An Introduction to the ALMA Simulations

Chelsea Sharon

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Very Long Baseline Array

CASA Simulations

- CASA can take any input image, smooth it, change it's location/resolution, and make mock observations+images
- Helps demonstrate to the TAC that the observations are feasible, they will achieve desired results, and you have expertise in dealing with radio data
- We'll be (mostly) following a CASA guide on simulation data for a star-forming region

CASA Guides

log in

page | discussion | view source | history

- Main Page

search

Search

Go | Search

tools

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Page information

Welcome to CASA Guides

CASA (Common Astronomy Software Applications) is a comprehensive softwar radioastronomical data VLA, both shown bel wiki provides examples

CASA News

- 23 March 2017: CASA Release 4.7.2 is now available
- 11 October 2016: CASA Newsletter #4

Events

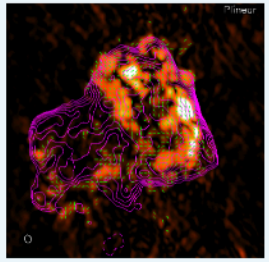
- 16 March 2017: ALMA Community Day Event at the University of Florida (Gainesville, FL)
- 28/29 March 2017: ALMA Community Day

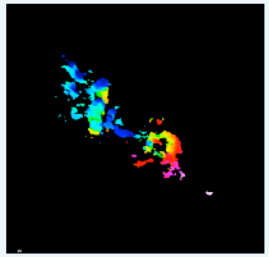
Using CASA

- CASA Basics
  - CASA Homepage Information on the latest releases, documentation, and support
  - CASA mailing lists Please subscribe to receive information on releases, critical bugs, etc.
  - Installing CASA Where to obtain CASA, and how to install it in different operating systems
  - Overviews
    - Guide to CASA syntax, task execution, and scripting
    - CASA calibration, imaging, and a description of basic tasks
    - CASA Python Overview Includes basics of python, and guides to arrays and plotting
- CASA Documentation
  - CASA Reference Manual & Cookbook HTML and the PDF Version
  - CASA Task Reference
  - CASA Toolkit Manual

CASA Tutorials

- ALMA Guides/Tutorials
- Karl G. Jansky VLA Tutorials
- Simulating Observations
- pre-upgrade VLA Tutorials
- ATCA Tutorials
- CARMA Tutorials
- SMA Tutorials
- Extracting Scripts from Tutorials





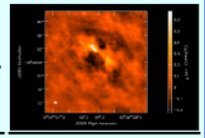
https://casaguides.nrao.edu

Simulations in the CASA Guides

- Scroll down to “A Tutorial for Simulating ALMA Data”
- Click “Simulation Examples in CASA 4.3”
- Scroll down to “Tutorials”

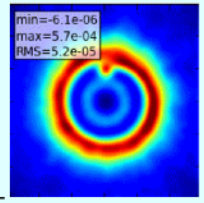
Simulation Guide for New Users (CASA 4.3)

A fully annotated tutorial that uses a Spitzer SAGE 8 micron continuum image of 30 Doradus and scales it to greater distance. A good place for new users to start.



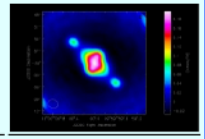
Protoplanetary Disk Simulation (CASA 4.3)

A sky model with a lightly annotated script that simulates a protoplanetary disk. Uses a theoretical model of dust continuum from Sebastian Wolff, scaled to the distance of a nearby star. This is another fairly generic simulation - if you're short on time, you probably don't need to go through this one and the New Users guide, but it can be useful to go through multiple examples.



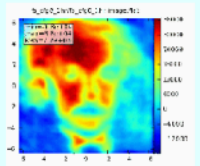
Simulation Guide Component Lists (CASA 4.3)

Tutorial for simulating data based on multiple sources (using both a FITS image and a component list). If you are interested in simulating from a list of simple sources (point, Gaussian, disk), rather than or in addition to a sky model image, then read the considerations here.



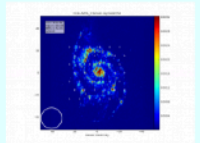
Einstein-Face (CASA 4.3)

A sky model and lightly annotated script that simulates the face of Einstein as seen by ALMA. This simulation is particularly useful for those who wish to better understand spatial filtering by an interferometer, but doesn't demonstrate new capabilities of the simulation tasks beyond those described above.



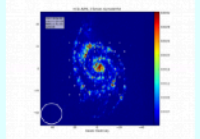
ACA Simulation (CASA 4.3)

A tutorial for simulating ALMA observations that use multiple configurations or use the 12-meter array in combination with the ALMA Compact Array. This tutorial demonstrates combining data from each ALMA component "by hand". This guide is of particular interest to those wishing to explore using the 12-m array in combination with the ACA, and those interested in combining data from multiple 12-m array configurations.



Simalma (CASA 4.3)

This tutorial demonstrates how to use **simalma**, a task that simplifies simulations that include the main 12-m array plus the ACA. Like the previous guide, this one is of particular interest to those wishing to explore multi-component ALMA observations.



Quick Check!

- Do you have CASA installed?
  - https://casa.nrao.edu/casa_obtaining.shtml
- Can you run it?
  - Within working directory, command line: casapy
- Do you have a model image to use?
  - Fits is preferable, but can convert jpeg etc. (https://casaguides.nrao.edu/index.php/Convert_jpg_to_fits)
  - Tutorial has link to file in "Getting Started"
- Do you have the ALMA configuration files?
  - If you have newest CASA or ran !update-data these may already exist within your CASA installation (https://almascience.nrao.edu/tools/casa-simulator)

Start CASA!

```
CASA <2>: tasklist
--------> tasklist()
Available tasks, organized by category (experimental tasks in parenthesis ()
  deprecated tasks in curly brackets {}).

Import/export        Information          Editing
------------------   ------------------   ---------------
exportasdm           imhead               fixplanets
exportfits           imreframe            fixvis
exportuvfits         imstat               flagcmd
importasdm           imval                flagdata
importatca           listcal              flagmanager
importfits           listfits             msview
importfitsidi        listhistory          plotms
importmiriad         listobs
importuvfits         listpartition
importula            listuis
```

```
Simulation
------------------
simanalyze
simobserve
(simalma)
```

- simobserve: creates mock uv data for input image
- simanalyze: images that data and creates useful diagnostic plots
- simalma: combines the two, particularly useful for combining 12m+7m+TP array data, but still in progress

simobserve parameters

```
CASA <5>: inp simobserve
--------> inp(simobserve)
#  simobserve :: visibility simulation task
project            =       'sim'        #  root prefix for output file names
skymodel           =         ''         #  model image to observe
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'

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
```

inp is not exhaustive. help(simobserve) is. So is Cookbook.

simobserve parameters: output

- **project** = root name for output files:
  - project.[cfg].skymodel = 4D input sky model image (optionally) scaled.
  - project.[cfg].skymodel.flat.regrid.conv = Input sky regridded to match the output image, and convolved with the output clean beam (PSF).
  - project.[cfg].skymodel.png = Diagnostic figure of sky model with pointings.
  - project.[cfg].ptg.txt = List of mosaic pointings.
  - project.[cfg].quick.psf = PSF calculated from uv coverage
  - project.[cfg].ms = Noise-free measurement set.
  - project.[cfg].noisy.ms = Corrupted measurement set.
  - project.[cfg].observe.png = Diagnostic figure of uv coverage and visibilities .
  - project.[cfg].simobserve.last = Saved input parameters for simobserve task.

simobserve parameters: model

- **skymodel**: input model image, which can be re-scaled using:
  - **inbright** = Peak brightness, assumes Jy/pixel always (e.g. “3.2mJy/pixel”)
  - **indirection** = Map center (e.g. “J2000 19h00m00 -40d00m00”)
  - **incell** = Spatial pixel size (e.g. “0.1arcsec”)
  - **incenter** = Frequency of center channel (e.g. “89GHz")
  - **inwidth** = Width of channels in frequency units (e.g. “10MHz")

simobserve parameters: components

```
CASA <5>: inp simobserve
--------> inp(simobserve)
#  simobserve :: visibility simulation task
project            =      'sim'        #  root prefix for output file names
skymodel           =        ''         #  model image to observe
complist           =        ''         #  componentlist to observe
setpointings       =      True
     integration   =      '10s'        #  integration (sampling) time
     direction     =        ''         #  "J2000 19h00m00 -40d00m00" or "" to
     mapsize       =   ['', '']        #  angular size of map or "" to cover
     maptype       =     'ALMA'        #  hexagonal, square (raster), ALMA, e
     pointingspacing =       ''        #  spacing in between pointings or "0.                    (3),
                                       #   SD=lambda/D/3

obsmode            =      'int'        #  observation mode to simulate [int(i
     antennalist   = 'alma.out10.cfg'  #  interferometer antenna position fil
     refdate       = '2014/05/21'      #  date of observation - not critical
     hourangle     =  'transit'        #  hour angle of observation center e.                    units
                                       #   will be interpreted as hours), or
     totaltime     =    '7200s'        #  total time of observation or number
     caldirection  =        ''         #  pt source calibrator [experimental]
     calflux       =      '1Jy'

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
```

complist & compwidth: For generating a sky model from a list of points/Gaussians (see: Simulation Guide Components List at https://casaguides.nrao.edu/index.php/Simulation_Guide_Component_Lists_(CASA_4.3))

simobserve parameters: mosaics

- **setpointings** = calculate a map of pointings; if false, provide ptgfile parameter (see “help”)
- **integration** = Time interval for each integration (e.g “10s”). NOT the total time on source. 10s is realistic, but longer will speed things up.
- **direction** = Mosaic center direction. Defaults to input image center. Can also be a list of pointings.
- **mapsize** = Angular size of map. Defaults to span model image.
- **maptype** = Sets pattern for mosaic if not specified elsewhere.
- **pointingspacing** = Spacing in between pointings (e.g “1arcsec”, "0.25PB" to use 1/4 of the primary beam FWHM, “nyquist”, etc.).

simobserve parameters: observations

- **obsmode** ="int" for interferometers or "sd" for singledish.
- **refdate** = Date of simulated observation (e.g. "yyyy/mm/dd")
- **hourangle** = Hour angle of observation (e.g. '-3h')
- **totaltime** = Total time of on source (e.g "7200s")
- **antennalist** = Path to ascii file containing antenna positions. If you have newest CASA version and ran !update-data, then should be in your CASA installation (os.getenv("CASAPATH").split()[0]+"/data/alma/simmos/alma.cycle5.N.cfg")
- **caldirection** = An unresolved calibrator (with flux set by cal flux parameter) that can be observed interleaved with the science pointings.
- **sdant** = The index of the antenna in the list to use for total power. Defaults to the first antenna in the list.

simobserve parameters: noise

- **thermalnoise** = “tsys-atm” for a model of the ALMA site atmosphere, or “tsys-manual” to specify the zenith sky brightness and opacity manually (using parameter t-sky and tau0).
- **t_ground** = Ground/spillover temperature in K.
- **user_pwv** = Precipitable water vapor if constructing an atmospheric model.
- **seed** = Random number seed for noise generation.

simobserve parameters

```
CASA <5>: inp simobserve
--------> inp(simobserve)
#  simobserve :: visibility simulation task
project            =       'sim'        #  root prefix for output file names
skymodel           =         ''         #  model image to observe
complist           =         ''         #  componentlist to observe
setpointings       =       True
     integration   =       '10s'        #  integration (sampling) time
     direction     =         ''         #  "J2000 19h00m00 -40d00m00" or "" to
     mapsize       =   ['', '']         #  angular size of map or "" to cover
     maptype       =      'ALMA'        #  hexagonal, square (raster), ALMA, e
     pointingspacing =        ''        #  spacing in between pointings or "0.                (3),
                                        #   SD=lambda/D/3

obsmode            =       'int'        #  observation mode to simulate [int(i
     antennalist   = 'alma.out10.cfg'   #  interferometer antenna position fil
     refdate       = '2014/05/21'       #  date of observation - not critical
     hourangle     =  'transit'         #  hour angle of observation center e.                units
                                        #   will be interpreted as hours), or
     totaltime     =     '7200s'        #  total time of observation or number
     caldirection  =         ''         #  pt source calibrator [experimental]
     calflux       =       '1Jy'

thermalnoise       = 'tsys-atm'         #  add thermal noise: [tsys-atm|tsys-m
     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
graphics           =       'both'       #  display graphics at each stage to [
verbose            =       False
overwrite          =        True        #  overwrite files starting with $project
```

leakage = Add cross polarization corruption of this fractional magnitude.

graphics = View plots on the screen, saved to file, both, or neither.

verbose = Print extra information to the logger and terminal.

overwrite = Overwrite existing files in the project subdirectory.

Run simobserve

- Say we want to observe a star-forming region in a nearby galaxy in 230GHz dust continuum, and have reason to believe it will look something like 30 Doradus
- We need to:
  - Scale and reposition the image so it's the right angular scale, on the right part of the sky, and at the right frequency
  - Choose appropriate mosaic setups
  - Set up your observation's integration time, resolution, etc.
  - Give it some noise

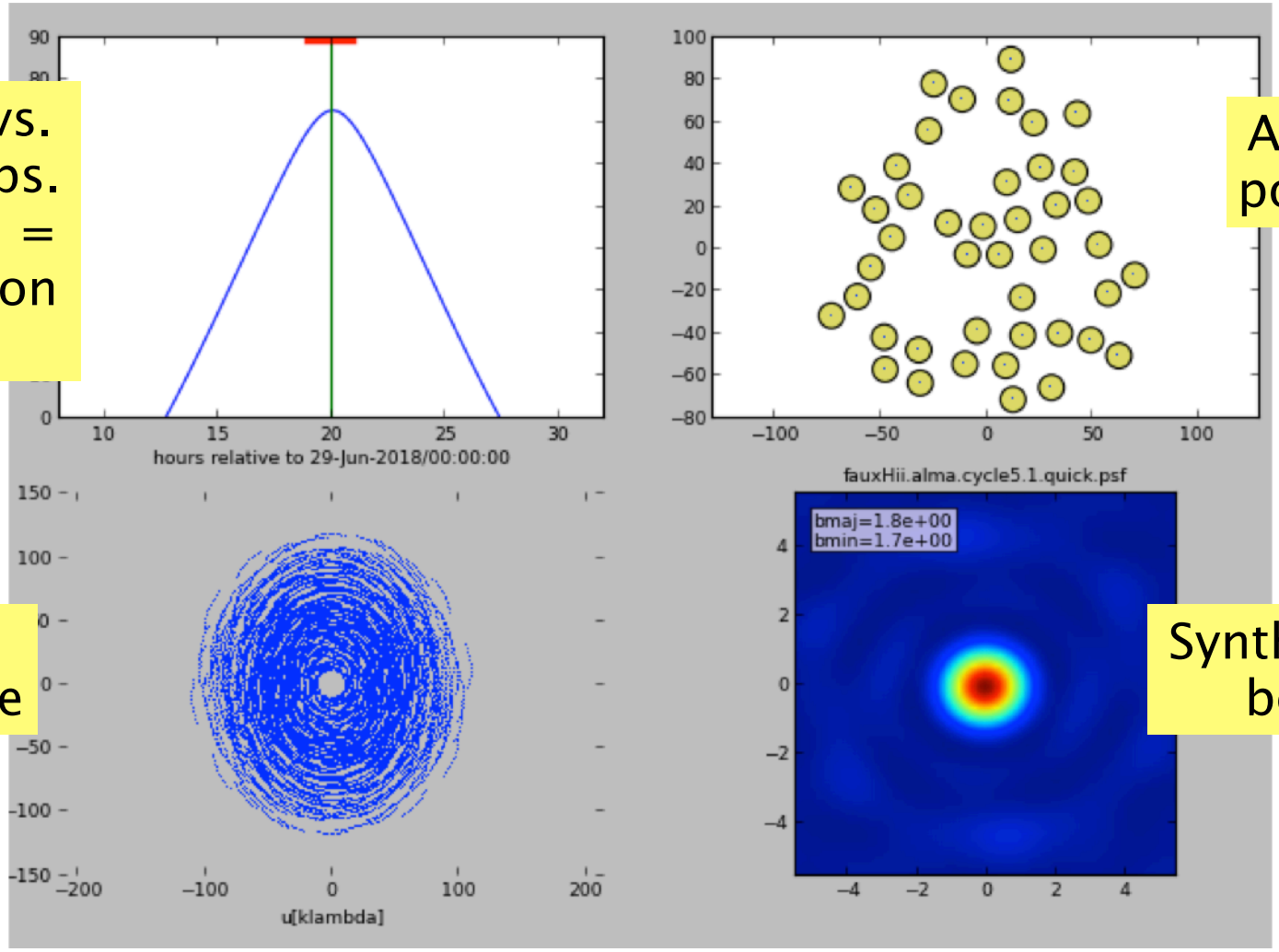
Run simobserve

- skymodel='30dor.fits'
- project='fauxHii'
- incell='0.15arcsec'
- indirection='J2000 10h00m00 -40d00m00'
- incenter='230GHz'
- inwidth='2GHz'
- inbright='0.06mJy/pixel'
- integration='600s'
- obsmode='int'
- refdate='2018/06/30'
- totaltime='7200s'
- antennalist=os.getenv("CASAPATH").split()[0]+"/data/alma/simmos/alma.cycle5.1.cfg"
- thermalnoise='tsys-atm'
- seed=1234

```
CASA <6>: simobserve(skymodel='30dor.fits',project='fauxHii',incell='0.15arcsec',indirection='J2000 10h00m00 -40d00m00',incenter='
230GHz',inwidth='2GHz',inbright='0.06mJy/pixel',integration='600s',obsmode='int',refdate='2018/06/30',totaltime='7200s',antennalis
t=os.getenv("CASAPATH").split()[0]+"/data/alma/simmos/alma.cycle5.1.cfg",thermalnoise='tsys-atm',seed=1234,overwrite=True)
```

simobserve output: plots

Altitude vs. LST for obs. date. Red = observation period.



Antenna positions

uv coverage

Synthesized beam

simobserve output: logger

- Spits out some information about the observations, the predicted beam shape, details about the atmospheric model, etc.

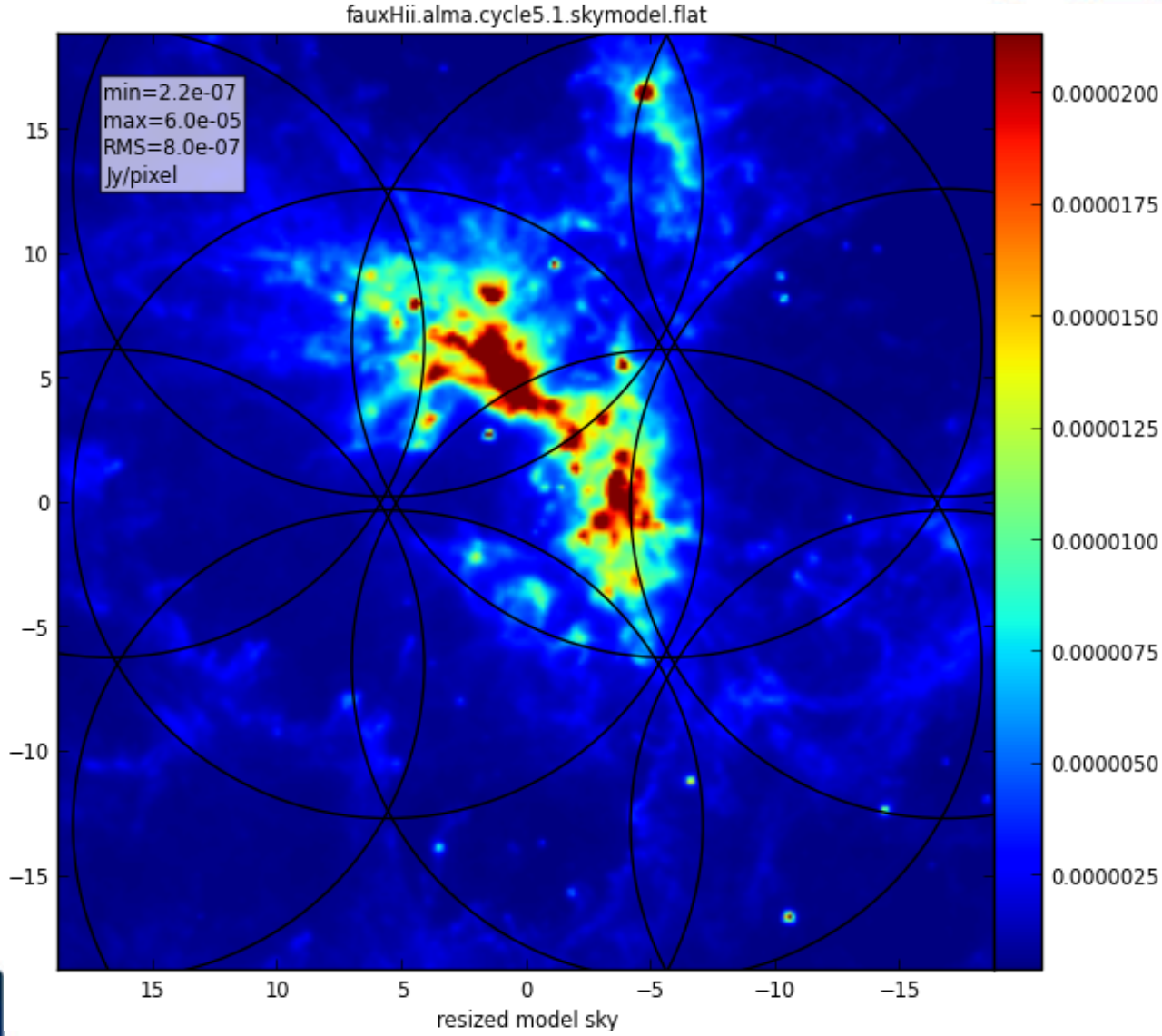
```
Beam fit: 1.83209 by 1.65421 (arcsec) at pa -88.0757 (deg)
```

```
INFOnoise::SVC::sizeUpSim()  For simint = integration, found 12 solution intervals.
INFOnoise::AtmCorr::initAtm  Initializing ATM
INFOnoise::AtmCorr::initAtm  altitude=5000m, Pground=560mb, Tground=269K, humidity= 20%, water scale height=2000m
INFOnoise::AtmCorr::initAtm  Spectral window 230(ch 10/20)
INFOnoise::AtmCorr::initAtm  After setting WH2O to 0.5
INFOnoise::AtmCorr::initAtm  Dry and Wet Opacity from RefractiveIndexProfile = 0.0112762, 0.021669 at 230 GHz (ch0)
INFOnoise::AtmCorr::initAtm  Dry and Wet Opacity from SkyStatus              = 0.0112762, 0.021669
INFOnoise::AtmCorr::initAtm  Sky plus ground and CMB Brightness Temp across Spw 0, for spill=0.95 Tground=269 pwv=0.5
INFOnoise::AtmCorr::initAtm   Zenith Tebb[229,230,230.9]=[25.8613,26.0682,26.9084]
```

simobserve output: files

- fauxHii.alma.cycle5.1.ms
- fauxHii.alma.cycle5.1.noisy.ms
- fauxHii.alma.cycle5.1.observe.png
- fauxHii.alma.cycle5.1.ptg.txt
- fauxHii.alma.cycle5.1.quick.psf
- fauxHii.alma.cycle5.1.simobserve.last
- fauxHii.alma.cycle5.1.skymodel
- fauxHii.alma.cycle5.1.skymodel.flat
- fauxHii.alma.cycle5.1.skymodel.png

simobserve output: files



Hmm… perhaps the default mosaic parameters aren't what I want?

simobserve output: listobs

Also, if you run listobs on the output measurement set,
listobs(vis='fauxHii/fauxHii.alma.cycle5.1.noisy.ms',verbose=True)
and look in the logger…

Date	Timerange (UTC)	Scan	FldId	FieldName
29-Jun-2018/	19:00:19.0 - 19:10:19.0	1	0	fauxHii.alma.cycle5*
	19:10:19.0 - 19:20:19.0	2	1	fauxHii.alma.cycle5*
	19:20:19.0 - 19:30:19.0	3	2	fauxHii.alma.cycle5*
	19:30:19.0 - 19:40:19.0	4	3	fauxHii.alma.cycle5*
	19:40:19.0 - 19:50:19.0	5	4	fauxHii.alma.cycle5*
	19:50:19.0 - 20:00:19.0	6	5	fauxHii.alma.cycle5*
	20:00:19.0 - 20:10:19.0	7	6	fauxHii.alma.cycle5*
	20:10:19.0 - 20:20:19.0	8	7	fauxHii.alma.cycle5*
	20:20:19.0 - 20:30:19.0	9	8	fauxHii.alma.cycle5*
	20:30:19.0 - 20:40:19.0	10	9	fauxHii.alma.cycle5*
	20:40:19.0 - 20:50:19.0	11	0	fauxHii.alma.cycle5*
	20:50:19.0 - 21:00:19.0	12	1	fauxHii.alma.cycle5*

Spent 600s on each pointing, but that didn't add up to 7200s, so 2 pointings are repeated…

simanalyze parameters

```
CASA <2>: inp(simanalyze)
#  simanalyze :: image and analyze measurement sets created with simobserve
project             =      'sim'        #  root prefix for output file names
image               =       True        #  (re)image $project.*.ms to $project.image
     vis            =  'default'        #  Measurement Set(s) to image
     modelimage     =         ''        #  lower resolution prior image to use in clean e.g. existing total power image
     imsize         =          0        #  output image size in pixels (x,y) or 0 to match model
     imdirection    =         ''        #  set output image direction, (otherwise center on the model)
     cell           =         ''        #  cell size with units e.g. "10arcsec" or "" to equal model
     interactive    =      False        #  interactive clean?  (make sure to set niter>0 also)
     niter          =          0        #  maximum number of iterations (0 for dirty image)
     threshold      =   '0.1mJy'        #  flux level (+units) to stop cleaning
     weighting      =  'natural'        #  weighting to apply to visibilities.  briggs will use robust=0.5
     mask           =         []        #  Cleanbox(es), mask image(s), region(s), or a level
     outertaper     =         []        #  uv-taper on outer baselines in uv-plane
     pbcor          =       True        #  correct the output of synthesis images for primary beam response?
     stokes         =        'I'        #  Stokes params to image
     featherimage   =         ''        #  image (e.g. total power) to feather with new image

analyze             =      False        #  (only first 6 selected outputs will be displayed)
graphics            =     'both'        #  display graphics at each stage to [screen|file|both|none]
verbose             =      False
overwrite           =       True        #  overwrite files starting with $project
dryrun              =      False        #  only print information [experimental; only for interfermetric data]
logfile             =         ''
```

Neither inp or help is exhaustive.
Cookbook is.

simanalyze parameters: I/O

- **project** = input and output files' root name:
  - project.[cfg].skymodel.flat.regrid.conv = Input model regridded to match the output image, convolved with the output beam.
  - project.[cfg].image = Synthesized image.
  - project.[cfg].flux.pbcoverage = Primary beam correction.
  - project.[cfg].residual = Residual image after cleaning.
  - project.[cfg].clean.last = Parameters used in the clean task.
  - project.[cfg].psf = Synthesized (dirty) beam from weighted uv distribution.
  - project.[cfg].image.png = Diagnostic figure of clean image and residual.
  - project.[cfg].fidelity = Fidelity image.
  - project.[cfg].analysis.png = Diagnostic figure of difference and fidelity.
  - project.[cfg].simanalyze.last = Saved input parameters for simanalyze task

simanalyze parameters: imaging

- **image** = False
  - I cleaned everything myself and just want the diagnostic plots (set **imagename** parameter)
- **image** = True
  - I want simanalyze to handle the imaging
  - **vis** = The output corrupted MS from simobserve.
  - But there are other imaging parameters you need to set…

simanalyze parameters: imaging

- Ignore: **modelimage**, **imsize**, **imdirection**, and **cell** since that will default to match your input model image.
- **interactive** = Whether you want to guide the clean regions and depth during imaging. Alternatively…
  - **niter** = Number of iterations in making your map.
  - **threshold** = The flux depth that your imaging should treat as real emission.
- **weighting** = Changes the resolution of your image by changing the weighting assigned to different baselines (use default of "natural"; see also **outertaper**).
- **mask** = The region where the imaging procedure (clean) should look for emission.
- **pbcor** = Applies primary beam response correction.
- Ignore **stokes** and **featherimage**.

simanalyze parameters: analysis

- **analysis** = Do analysis on the output images.
- **showuv** = Show the uv plane coverage.
- **showpsf** = Show the synthesized beam.
- **showmodel** = Show the model image.
- **showconvolved** = Show the model image convolved to the same resolution as the output.
- **showclean** = Show the final map.
- **showresidual** = Show difference between final map and the clean component image.
- **showdifference** = Show the difference between the final map and convolved model image.
- **showfidelity** = Show abs(input)/max(abs(input-output), 0.7*rms(output))

simanalyze parameters: output

```
CASA <2>: inp(simanalyze)
#  simanalyze :: image and analyze measurement sets created u
project            =      'sim'        #  root prefix for ou
image              =       True        #  (re)image $project
     vis           =  'default'        #  Measurement Set(s)
     modelimage    =         ''        #  lower resolution p
     imsize        =          0        #  output image size
     imdirection   =         ''        #  set output image c
     cell          =         ''        #  cell size with uni
     interactive   =      False        #  interactive clean?
     niter         =          0        #  maximum number of
     threshold     =   '0.1mJy'        #  flux level (+units
     weighting     =  'natural'        #  weighting to apply
     mask          =         []        #  Cleanbox(es), mask
     outertaper    =         []        #  uv-taper on outer
     pbcor         =       True        #  correct the output
     stokes        =        'I'        #  Stokes params to i
     featherimage  =         ''        #  image (e.g. total

analyze            =      False        #  (only first 6 sele
graphics           =     'both'        #  display graphics a
verbose            =      False
overwrite          =       True        #  overwrite files st
dryrun             =      False        #  only print informa
logfile            =         ''
```

graphics = View plots on the screen, saved to file, both, or neither.

verbose = Print extra information to the logger and terminal.

overwrite = Overwrite existing files in the project subdirectory (doesn't work for image products).

dryrun = Print the information without doing anything as a test (not for single dish).

logfile = Output the information to a different log file.

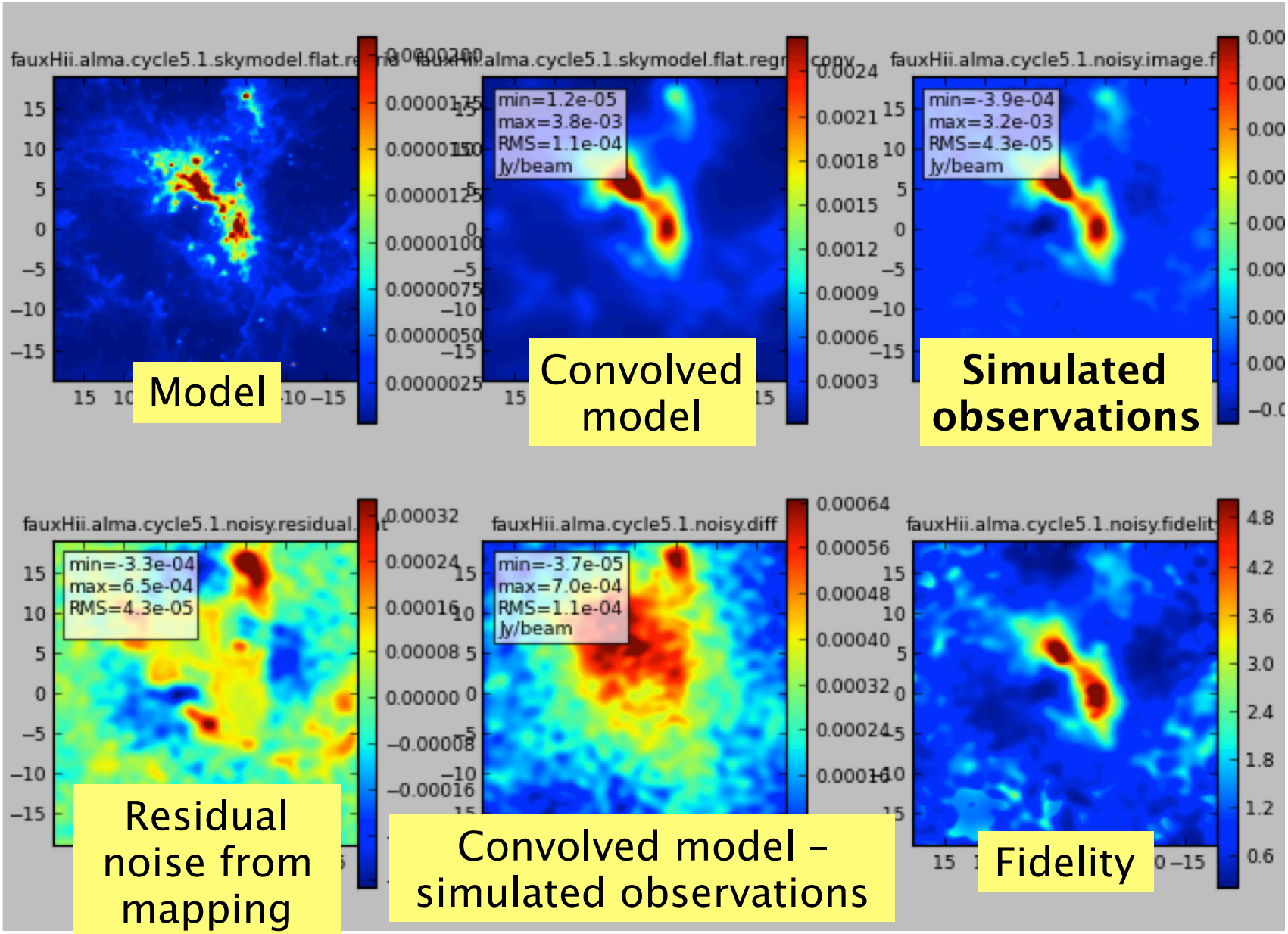
Run simanalyze

Let's map the source and do the analysis!

- **project**='fauxHii'
- **image**=True
- **vis**='fauxHii.alma.cycle5.1.noisy.ms'
- **interactive**=False
- **niter**=1000
- **threshold**='0.04mJy'
- **weighting**='natural'
- **mask**=[[80,135,135,195],[136,90,170,160]]
- **pbcor**=True
- **analyze**=True
- **graphics**='both'
- **showmodel**, **showconvolved**, **showclean**, **showresidual**, **showdifference**, **showfidelity** = True
- **showuv**, **showpsf** = False

```
CASA <12>: simanalyze(project='fauxHii',image=True,vis='fauxHii.alma.cycle5.1.noisy.ms',interactive=False,niter=1000,threshold='0.04mJy',weighting='natural',mask=[[80,135,135,195],[136,90,170,160]],pbcor=True,analyze=True,graphics='both',showmodel=True,showconvolved=True,showclean=True,showresidual=True,showdifference=True,showfidelity=True,showuv=False,showpsf=False,overwrite=True)
```

simanalyze output: plots



simanalyze output: logger

- Spits out some information about how the imaging went and some summary statistics…

```
…odel::solve  Finished Clark clean inner cycle
…odel::solve  Clean used 1000 iterations to approach a threshhold of 3.8e-05
…odel::solve  0.0071018 Jy <- cleaned in this cycle for  model 0 (Total flux : 0.0217233Jy)
…odel::solve  Final maximum residual = 0.000109691
…odel::solve  Model 0: max, min residuals = 0.000109691, 3.2521e-06 clean flux 0.0217233
…er::clean()  Threshhold not reached yet.
```

```
…:analysis::  Simulation rms: [  2.83637408e-07] Jy/pix = [  4.32568741e-05] Jy/bm
…:analysis::  Simulation max: [  2.06751380e-05] Jy/pix = [ 0.00315312] Jy/bm
…:analysis::  Beam bmaj: 1.83114099503 bmin: 1.65380430222 bpa: 92.8190002441
```

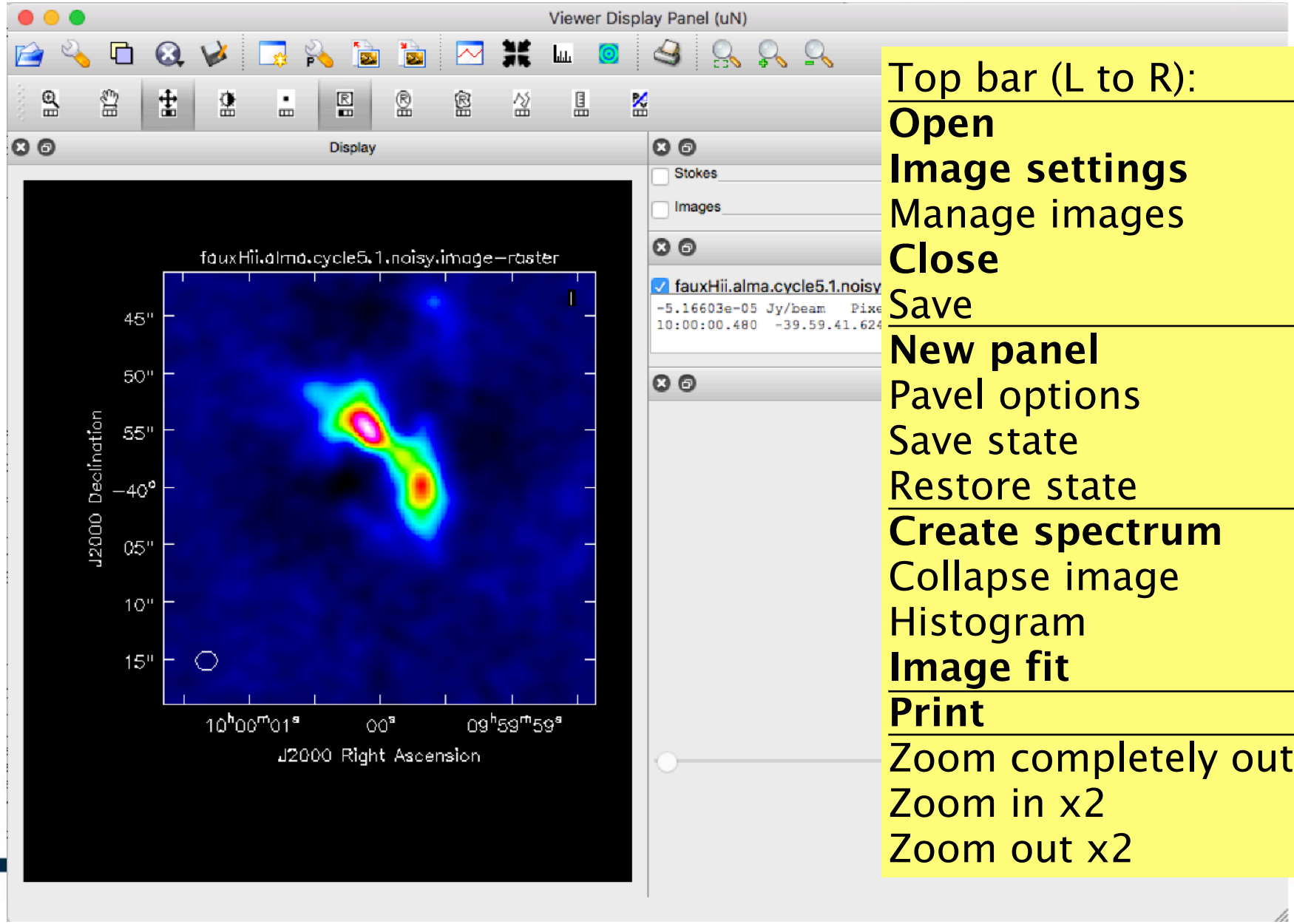
simobserve output: files

- fauxHii.alma.cycle5.1.noisy.analysis.png
- fauxHii.alma.cycle5.1.noisy.clean.last
- fauxHii.alma.cycle5.1.noisy.diff
- fauxHii.alma.cycle5.1.noisy.fidelity
- fauxHii.alma.cycle5.1.noisy.flux
- fauxHii.alma.cycle5.1.noisy.flux.pbcoverage
- fauxHii.alma.cycle5.1.noisy.image
- fauxHii.alma.cycle5.1.noisy.image.flat
- fauxHii.alma.cycle5.1.noisy.image.png
- fauxHii.alma.cycle5.1.noisy.mask
- fauxHii.alma.cycle5.1.noisy.model
- fauxHii.alma.cycle5.1.noisy.psf
- fauxHii.alma.cycle5.1.noisy.residual
- fauxHii.alma.cycle5.1.skymodel.flat.regrid
- fauxHii.alma.cycle5.1.skymodel.flat.regrid.conv
- fauxHii.simanalyze.last

Most of these are images you can pull up in the viewer.
In CASA run: viewer()

Note: No difference between flat and not-flat images (at least for pbcor=True).

simobserve output: viewer



Top bar (L to R):

Open
Image settings
Manage images
Close
Save

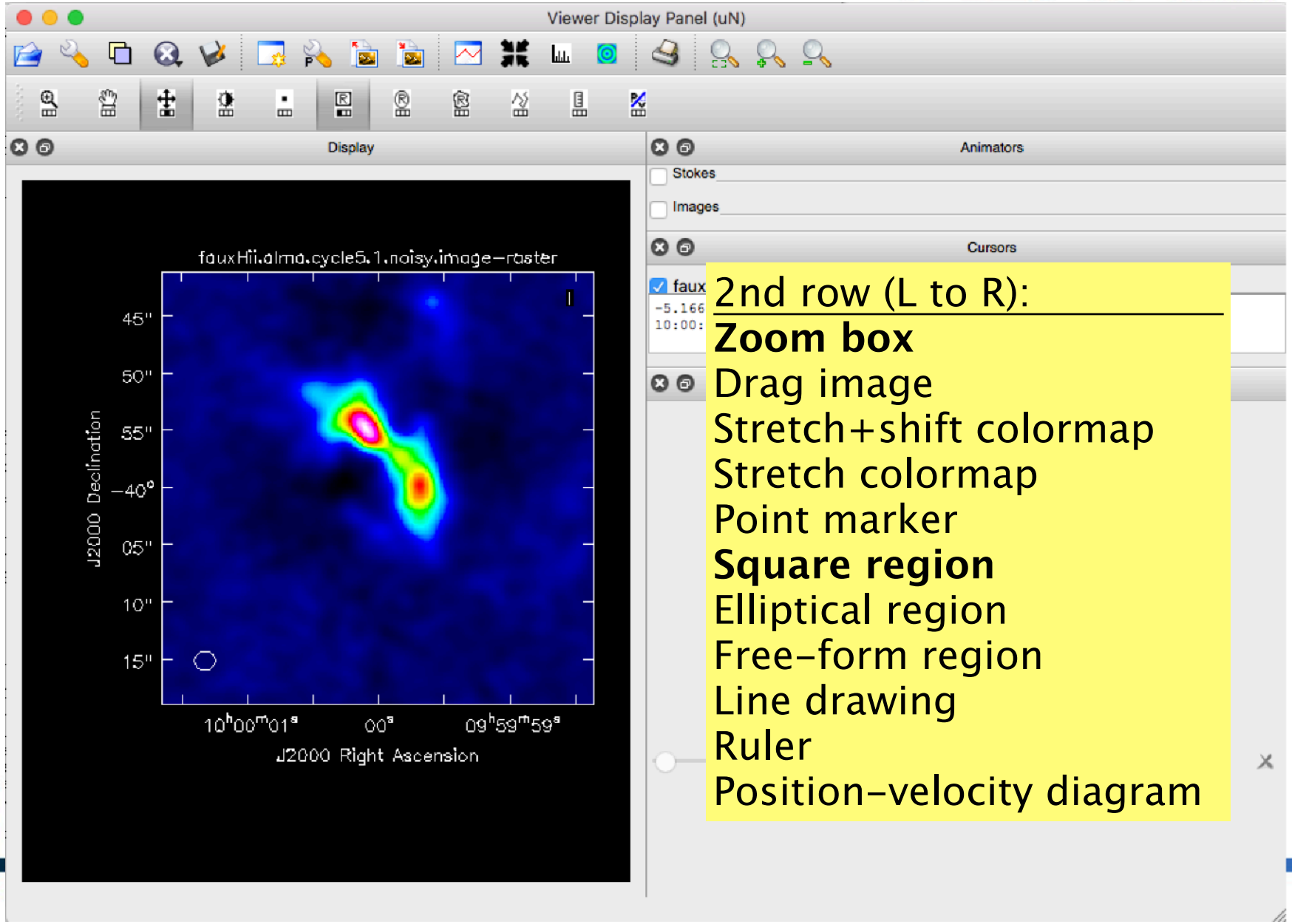
New panel
Pavel options
Save state
Restore state

Create spectrum
Collapse image
Histogram
Image fit

Print

Zoom completely out
Zoom in x2
Zoom out x2

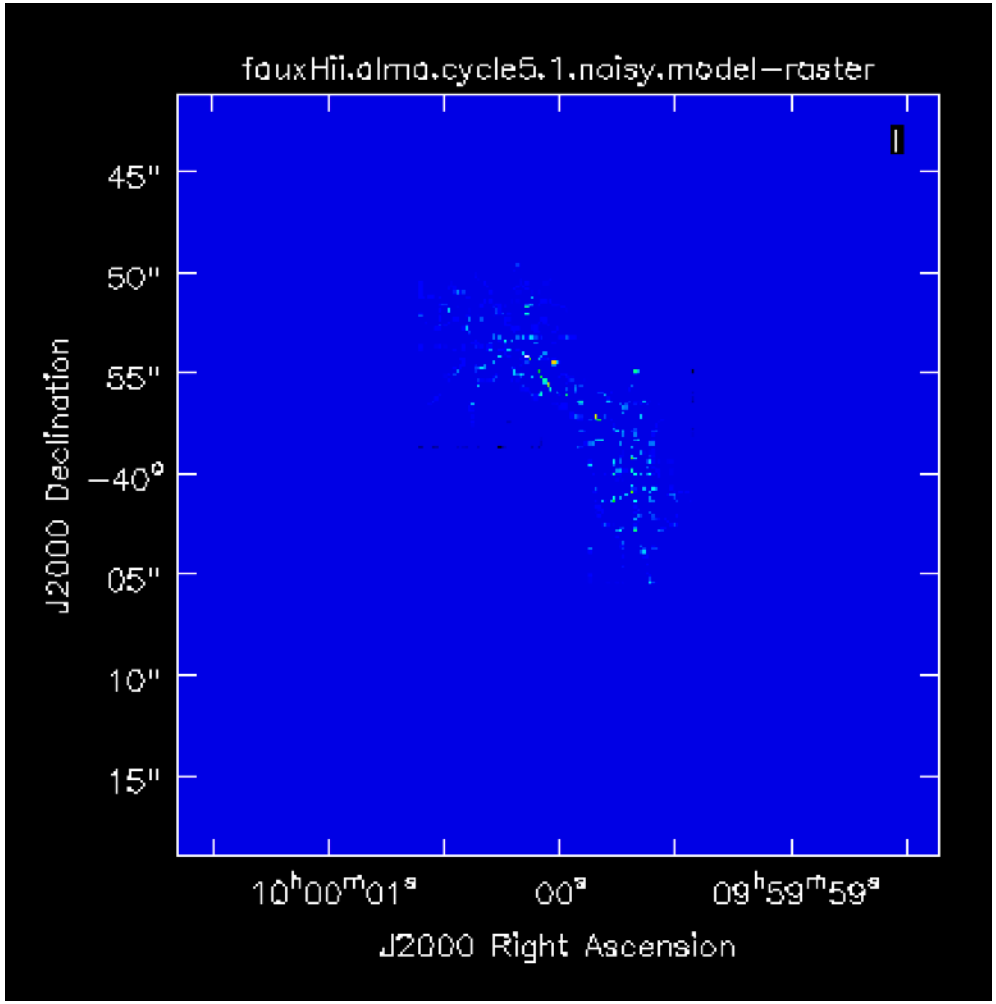
simobserve output: viewer



2nd row (L to R):

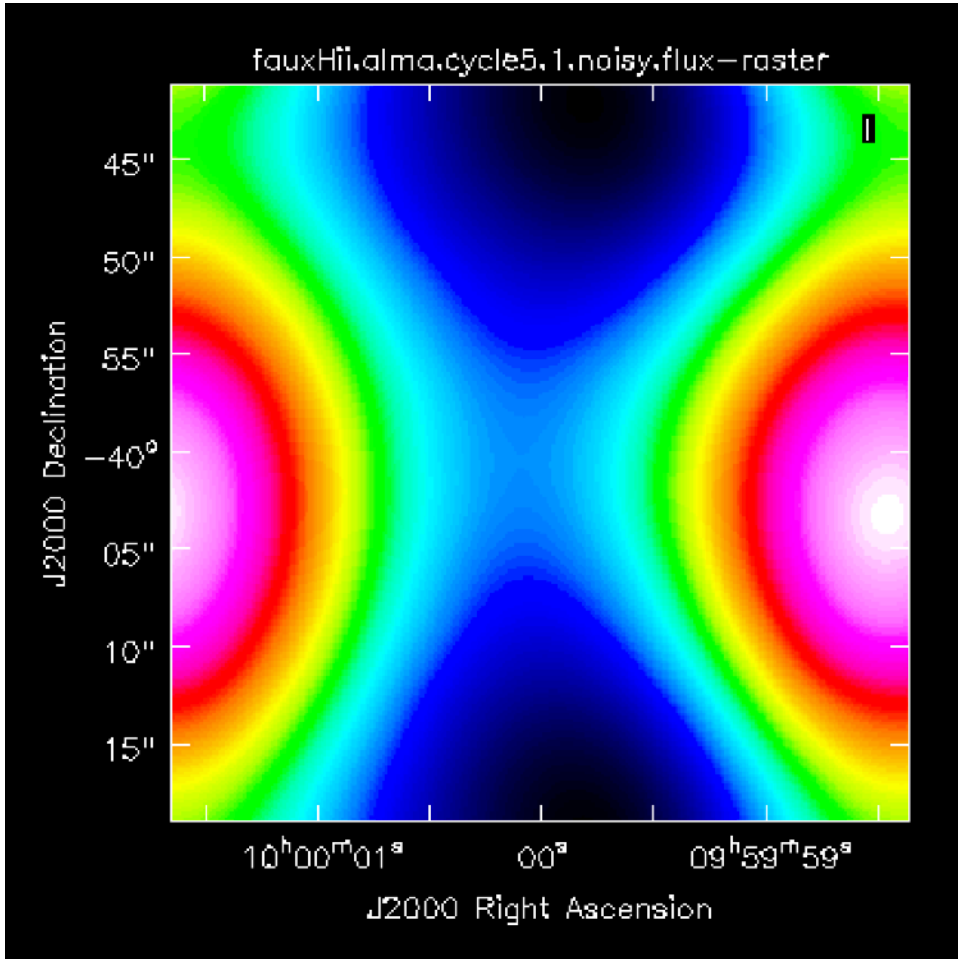
Zoom box
Drag image
Stretch+shift colormap
Stretch colormap
Point marker
Square region
Elliptical region
Free-form region
Line drawing
Ruler
Position-velocity diagram

simobserve output: model image



Note how modeling extended emission with a bunch of points is imperfect…

simobserve output: model image



And something has gone wrong with the primary beam coverage…
Need to ask it to map area that encompasses the mosaic region (or do the imaging yourself).

CASA Simulating is useful…

- Not only to demonstrate to the TAC that the observations are feasible, they will achieve desired results, and you have experience in dealing with radio data.
- But also to test whether the mosaic setup is what you want,
- and see if you really need those ACA/TP observations.

Useful exercise: Using what you learned from this test run, can you implement improvements on the observations?