MS (or CalTable) (blank for none)
xaxis               =    'uvwave'       #  plot x-axis (blank for default/current)
yaxis               =       'amp'       #  plot y-axis (blank for default/current)
     ydatacolumn    = 'corrected'       #  data column to use for y-axis (blank for default/current)

selectdata          =        True       #  data selection parameters
     field          =         '3'       #  field names or field index numbers (blank for all)
     spw            =          ''       #  spectral windows:channels (blank for all)
     timerange      =          ''       #  time range (blank for all)
     uvrange        =          ''       #  uv range (blank for all)
     antenna        =          ''       #  antenna/baselines (blank for all)
     scan           =          ''       #  scan numbers (blank for all)
     correlation    =     'LL,RR'       #  correlations (blank for all)
     array          =          ''       #  (sub)array numbers (blank for all)
     observation    =          ''       #  Select by observation ID(s)
     msselect       =          ''       #  MS selection (blank for all)
```

(And many other possible parameters…)

Other Useful Tasks

Example: Plot of amplitude vs. uv distance (in meters)



NRAO

A Few More Notes:

1. CASA routines can also be run as Python functions:

tclean(vis : "HD10647.alma.cycle5.3.noisy.ms", **imagename=**"HD10647_natural",
imsize=250, **cell=**"0.1arcsec"**, specmode=**"mfs", **gridder=**"standard",
deconvolver="hogbom"**, weighting=**"natural", **niter=**1000,
threshold="0.042mJy", **interactive=**True**)**

2. You can create scripts with sets of commands like the one above and run through them all automatically in CASA:

execfile("script.py") **or** **casa -c script.py**

NRAO

A Few Things to Remember

1. Use your online resources. There are lots of them!
2. While CASA is a very powerful tool, it can also be very buggy. Don't get frustrated. Try quitting and restarting.
3. The tasks you are more likely to use are…

   simobserve: simulating observations
   tclean: deconvolving your image
   viewer: displaying your image
   exportfits: exporting your image
   exportuvfits: exporting your visibilities
   plotms: examining your visibilities

4. CASA is built in iPython, so use your python intuition. Variables and lists are all defined using Python syntax.

Check out the hands-on activity later this afternoon to try all of this yourself!

NRAO

www.nrao.edu
science.nrao.edu

NRAO