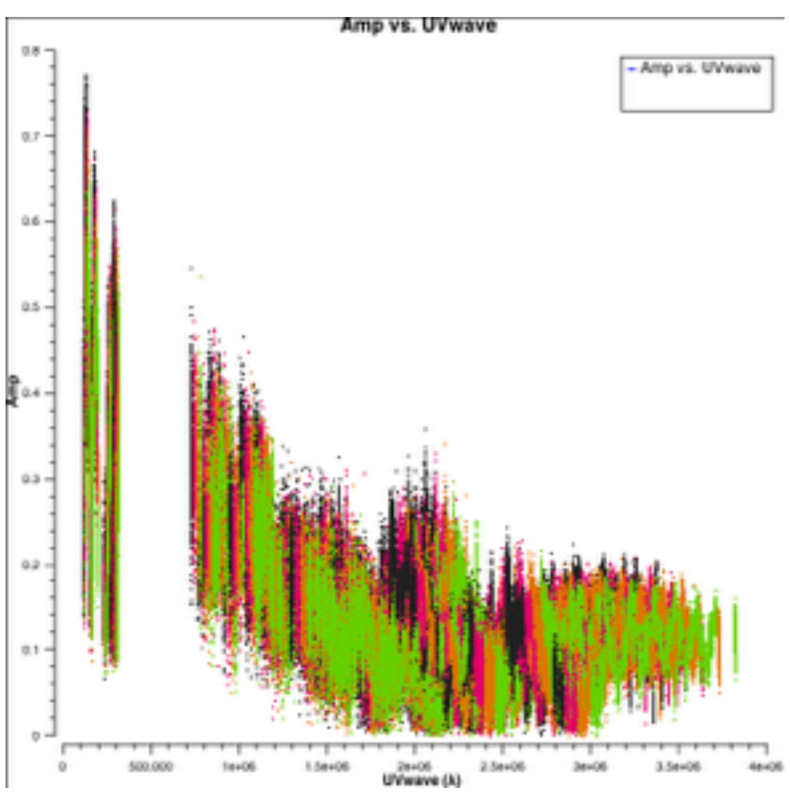
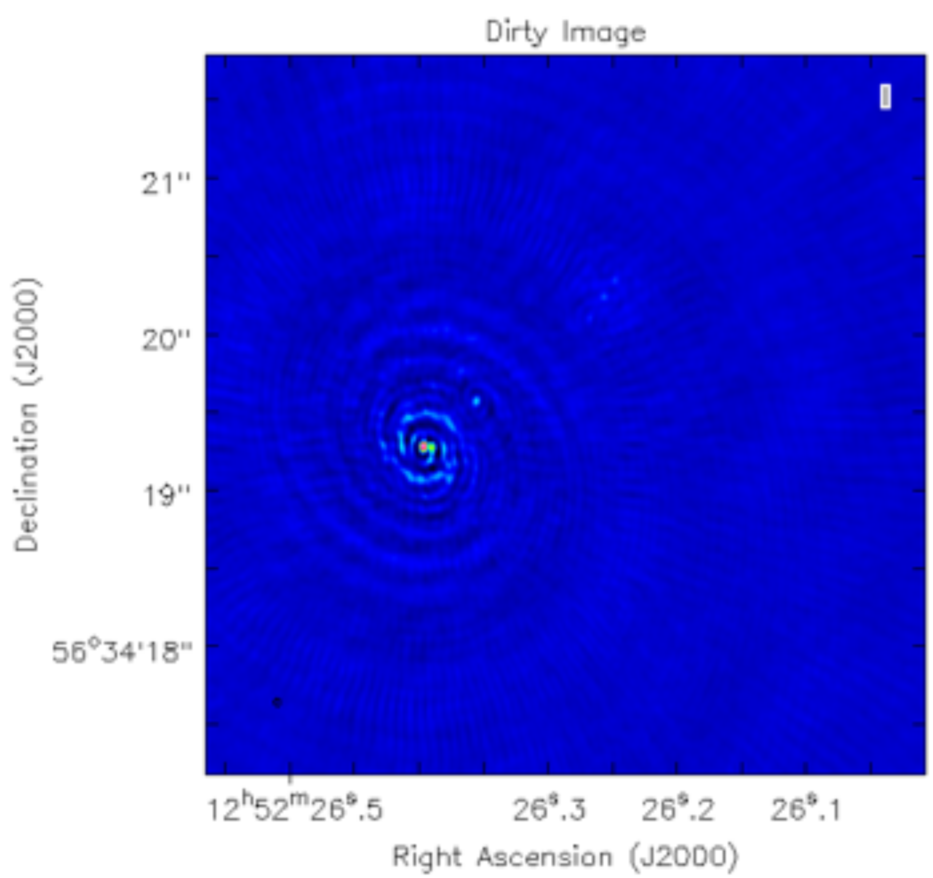
Basic Imaging and Self-Calibration (T4 + T7)

John McKean

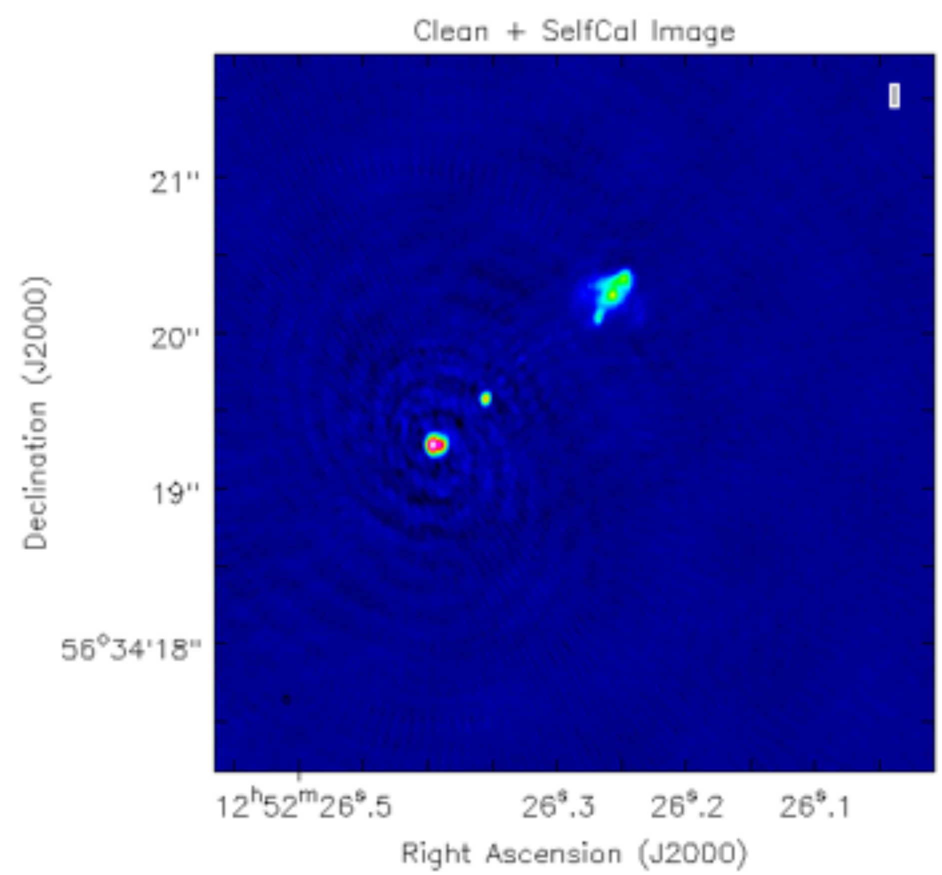
Visibilities



Fourier Transform



Deconvolution



AIM:

1. To make an image by taking the fast Fourier transform of the visibility data.
2. Carry out deconvolution using the CLEAN algorithm with CASA.
3. Use the new model obtained for the sky brightness distribution to carry out self-calibration.

During this process, we will make a ScriptForImaging.py file that can be used within CASA to make images automatically.

In the following "`> command`" is used to show inputs to the terminal and # comment # is used to explain where possible what is going on.

We will use the e-MERLIN data set on J1252+5634 that was edited and calibrated during the earlier tutorials (T2 and T3).

If you had problems during T3, download the calibrated dataset from,

http://almanas.jb.man.ac.uk/amsr/3C277p1/1252+5634.ms.tar

STEP 1 - Set-up the script

We will add our commands to a new ScriptForImaging.py, This script allows us to re-do what we have done, or parts of the process, automatically (useful for checking mistakes). Download the template from,

http://www.astron.nl/~mckean/ScriptForImaging.py

We can edit this file using your favourite text editor, e.g. emacs, pico, etc.

```
> pico ScriptForImaging.py
```

We will edit the script as we go.

ScriptForImaging — Edited

ScriptForImaging.py 〉 No Selection

```
# e-MERLIN imaging script for J1252+5634 (4 spws x 64 channels) in CASA 4.4.0

#Calibration steps
thesteps = [0]
step_title = {0: 'Title of step 0 (casa task)',
              1: 'Title of step 1 (casa task)'}

try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
  print 'global variable mysteps not set.'
if (thesteps==[]):
  thesteps = range(0,len(step_title))
  print 'Executing all steps: ', thesteps


# The Python variable 'mysteps' will control which steps
# are executed when you start the script using
#   execfile('scriptForCalibration.py')
# e.g. setting
#   mysteps = [2,3,4]# before starting the script will make the script execute
# only steps 2, 3, and 4
# Setting mysteps = [] will make it execute all steps.

print 'Write the value for variables -> run the script from the beginning'
#definitions

msfile = '1252+5634.ms' #ms multisource file
myspw = '0~3' #spw of interest, use myspw = '3' is your computer is slow

# description of step
mystep = 0
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]


# description of step
mystep = 1
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]
```

Here we enter our steps

Here we enter our variables

Here we will enter our commands

Here we will enter our commands

To start CASA,

```
> casapy      # start CASA #
```

You should see the following in your terminal

```
xterm
mckean@macbook /Work/ERIS-2015 > casapy
==>  /Applications/CASA.app/Contents

=========================================
The start-up time of CASA may vary
depending on whether the shared libraries
are cached or not.
=========================================

CASA Version 4.4.0-REL (r33623)
  Compiled on: Fri 2015/06/12 20:55:45 UTC

___________________________________________________________________
    For help use the following commands:
    tasklist               - Task list organized by category
    taskhelp               - One line summary of available tasks
    help taskname          - Full help for task
    toolhelp               - One line summary of available tools
    help par.parametername - Full help for parameter name

___________________________________________________________________
Activating auto-logging. Current session state plus future input saved.
Filename       : ipython-20150905-201047.log
Mode           : backup
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (4.3.0a rev#33168) import complete ***

CASA <2>: 
```

To run the script,

```
> mysteps = [0, 1]      # this will run steps 0 and 1 #
> execfile('ScriptForImaging.py')      # this run the script#
```

Nothing will happen because we have no commands yet, but msfile and myspw alias have been set.

STEP 2 - Determine our imaging field-of-view and pixel size

We will make an image by taking the fast Fourier transform (FFT) of the visibility data. This will involve projecting the sky surface brightness distribution onto a regular grid of pixels. We have some choices to make,

1. What is the size of the image that we would like to make?
2. How large should the pixels be?

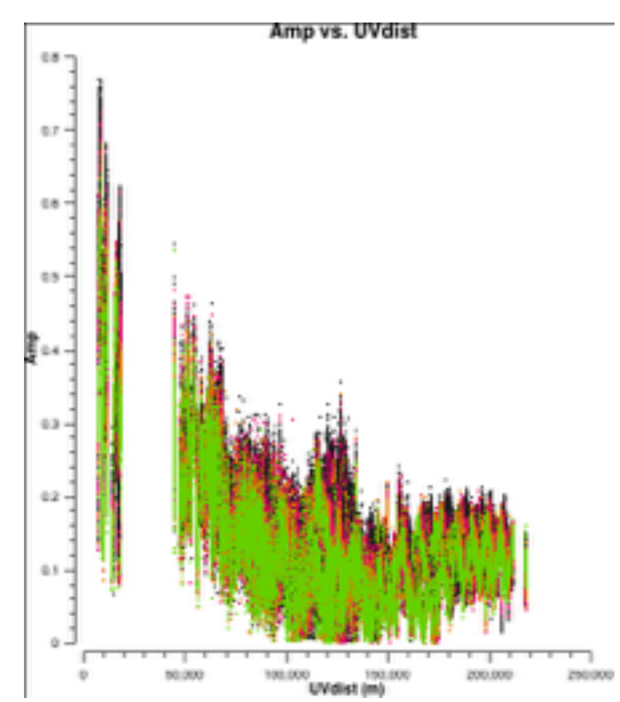
Image size: The visibilities contain information from all of the sources in the field-of-view. Technically we should make an image that is equal to this field-of-view. Our array is 6 antennas that are 25 m in size.

What is the field of view or a 25 m telescope at ~5 GHz?

```
> 3600 * (180 / pi) * (3e8 / 5.265e9) / 25    # arcsec * (rad->deg) * (c / ν) / D #
Out: 470.11921651759855      # Full width half max in arcsec #
```

This should be the field-of-view that we image, but we will use ~5 arcsec for speed.

Pixel Size: We need to Nyqvist sample the data when we projected it onto a regular grid so that we do not lose information. We can estimate pixel-size by considering the longest baseline in our data set using plotms and plotting AMP versus UVDIST (colourise SPW, corr='RR, LL', argchannel = '64').



We see that the longest baseline is at ~220 km. So we can estimate the synthesised beam with,

```
> 3600 * (180 / pi) * (3e8 / 5.265e9) / 220e3
Out: 0.05347342021615011      # max resolution in arcsec #
```

This is approximately what we would expect, so we take 10.7 mas pixels for safety (1/5 sampling).

STEP 3 - Make an image

We will start by making our first image, which will be the FFT of the visibility data. All deconvolution is carried out in CASA using the CLEAN task.

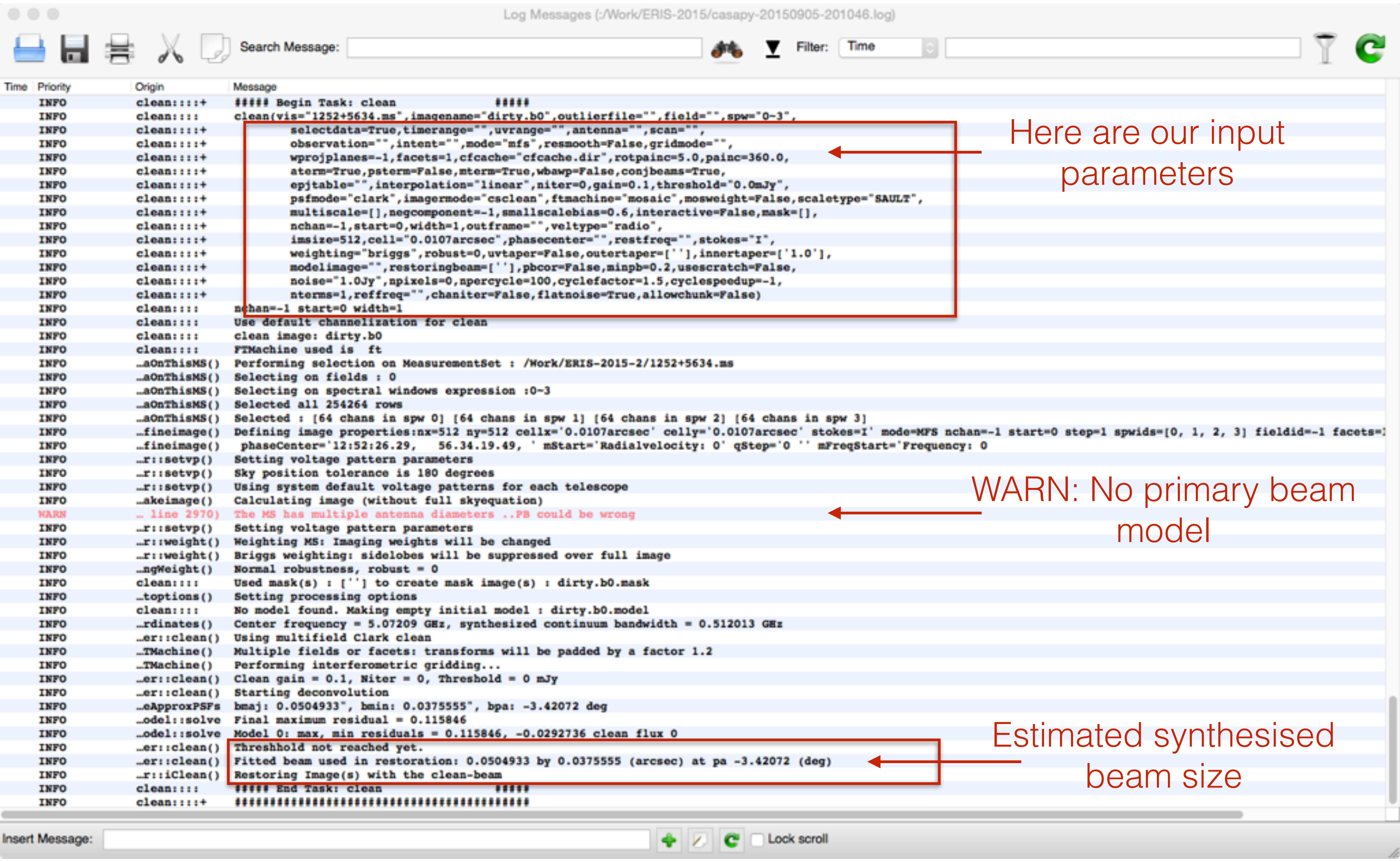
```
> help clean
```

This will give you a full summary of the task and suggested input parameters. Many of them we will not use here for this tutorial.

Start by replacing all of the parameters back to their defaults

```
> default clean

> vis = msfile                # Name of the visibility MS file #
> spw = myspw                 # spectral windows that we will use #
> cell = “0.0107arcsec”       # pixel-size we will use #
> imsize = 512                # image size we will use (~5 arcsec) #
> weighting = “briggs”        # set visibility weighting #
> robust = 0                  # set robust parameter (balance between nat/uni) #
> niter = 0                   # we will do no convolution #
> imagename = “dirty.b0”      # call your image something #
> inp                         # check your inputs (nothin should be red) #
> go clean                    # run the task #
```

Look at your logger window to view the progress of CLEAN.



Log Messages (:/Work/ERIS-2015/casapy-20150905-201046.log)
Search Message:
Filter: Time
Time Priority Origin Message
INFO clean::::+ ##### Begin Task: clean #####
INFO clean:::: clean(vis="1252+5634.ms",imagename="dirty.b0",outlierfile="",field="",spw="0-3",
INFO clean::::+ selectdata=True,timerange="",uvrange="",antenna="",scan="",
INFO clean::::+ observation="",intent="",mode="mfs",resmooth=False,gridmode="",
INFO clean::::+ wprojplanes=-1,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,
INFO clean::::+ aterm=True,psterm=False,mterm=True,wbawp=False,conjbeams=True,
INFO clean::::+ epjtable="",interpolation="linear",niter=0,gain=0.1,threshold="0.0mJy",
INFO clean::::+ psfmode="clark",imagermode="csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",
INFO clean::::+ multiscale=[],negcomponent=-1,smallscalebias=0.6,interactive=False,mask=[],
INFO clean::::+ nchan=-1,start=0,width=1,outframe="",veltype="radio",
INFO clean::::+ imsize=512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",
INFO clean::::+ weighting="briggs",robust=0,uvtaper=False,outertaper=[''],innertaper=['1.0'],
INFO clean::::+ modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=False,
INFO clean::::+ noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,
INFO clean::::+ nterms=1,reffreq="",chaniter=False,flatnoise=True,allowchunk=False)
INFO clean:::: nchan=-1 start=0 width=1
INFO clean:::: Use default channelization for clean
INFO clean:::: clean image: dirty.b0
INFO clean:::: FTMachine used is ft
INFO ...aOnThisMS() Performing selection on MeasurementSet : /Work/ERIS-2015-2/1252+5634.ms
INFO ...aOnThisMS() Selecting on fields : 0
INFO ...aOnThisMS() Selecting on spectral windows expression :0-3
INFO ...aOnThisMS() Selected all 254264 rows
INFO ...aOnThisMS() Selected : [64 chans in spw 0] [64 chans in spw 1] [64 chans in spw 2] [64 chans in spw 3]
INFO ...fineimage() Defining image properties:nx=512 ny=512 cellx='0.0107arcsec' celly='0.0107arcsec' stokes=I' mode=MFS nchan=-1 start=0 step=1 spwids=[0, 1, 2, 3] fieldid=-1 facets=1
INFO ...fineimage() phaseCenter='12:52:26.29, 56.34.19.49, ' mStart='Radialvelocity: 0' qStep='0 '' mFreqStart='Frequency: 0
INFO ...r::setvp() Setting voltage pattern parameters
INFO ...r::setvp() Sky position tolerance is 180 degrees
INFO ...r::setvp() Using system default voltage patterns for each telescope
INFO ...akeimage() Calculating image (without full skyequation)
WARN ... line 2970) The MS has multiple antenna diameters ..PB could be wrong
INFO ...r::setvp() Setting voltage pattern parameters
INFO ...r::weight() Weighting MS: Imaging weights will be changed
INFO ...r::weight() Briggs weighting: sidelobes will be suppressed over full image
INFO ...ngWeight() Normal robustness, robust = 0
INFO clean:::: Used mask(s) : [''] to create mask image(s) : dirty.b0.mask
INFO ...toptions() Setting processing options
INFO clean:::: No model found. Making empty initial model : dirty.b0.model
INFO ...rdinates() Center frequency = 5.07209 GHz, synthesized continuum bandwidth = 0.512013 GHz
INFO ...er::clean() Using multifield Clark clean
INFO ...TMachine() Multiple fields or facets: transforms will be padded by a factor 1.2
INFO ...TMachine() Performing interferometric gridding...
INFO ...er::clean() Clean gain = 0.1, Niter = 0, Threshold = 0 mJy
INFO ...er::clean() Starting deconvolution
INFO ...eApproxPSFs bmaj: 0.0504933", bmin: 0.0375555", bpa: -3.42072 deg
INFO ...odel::solve Final maximum residual = 0.115846
INFO ...odel::solve Model 0: max, min residuals = 0.115846, -0.0292736 clean flux 0
INFO ...er::clean() Threshhold not reached yet.
INFO ...er::clean() Fitted beam used in restoration: 0.0504933 by 0.0375555 (arcsec) at pa -3.42072 (deg)
INFO ...r::iClean() Restoring Image(s) with the clean-beam
INFO clean:::: ##### End Task: clean #####
INFO clean::::+ ##
Insert Message:
Lock scroll
Here are our input parameters
WARN: No primary beam model
Estimated synthesised beam size

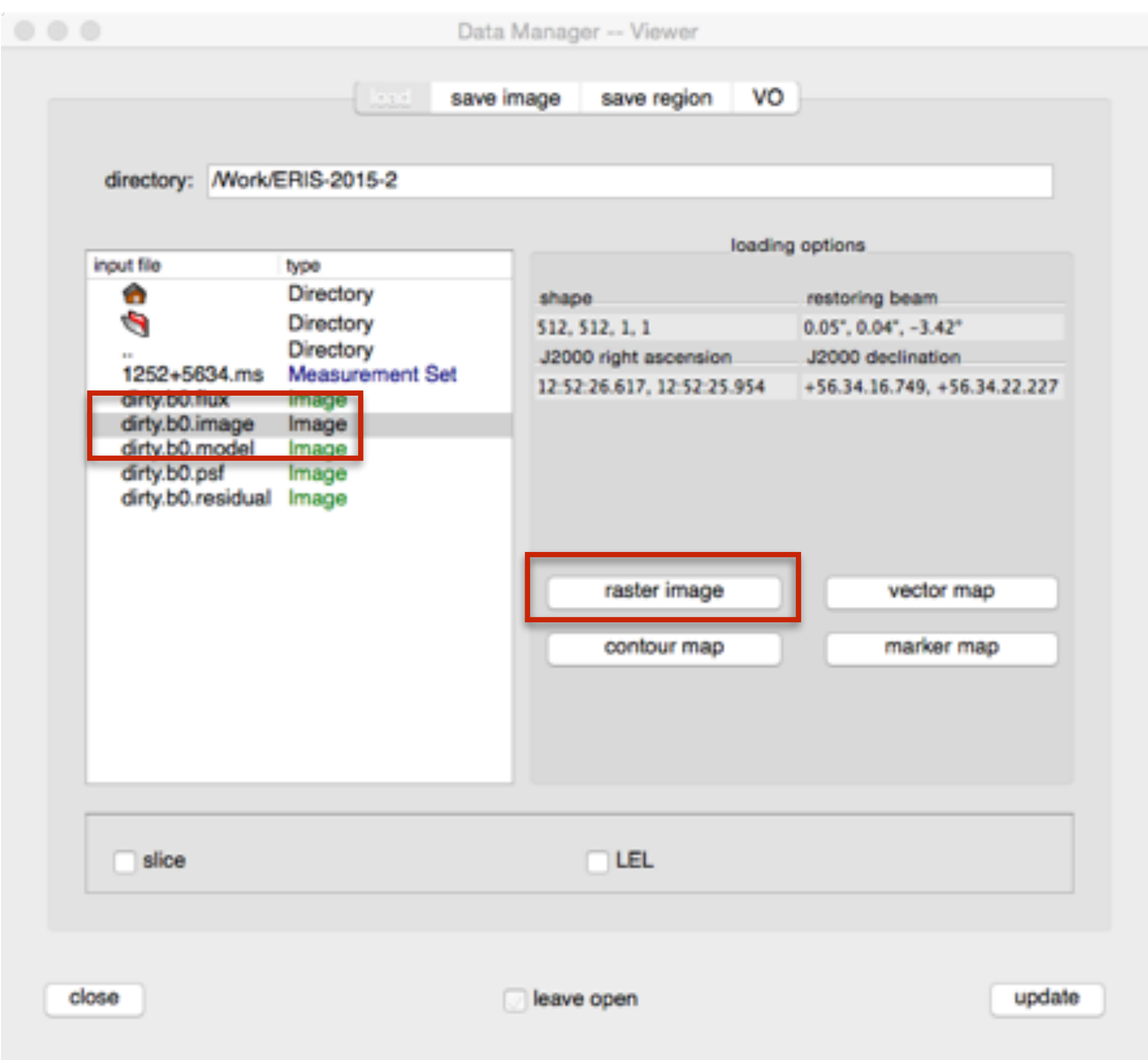
Lets look at the output. We have generated 5 images that are all on the same grid

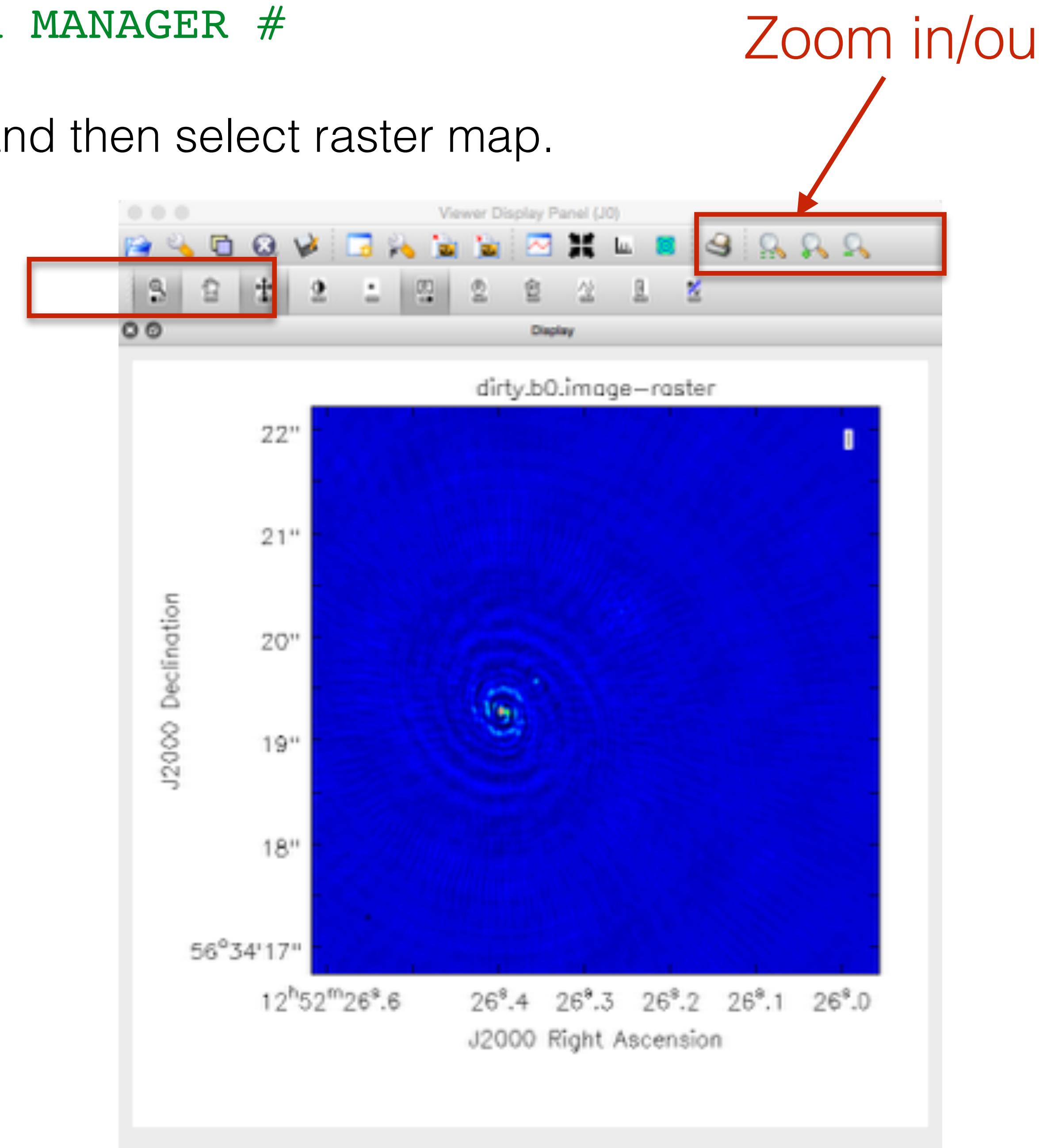
dirty.b0.image # The 'deconvolved' image #
dirty.b0.psf # The image of the point spread function (FFT of the uv-sampling function) #
dirty.b0.model # The image containing your model components (delta functions, truncated Gaussians) #
dirty.b0.residual # The image made by subtracting the model from the visibility of doing an FFT #
dirty.b0.flux # An image of the expected primary beam response #

We can look at each of these images using the CASA VIEWER (run interactively or from the command line).

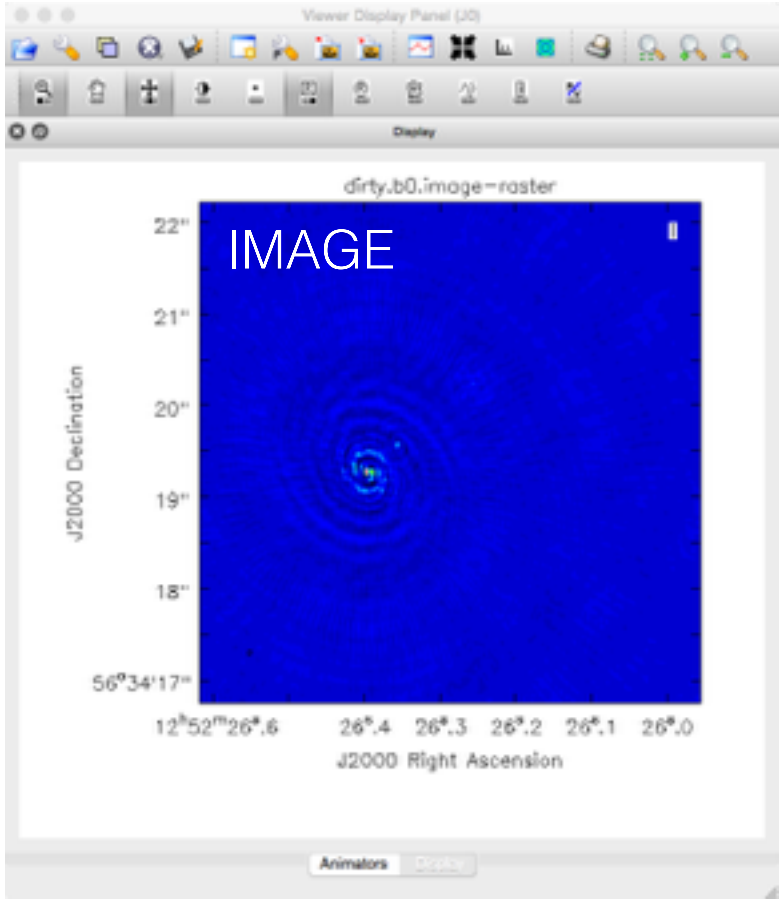
```
> viewer    # start the viewer GUI and DATA MANAGER #
```

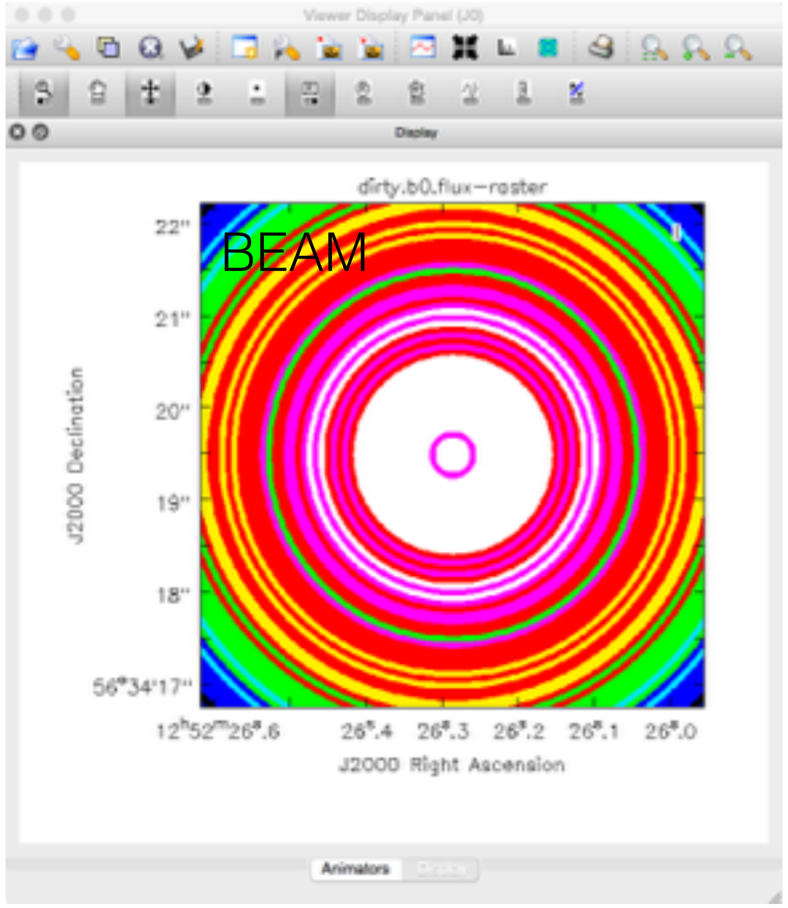
This will start the VIEWER GUI, select the dirty.b0.image and then select raster map.

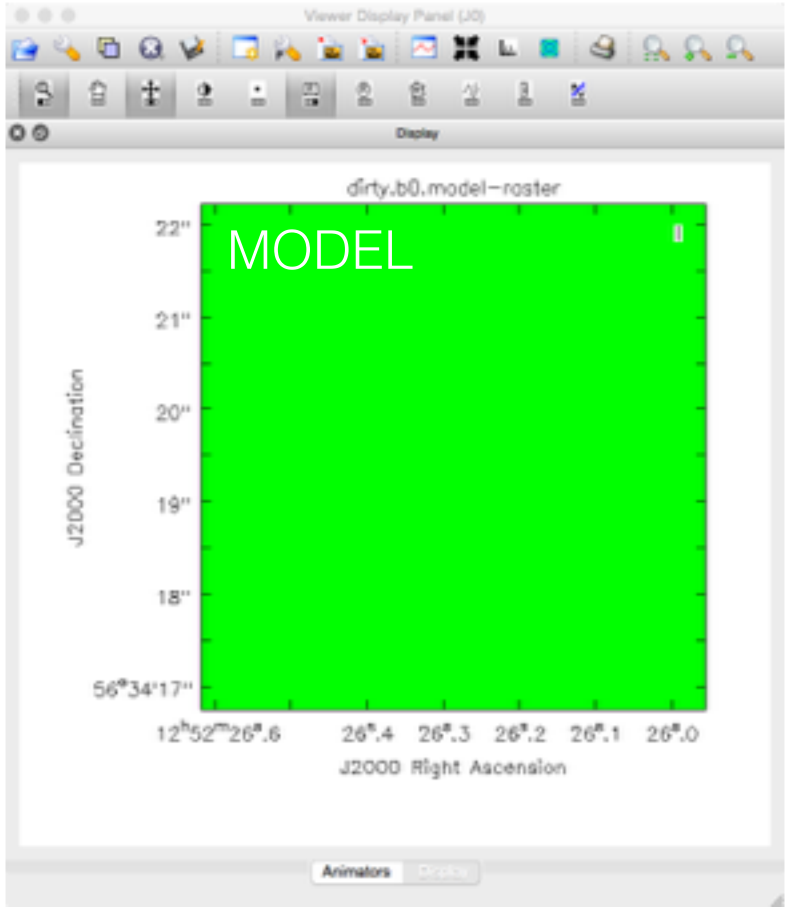


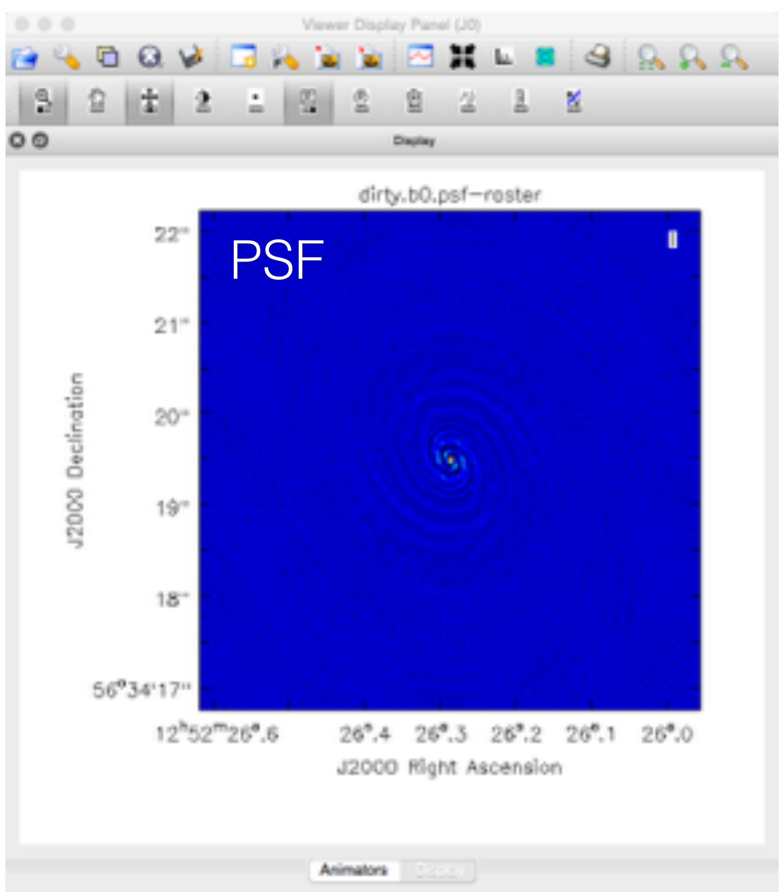


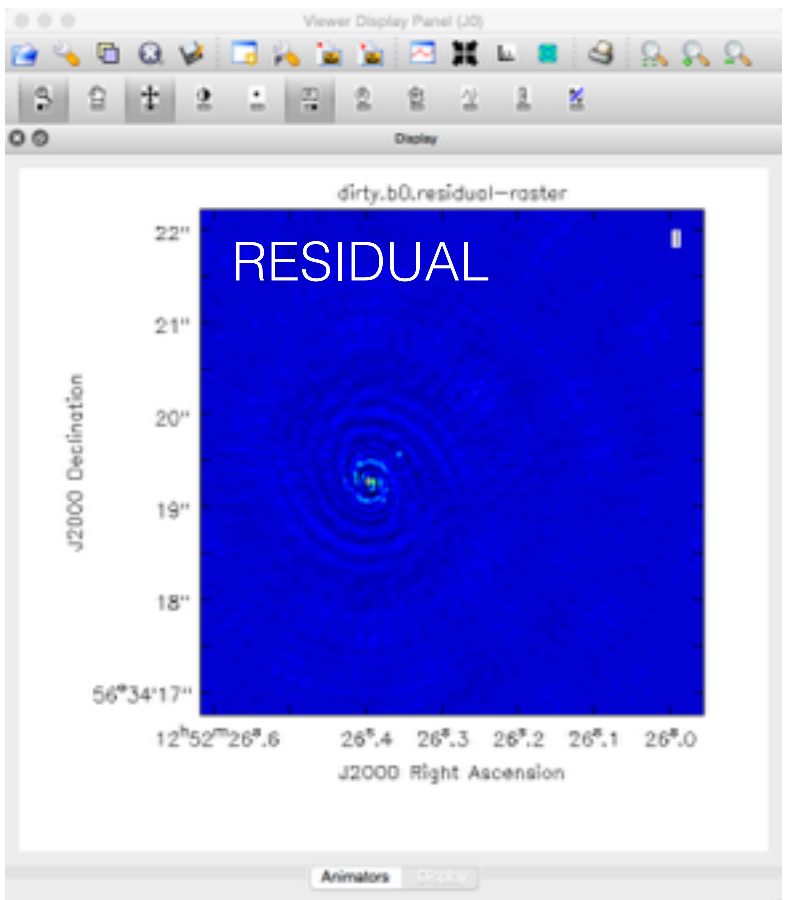
Lets look at each of the output images (either start a new VIEWER or add multiple images to the same VIEWER and use the ANIMATOR option - top menu -> VIEW -> ANIMATOR).









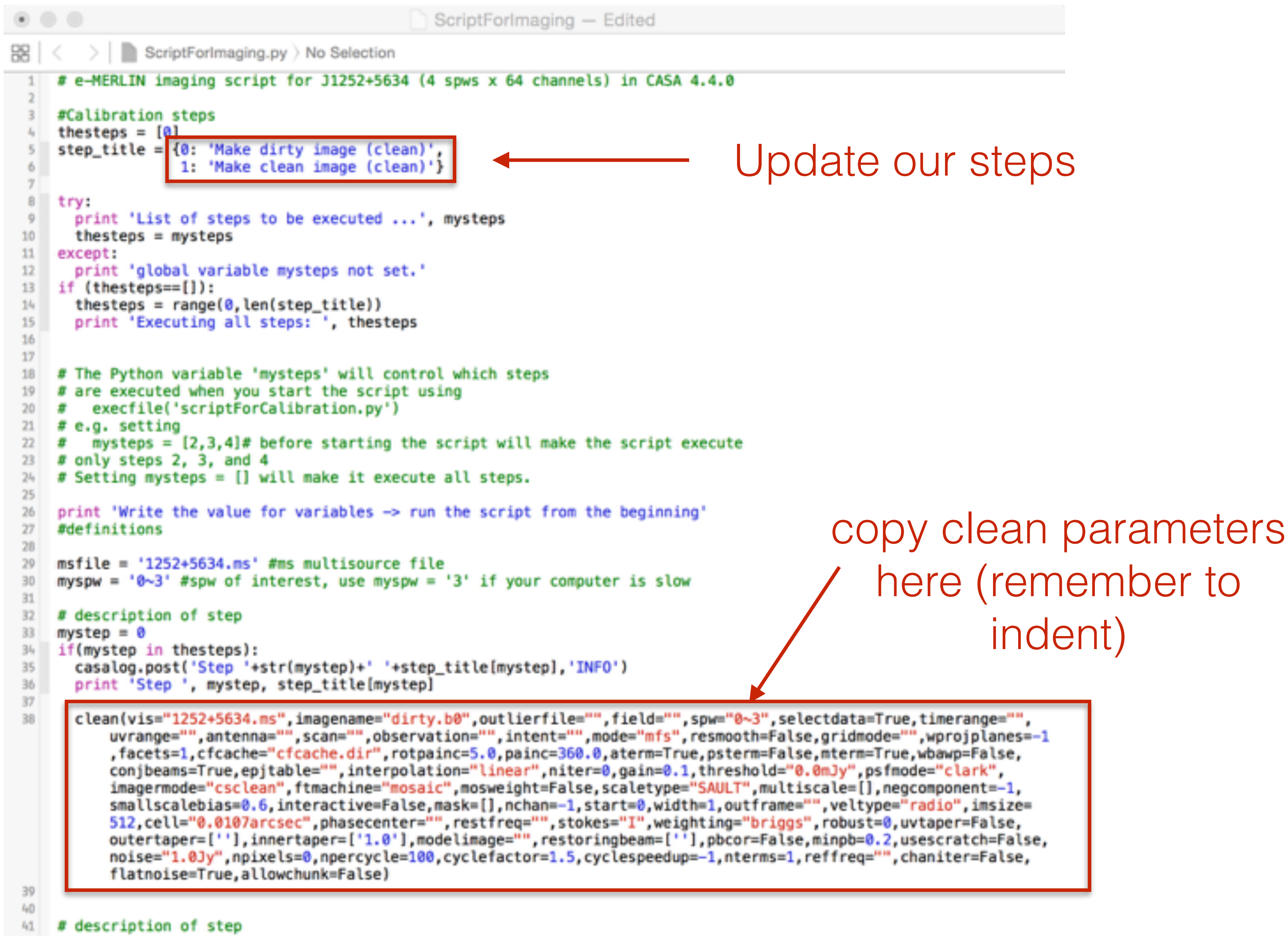


All that seemed to work well, so lets add the parameters of our CLEAN run to our script. Every time we run a task in CASA we generate a, for example clean.last file

```
> !more clean.last
```

and copy the final part to our script, and if we wanted, do what we just did using our script,

```
> mysteps = [0]      # this will run step 0 #
> execfile('ScriptForImaging.py')      # this run the script#
```



```
# e-MERLIN imaging script for J1252+5634 (4 spws x 64 channels) in CASA 4.4.0

#Calibration steps
thesteps = [0]
step_title = {0: 'Make dirty image (clean)',
              1: 'Make clean image (clean)'}

try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
  print 'global variable mysteps not set.'
if (thesteps==[]):
  thesteps = range(0,len(step_title))
  print 'Executing all steps: ', thesteps


# The Python variable 'mysteps' will control which steps
# are executed when you start the script using
#   execfile('scriptForCalibration.py')
# e.g. setting
#   mysteps = [2,3,4]# before starting the script will make the script execute
# only steps 2, 3, and 4
# Setting mysteps = [] will make it execute all steps.

print 'Write the value for variables -> run the script from the beginning'
#definitions

msfile = '1252+5634.ms' #ms multisource file
myspw = '0~3' #spw of interest, use myspw = '3' if your computer is slow

# description of step
mystep = 0
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  clean(vis="1252+5634.ms",imagename="dirty.b0",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",
      uvrange="",antenna="",scan="",observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1
      ,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,psterm=False,mterm=True,wbawp=False,
      conjbeams=True,epjtable="",interpolation="linear",niter=0,gain=0.1,threshold="0.0mJy",psfmode="clark",
      imagermode="csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",multiscale=[],negcomponent=-1,
      smallscalebias=0.6,interactive=False,mask=[],nchan=-1,start=0,width=1,outframe="",veltype="radio",imsize=
      512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="briggs",robust=0,uvtaper=False,
      outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=False,
      noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,
      flatnoise=True,allowchunk=False)


# description of step
```

STEP 4 - What about image weights

So far we have only used robust = 0, but lets try the case of natural and uniform weights (robust = 2 and = -2).

```
> tget clean                # recover the last set of parameters used #
> robust = 2                # set robust parameter to 2 (natural weighting) #
> imagename = “dirty.b2” # set new image name to make new file #
> go clean                  # start FFT #
```

And once that is completed, we can add the clean.last command to our script. The run with robust = -2

```
> tget clean                 # recover the last set of parameters used #
> robust = -2                # set robust parameter to -2 (uniform weighting) #
> imagename = “dirty.b-2” # set new image name to make new file #
> go clean                   # start FFT #
```

Note the synthesised beam sizes that are estimated by CASA for the different weights.

Next lets look at the dirty images and psf images using the VIEWER.

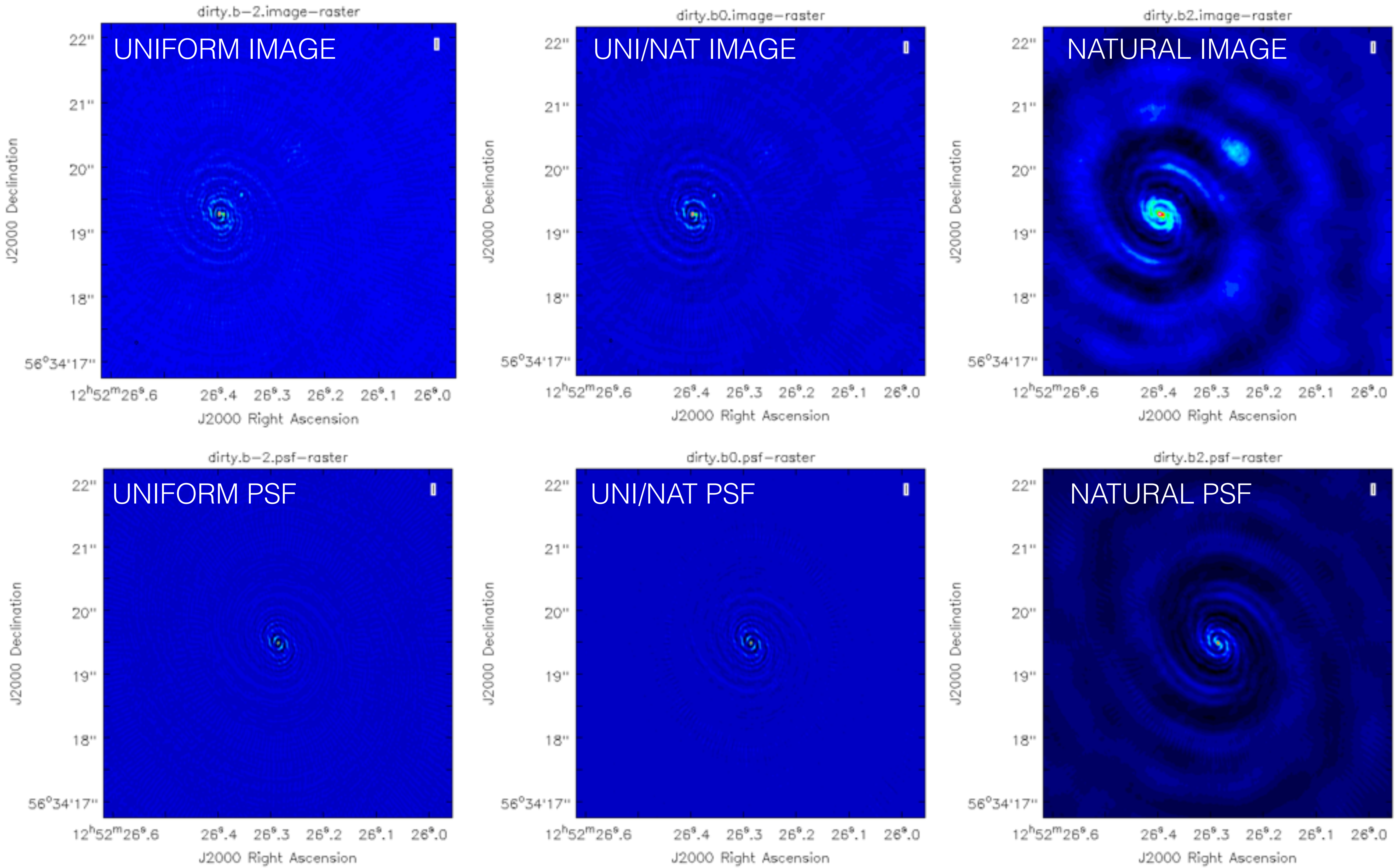
ScriptForImaging

ScriptForImaging.py 〉 No Selection

```
# Setting mysteps = [] will make it execute all steps.

print 'Write the value for variables -> run the script from the beginning'
#definitions

msfile = '1252+5634.ms' #ms multisource file
myspw = '0~3' #spw of interest, use myspw = '3' if your computer is slow

# description of step
mystep = 0
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  clean(vis="1252+5634.ms",imagename="dirty.b0",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",
      uvrange="",antenna="",scan="",observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1
      ,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,psterm=False,mterm=True,wbawp=False,
      conjbeams=True,epjtable="",interpolation="linear",niter=0,gain=0.1,threshold="0.0mJy",psfmode="clark",
      imagermode="csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",multiscale=[],negcomponent=-1,
      smallscalebias=0.6,interactive=False,mask=[],nchan=-1,start=0,width=1,outframe="",veltype="radio",imsize=
      512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="briggs",robust=0,uvtaper=False,
      outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=False,
      noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,
      flatnoise=True,allowchunk=False)

  clean(vis="1252+5634.ms",imagename="dirty.b2",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",
      uvrange="",antenna="",scan="",observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1
      ,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,psterm=False,mterm=True,wbawp=False,
      conjbeams=True,epjtable="",interpolation="linear",niter=0,gain=0.1,threshold="0.0mJy",psfmode="clark",
      imagermode="csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",multiscale=[],negcomponent=-1,
      smallscalebias=0.6,interactive=False,mask=[],nchan=-1,start=0,width=1,outframe="",veltype="radio",imsize=
      512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="briggs",robust=2,uvtaper=False,
      outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=False,
      noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,
      flatnoise=True,allowchunk=False)

  clean(vis="1252+5634.ms",imagename="dirty.b-2",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",
      uvrange="",antenna="",scan="",observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1
      ,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,psterm=False,mterm=True,wbawp=False,
      conjbeams=True,epjtable="",interpolation="linear",niter=0,gain=0.1,threshold="0.0mJy",psfmode="clark",
      imagermode="csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",multiscale=[],negcomponent=-1,
      smallscalebias=0.6,interactive=False,mask=[],nchan=-1,start=0,width=1,outframe="",veltype="radio",imsize=
      512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="briggs",robust=-2,uvtaper=False,
      outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=False,
      noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,
      flatnoise=True,allowchunk=False)


# description of step
mystep = 1
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]
```

TIP: It is useful to first make a dirty image to see if you choice of pixel size (cell) and image size (imsize) is appropriate given your target observation.

Also, look at the side-lobe structure of the PSF as it will help you when you are de-convolving the image,



STEP 5 - Deconvolution

The ripples that we see in the dirty images are due to the side-lobe structure of the PSF. This is dependent on the uv-coverage (sampling function) and our choice of weighting. For the remainder of the tutorial, we will use Briggs weighting with robust = 0.

```
> tget clean      # recover the last set of parameters used #
> robust = 0      # set robust parameter to 0 (uniform/natural weighting) #
```

We deconvolve using the CLEAN algorithm, and in this case we will use delta functions to make a model for the source. Other options, for example, truncated Gaussians are possible, but we will not use here.

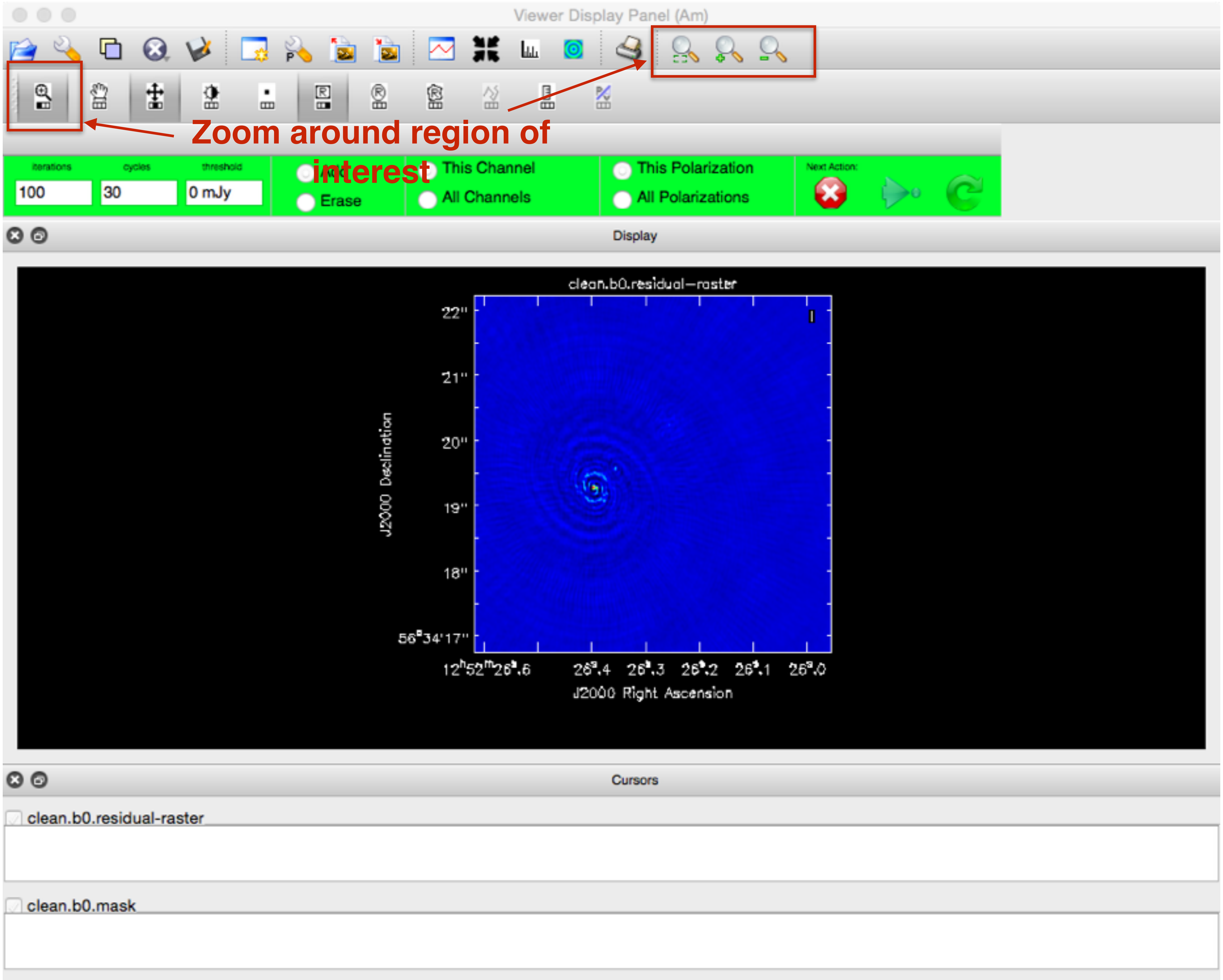
The CLEAN algorithm has the following steps:

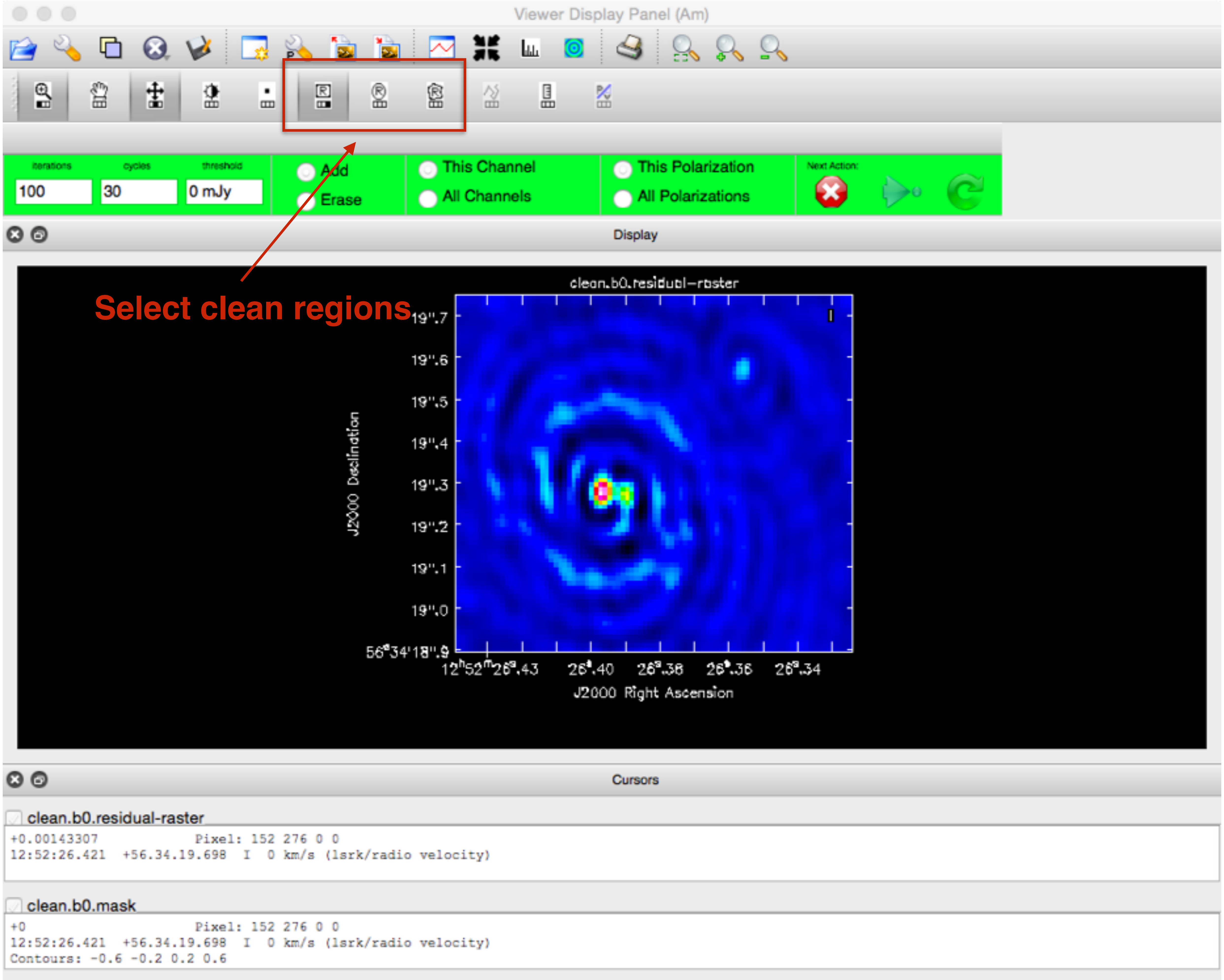
1. Identify the surface brightness peak in the map.
2. Fit a delta function to this position that has a value of the peak surface brightness * gain factor.
3. Subtract the delta function from the image.
4. Identify the next brightness peak and repeat steps 2 and 3 (Minor Cycle).
5. Subtract the collection of delta functions from the uv-data and re image.
6. Repeat steps 1-5 until some threshold is reached.

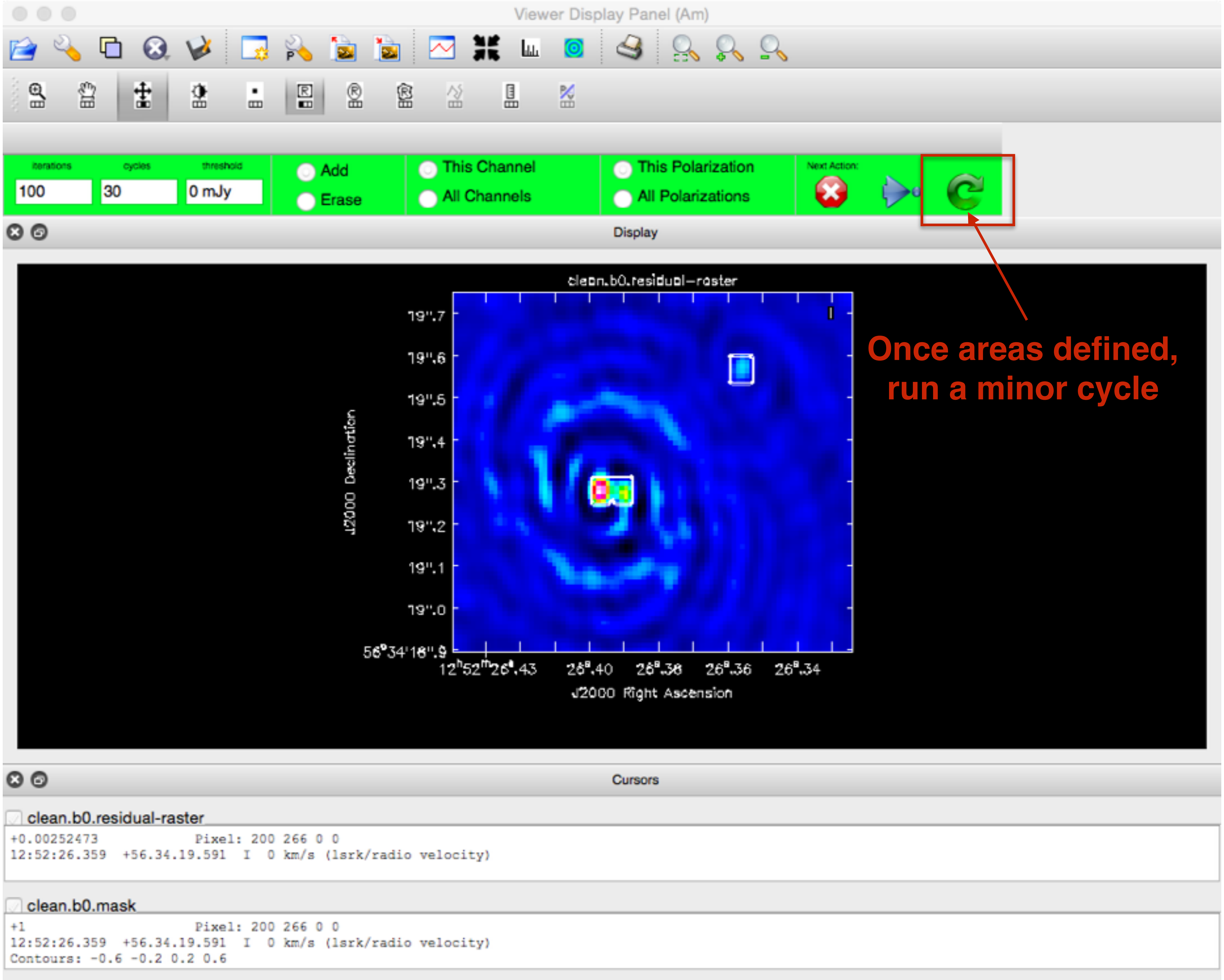
Now we need to define two new parameters for CLEAN

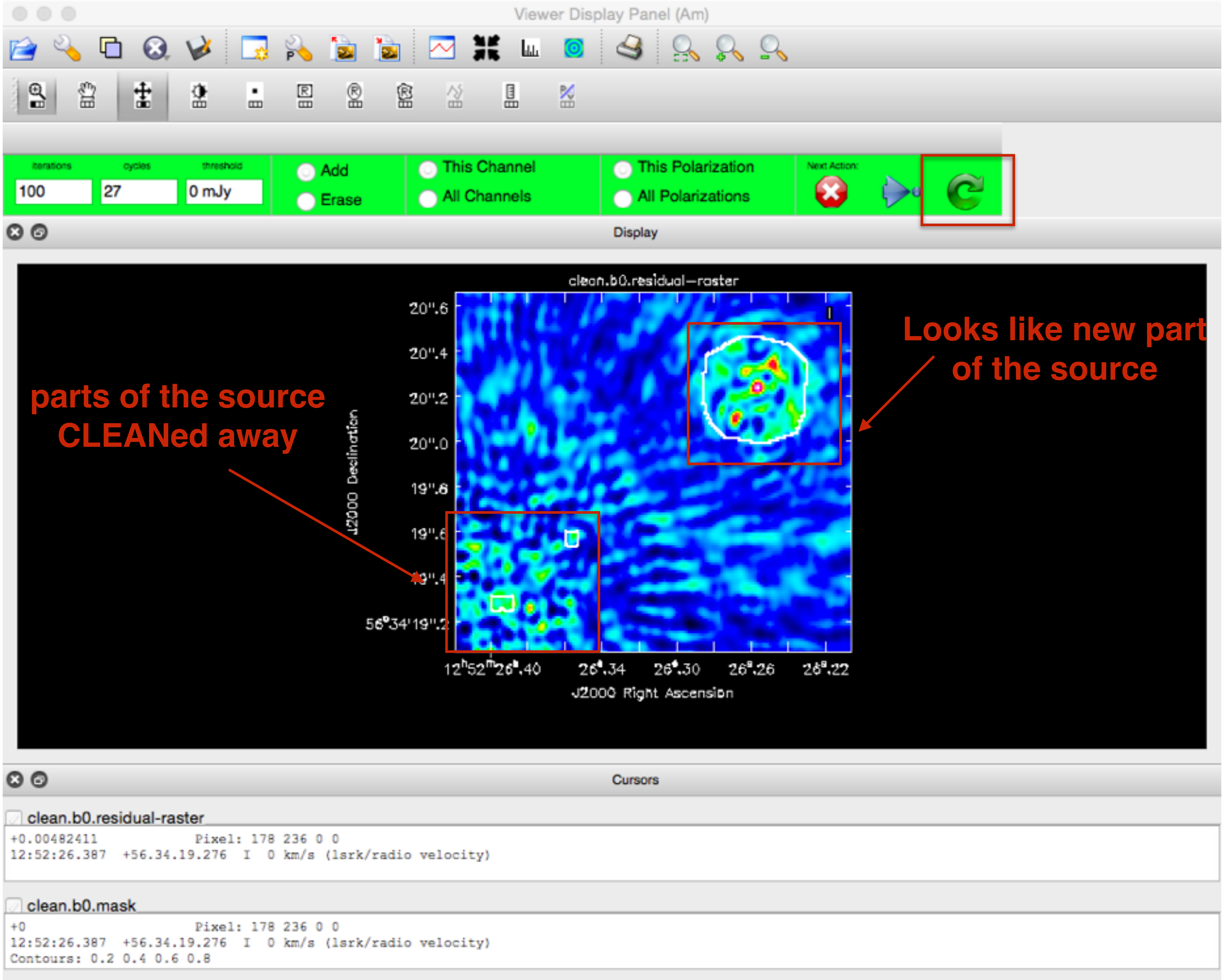
```
> niter = 3000                # number of interactions (trial / error) #
> gain = 0.05                 # factor of the peak brightness to be subtracted #
> interactive = T             # to allow interactive cleaning #
> imagename = “clean.b0”      # set new image name to make new file #
> inp                         # review parameters #
> go clean                    # start FFT and deconvolution #
```

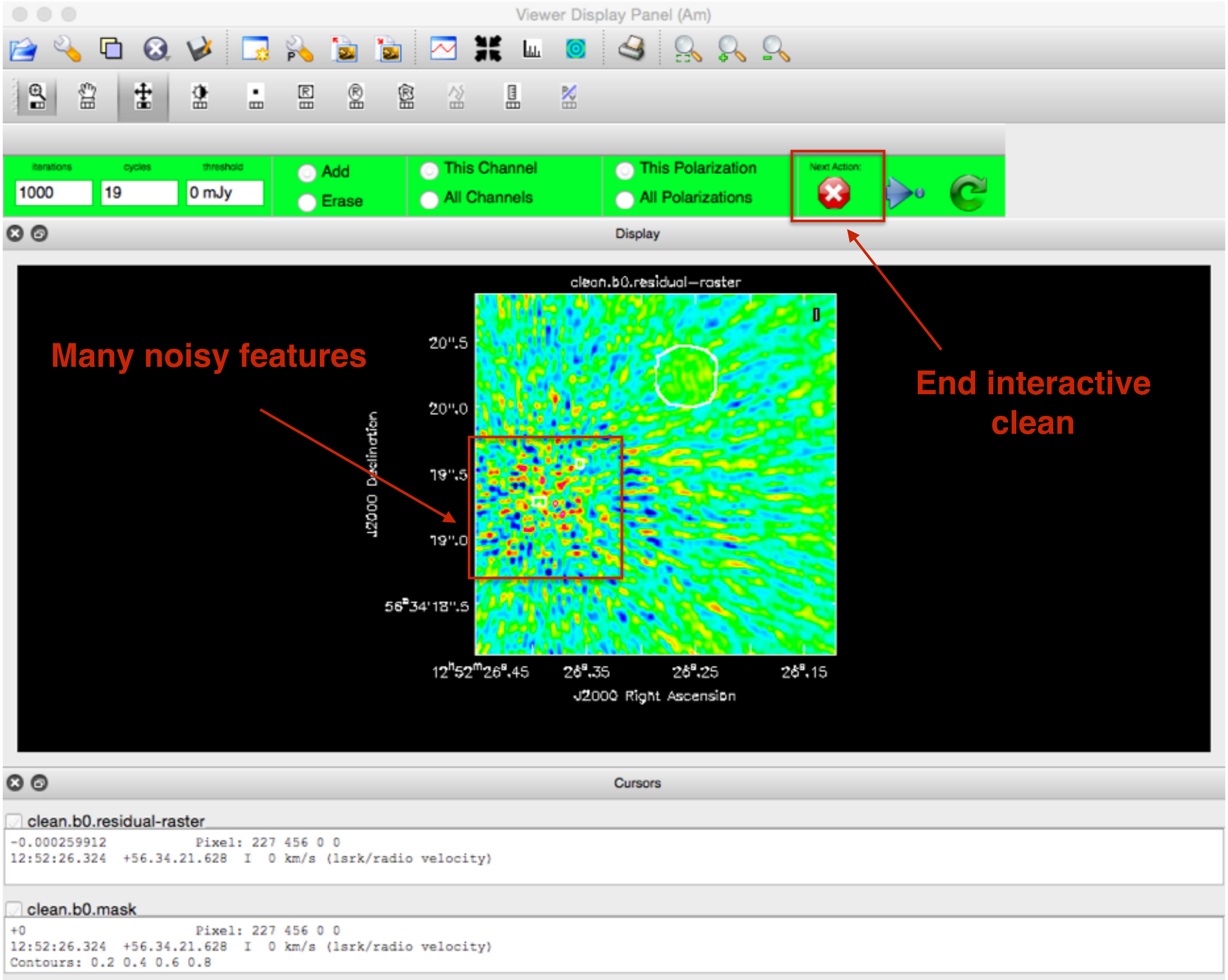
Remember to look at your logger for information.

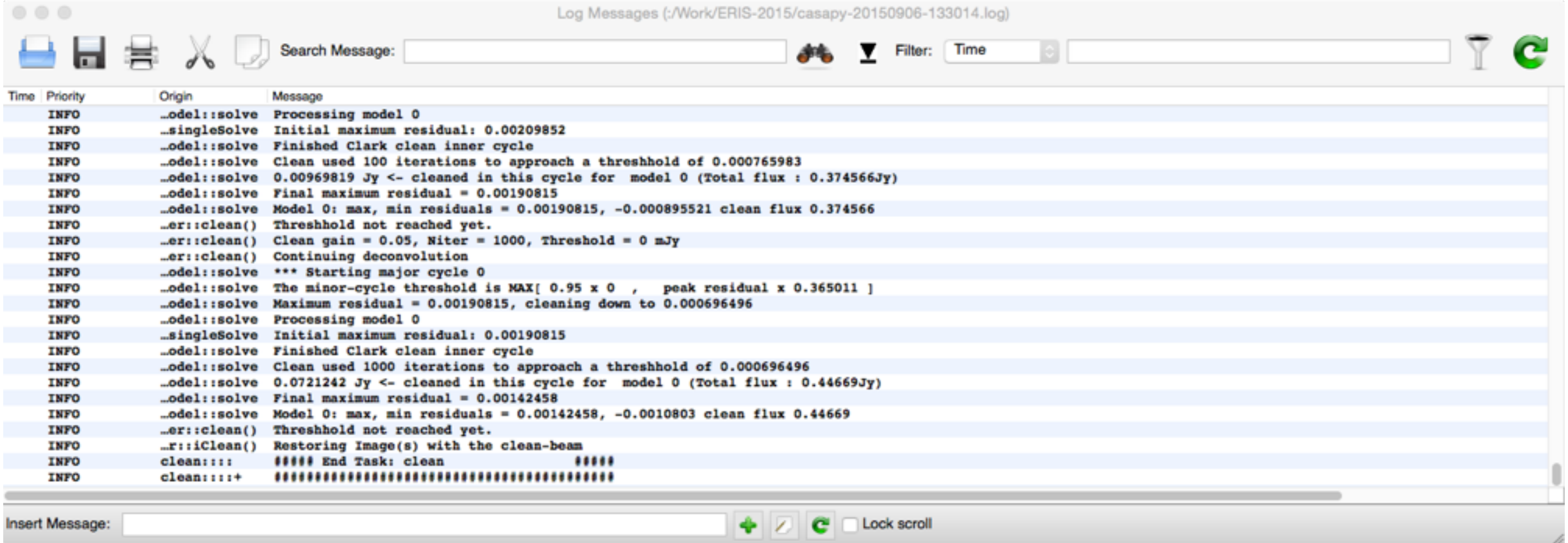

Viewer Display Panel (Am)
Zoom around region of interest
iterations
cycles
threshold
100
30
0 mJy
Erase
This Channel
All Channels
This Polarization
All Polarizations
Next Action:
Display
clean.b0.residual–raster
22"
21"
20"
19"
18"
56°34'17"
J2000 Declination
12h52m26s.6
26s.4
26s.3
26s.2
26s.1
26s.0
J2000 Right Ascension
Cursors
clean.b0.residual-raster
clean.b0.mask


Viewer Display Panel (Am)
iterations
cycles
threshold
100
30
0 mJy
Add
Erase
This Channel
All Channels
This Polarization
All Polarizations
Next Action:
Display
Select clean regions
clean.b0.residual-raster
J2000 Declination
J2000 Right Ascension
Cursors
clean.b0.residual-raster
+0.00143307 Pixel: 152 276 0 0
12:52:26.421 +56.34.19.698 I 0 km/s (lsrk/radio velocity)
clean.b0.mask
+0 Pixel: 152 276 0 0
12:52:26.421 +56.34.19.698 I 0 km/s (lsrk/radio velocity)
Contours: -0.6 -0.2 0.2 0.6


Viewer Display Panel (Am)
iterations
cycles
threshold
100
30
0 mJy
Add
Erase
This Channel
All Channels
This Polarization
All Polarizations
Next Action:
Display
clean.b0.residual-raster
J2000 Declination
J2000 Right Ascension
Once areas defined, run a minor cycle
Cursors
clean.b0.residual-raster
+0.00252473 Pixel: 200 266 0 0
12:52:26.359 +56.34.19.591 I 0 km/s (lsrk/radio velocity)
clean.b0.mask
+1 Pixel: 200 266 0 0
12:52:26.359 +56.34.19.591 I 0 km/s (lsrk/radio velocity)
Contours: -0.6 -0.2 0.2 0.6


Viewer Display Panel (Am)
iterations
cycles
threshold
100
27
0 mJy
Add
Erase
This Channel
All Channels
This Polarization
All Polarizations
Next Action:
Display
clean.b0.residual-raster
parts of the source CLEANed away
Looks like new part of the source
J2000 Declination
J2000 Right Ascension
Cursors
clean.b0.residual-raster
+0.00482411 Pixel: 178 236 0 0
12:52:26.387 +56.34.19.276 I 0 km/s (lsrk/radio velocity)
clean.b0.mask
+0 Pixel: 178 236 0 0
12:52:26.387 +56.34.19.276 I 0 km/s (lsrk/radio velocity)
Contours: 0.2 0.4 0.6 0.8


Viewer Display Panel (Am)
iterations
cycles
threshold
1000
19
0 mJy
Add
Erase
This Channel
All Channels
This Polarization
All Polarizations
Next Action:
Display
clean.b0.residual-raster
Many noisy features
End interactive clean
20".5
20".0
19".5
19".0
56°34'18".5
J2000 Declination
12h52m26s.45
26s.35
26s.25
26s.15
J2000 Right Ascension
Cursors
clean.b0.residual-raster
-0.000259912 Pixel: 227 456 0 0
12:52:26.324 +56.34.21.628 I 0 km/s (lsrk/radio velocity)
clean.b0.mask
+0 Pixel: 227 456 0 0
12:52:26.324 +56.34.21.628 I 0 km/s (lsrk/radio velocity)
Contours: 0.2 0.4 0.6 0.8



We end clean when we think we have reached a reasonable noise limit.

Note that we have cleaned a total flux of ~0.45 Jy and the threshold is 0.0007 Jy (we will use these values for running CLEAN non-INTERACTIVELY).

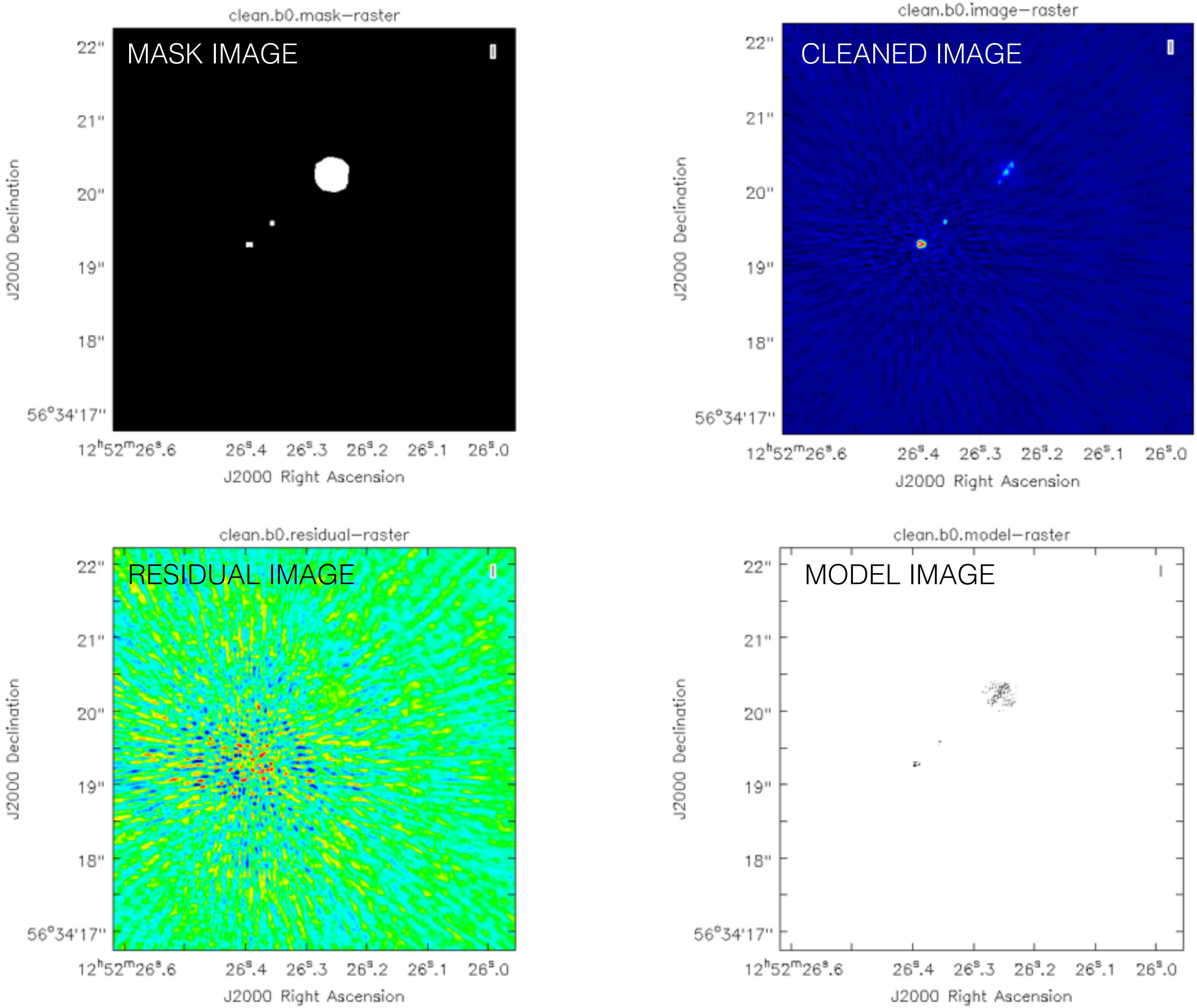
We have also generated a new file,

clean.b0.mask   # The mask image that defines the CLEAN regions #

Let's look at the final images using the VIEWER

```
> viewer        # start the viewer GUI and DATA MANAGER #
```

Load the RASTER map of the image, model, residual, mask.


clean.b0.mask-raster
MASK IMAGE
clean.b0.image-raster
CLEANED IMAGE
clean.b0.residual-raster
RESIDUAL IMAGE
clean.b0.model-raster
MODEL IMAGE
J2000 Declination
J2000 Right Ascension
22"
21"
20"
19"
18"
56°34'17"
12h52m26s.6
26s.4
26s.3
26s.2
26s.1
26s.0

All that seemed to work well, so lets add the parameters of our CLEAN run to our script. First, we add the threshold, give a new image name and set not to run interactively,

```
> tget clean                   # recover the last set of parameters used #
> imagename = “clean.b0.auto” # set new image name to make new file #
> interactive = F              # don’t allow interactive cleaning #
> threshold = “0.7mJy”         # set threshold to stop cleaning #
> mask = “clean.b0.mask”       # use of pre-defined mask #
> inp                          # review parameters #
> tput clean                   # save parameters to the .last file #

> !more clean.last
```

and copy the final part to our script (step 2).

```
# description of step
mystep = 1
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  clean(vis="1252+5634.ms",imagename="clean.b0.auto",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",uvrange="",
      antenna="",scan="",observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1,facets=1,cfcache="cfcach
      e.dir",rotpainc=5.0,painc=360.0,aterm=True,psterm=False,mterm=True,wbawp=False,conjbeams=True,epjtable="",interpolation="
      linear",niter=3000,gain=0.05,threshold="0.7mJy",psfmode="clark",imagermode="csclean",ftmachine="mosaic",mosweight=False,
      scaletype="SAULT",multiscale=[],negcomponent=-1,smallscalebias=0.6,interactive=False,mask="clean.b0.mask",nchan=-1,start=
      0,width=1,outframe="",veltype="radio",imsize=512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="bri
      ggs",robust=0,uvtaper=False,outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,
      usescratch=False,noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False
      ,flatnoise=True,allowchunk=False)
```

Lets try running everything using our script (this will overwrite our dirty images and make a new clean image). Depending on your computer, this should take about ~5 mins to run.

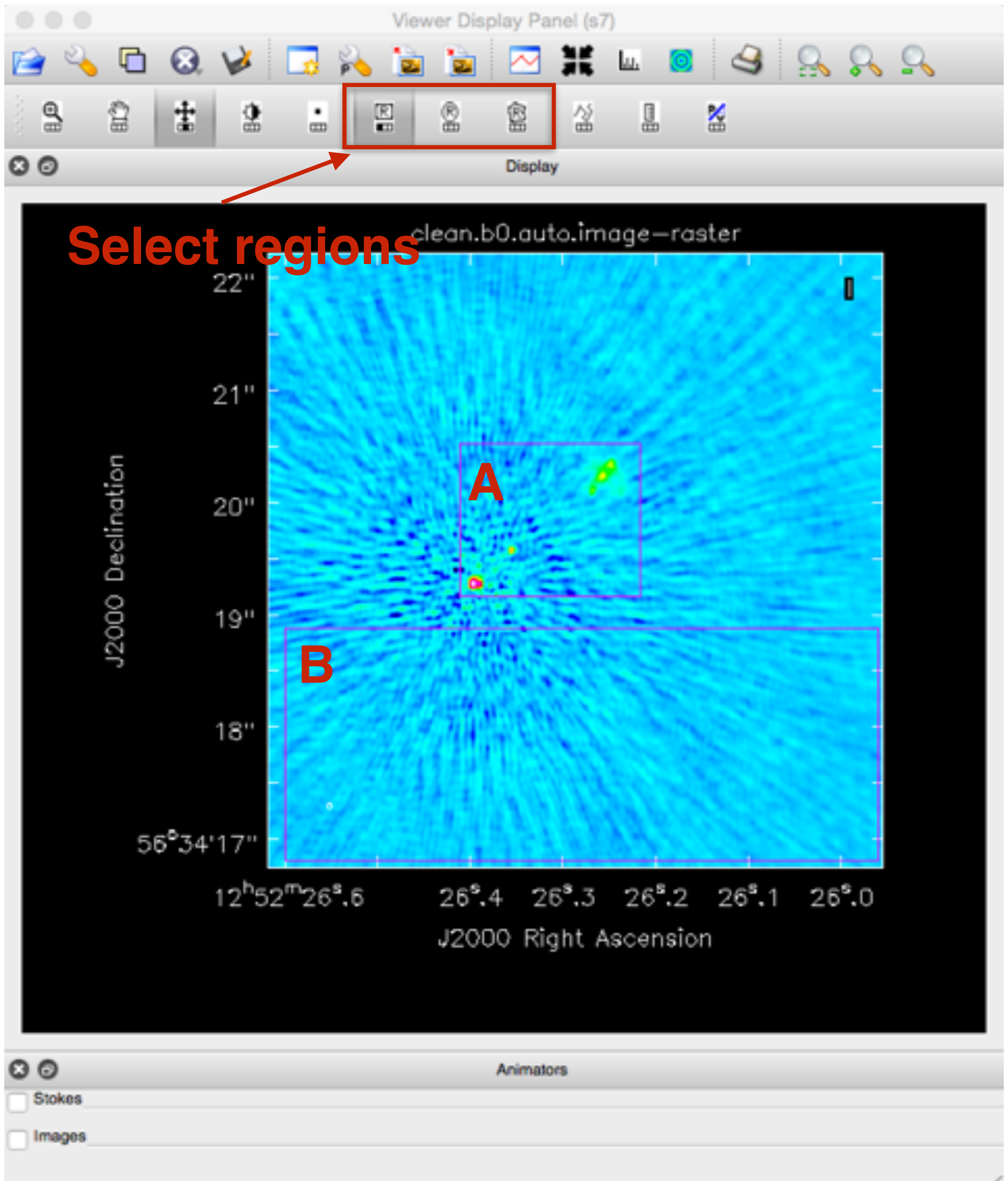
```
> mysteps = [0, 1]       # this will run step 0 #
> execfile(‘ScriptForImaging.py’)      # this run the script#
```

STEP 6 - Image properties

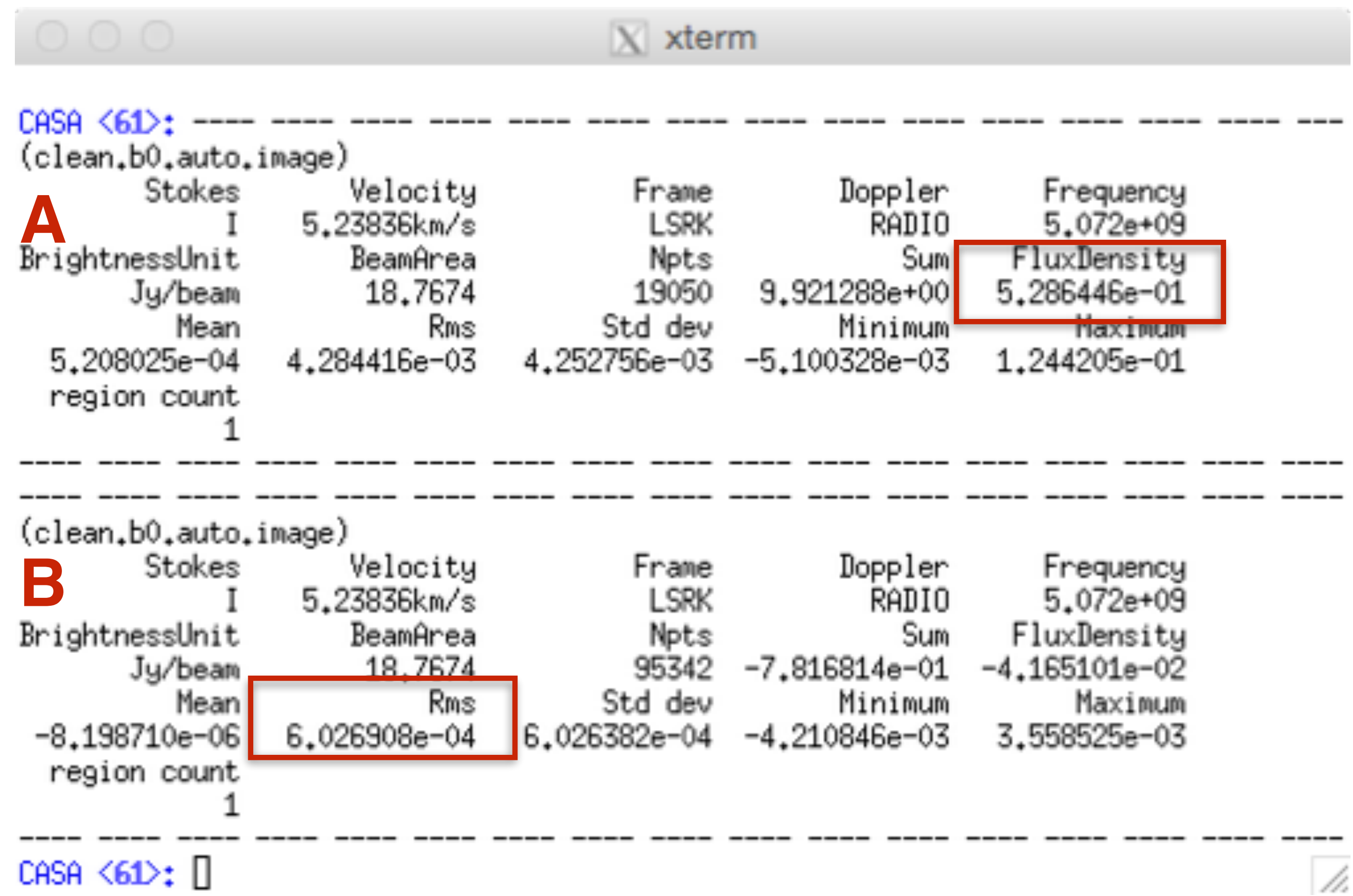
We can use the VIEWER to estimate some image statistics based on our new clean image.

```
> viewer      # start the viewer GUI and DATA MANAGER #
```

Load the RASTER “clean.b0.auto.image” map of the VIEWER.



Double click inside the regions to get the statistics.

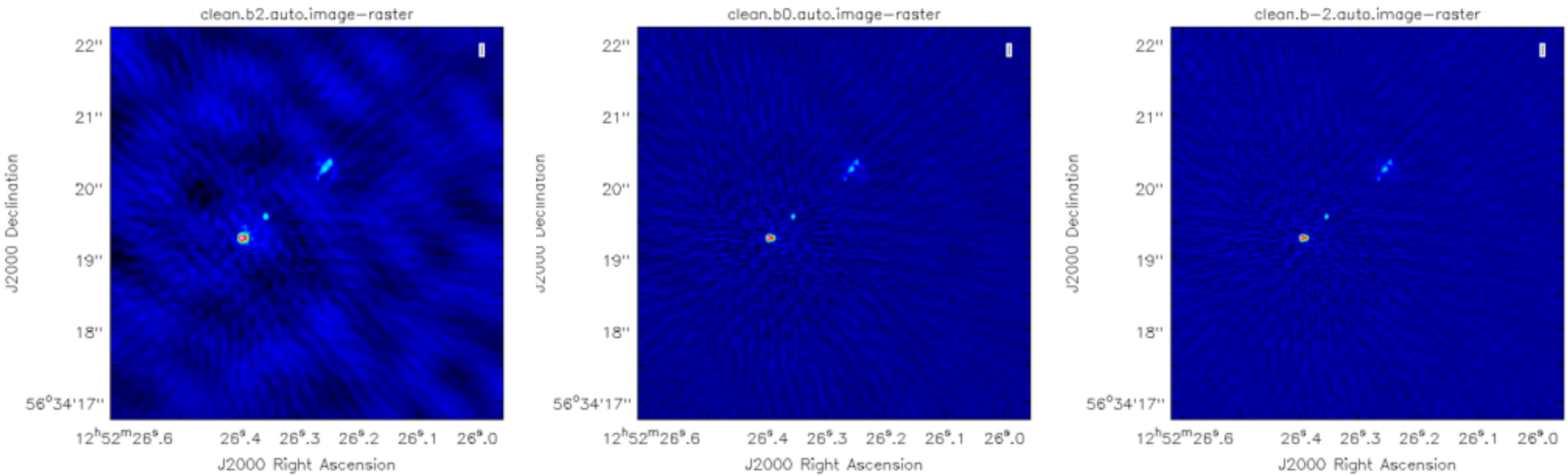


Note the flux-density of our target and the rms noise of the image

STEP 7 - Student exercise

Try making an image of the source using uniform and natural weighting (robust = 2 and -2), do this by making a new step 2 and 3 in your imaging script, and run it over lunch.

Remember to change the image name, otherwise you will overwrite your previous images.



Measure the flux-density and rms noise of each map, how do they compare.

What we find is that there are still strong image residuals post imaging. Where do these come from?

They are party due to residual phase and amplitude errors in the data.

Phase Error: Moves the source around, poor deconvolution
Amplitude Error: Results in a different psf than expected, poor deconvolution.

STEP 8 - Self-calibration

After transferring the solutions from a calibrator we may find that there are residual errors in our data.

Why?

Our calibrators are observed at a different time (except for simultaneous observations; in beam-calibration) and position on the sky than our target.

Use the process of self-calibration:

1) Make an image of your target (after applying calibrator solutions).
2) Use this model to calibrate the data over some solution interval.
3) Make an image of your target (after applying self-calibration solutions).
4) Use this model to calibrate the data over some solution interval.
5) Iterate this process until no major improvement on image quality.

Advantages:

1) Can correct for residual amplitude and phase errors.
2) Can correct for direction dependent effects (see later).

Disadvantages:

1) Errors in the model or low SNR can propagate into your self-calibration solutions, and you can diverge from the correct model.

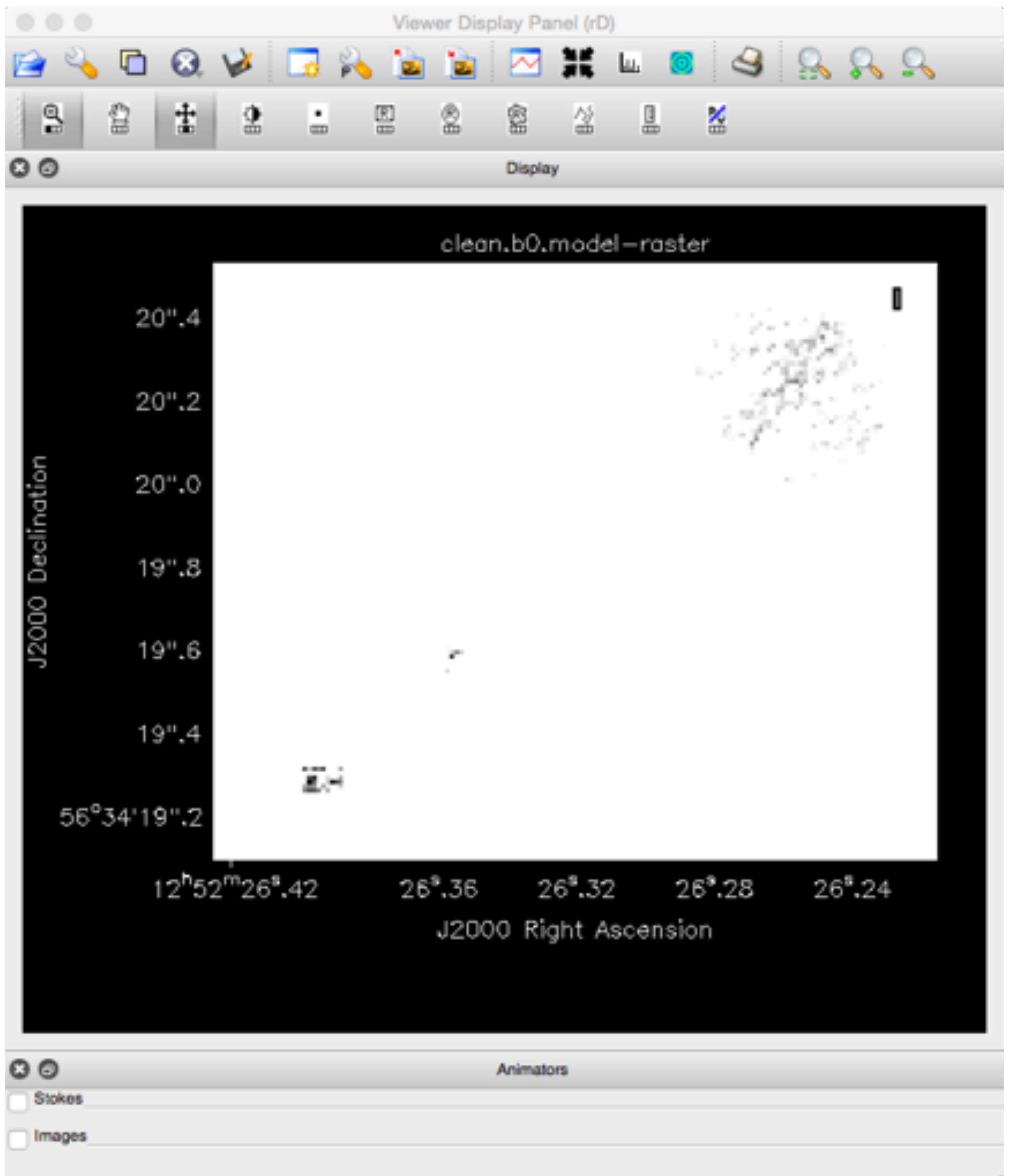
We will use our model for the source that we made during the previous clean process.

Our first step is to blank the MODEL COLUMN of our MS file, to limit any problems from previous work.

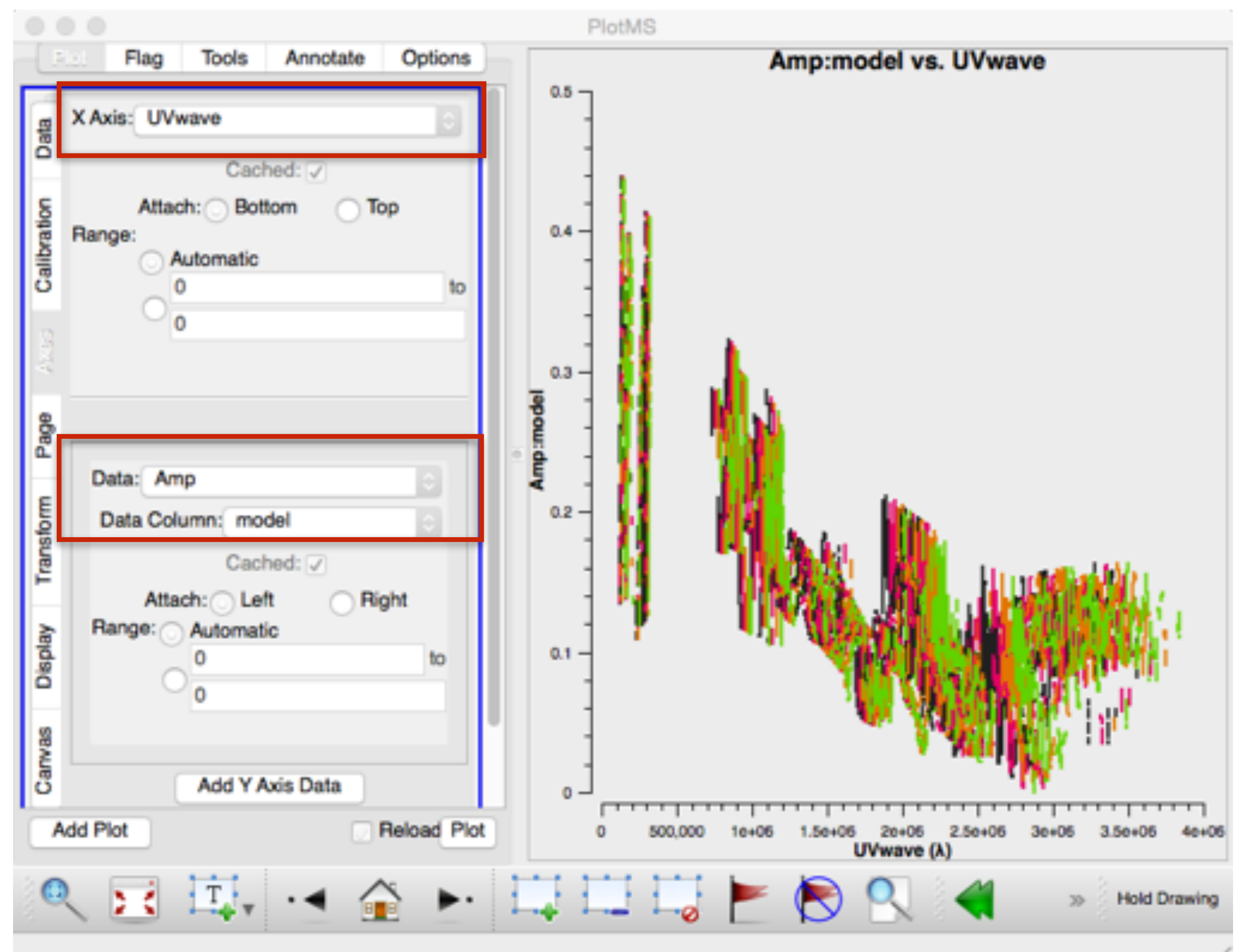
```
> tget clearcal          # recover the last set of parameters used #
> vis = “1252+5634.ms” # select the visibility dataset #
> inp                    # review parameters #
> go clearcal            # start FFT #
```

We will use the FT task to take the FFT of our model and generate a set of model visibilities. Note that CLEAN can do this automatically for you.

```
> mymodel = "clean.b0.model" # set alias to point to best current model #
> tget ft                    # recover the last set of parameters used #
> model = mymodel            # set the input model to my model #
> usescratch = T             # write the model visibilities into the MODEL column #
> inp                        # review parameters #
> go ft                      # start FFT and sampling #
```



Plot the model visibilities (avgchannel = 64), colourise by SPW



Lets add this to our script (a new step)

> `!more ft.last`

and copy the final part to our script (step 4).

ScriptForImaging

ScriptForImaging.py 〉 No Selection

```
# e-MERLIN imaging script for J1252+5634 (4 spws x 64 channels) in CASA 4.4.0

#Calibration steps
thesteps = [0]
step_title = {0: 'Make dirty image (clean)',
              1: 'Make clean image (clean)',
              2: 'Make natural weighting clean image (clean)',
              3: 'Make uniform weighting clean image (clean)',
              4: 'Insert intial model (ft)'}

try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
```

Add new step (4 is you did the homework, 2 if you did not)

```
"linear",niter=3000,gain=0.05,threshold="0.7mJy",psfmode="clark",imagermode="csclean",ftmachine="mosaic",mosweight=False,
scaletype="SAULT",multiscale=[],negcomponent=-1,smallscalebias=0.6,interactive=False,mask="clean.b0.mask",nchan=-1,start=
0,width=1,outframe="",veltype="radio",imsize=512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="bri
ggs",robust=-2,uvtaper=False,outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,
usescratch=False,noise="1.0Jy",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False
,flatnoise=True,allowchunk=False)


# description of step
mystep = 4
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  mymodel = 'clean.b0.model'

  ft(vis="1252+5634.ms",field="",spw="",model=mymodel,nterms=1,reffreq="",complist="",incremental=False,usescratch=True)
```

Here we enter our variables

Here we enter the ft parameters

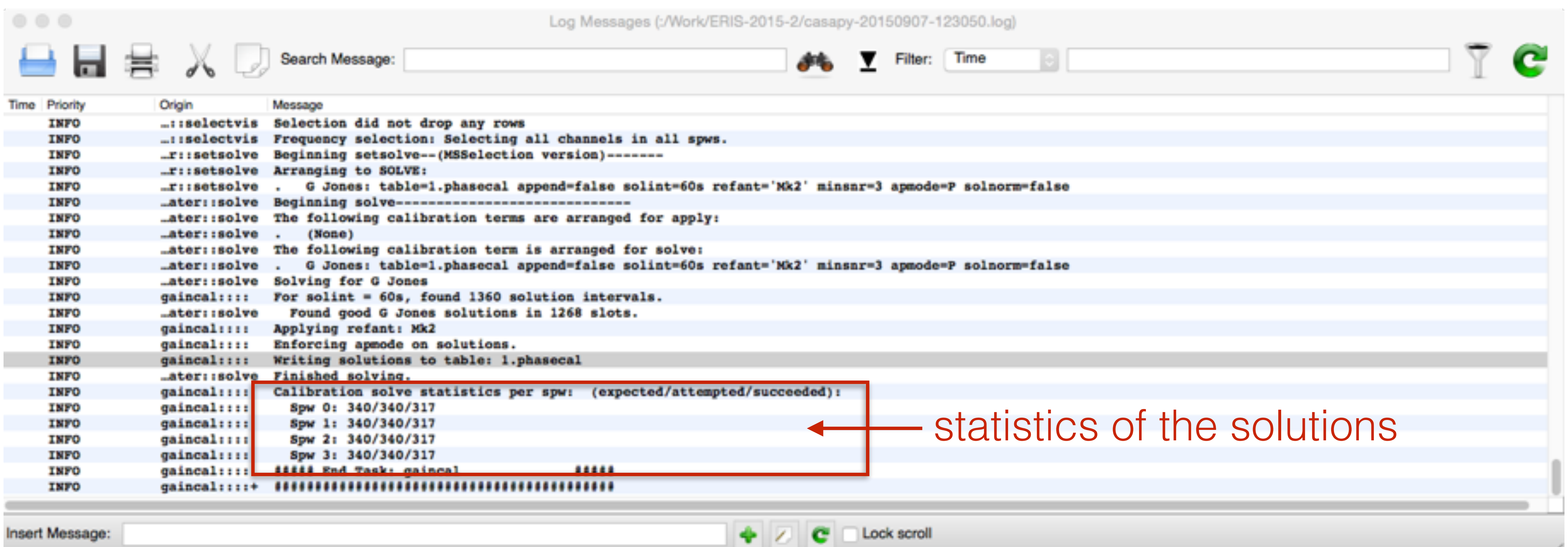
We will first carry out PHASE-ONLY self-calibration using this model. Remember,

An error in your model can be absorbed in the calibration

$$\vec{V}_{ij} = J_{ij}\vec{V}_{ij}^{\mathrm{IDEAL}}$$

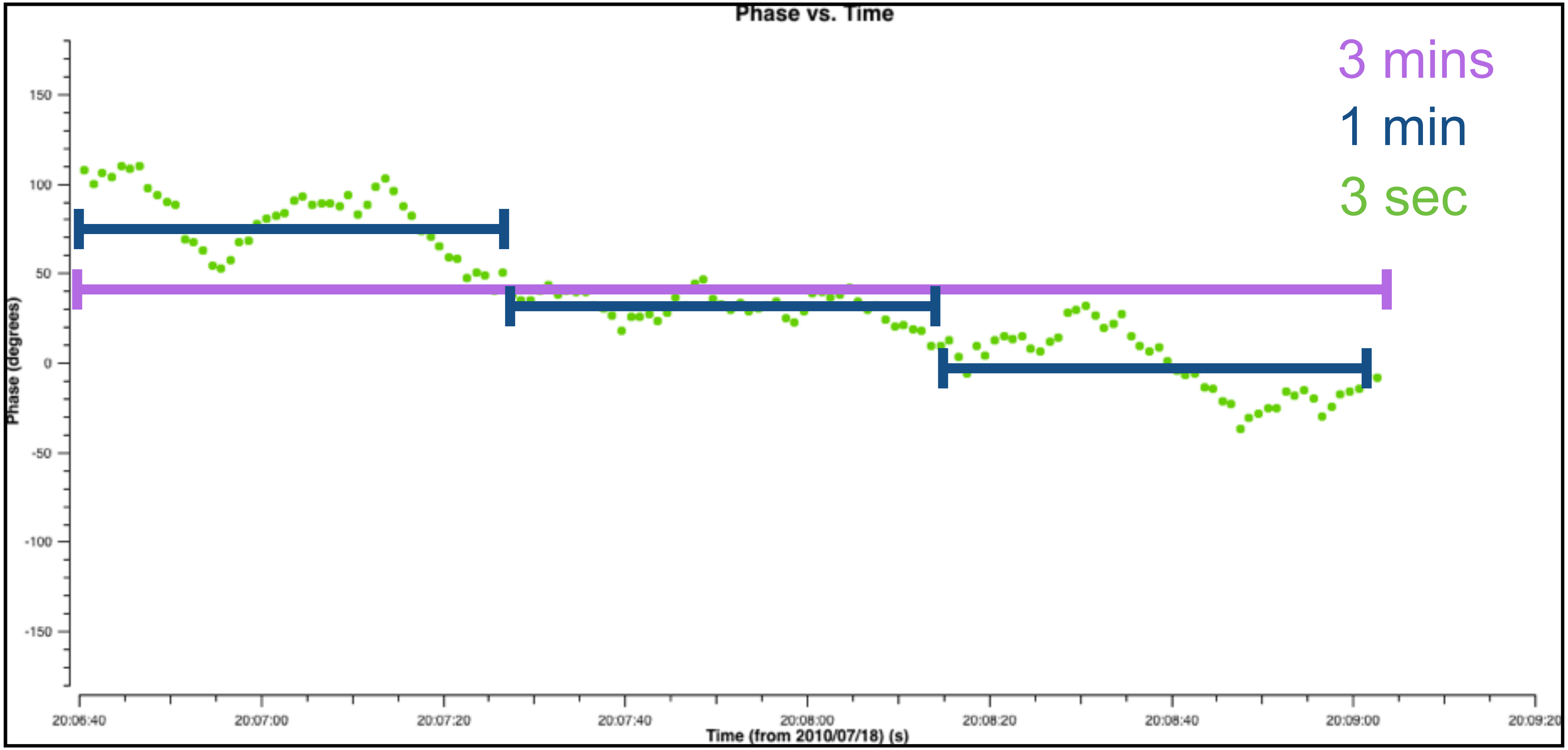
Our model will be used to determine a new calibration table which will describe the phase variations as a function of time.

```
> default gaincal            # reset the calibration parameters #
> vis = "1252+5634.ms"       # select the visibility dataset #
> caltable = "1.phasecal"    # make a new calibration table #
> solint = "60s"             # we will start by using a solution interval of 60 s #
> refant = "Mk2"             # select MarkII as the reference antenna #
> calmode = "p"              # phase-only self cal #
> inp                        # review parameters #
> go gaincal                 # start FFT #
```



statistics of the solutions

What is an appropriate solution time?

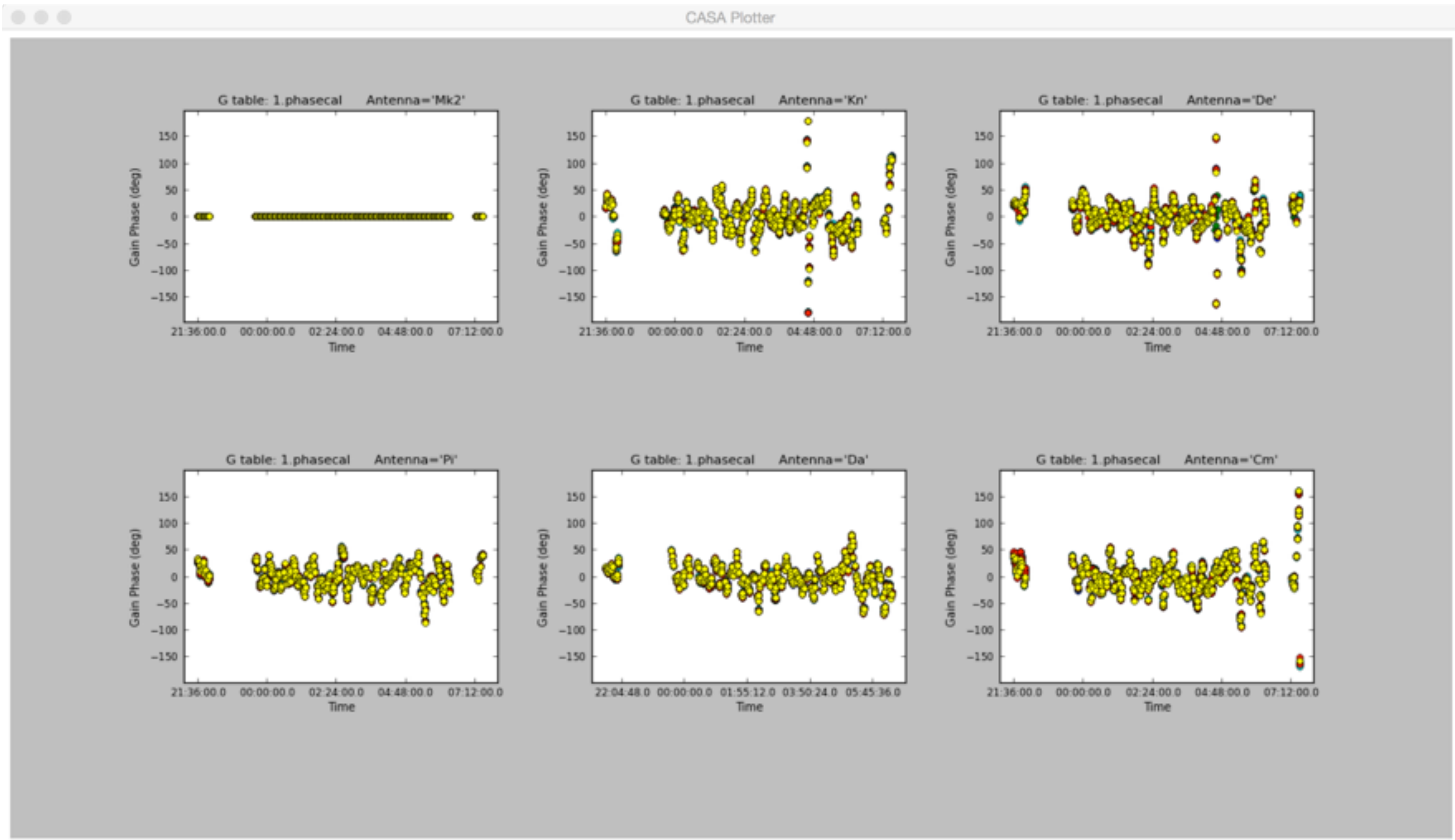


Want to have,

Shortest possible time-scale to **track** the gain variations, whist being **long enough** to have a sufficient **signal-to-noise ratio.**

Lets look at the quality of the solutions

```
> default plotcal                  # reset the plotting parameters #
> caltable = “1.phasecal”          # use our new calibration table #
> xaxis = “time”                   # plot as a function of time #
> yaxis = “phase”                  # plot the phase solutions #
> subplot = 231                    # plot the 6 antennas on one plot #
> plotrange = [-1,-1,-180,180]     # plot all time and +/- 180 deg #
> iteration = “antenna”            # lets look at each antenna sep #
> go gaincal                       # determine calibration parameters #
```



As the solutions look quite good, lets apply them to the data,

```
> default applycal          # reset the calibration parameters #
> vis = "1252+5634.ms"      # select the visibility dataset #
> gaintable = "1.phasecal" # select tables to apply (in correct order) #
> calwt = F                 # lets not calibrate the weights… #
> inp                       # review parameters #
> go applycal               # apply calibration tables and write CORRECTED column #
```

Now we can make a new image and model for the source using CLEAN (non-interactively).

```
> tget clean                   # recover the last set of parameters used #
> imagename = "clean.b0.self" # set new image name to make new file #
> robust = 0                   # set robust parameter to 0 #
> interactive = F              # don't allow interactive cleaning #
> threshold = "0.7mJy"         # set threshold to stop cleaning #
> mask = "clean.b0.mask"       # use of pre-defined mask #
> usescratch = T               # write the model visibilities to MODEL column #
> inp                          # review parameters #
> go clean                     # start deconvolution #
```

At this point, we have now completed a self-calibration loop (GAINCAL -> APPLYCAL -> CLEAN), this will be step 5 of our script.

Lets add these task to a new step of our script.

```
> !more gaincal.last           # start deconvolution #
> !more applycal.last          # start deconvolution #
> !more clean.last             # start deconvolution #
```

add `os.system('rm -rf clean.b0.self.*')` to the first part of the step.

ScriptForImaging

ScriptForImaging.py 〉 No Selection

```
# e-MERLIN imaging script for J1252+5634 (4 spws x 64 channels) in CASA 4.4.0

#Calibration steps
thesteps = [0]
step_title = {0: 'Make dirty image (clean)',
              1: 'Make clean image (clean)',
              2: 'Make natural weighting clean image (clean)',
              3: 'Make uniform weighting clean image (clean)',
              4: 'Insert intial model (ft)',
              5: 'Phase-only self-calibration loop (gaincal, applycal, clean)'}

try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
```

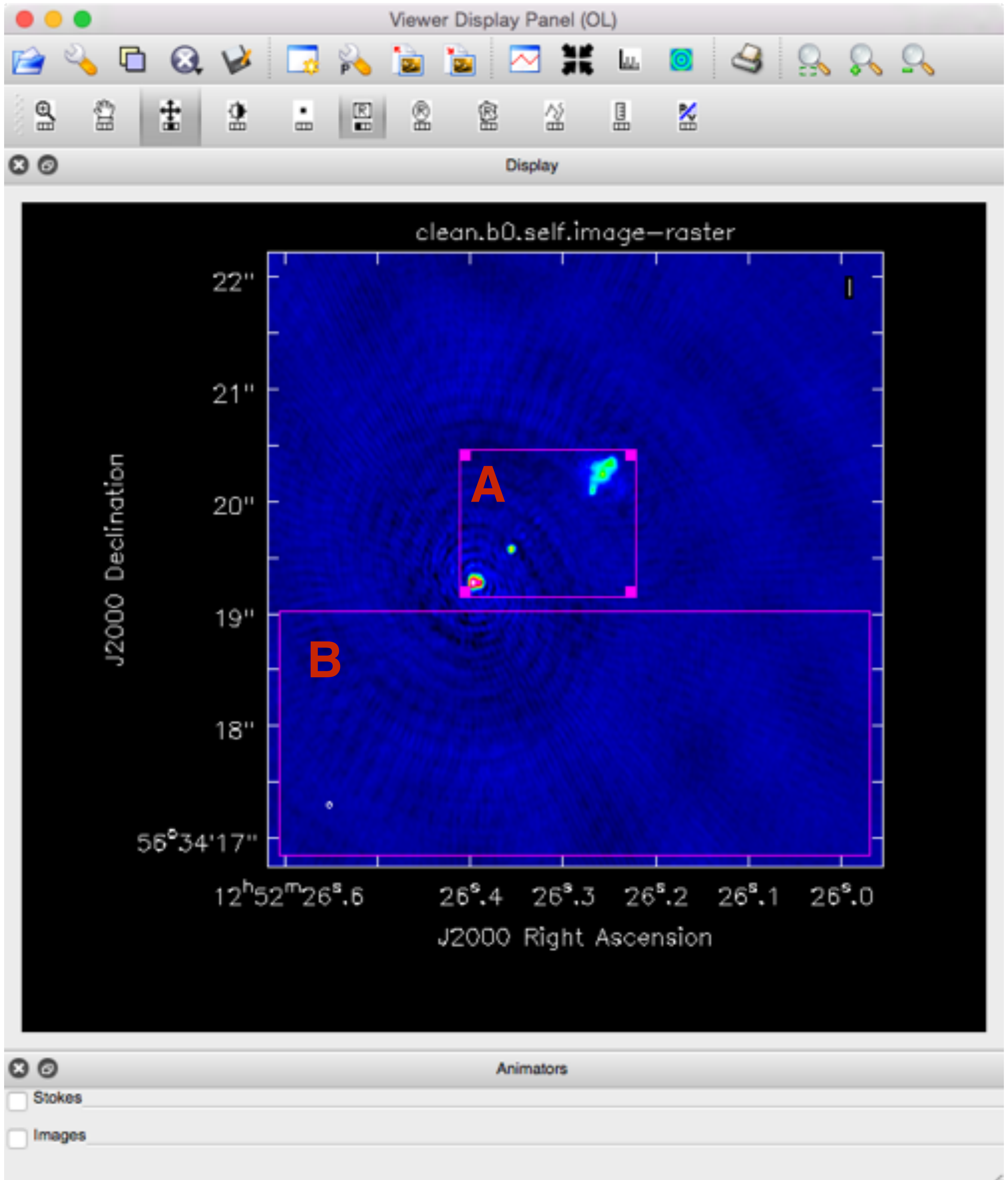
Add new step 5

```
  mymodel = 'clean.b0.model'

  ft(vis="1252+5634.ms",field="",spw="",model=mymodel,nterms=1,reffreq="",complist="",incremental=False,usescratch=True)


# description of step
mystep = 5
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  os.system('rm -rf clean.b0.self.*')
  mysolint = '60s'

  gaincal(vis="1252+5634.ms",caltable="1.phasecal",field="",spw="",intent="",selectdata=True,timerange="",uvrange="",antenna="",scan="",observation="",
      msselect="",solint=mysolint,combine="",preavg=-1.0,refant="Mk2",minblperant=4,minsnr=3.0,solnorm=False,gaintype="G",smodel=[],calmode="p",append=
      False,splinetime=3600.0,npointaver=3,phasewrap=180.0,docallib=False,callib="",gaintable=[],gainfield=[],interp=[],spwmap=[],parang=False)

  applycal(vis="1252+5634.ms",field="",spw="",intent="",selectdata=True,timerange="",uvrange="",antenna="",scan="",observation="",msselect="",docallib=
      False,callib="",gaintable=['1.phasecal'],gainfield=[],interp=[],spwmap=[],calwt=False,parang=False,applymode="",flagbackup=True)

  clean(vis="1252+5634.ms",imagename="clean.b0.self",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",uvrange="",antenna="",scan="",
      observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,
      psterm=False,mterm=True,wbawp=False,conjbeams=True,epjtable="",interpolation="linear",niter=3000,gain=0.05,threshold="0.7mJy",psfmode="clark",
      imagermode="csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",multiscale=[],negcomponent=-1,smallscalebias=0.6,interactive=False,mask="c
      lean.b0.mask",nchan=-1,start=0,width=1,outframe="",veltype="radio",imsize=512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="b
      riggs",robust=0,uvtaper=False,outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=True,noise="1.0Jy
      ",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,flatnoise=True,allowchunk=False)
```

Here we enter our variables

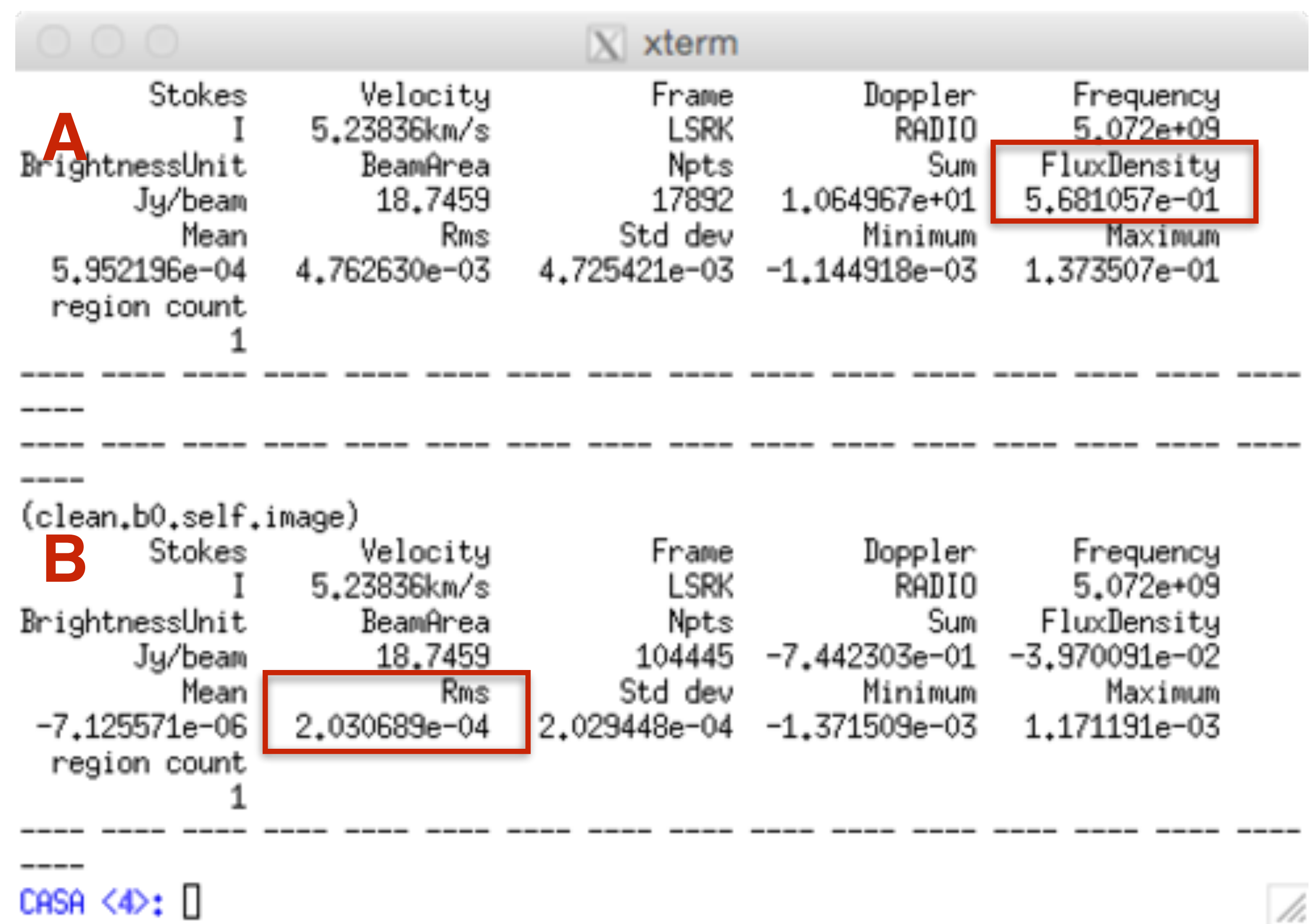
Lets look at our first self-calibrated image

```
> viewer
```



Double click inside the regions to get the statistics.

We find that self-calibration has lowered the noise, and increased the removed flux of the sources



```
A
        Stokes        Velocity           Frame         Doppler       Frequency
             I     5.23836km/s            LSRK           RADIO       5.072e+09
BrightnessUnit        BeamArea            Npts             Sum     FluxDensity
       Jy/beam         18.7459           17892    1.064967e+01    5.681057e-01
          Mean             Rms         Std dev         Minimum         Maximum
  5.952196e-04    4.762630e-03    4.725421e-03   -1.144918e-03    1.373507e-01
  region count
             1
---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
----
---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
----
(clean.b0.self.image)
B
        Stokes        Velocity           Frame         Doppler       Frequency
             I     5.23836km/s            LSRK           RADIO       5.072e+09
BrightnessUnit        BeamArea            Npts             Sum     FluxDensity
       Jy/beam         18.7459          104445   -7.442303e-01   -3.970091e-02
          Mean             Rms         Std dev         Minimum         Maximum
 -7.125571e-06    2.030689e-04    2.029448e-04   -1.371509e-03    1.171191e-03
  region count
             1
---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
----
CASA <4>:
```

STEP 9 - Self-calibration loops (Phase)

We will now attempt a self-calibration loop using our script.

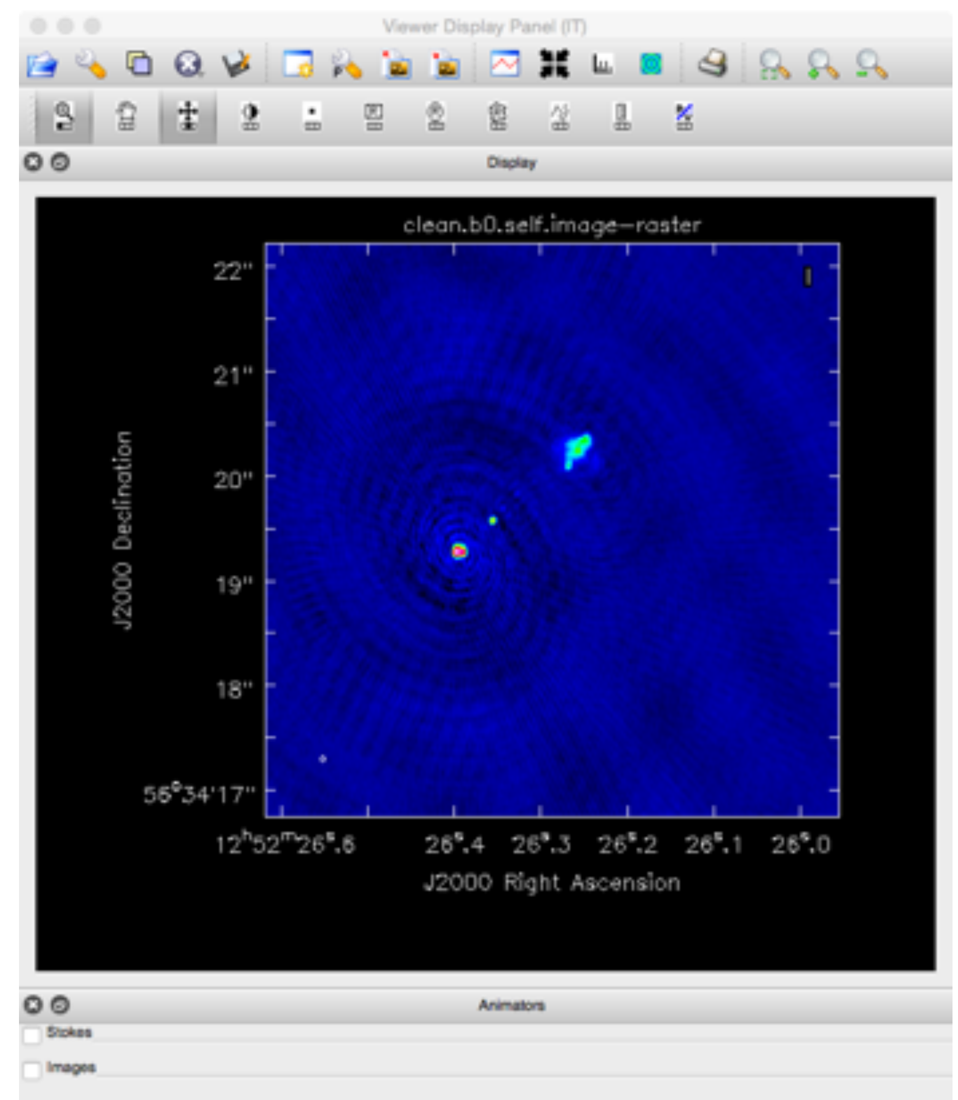
```
> mysteps = [5]                    # this will run step 5 only #
> execfile('ScriptForImaging.py') # this run the script#
```

** If you see an error, it is likely due to SYNTAX issues. Make sure that you have the correct indentation for each step (double space), see the line of the script that reports the error message **

** Also make sure that you are running only STEP 5 - if another step is running, then it means that you haven't indent the commands within other steps correctly. **

Lets look at our next self-calibrated image

```
> viewer
```



This looks similar to before, how does the noise and peak flux compare?

Lets try a final phase-only self-calibration but, lets change of script to use a shorter solution interval, i.e. track phase changes on shorter time-scales, but now we have a better model.

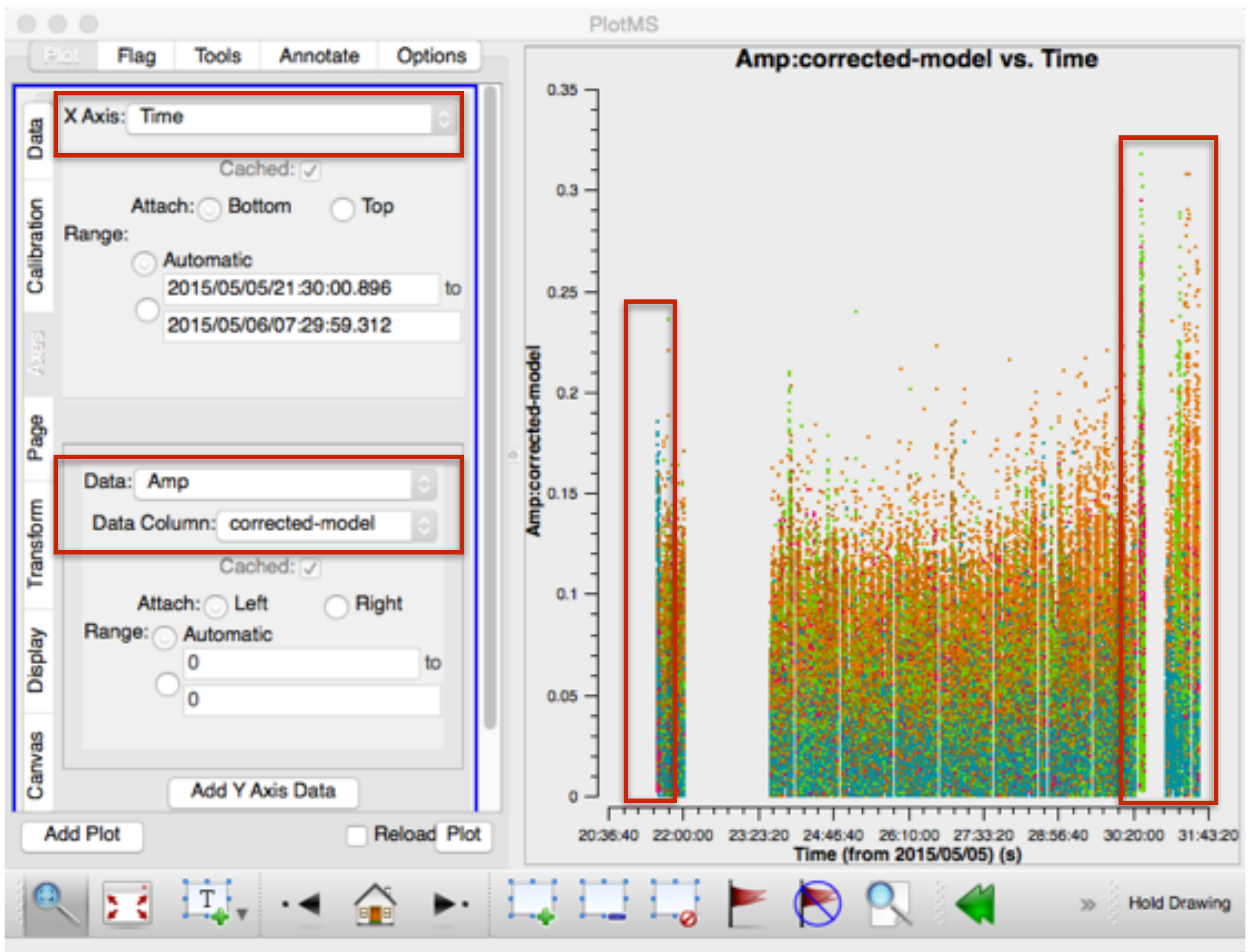
```
mysolint = '30s'
```

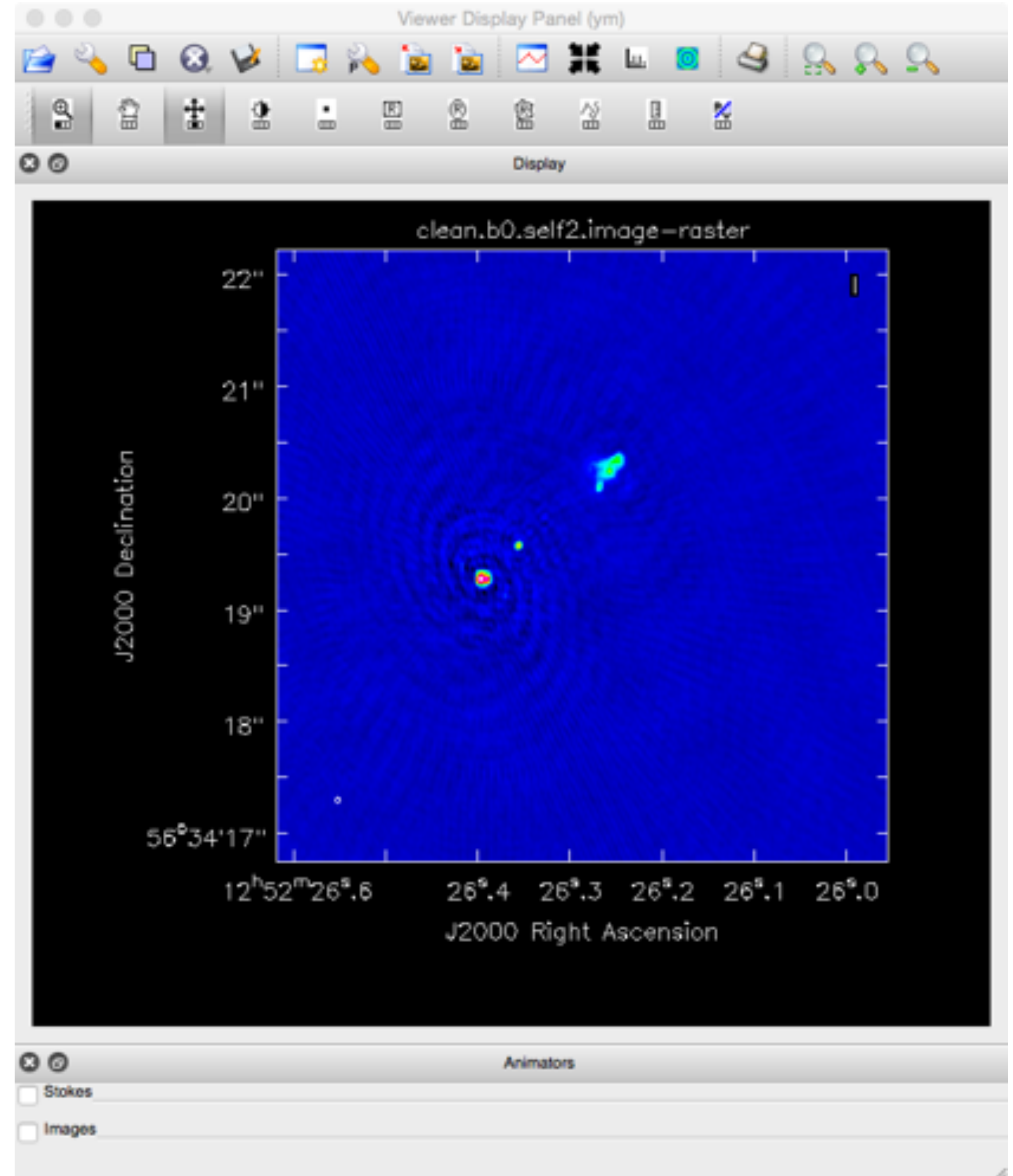
```
> mysteps = [5]
> execfile('ScriptForImaging.py')
```

We are not seeing any major improvement, could our problems be due to bad data, lets check

Lets inspect the residual visibilities,

```
> default plotms
> vis = "1252+5634.ms"
> correlation = 'rr, ll'
> avgchannel = '64'
> inp
> plotms
```





Residual visibilities are corrected - model

It looks like there are some time ranges when the amplitudes increase

1) 21:30:00 to 21:32:00
2) 30:20:00 to 32:00:00

Lets flag these time ranges

STEP 10 - Post-calibration flagging

We will now flag the bad time ranges

```
> default flagdata                  # restore the parameters to defaults #
> vis = “1252+5634.ms”              # select the visibility dataset #
> timerange = “21:30:00~21:32:00” # select the time range to flag #
> inp                               # review parameters #
> go flagdata                       # carry out flagging #
```

Add this to our script, but at which point? (at the beginning)

```
> !more flagdata.last
```

```
# description of step
mystep = 0
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

    flagdata(vis="1252+5634.ms",mode="manual",autocorr=False,inpfile="",reason="any",tbuff=0.0,spw="",field="",antenna="",uvrange="",timerange="21:30:00
    ~21:32:00",correlation="",scan="",intent="",array="",observation="",feed="",clipminmax=[],datacolumn="DATA",clipoutside=True,channelavg=False,
    clipzeros=False,quackinterval=1.0,quackmode="beg",quackincrement=False,tolerance=0.0,addantenna="",lowerlimit=0.0,upperlimit=90.0,ntime="scan",
    combinescans=False,timecutoff=4.0,freqcutoff=3.0,timefit="line",freqfit="poly",maxnpieces=7,flagdimension="freqtime",usewindowstats="none",halfwin=1
    ,extendflags=True,winsize=3,timedev="",freqdev="",timedevscale=5.0,freqdevscale=5.0,spectralmax=1000000.0,spectralmin=0.0,extendpols=True,growtime=
    50.0,growfreq=50.0,growaround=False,flagneartime=False,flagnearfreq=False,minrel=0.0,maxrel=1.0,minabs=0,maxabs=-1,spwchan=False,spwcorr=False,
    basecnt=False,name="Summary",action="apply",display="",flagbackup=True,savepars=False,cmdreason="",outfile="")

    clean(vis="1252+5634.ms",imagename="dirty.b0",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",uvrange="",antenna="",scan="",
    observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,
    psterm=False,mterm=True,wbawp=False,conjbeams=True,epjtable="",interpolation="linear",niter=0,gain=0.1,threshold="0.0mJy",psfmode="clark",imagermode
```

```
> tget flagdata                     # restore the parameters from last usage #
> timerange = “30:20:00 to 32:00:00” # select the time range to flag #
> inp                               # review parameters #
> go flagdata                       # carry out flagging #
```

Add this to our script.

```
> !more flagdata.last
```

STEP 11 - Self-calibration (Amp)

Lets try a loop of amplitude self-calibration to fix the residual amplitude errors

```
> tget gaincal                # recover the last set of parameters used #
> caltable = “1.ampcal”       # make a new calibration table #
> solint = “inf”              # we will use a very large solution interval
> combine = “scan”              that spans over all scans #
> calmode = “a”               # amplitude-only self-cal #
> gaintable = [“1.phasecal”]  # apply previous phase solutions #
> inp                         # review parameters #
> go gaincal                  # determine calibration parameters #

> tget applycal               # recover the last set of parameters used #
> gaintable = [“1.phasecal”,“1.ampcal”] # select tables to apply #
> inp                         # review parameters #
> go applycal                # apply calibration tables and write CORRECTED column #
```

Now we can make a new image and model for the source using CLEAN (non-interactively).

```
> tget clean                    # recover the last set of parameters used #
> imagename = “clean.b0.self2”  # set new image name to make new file #
> interactive = T               # don’t allow interactive cleaning #
> threshold = “”                # stop cleaning when based on the residuals #
> inp                           # review parameters #
> go clean                      # start deconvolution #
```

We will now go through a process of interactive CLEAN to make our next map.

At this point, we have now completed a self-calibration loop (GAINCAL -> APPLYCAL -> CLEAN), this will be step 6 of our script.

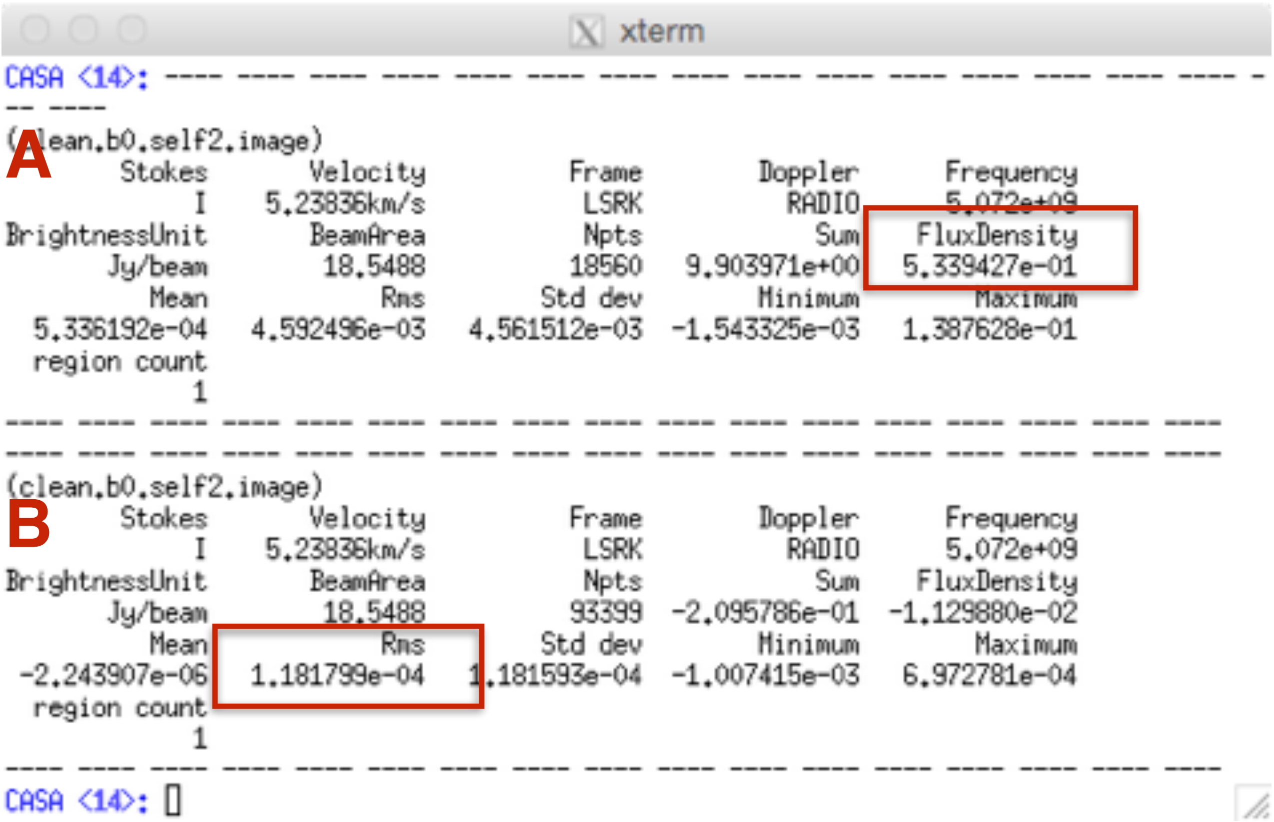
ScriptForImaging — Edited

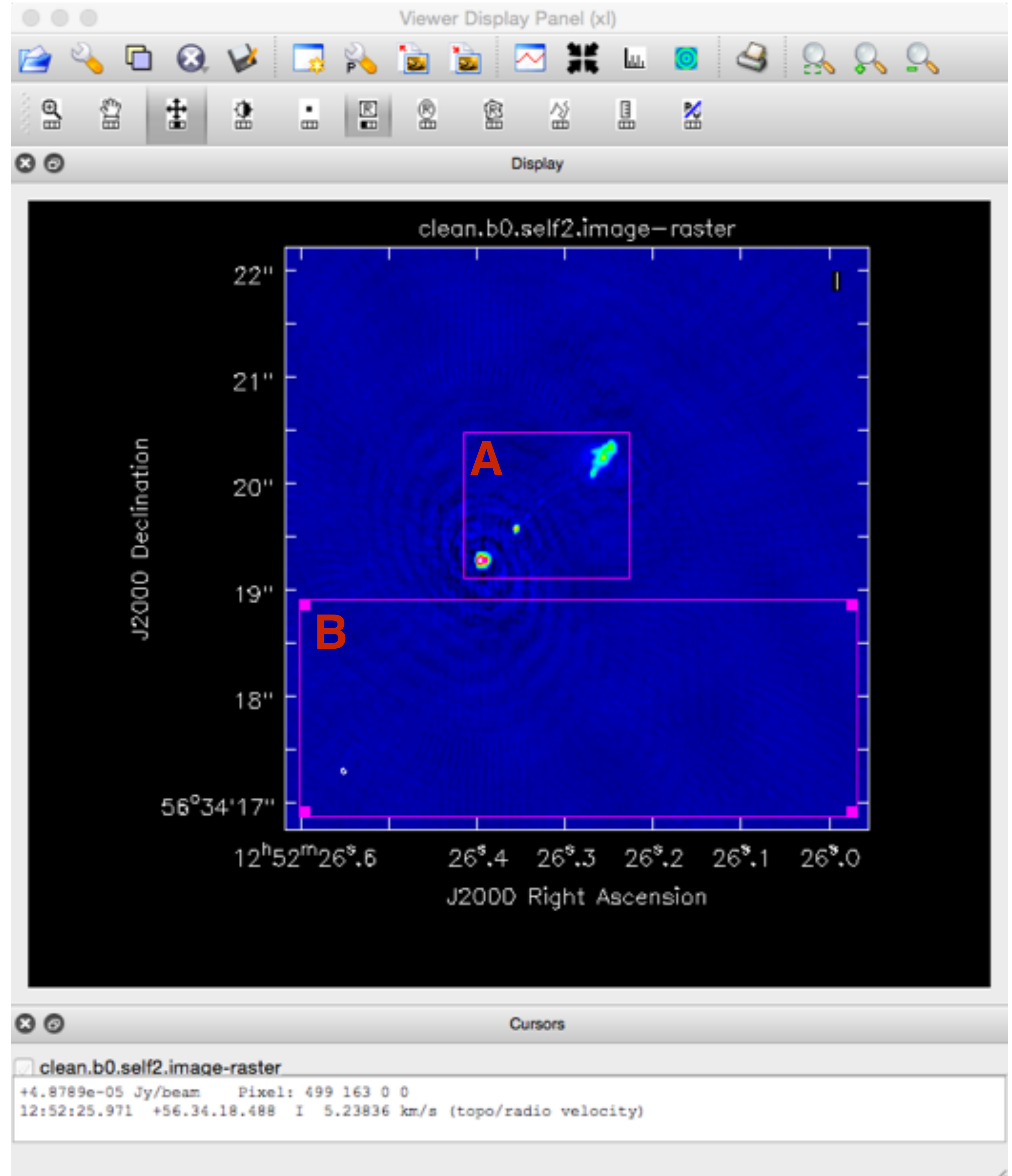
ScriptForImaging.py 〉 No Selection

```
# e-MERLIN imaging script for J1252+5634 (4 spws x 64 channels) in CASA 4.4.0

#Calibration steps
thesteps = [0]
step_title = {0: 'Make dirty image (clean)',
              1: 'Make clean image (clean)',
              2: 'Make natural weighting clean image (clean)',
              3: 'Make uniform weighting clean image (clean)',
              4: 'Insert intial model (ft)',
              5: 'Phase-only self-calibration loop (gaincal, applycal, clean)',
              6: 'Amplitude-only self-calibration loop (gaincal, applycal, clean)'}

try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
  print 'global variable mysteps not set.'
```

Add new step 6

```
    ",npixels=0,npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,flatnoise=True,allowchunk=False)


# description of step
mystep = 6
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  os.system('rm -rf clean.b0.self2.*')
  mysolint = 'inf'

  gaincal(vis="1252+5634.ms",caltable="1.ampcal",field="",spw="",intent="",selectdata=True,timerange="",uvrange="",antenna="",scan="",observation="",
      msselect="",solint=mysolint,combine="scan",preavg=-1.0,refant="Mk2",minblperant=4,minsnr=3.0,solnorm=False,gaintype="G",smodel=[],calmode="a",append
      =False,splinetime=3600.0,npointaver=3,phasewrap=180.0,docallib=False,callib="",gaintable=[],gainfield=[],interp=[],spwmap=[],parang=False)

  applycal(vis="1252+5634.ms",field="",spw="",intent="",selectdata=True,timerange="",uvrange="",antenna="",scan="",observation="",msselect="",docallib=
      False,callib="",gaintable=['1.phasecal'],gainfield=[],interp=[],spwmap=[],calwt=False,parang=False,applymode="",flagbackup=True)

  clean(vis="1252+5634.ms",imagename="clean.b0.self2",outlierfile="",field="",spw="0~3",selectdata=True,timerange="",uvrange="",antenna="",scan="",
      observation="",intent="",mode="mfs",resmooth=False,gridmode="",wprojplanes=-1,facets=1,cfcache="cfcache.dir",rotpainc=5.0,painc=360.0,aterm=True,
      psterm=False,mterm=True,wbawp=False,conjbeams=True,epjtable="",interpolation="linear",niter=3000,gain=0.05,threshold="",psfmode="clark",imagermode="
      csclean",ftmachine="mosaic",mosweight=False,scaletype="SAULT",multiscale=[],negcomponent=-1,smallscalebias=0.6,interactive=True,mask="clean.b0.mask"
      ,nchan=-1,start=0,width=1,outframe="",veltype="radio",imsize=512,cell="0.0107arcsec",phasecenter="",restfreq="",stokes="I",weighting="briggs",robust
      =0,uvtaper=False,outertaper=[''],innertaper=['1.0'],modelimage="",restoringbeam=[''],pbcor=False,minpb=0.2,usescratch=True,noise="1.0Jy",npixels=0,
      npercycle=100,cyclefactor=1.5,cyclespeedup=-1,nterms=1,reffreq="",chaniter=False,flatnoise=True,allowchunk=False)
```





Our dynamic range is peak / rms ~ 1000, which is limited by deconvolution errors in the complex bright component.

Further careful imaging (with a smaller cell size) my improve this.

Our final model should be copied into our script, at step 4,

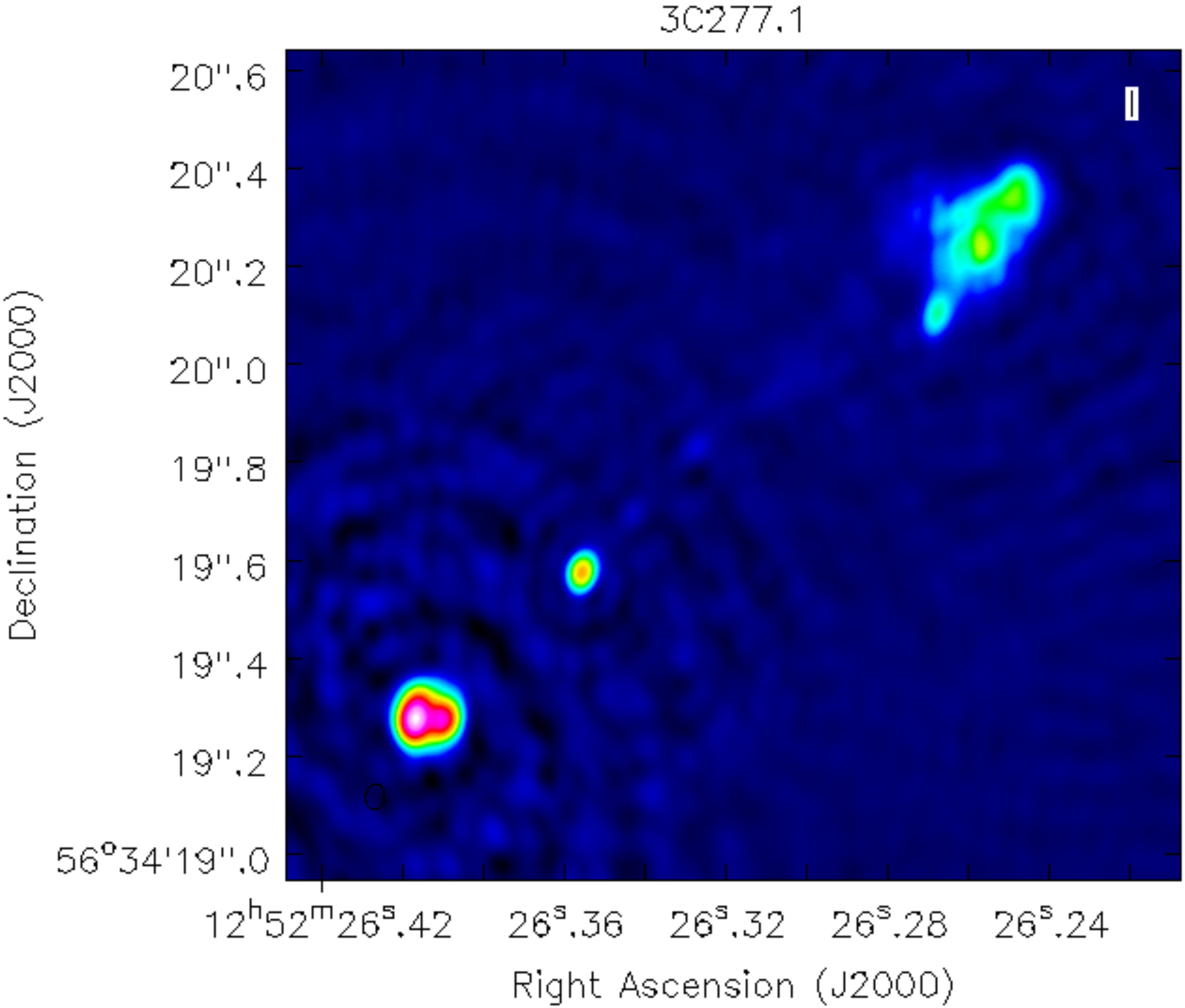
```
> ! cp -rf clean.b0.self2.model best-model.model
> ! cp -rf clean.b0.self2.mask best-mask.mask
```

```
# description of step
mystep = 4
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]

  mymodel = 'best-model.model'

  ft(vis="1252+5634.ms",field="",spw="",model=mymodel,nterms=1,reffreq="",complist="",incremental=False,
      usescratch=True)


# description of step
mystep = 5
if(mystep in thesteps):
  casalog.post('Step '+str(mystep)+' '+step_title[mystep],'INFO')
  print 'Step ', mystep, step_title[mystep]
```

STEP 12 - Student exercise

1) Make a new directory
2) Copy the 1252+5634.ms.tar file to the directory and untar it (cp … tar xvf)
3) Copy the best-model.model file to the directory (cp -rf …)
4) Copy the best-mask.mask file to the directory (cp -rf …)
5) Copy the ScriptForImaging.py file to the directory (cp …)
6) Change to the directory (cd)
7) Rename best-mask.mask to clean.b0.mask (mv)

Start casapy and run the script using all steps.



3C277.1
I
20''.6
20''.4
20''.2
20''.0
19''.8
19''.6
19''.4
19''.2
56°34'19''.0
Declination (J2000)
12h52m26s.42
26s.36
26s.32
26s.28
26s.24
Right Ascension (J2000)