Tutorial 4 - LOFAR (simple imaging)

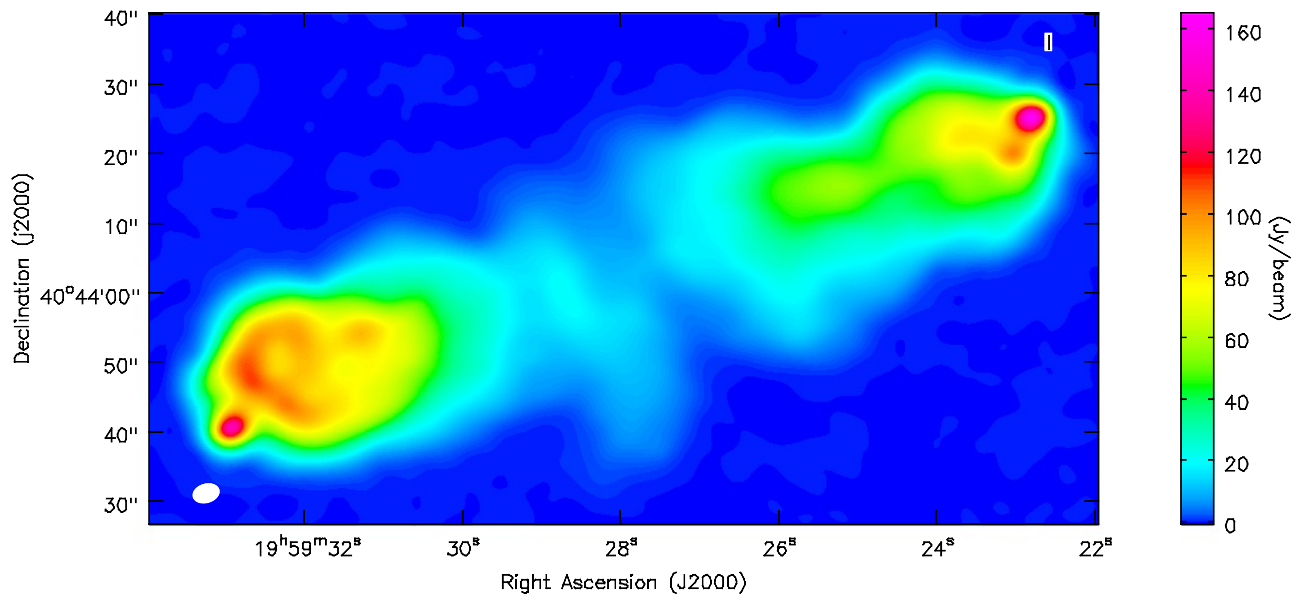
Aim: To inspect and reduce a LOFAR dataset of Cygnus A using CASA.

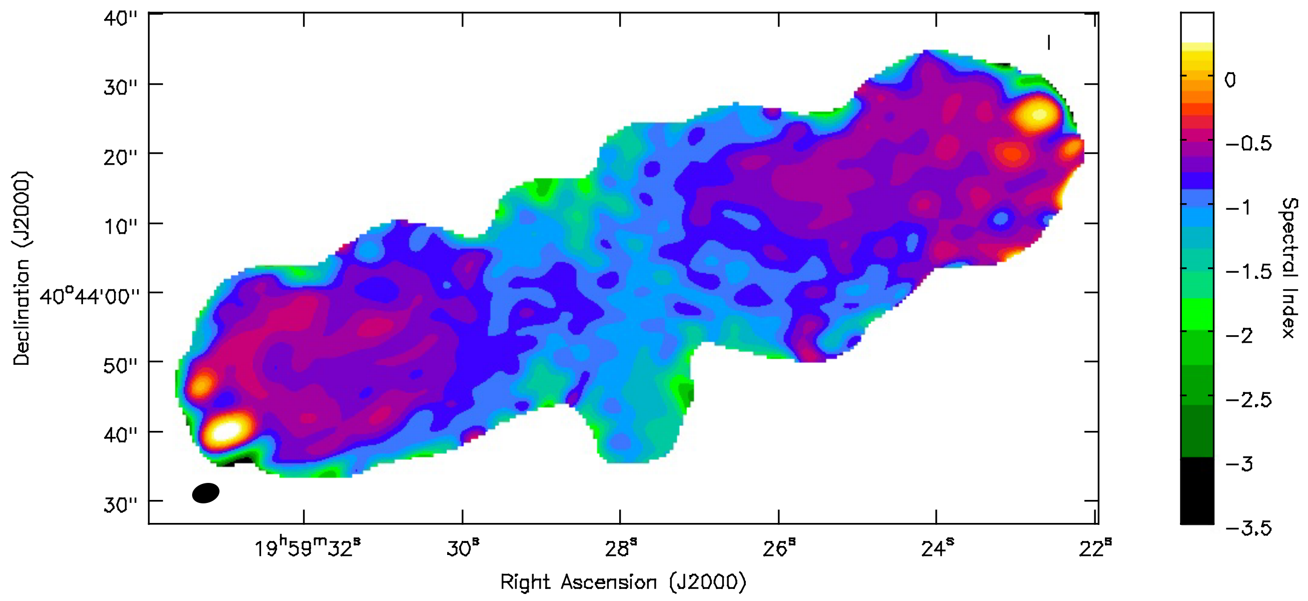
Learning Objectives:

i) To inspect the provided dataset and to identify and flag bad data.

ii) Use CASA to calibrate the data set and produce a flux-calibrated image of the target

iii) Use self-calibration to improve the calibration of the data set.





Learning Specifics:

The calibration and imaging of LOFAR data are carried out using LOFAR specific software, which allows direction dependent gain computation over a multi-node cluster.

However, for objects that dominate the sky brightness distribution, classic data reduction packages can be employed (e.g. AIPS, MIRIAD, CASA, DIFMAP). In this tutorial, the student will produce an image of the brightest radio galaxy in the LOFAR sky, Cygnus A, using CASA. They will learn the basic use of CASA and the application to LOFAR data.

The preliminaries

The Data: You will use a training dataset taken with the High Band Antenna (HBA) array of LOFAR on 2013 March 2.

Unpack the reduction files obtained from the ERIS2013 website in your present working directory (pwd),

```
> tar xvf tutorial4.tar
```

This will then unpack the following files in the pwd,

L102078_SB104_uv.dppp.MS	The MeasurementSet (MS)
151mhz.b-2.1.model	The model used for calibration

Note that the MS has already been preprocessed through DPPP (Default Pre-Processing Pipeline) by staff at the LOFAR Radio Observatory.

This process carries out an initial flagging of the radio frequency interference (RFI) and averages the data in time and frequency to a manageable size for our tutorial (172MB).

The model file contains the expected surface brightness distribution of the source at the frequency of our dataset.

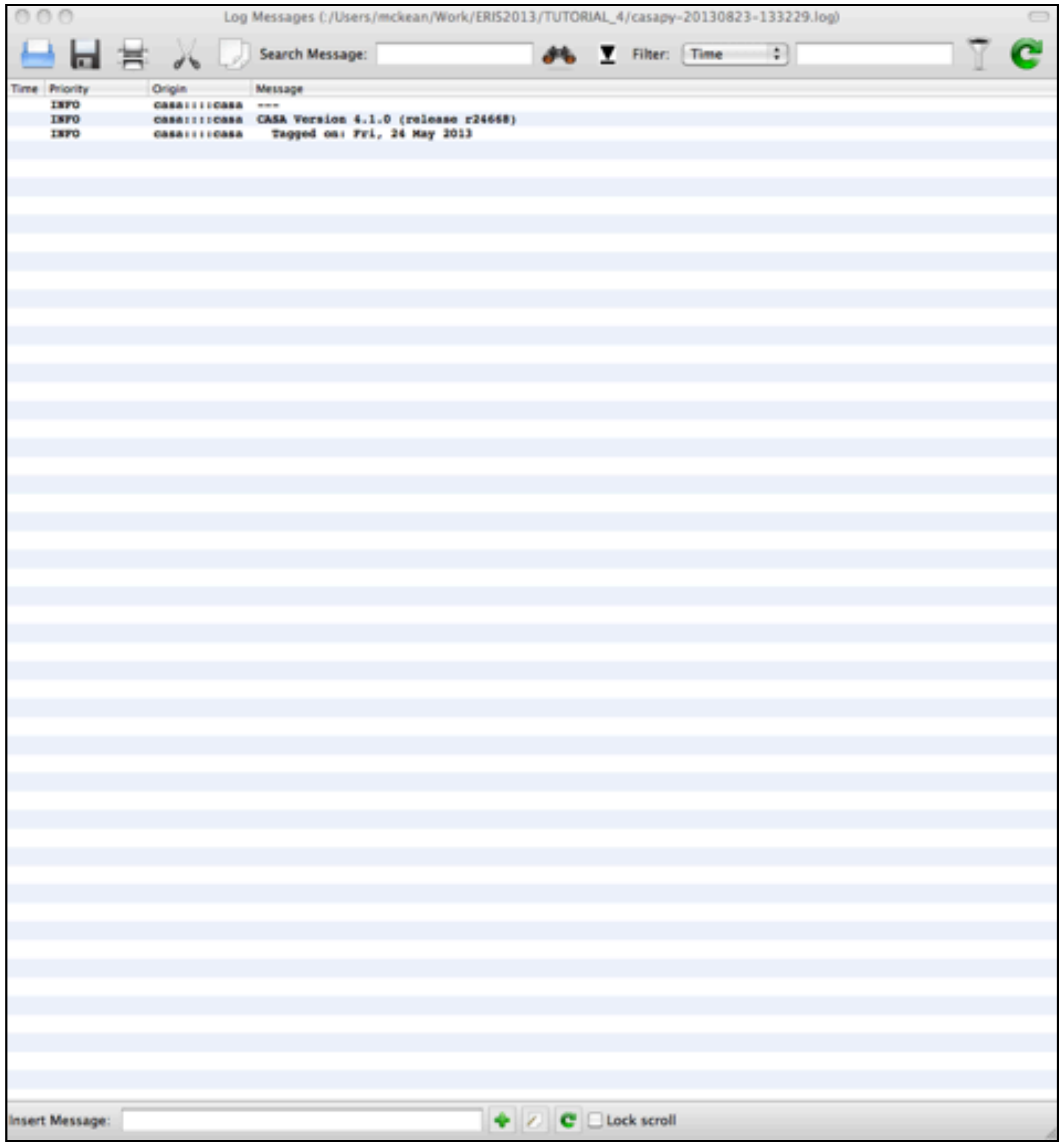
Lets start by inspecting these files.

The preliminaries

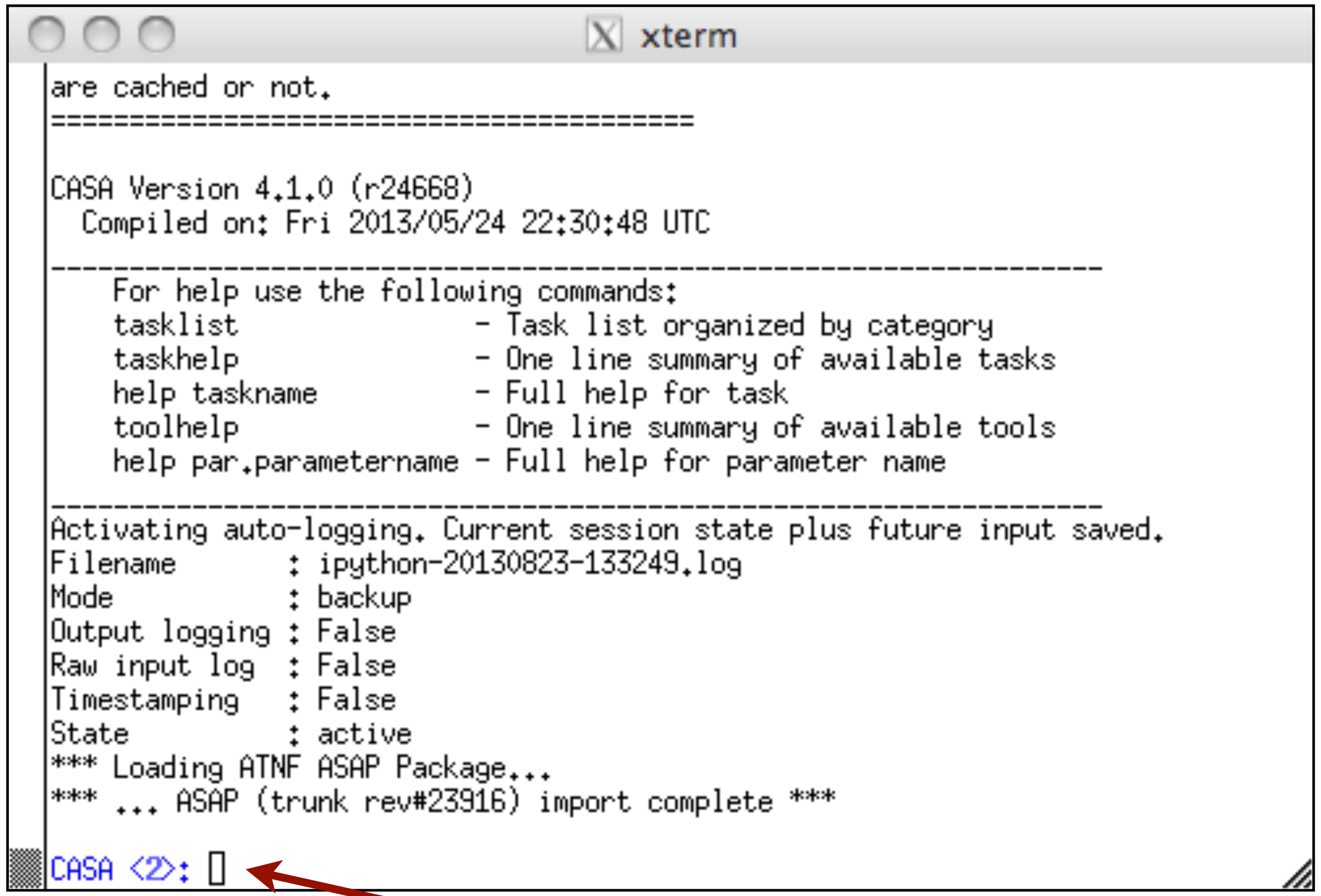
Reduction Package: The data reduction package that will be used for this tutorial is `CASA`. To start this package, use at the terminal/X11 prompt.

```
> casapy
```

`CASA` will now start, bringing up a log messages and a terminal window.


Log Messages (:/Users/mckean/Work/ERIS2013/TUTORIAL_4/casapy-20130823-133229.log)
Search Message:
Filter: Time
Insert Message:
Lock scroll

```
Time Priority Origin          Message
     INFO     casa::::casa    ---
     INFO     casa::::casa    CASA Version 4.1.0 (release r24668)
     INFO     casa::::casa      Tagged on: Fri, 24 May 2013
```



```
are cached or not.
=========================================

CASA Version 4.1.0 (r24668)
  Compiled on: Fri 2013/05/24 22:30:48 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-20130823-133249.log
Mode           : backup
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (trunk rev#23916) import complete ***

CASA <2>: 
```

The commands that we type are shown at the prompt.

A closer look

LISTOBS: We can obtain a summary of the MS using,

```
> listobs(vis="L102078_SB104_uv.dppp.MS",    \\
selectdata=True,spw="",field="",antenna="", \\
uvrange="",timerange="",correlation="",     \\
scan="",intent="",feed="",array="",         \\
observation="",verbose=True,listfile="",    \\
listunfl=False,cachesize=50)
```

The summary is printed to the log messages window, but can also be printed to a file for inspecting later by setting,

```
> listfile="summary.txt"
```

From our summary file, we see that,

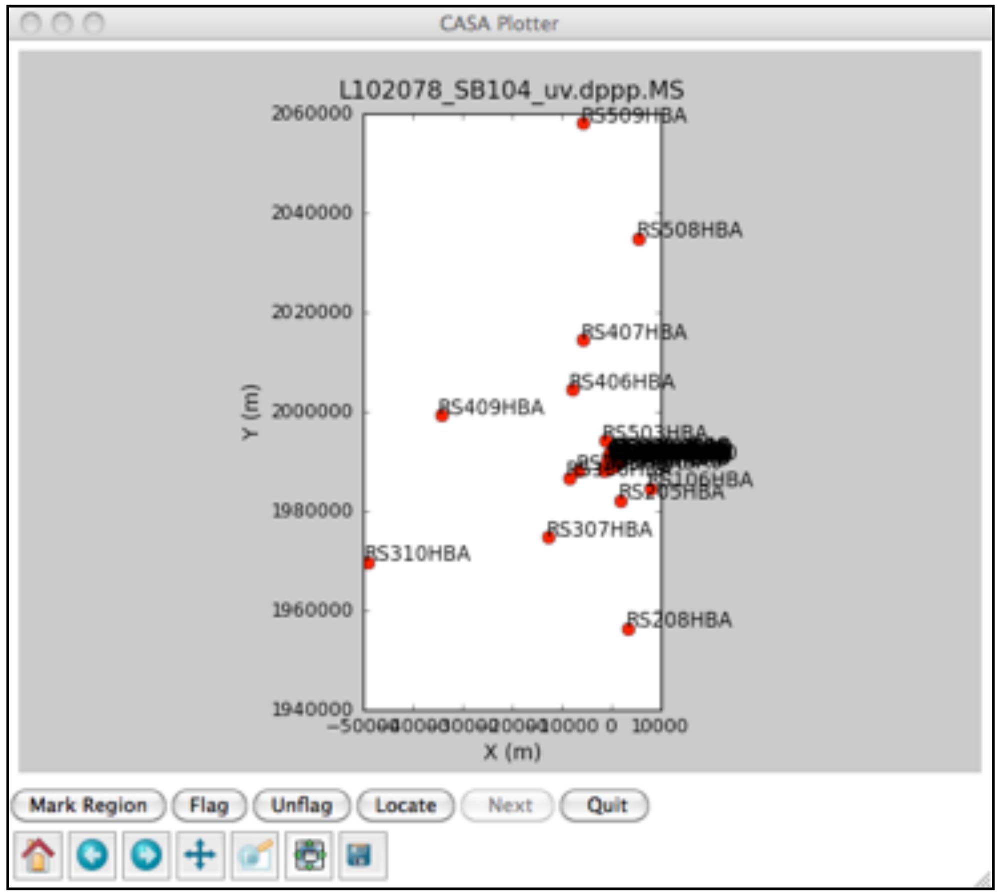
Observing time	= 6 hours
Observing frequency	= 151.170 MHz
Channels	= 1
Polarisation	= 4 (XX, YY, XY, YX)
Stations	= 36 (23 core and 13 remote)

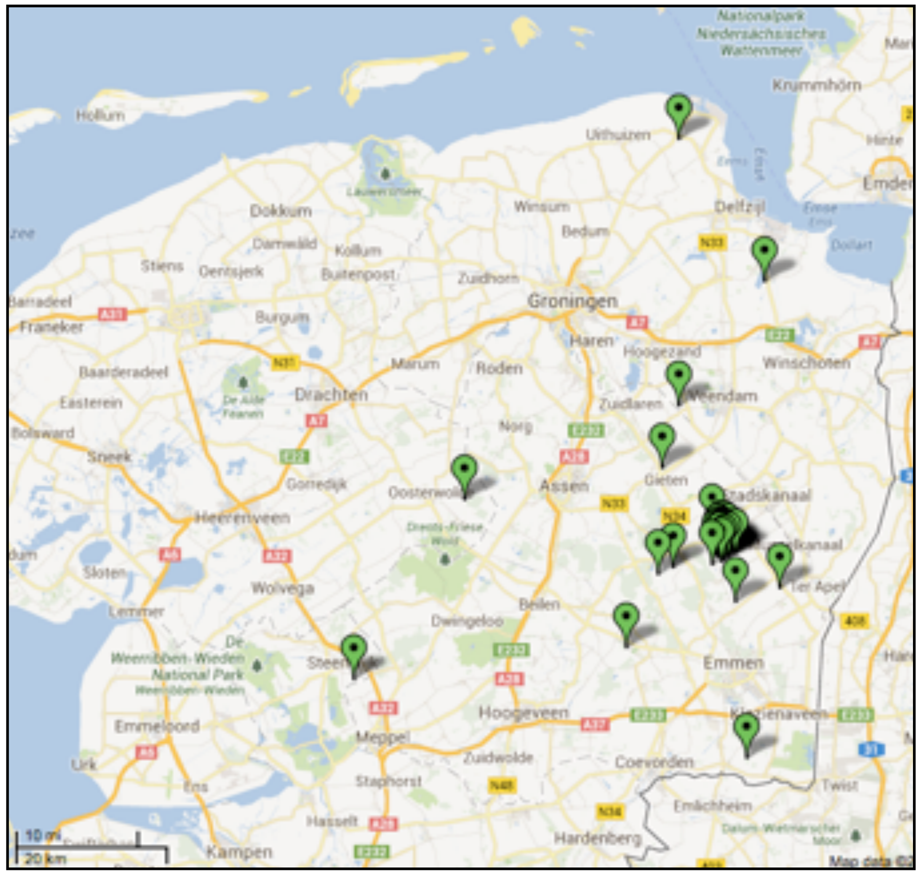
PLOTANTS: We can look at the array configuration using,

```
> plotants(vis="L102078_SB104_uv.dppp.MS",   \\
figfile="")
```

Alternatively, see where the stations are located on the map,

http://www.astron.nl/~heald/lofarStatusMap.html





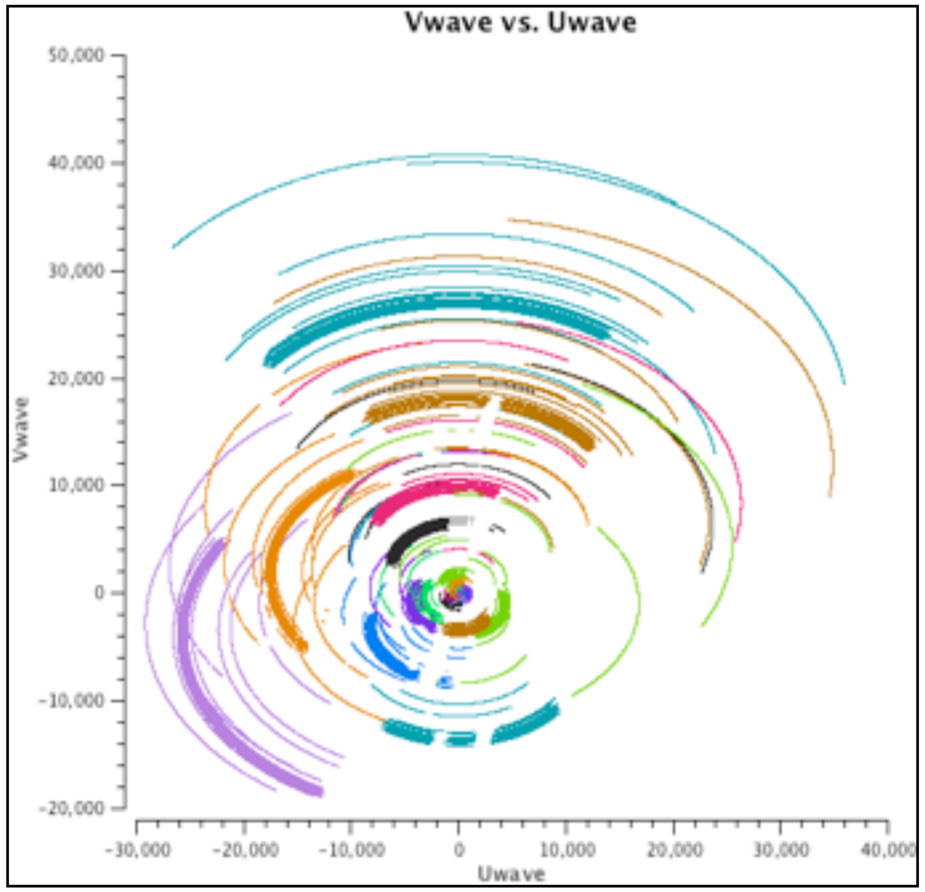
A closer look

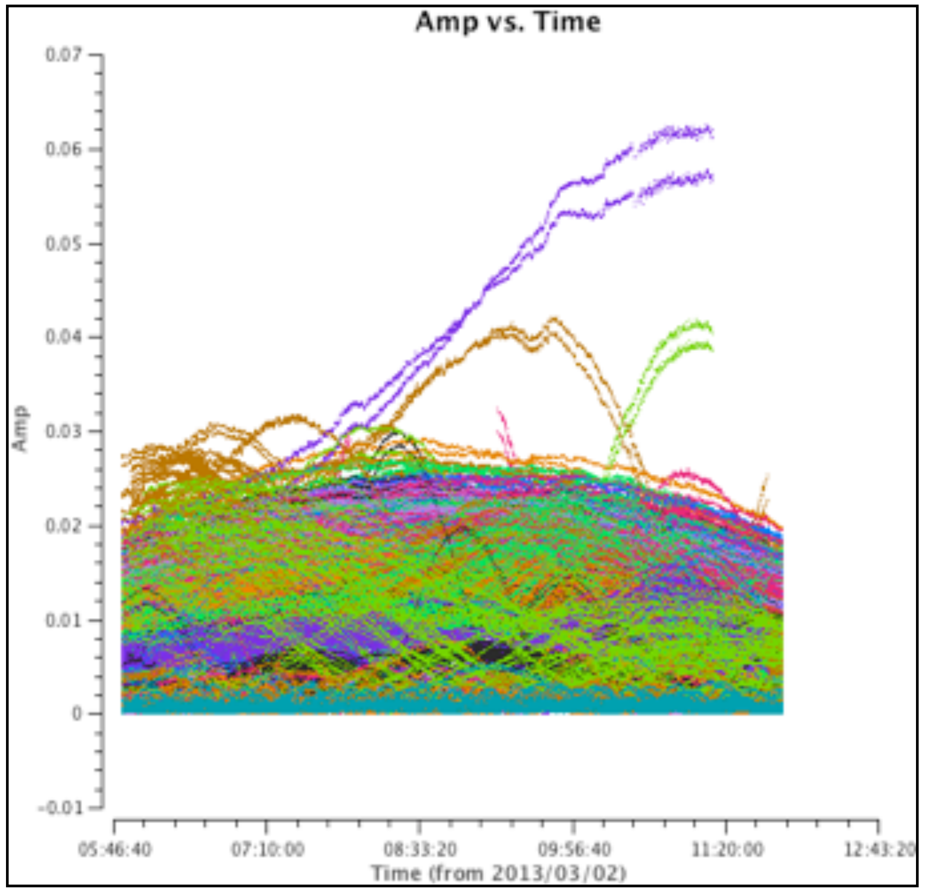
PLOTMS: We can look at the visibility data using,

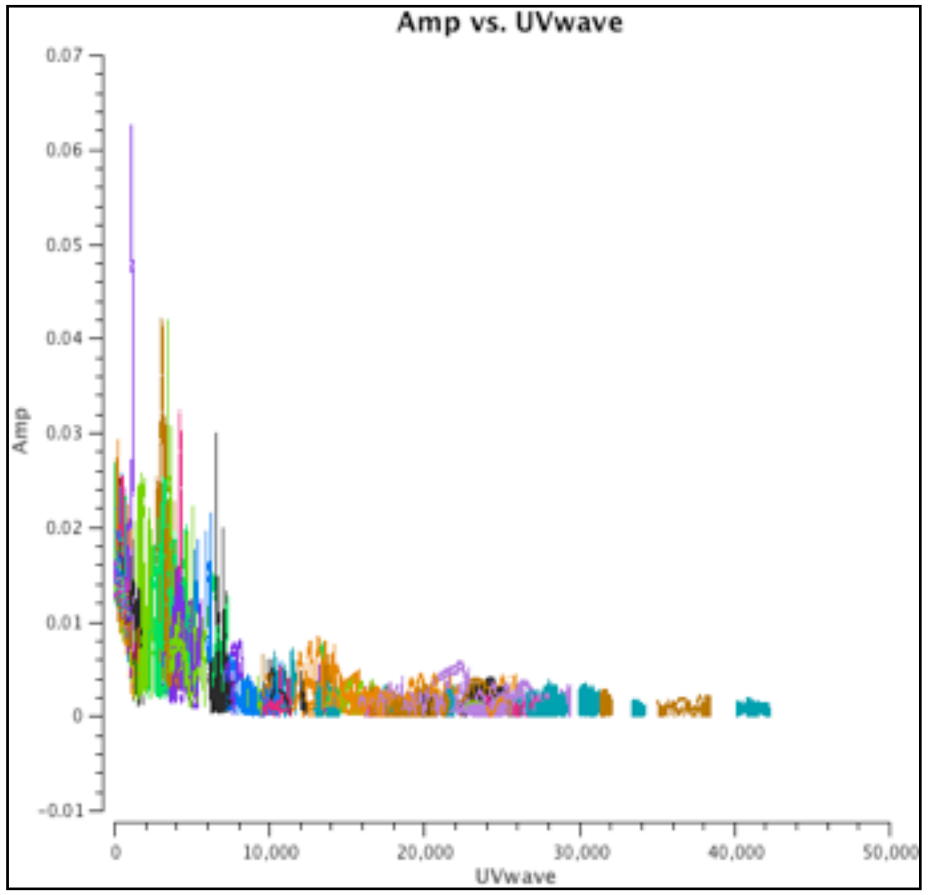
```
> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="time",yaxis="amp",                   \\
correlation="xx,yy",coloraxis="antenna2")

> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="uvwave",yaxis="amp",                 \\
correlation="xx,yy",coloraxis="antenna2")

> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="uwave",yaxis="vwave",                \\
correlation="xx,yy",coloraxis="antenna2")
```







Our model

VIEWER: Our model file is based on a previous calibration and uses a combination of delta functions and truncated Gaussians to describe the surface brightness distribution of the source at 151 MHz.

```
> viewer(infile="151mhz.b-2.1.model",           \\
displaytype="raster",channel=0,zoom=2,          \\
outfile="",outscale=1.0,outdpi=300,             \\
outformat="jpg",outlandscape=False,gui=True)
```

Try playing around with the image settings.

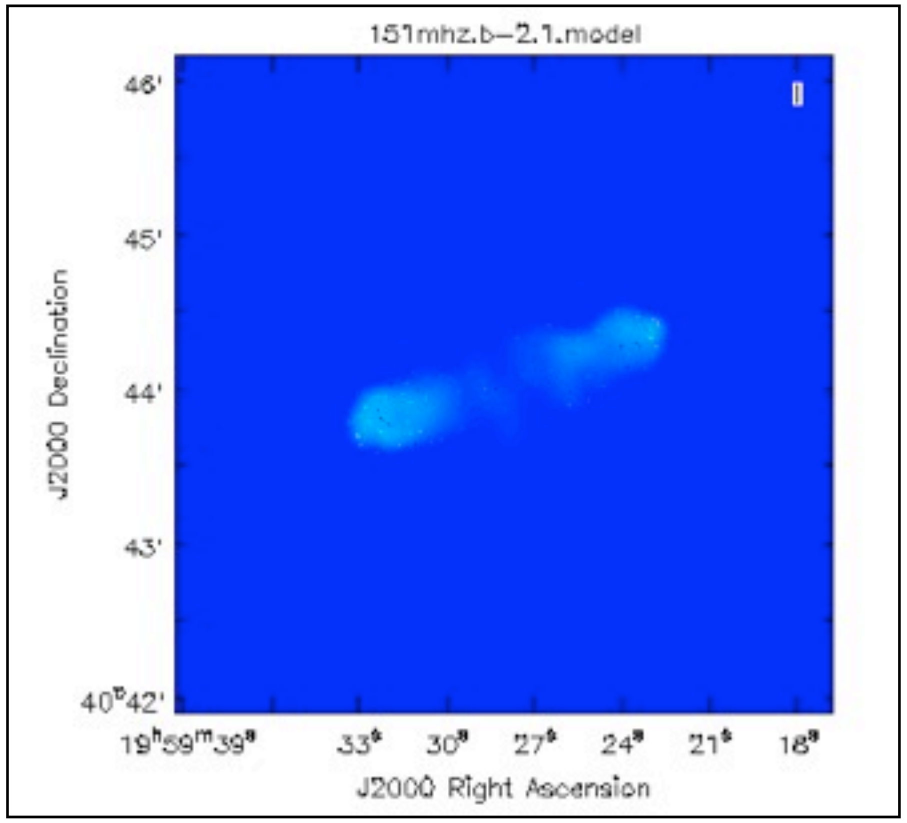
FT: Our calibration is carried out by comparing the expected MODEL visibilities to the observed DATA visibilities. First we need to take the Fourier transform of our model file to generate the MODEL visibiliites

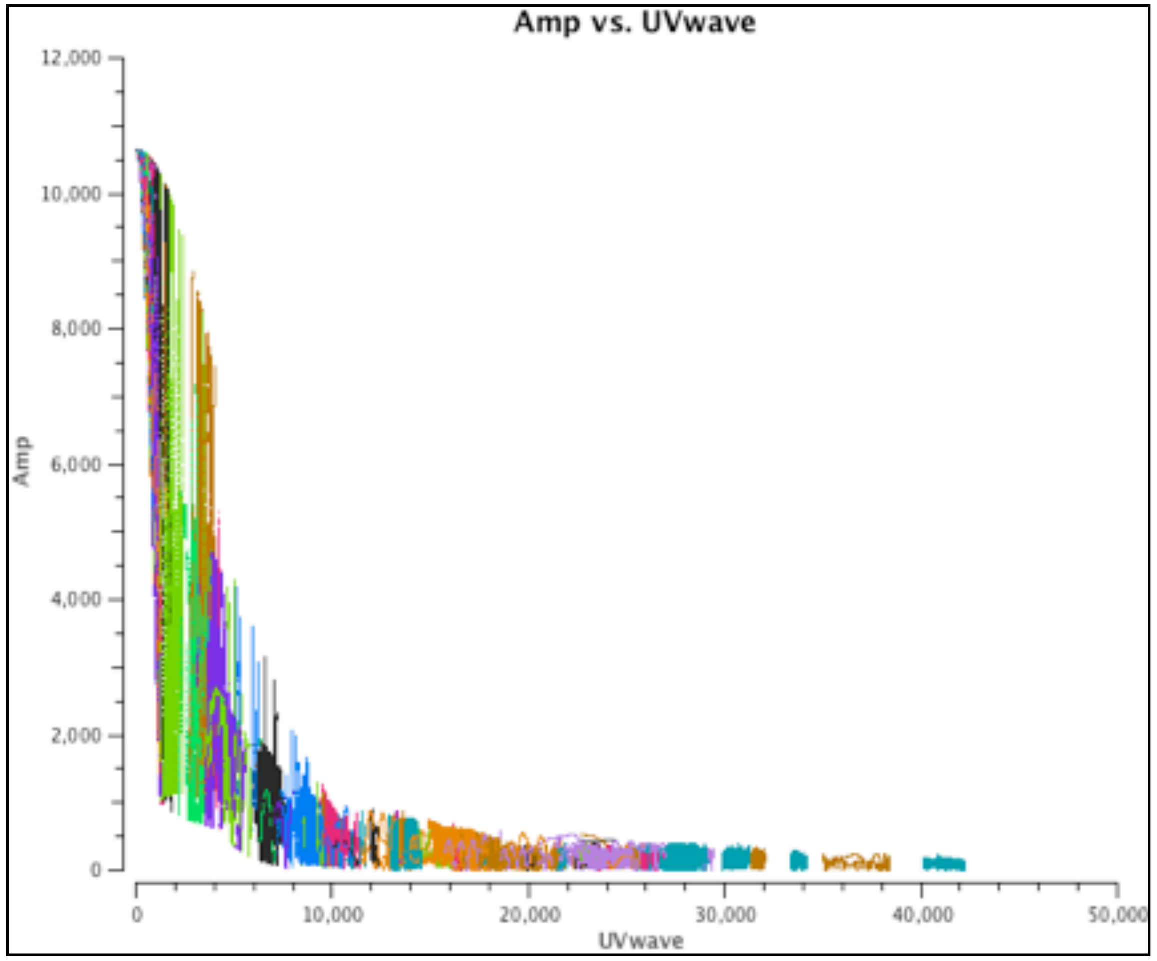
```
> ft(vis="L102078_SB104_uv.dppp.MS",            \\
field="",spw="",model="151mhz.b-2.1.model",  \\
nterms=1,reffreq="",complist="",                \\
incremental=False,usescratch=True)
```

PLOTMS: Lets plot our model visibilities,

```
> plotms(vis="L102078_SB104_uv.dppp.MS",        \\
xaxis="uvwave",yaxis="amp",                     \\
ydatacolumn="model",correlation="xx,yy",        \\
coloraxis="antenna2")
```

Try plotting the visibilities as a function of time.





Amplitude and phase calibration

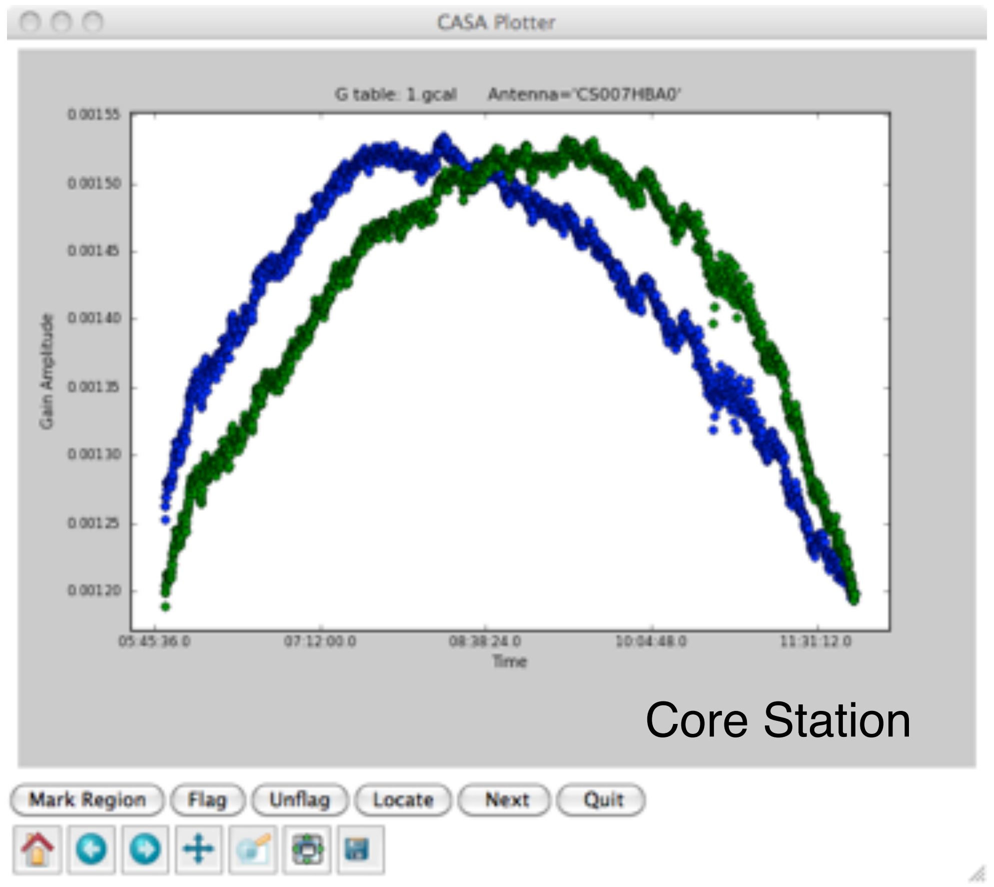
GAINCAL: We determine the gain (amplitude and phase) changes as a function of time.

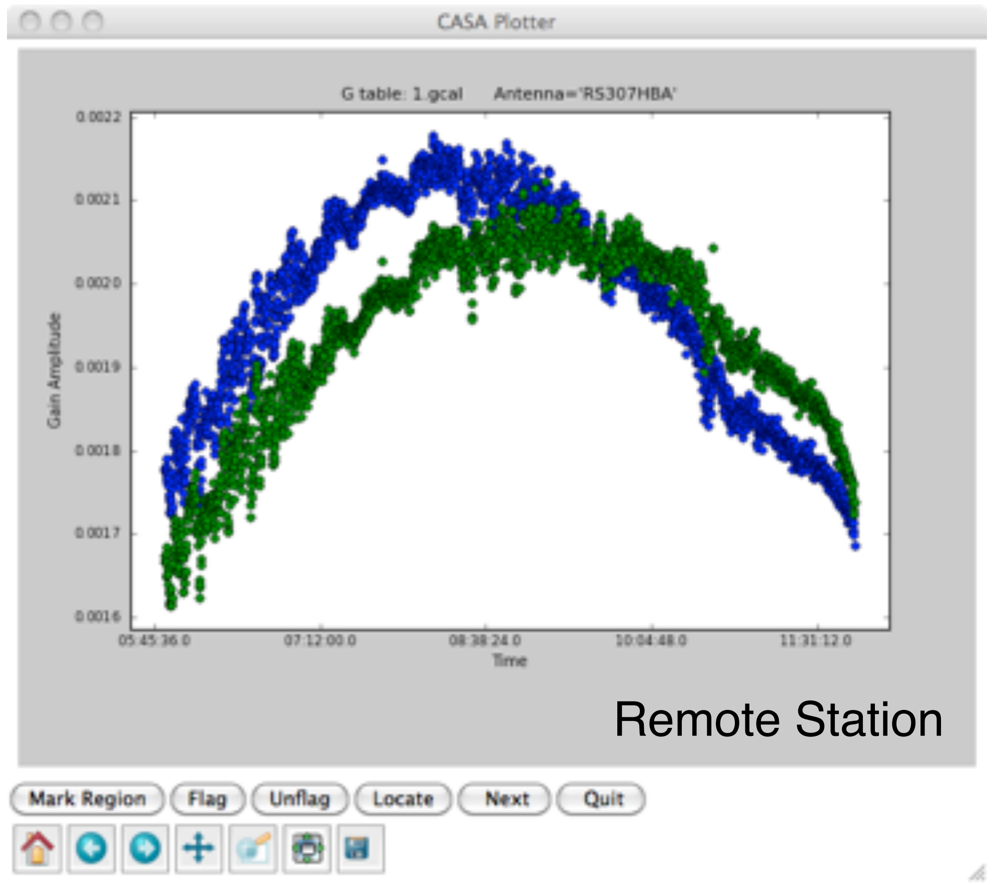
```
> gaincal(vis="L102078_SB104_uv.dppp.MS",     \\
caltable="1.gcal",field="",spw="",            \\
intent="",selectdata=False,timerange="",      \\
uvrange="",antenna="",scan="",                \\
observation="",msselect="",solint="int",      \\
combine="",preavg=-1.0,refant="",             \\
minblperant=4,minsnr=3.0,solnorm=False,       \\
gaintype="G",smodel=[],calmode="ap",          \\
append=False,splinetime=3600.0,               \\
npointaver=3,phasewrap=180.0,gaintable=[''] \\
,gainfield=[''],interp=[''],spwmap=[],        \\
gaincurve=False,opacity=[],parang=False)
```

Our signal-to-noise should be quite good, so lets solve for every visibility integration (10 s).

PLOTCAL: Now, lets inspect our amplitude solutions for each antenna.

```
> plotcal(caltable="1.gcal",xaxis="time",     \\
yaxis="amp",poln="",field="",antenna="",      \\
spw="",timerange="",subplot=111,              \\
overplot=False,clearpanel="Auto",             \\
iteration="antenna",plotrange=[],             \\
showflags=False,plotsymbol="o",               \\
plotcolor="blue",markersize=5.0,              \\
fontsize=10.0,showgui=True,figfile="")        \\
```





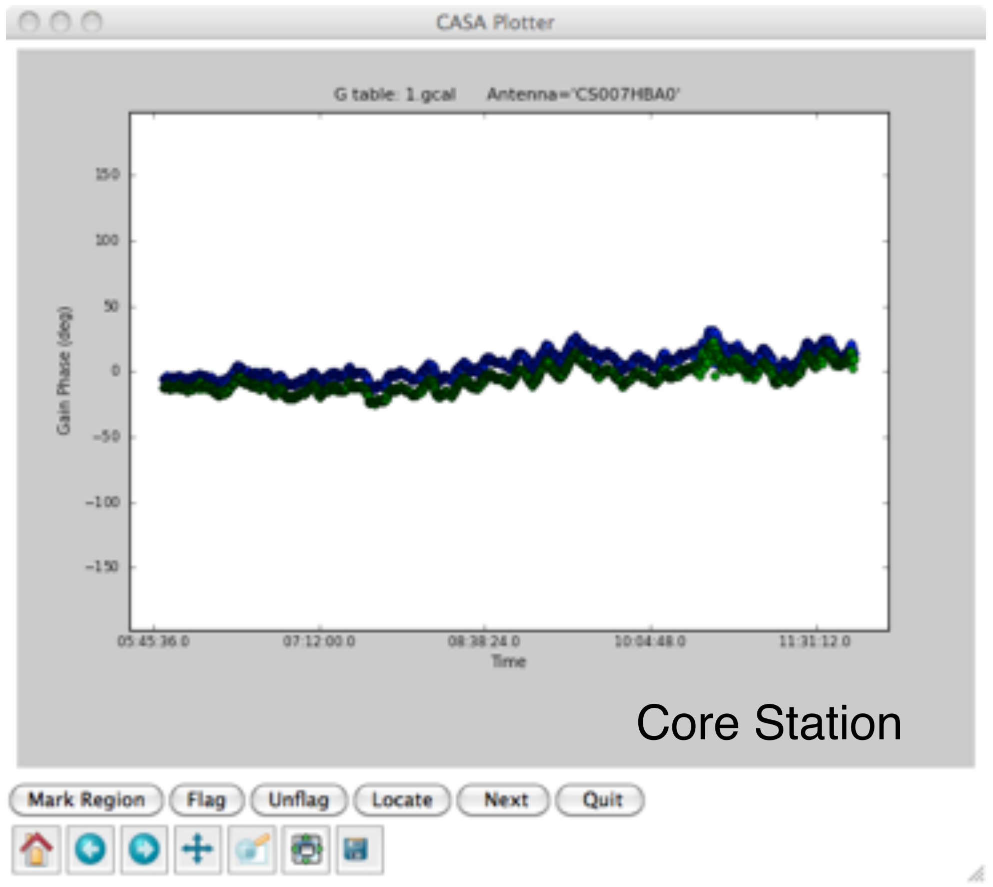
Amplitude and phase calibration

PLOTCAL: Now, lets inspect our phase solutions for each antenna.

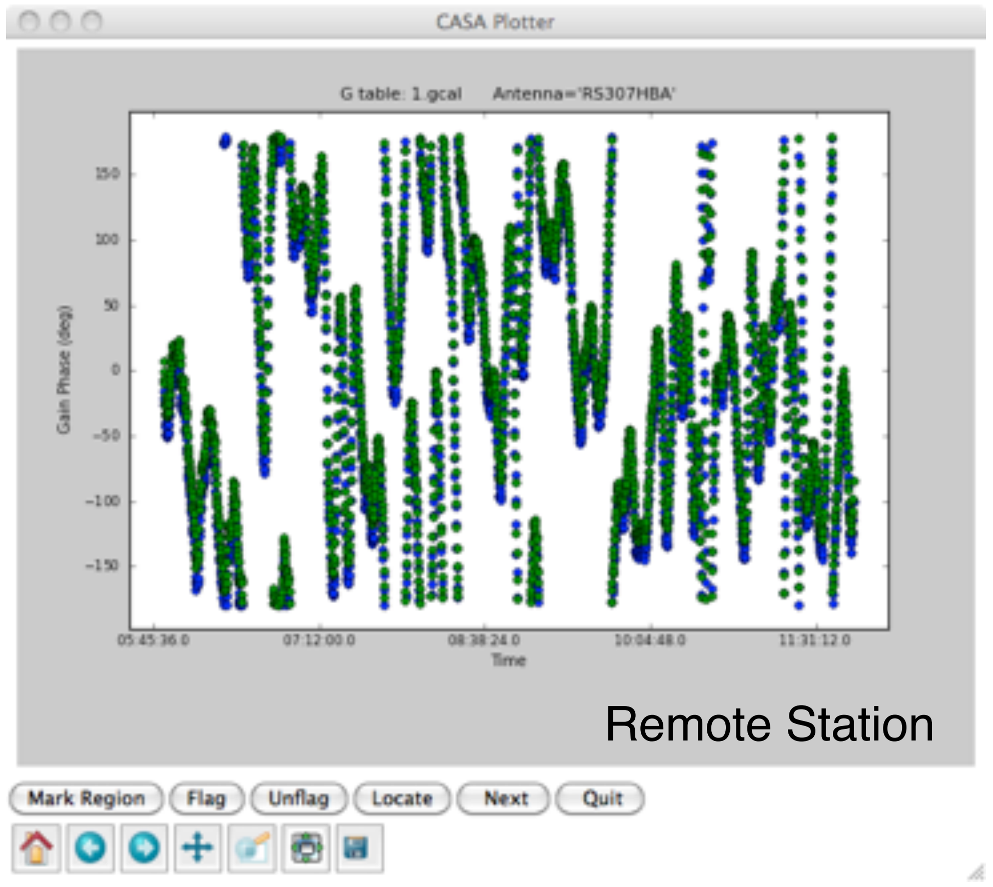
```
> plotcal(caltable="1.gcal",xaxis="time",    \\
yaxis="phase",poln="",field="",antenna="",   \\
spw="",timerange="",subplot=111,             \\
overplot=False,clearpanel="Auto",            \\
iteration="antenna",                         \\
plotrange=[-1,-1,-180,180],showflags=False,  \\
plotsymbol="o",plotcolor="blue",             \\
markersize=5.0,fontsize=10.0,showgui=True,   \\
figfile="")                                  \\
```

APPLYCAL: We now apply our solution tables and write a corrected data column to the MS.

```
> applycal(vis="L102078_SB104_uv.dppp.MS",   \\
field="",spw="",intent="",selectdata=False,  \\
timerange="",uvrange="",antenna="",scan="",  \\
observation="",msselect="",                  \\
gaintable="1.gcal",gainfield=[''],           \\
interp=[''],spwmap=[],gaincurve=False,       \\
opacity=[],parang=False,calwt=[True],        \\
applymode="",flagbackup=True)                \\
```



Core Station



Remote Station

Inspecting the calibrated data

PLOTMS: We can look at the visibility data using,

```
> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="uvwave",yaxis="amp",                  \\
ydatacolumn="corrected",                     \\
correlation="xx,yy",coloraxis="antenna2")
```

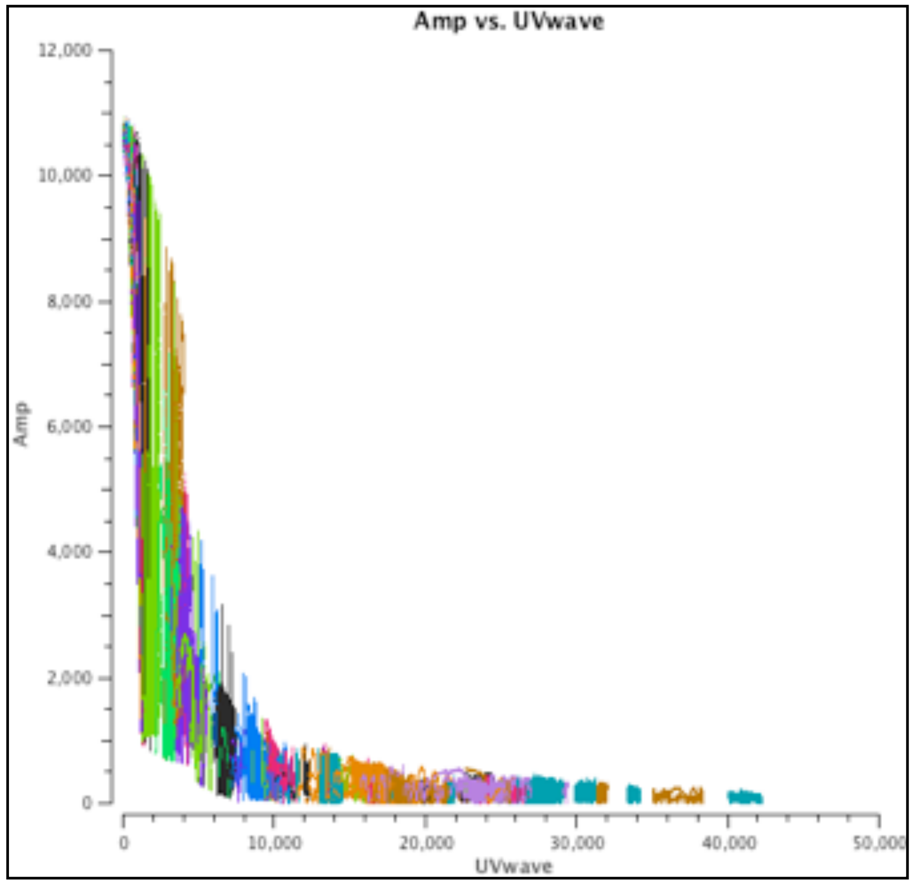
lets compare with what we expected to see,

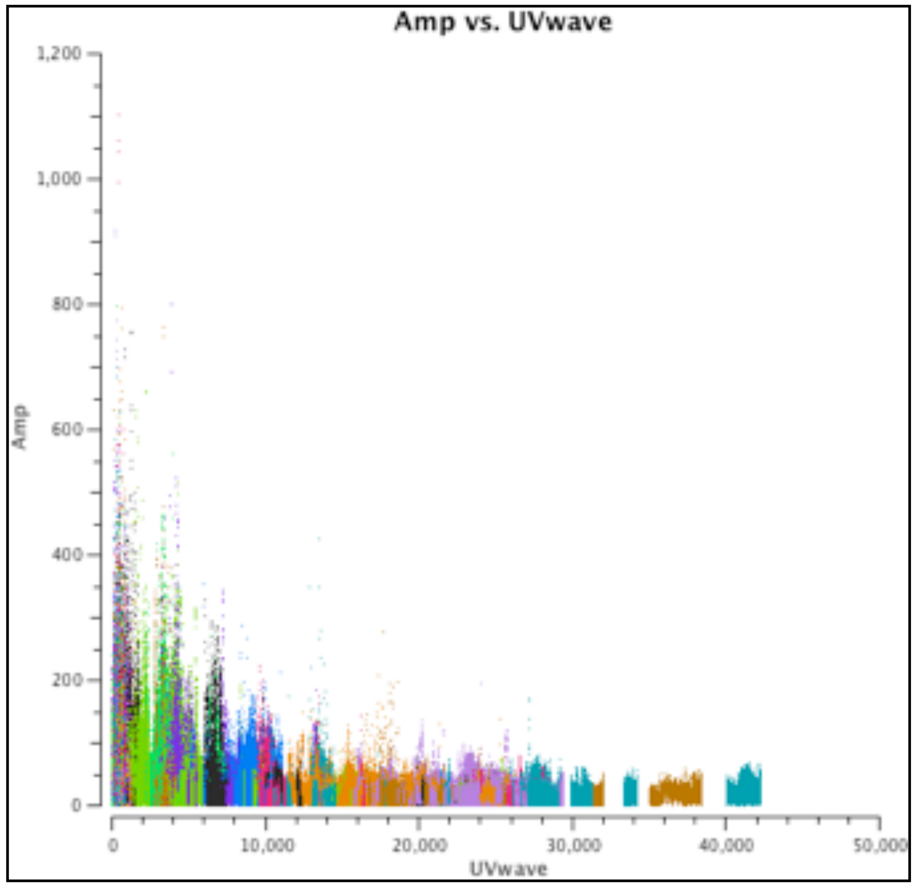
```
> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="uvwave",yaxis="amp",                  \\
ydatacolumn="corrected-model",               \\
correlation="xx,yy", coloraxis="antenna2")
```

Check how the corrected and residual visibilities compare as a function of time.

```
> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="uvwave",yaxis="time",                 \\
ydatacolumn="corrected",                     \\
correlation="xx,yy", coloraxis="antenna2")

> plotms(vis="L102078_SB104_uv.dppp.MS",     \\
xaxis="uvwave",yaxis="time",                 \\
ydatacolumn="corrected-model",               \\
correlation="xx,yy", coloraxis="antenna2")
```





Making an image

CLEAN: Due to our incomplete uv-coverage of image (the Fourier transform of the visibilities) will show large side-lobes. Therefore, we employ a deconvolution algorithm to build up an image of the sky.

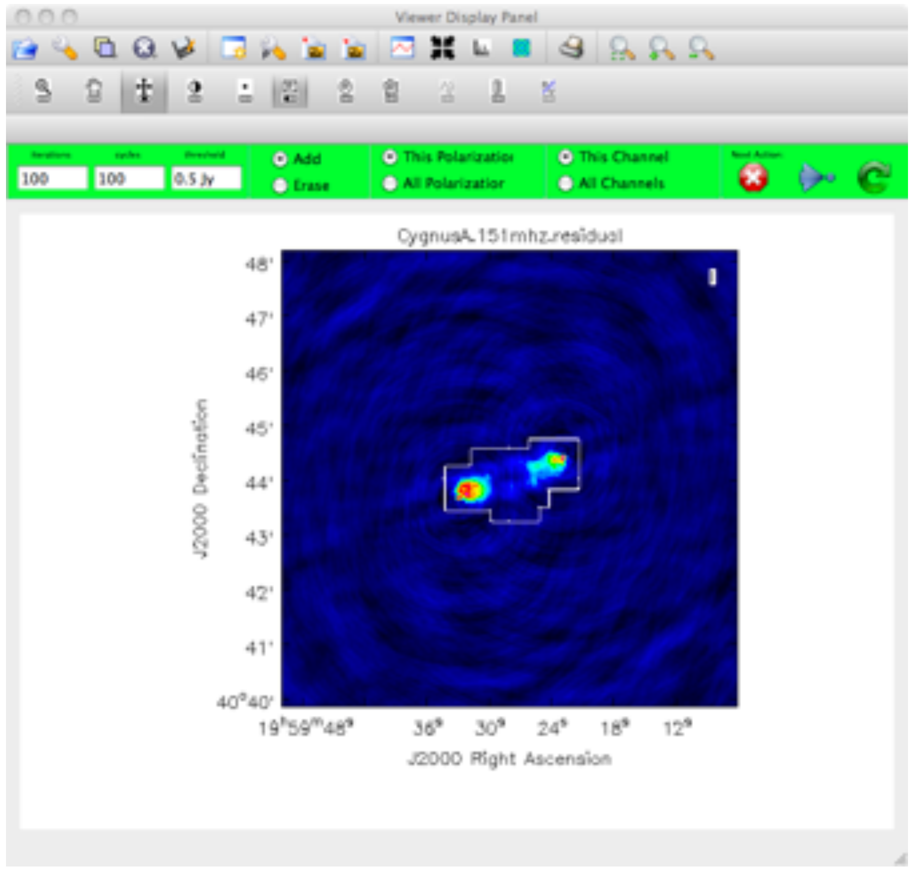
As Cygnus A has complicated structure, we will use multi-scale clean, i.e., use a set or truncated Gaussians to describe the sky surface brightness distribution.

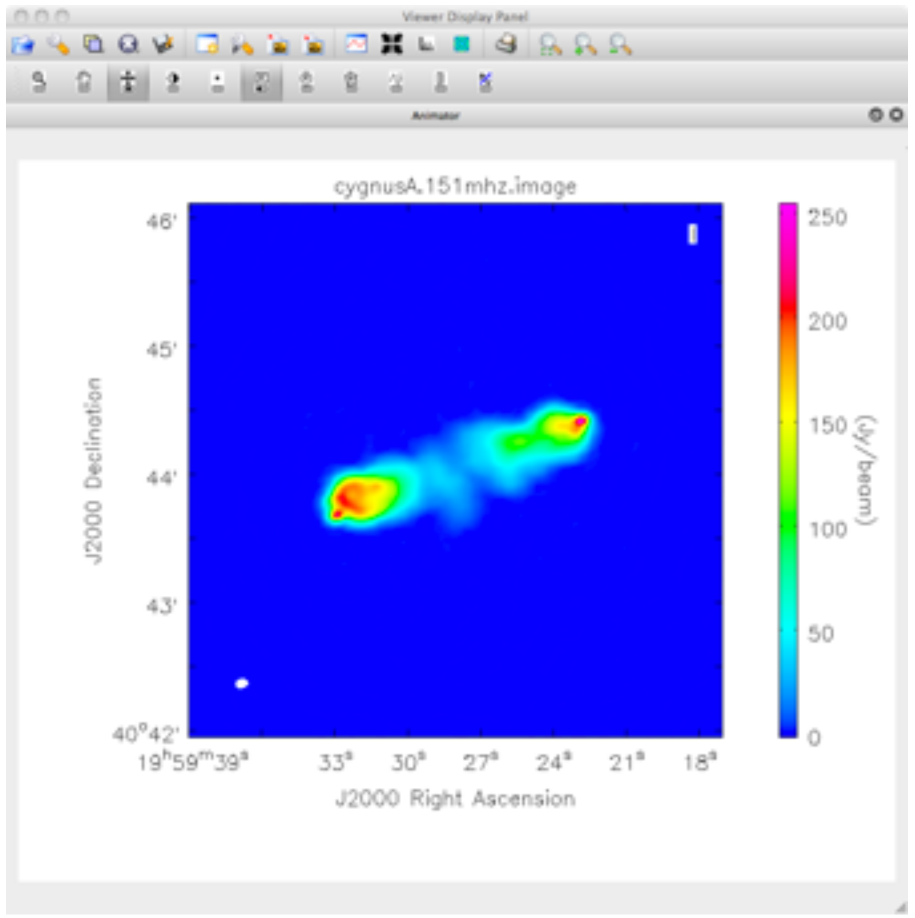
```
> clean(vis="L102078_SB104_uv.dppp.MS",      \\
imagename="CygnusA.151mhz",                  \\
niter=10000,gain=0.05,threshold="0.5Jy",     \\
multiscale=[0, 10, 20, 40],interactive=True,\\
imsize=[1000,1000],cell="0.5arcsec",         \\
weighting="briggs",robust=-2,usescratch=True)
```

VIEWER: We can look at our final image using,

```
> viewer(infile="cygnusA.151mhz.image",      \\
displaytype="raster",channel=0,zoom=2,       \\
outfile="",outscale=1.0,outdpi=300,          \\
outformat="jpg",outlandscape=False,gui=True)
```

FURTHER WORK:
i) Try redoing the calibration using your new model that was generated by running CLEAN,
ii) try smoothing the solutions, and
iii) try flagging any bad data.


Viewer Display Panel
100
100
0.5 Jy
Add
Erase
This Polarization
All Polarization
This Channel
All Channels
CygnusA.151mhz.residual
48'
47'
46'
45'
44'
43'
42'
41'
40°40'
J2000 Declination
19h59m48s
36s
30s
24s
18s
12s
J2000 Right Ascension


Viewer Display Panel
Animator
cygnusA.151mhz.image
46'
45'
44'
43'
40°42'
J2000 Declination
19h59m39s
33s
30s
27s
24s
21s
18s
J2000 Right Ascension
250
200
150
100
50
0
(Jy/beam)