

Basic CABB Continuum Data Reduction with CASA*

Tessa Vernstrom

2019
September

1 Introduction

This tutorial will describe the regular method of calibrating, flagging and imaging for ATCA CABB data. Most of what we do here will be applicable to pre-CABB data as well. The data we use here is from the ATCA experiment CX208. This experiment's aim was to measure the flux density of a source that appeared to have a very steep spectral index at low frequencies (below 1 GHz). The ATCA 16cm system (which covers the 1.1 – 3.1 GHz range) was used to get a measurement of the spectral index in this frequency range. The experiment was performed on 28 April 2011. The array was in the 6A configuration at the time, because it was thought that confusion may have been a possible explanation for the apparent spectral steepness. Each section in this tutorial will deal with a stage of the processing. The outline of the process is as follows:

1. load the data
2. flag the data - RFI flagging
3. initial calibration
4. calibrate
5. apply calibration and prepare for image
6. image and deconvolve

This tutorial will show these steps using the NRAO software CASA¹, however a version of this tutorial, with the same dataset, using the MIRIAD software package is also available.²

1.1 Using CASA

To open CASA from the command line, you should normally only have to type “casa” or “casapy” at terminal prompt and press enter. This will open essentially an interactive python CASA window, as well as a logger. This operates like a normal interactive or ipython window. You can import modules (e.g. “import numpy as np”) as well as perform other python functions.

A list of all available CASAtasks and information on how to use them is given at the CASA Documentation Homepage, <https://casa.nrao.edu/casadocs>.

In brief, there are at least three different ways to run CASA:

*with help from Jamie Stevens

¹ <https://casa.nrao.edu>

² http://www.narrabri.atnf.csiro.au/people/Jamie.Stevens/CX208/tutorial/basic_continuum_tutorial.pdf

- Interactively examining task inputs. In this mode, one types taskname to load the task, inp to examine the inputs, and go once those inputs have been set to your satisfaction. Allowed inputs are shown in blue and bad inputs are colored red. The input parameters themselves are changed one by one, e.g., selectdata=T. Screenshots of the inputs to various tasks used in the data reduction below are provided, to illustrate which parameters need to be set. More detailed help can be obtained on any task by typing help(taskname). Once a task is run, the set of inputs are stored and can be retrieved via tget taskname; subsequent runs will overwrite the previous tget file.

```

CASA <3>: inp listobs
-----> inp(listobs)
# listobs :: List the summary of a data set in the logger or in a file
vis                =      ''          # Name of input visibility file (MS)
selectdata         =      True        # Data selection parameters
  field            =      ''          # Selection based on field names or field index numbers. Default is all.
  spw               =      ''          # Selection based on spectral-window/frequency/channel.
  antenna           =      ''          # Selection based on antenna/baselines. Default is all.
  timerange         =      ''          # Selection based on time range. Default is entire range.
  correlation        =      ''          # Selection based on correlation. Default is all.
  scan              =      ''          # Selection based on scan numbers. Default is all.
  intent            =      ''          # Selection based on observation intent. Default is all.
  feed              =      ''          # Selection based on multi-feed numbers: Not yet implemented
  array             =      ''          # Selection based on (sub)array numbers. Default is all.
  uvrange           =      ''          # Selection based on uv range. Default: entire range. Default units: meters.
  observation        =      ''          # Selection based on observation ID. Default is all.

verbose           =      True        # Controls level of information detail reported. True reports more than False.
listfile          =      ''          # Name of disk file to write output. Default is none (output is written to logger only).
listunfl          =      False       # List unflagged row counts? If true, it can have significant negative performance impact.
cachesize         =      50          # EXPERIMENTAL. Maximum size in megabytes of cache in which data structures can be held.

CASA <4>: vis='visname.ms'
CASA <5>: listfile='lsitobs.dat'
CASA <6>: go

```

You can also see the manual help information for a particular task by typing: help(taskname).

- Pseudo-interactively via task function calls. In this case, all of the desired inputs to a task are provided at once on the CASA command line. This tutorial is made up of such calls, which were developed by looking at the inputs for each task and deciding what needed to be changed from default values. For task function calls, only parameters that you want to be different from their defaults need to be set.
- Non-interactively via a script. A series of task function calls can be combined together into a script, and run from within CASA via execfile('scriptname.py').

If you are a relative novice or just new to CASA, it is strongly recommended to work through this tutorial by cutting and pasting the task function calls provided below after you have read all the associated explanations (you can cut and paste them also into a text editor and save as a .py file for later use), **but** before executing commands inp and help the tasks to get more information about them and see the available parameters to set. Work at your own pace, look at the inputs to the tasks to see what other options exist, and read the help files. Later, when you are more comfortable, you might try to rerun the script, modify it for your purposes, and begin to reduce other data.

1.2 Formats

The native CASA format for data is called a Measurement Set (MS). There are a number of tasks for importing data from other formats to the CASA format (e.g. importuvfits, importatca, importgmrt, importmiriad, importasdm). The data are stored in columns, with the raw data

being stored in the “DATA” column. During calibration, or self-calibration, a “MODEL_DATA” column may be created and model visibilities are stored. The raw data is never altered. But once calibration solutions are derived, they are applied. This creates a “CORRECTED_DATA” column, where the corrected visibilities are stored. This does mean that a CASA measurement set can \sim triple in size, make sure you have enough free hard drive space for this. The data can be viewed as plots using the task **plotms**, or viewed as a table using the **browsetable** task.

The calibration solutions are stored in calibration tables. These can be viewed using either the **plotms** task or the **plotcal** task. These tables are not applied until the task **applycal** has been called.

Flags are generally applied across all the data (DATA, MODEL, CORRECTED). Different versions of the flagging may be saved or restored using the **flagmanager** task (which will allow you to manually save or restore the flags at any time), or when flagging the data with **flagdata** or when applying calibration with **applycal** set `flagbackup=True` to save a version of the flags before a task is run. This will create a new folder where the flags are stored having the name “vis-name.ms.flagversions”.

1.3 Plotting

The main utility for plotting or examining your data in CASA is the task **plotms**. Plots can be generated one of two ways: First by typing `plotms()` with no inputs. This will open up the plotms window. Here you can make your selections or inputs directly into the plotms window. You can also choose the inputs in the function call,

```
# In CASA
plotms(vis=visname, field='0', xaxis='frequency', yaxis='amp',
       ydatacolumn='data', correlation='xy,yx')
```

This again will open the plotms window but with the specified parameters chosen. The field column can be the field number or the field name. To see all the options type `inp plotms`.

Interactive flagging can be done in plotms, as well as locating of data (e.g. some bad data points may be highlighted and then the “locate” button selected, the exact time, baselines, correlations, channel, etc of the selected data will then be shown in the logger window).

2 Loading & Listing the Data

Once you have CASA up and running in the directory containing the data, then start your data reduction by getting the RPFITS file data into a Measurement Set, CASA’s native format. The task to do this, **importatca**, lets us choose which of the simultaneously recorded frequencies to load. In this case we just want to only load the data from the first IF (or in CASA spectral window, spw) since when observing with the 16cm band, the central continuum frequency is set to be 2.1 GHz in each IF, and each IF is therefore just a copy of the other (recall that the usable 16cm frequency range is 1.1 – 3.1 GHz). We can select with `spw=0`.

We specify the options ‘birdie’ to remove known bad channels and ‘noac’ to discard the autocorrelations. The edge parameter specifies how many edge channels to discard as a percentage of the number of channels in each band, e.g., the default value of 8 will discard 82 channels from the top and bottom of a 2048 channel spectrum. Here we set `edge=4` to discard 40 channels at the bottom of the band and 40 channels at the top of the band.

```
# In CASA  
rawname='2011-04-28_1858.CX208' #define the variable for the raw data name  
visname='cx208.0.ms' #define variable name for the output measurement set  
importatca(vis=visname,files=rawname,spw=0,options='birdie,noac',edge=4)
```

As the data loads a listing appears in the logger of the data encountered. Once this task is completed you should see the newly created measurement set in your directory.

The task **listobs** can be used to get a listing of the individual scans comprising the observation, the frequency setup, source list, and antenna locations. By default this information will be displayed in the logger, however, it can be useful to save this information to a file which can be set with the **listfile** parameter.

```
# In CASA  
listobs(vis=visname, listfile='listobs_cx208.dat')
```

The output should look like:

```

=====
Observer: obs      Project: CX208
Observation: ATCA
Data records: 15210      Total elapsed time = 10720 seconds
Observed from 28-Apr-2011/18:59:09.9 to 28-Apr-2011/21:57:49.9 (UTC)
=====

```

```

ObservationID = 0      ArrayID = 0
Date      Timerange (UTC)      Scan      FldId      FieldName      nRows      SpwIds      Average Interval(s)      ScanIntent
28-Apr-2011/18:59:09.9 - 19:03:59.9      0      0      1934-638      435      [0] [10]
19:04:39.9 - 19:13:19.9      1      1      2058-425      780      [0] [10]
19:13:39.9 - 19:23:49.9      2      2      2051-377      915      [0] [10]
19:24:09.9 - 19:25:19.9      3      1      2058-425      105      [0] [10]
19:25:39.9 - 19:35:49.9      4      2      2051-377      915      [0] [10]
19:36:09.9 - 19:37:19.9      5      1      2058-425      105      [0] [10]
19:37:39.9 - 19:47:49.9      6      2      2051-377      915      [0] [10]
19:48:09.9 - 19:49:19.9      7      1      2058-425      105      [0] [10]
19:49:39.9 - 19:59:49.9      8      2      2051-377      915      [0] [10]
20:00:09.9 - 20:01:19.9      9      1      2058-425      105      [0] [10]
20:01:39.9 - 20:11:49.9      10     2      2051-377      915      [0] [10]
20:12:09.9 - 20:13:19.9      11     1      2058-425      105      [0] [10]
20:13:39.9 - 20:23:49.9      12     2      2051-377      915      [0] [10]
20:24:09.9 - 20:25:19.9      13     1      2058-425      105      [0] [10]
20:25:39.9 - 20:35:49.9      14     2      2051-377      915      [0] [10]
20:36:09.9 - 20:37:19.9      15     1      2058-425      105      [0] [10]
20:37:39.9 - 20:47:49.9      16     2      2051-377      915      [0] [10]
20:48:09.9 - 20:49:29.9      17     1      2058-425      120      [0] [10]
20:49:49.9 - 21:00:09.9      18     2      2051-377      930      [0] [10]
21:00:29.9 - 21:01:49.9      19     1      2058-425      120      [0] [10]
21:02:09.9 - 21:12:39.9      20     2      2051-377      945      [0] [10]
21:12:59.9 - 21:14:19.9      21     1      2058-425      120      [0] [10]
21:14:39.9 - 21:25:09.9      22     2      2051-377      945      [0] [10]
21:25:29.9 - 21:26:49.9      23     1      2058-425      120      [0] [10]
21:27:09.9 - 21:37:29.9      24     2      2051-377      930      [0] [10]
21:37:49.9 - 21:39:09.9      25     1      2058-425      120      [0] [10]
21:39:29.9 - 21:49:49.9      26     2      2051-377      930      [0] [10]
21:50:09.9 - 21:51:29.9      27     1      2058-425      120      [0] [10]
21:51:49.9 - 21:57:49.9      28     2      2051-377      540      [0] [10]
(nRows = Total number of rows per scan)

```

Fields: 3

ID	Code Name	RA	Decl	Epoch	SrcId	nRows
0	C 1934-638	19:39:25.025995	-63.42.45.63000	J2000	0	435
1	C 2058-425	21:01:59.115000	-42.19.16.16006	J2000	1	2235
2	2051-377	20:54:54.400005	-37.33.38.99991	J2000	2	12540

Spectral Windows: (1 unique spectral windows and 1 unique polarization setups)

SpwID	Name	#Chans	Frame	Ch0(MHz)	ChanWid(kHz)	TotBW(kHz)	CtrFreq(MHz)	Corrs
0	2049	TOPO		3124.000	-1000.000	2048000.0	2100.0000	XX XY YX YY

Sources: 3

ID	Name	SpwId	RestFreq(MHz)	SysVel(km/s)
0	1934-638	0	2100	0
1	2058-425	0	2100	0
2	2051-377	0	2100	0

Antennas: 6:

ID	Name	Station	Diam.	Long.	Lat.	Offset from array center (m)			ITRF Geocentric coordinates (m)		
						East	North	Elevation	x	y	z
0	CA01	W4	22.0 m	+149.34.50.5	-30.08.43.7	2938.7745	1.7269	3.0055	-4752407.475000	2790374.102000	-3200483.763000
1	CA02	W45	22.0 m	+149.34.27.0	-30.08.43.7	2311.2246	1.3060	2.2668	-4752088.967000	2790914.817000	-3200483.756000
2	CA03	W102	22.0 m	+149.33.54.3	-30.08.43.7	1438.7848	0.7585	1.3280	-4751646.147000	2791666.524000	-3200483.758000
3	CA04	W173	22.0 m	+149.33.13.7	-30.08.43.7	352.0393	0.1942	0.3443	-4751094.576000	2792602.893000	-3200483.752000
4	CA05	W195	22.0 m	+149.33.01.1	-30.08.43.7	15.3063	0.0485	0.0497	-4750923.652000	2792893.021000	-3200483.730000
5	CA06	W392	22.0 m	+149.31.08.2	-30.08.43.8	-2999.9953	-0.8778	-1.6171	-4749393.198000	2795491.050000	-3200483.694000

Figure 1: Output from listobs

We can see there are 3 different sources : 1934-638 the primary calibrator , 2058-425 the secondary or phase calibrator, and 2051-377 the target. We also see the data has a 2048 MHz bandwidth, with 1 MHz channels, centred on 2.1 GHz with 4 linear polarisations. The antenna station names give away that this is a compact configuration, with CA01-CA05 close together and CA06 a long way away (multiply the station numbers by 15 to get distance in meters from station W0).

The task `plotants` can be used to plot the positions of the antenna. And the task `plotuv` will generate a plot of the uv coverage. Examples of these are show in Figure 2.

```

# In CASA
#make a plot of the antenna locations
plotants(vis=visname,figfile='cx208_ants.png')

#make a plot of the uv coverage
plotuv(vis=visname,field=tar,figfile='cx208_target_uv.png')

```

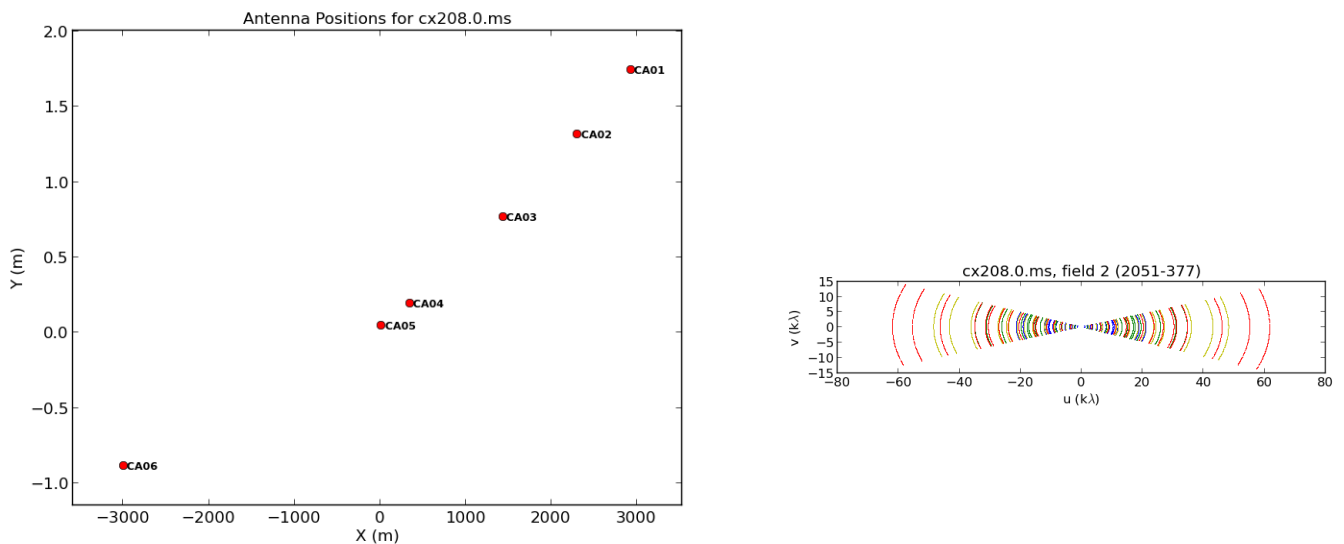


Figure 2: Left: Plot of the antenna locations. Right: plot the of uv coverage of the target.

It is also helpful to set some additional variables.

```
#In CASA
#define some variables from output of listobs for
#the field names and a reference antenna
pri='1934-638'
sec='2058-425'
tar='2051-377'
ref='CA04'
```

3 Flagging the Data

It is always a good idea to examine the data and do some basic flagging before jumping straight into calibration. For this the task `flagdata` will be used. First we flag any visibilities with zero amplitudes, as well as any where the antennas are in shadow. Then we “quack” (or flag) the first 5 seconds of the each scan.

```
# In CASA
#save the intitial flags with flagmanager, can set names for the saved flag versions.
flagmanager(vis=visname, mode='save', versionname='before_online_flagging')

#flag for zero values, where antennas in shadow and the first 5 seconds of the scans.
flagdata(vis=visname, mode='clip', clipzeros=True,flagbackup=False)
flagdata(vis=visname, mode='shadow', tolerance=0.0, flagbackup=False)
flagdata(vis=visname, mode='quack', quackinterval=5.0, quackmode='beg', flagbackup=False)

#save flags again
flagmanager(vis=visname, mode='save', versionname='after_online_flagging')
```

At low frequencies there is a lot of RFI, some of it is very strong and persistent so lets get of rid of that first. Use `plotms` to make a plot of the XY and YX correlations on the calibrator. These correlations should normally have little signal in them, certainly anything stronger than the calibrator flux will be interference.

```
# In CASA
plotms(vis=visname,field=pri,xaxis='channel',yaxis='amp',
       correlation='xy,yx',ydatacolumn='data')
```

The “range” parameter in the plotms window can be used to zoom in. An image of this can be seen in Figure 3.

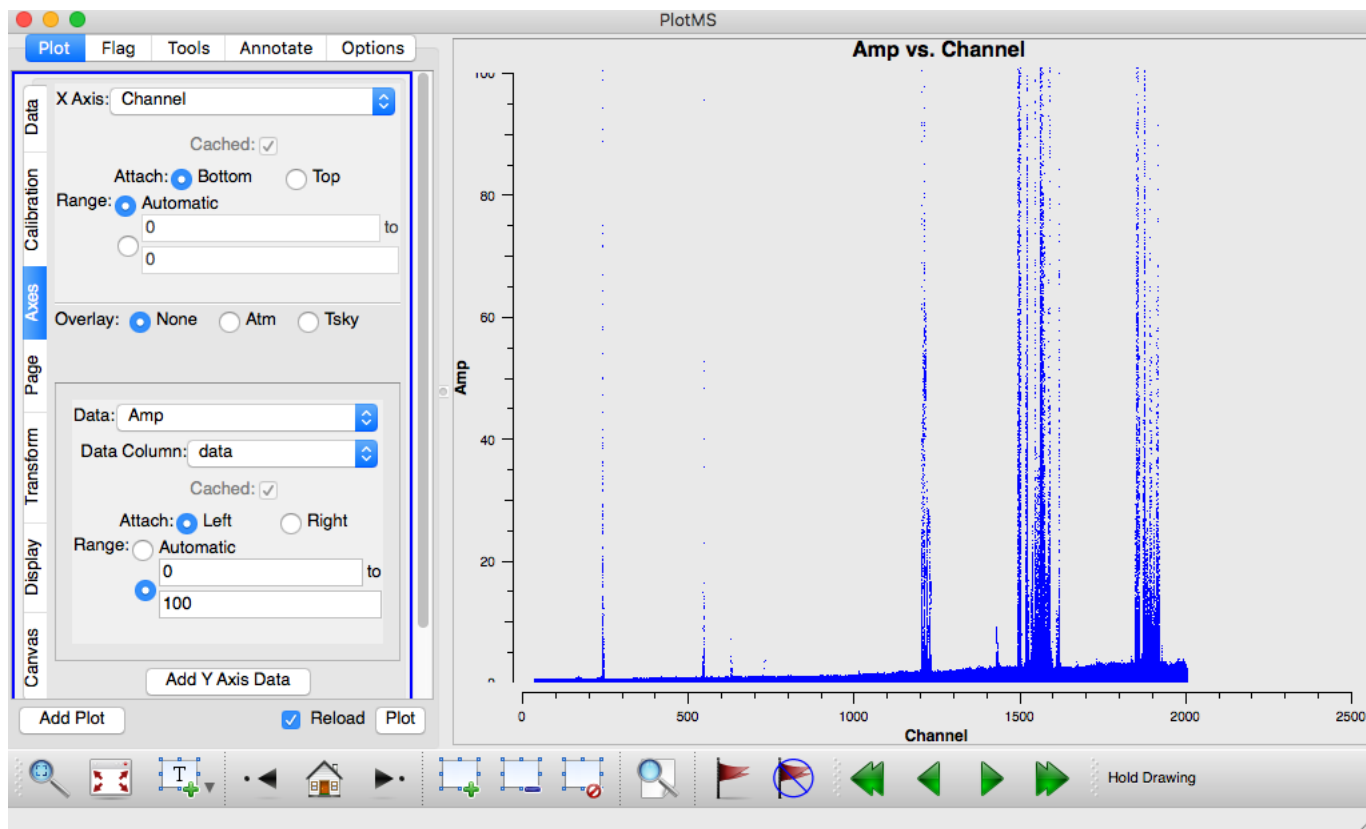


Figure 3: Plot of 1934 XY,YX amplitude vs channel before RFI flagging.

CASA has several modes for automatic RFI flagging, both use `flagdata`. Here will we look at using `mode='tfcrop'`.

```
# In CASA
#Calculate flags using tfcrop based on time and frequency window
flagdata(vis=visname, mode='tfcrop', datacolumn='data', action='apply', display='report',
        flagbackup=True, extendpols=True, correlation='', flagdimension='freqtime',
        growtime=95.0, growfreq=95.0, timecutoff=4., freqcutoff=3.5, timefit='line',
        freqfit='poly', maxnpieces=5, combinescans=False, ntime='scan', extendflags=False)

#extend the flags to all correlations
flagdata(vis=visname, mode='extend', action='apply', display='report', flagbackup=False,
        extendpols=True, correlation='', growtime=95.0, growfreq=95.0, growaround=True,
        flagneartime=False, flagnearfreq=False, combinescans=False, ntime='scan')
```

Now if we look at the same plot again (Figure 4) we can see how well the RFI flagging did (either re-enter the call to plotms from above or in the plotms window simply hit “plot” again).

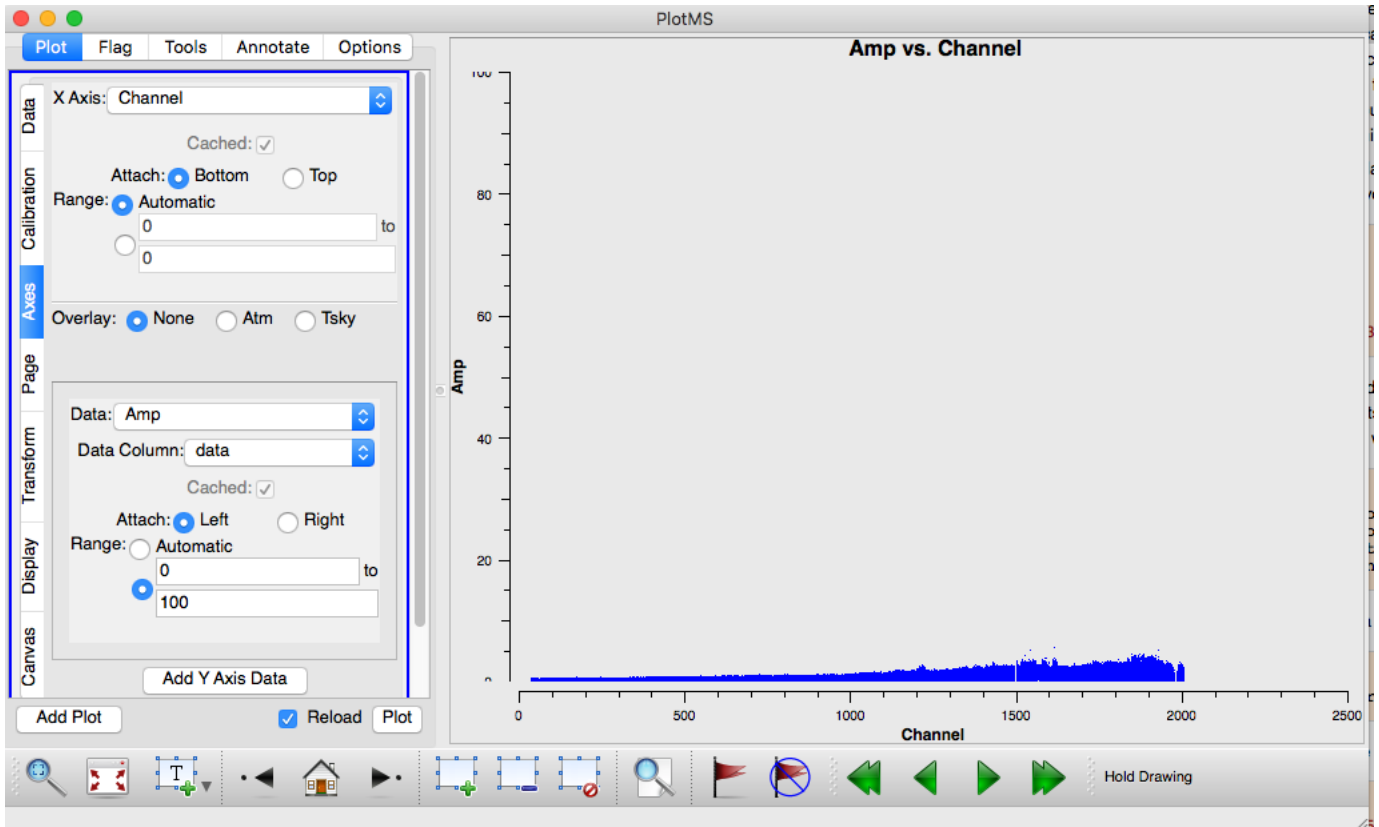


Figure 4: Plot of 1934 XY,YX amplitude vs channel after RFI flagging.

Other plots can be made or checked by changing the correlations, the field that is plotted, or what is plotted on the axes (feel free to play around).

4 Calibration

4.1 MS transform

The first thing we want to do is split the wide bandwidth of 2048 MHz up into 8 spectral windows of 256 MHz each so we can solve for the gains and polarisation leakages per spectral window. Make sure you are using CASA 4.7 or later for this step as earlier versions of `mstransform` had a bug that scrambled the baselines when used in this way

```
# In CASA
msname='cx208.ms' #define new name for transformed MS
mstransform(vis=visname,outputvis=msname,datacolumn='data',mode='channel',
            regridms=True,spw='0',nspw=8)

#can make a new listobs to see the difference
listobs(vis=msname,listfile='listobs_cx208_transform.dat')

#save new flag file
flagmanager(vis=msname,mode='save',versionname='after_transform')
```

4.2 Initial Calibration

Next we set the flux scale for the primary calibrator 1934-638 using the task `setjy`. We have a choice of standards: 'Perley-Butler 2010' works fine at frequencies below 11 GHz; 'Stevens-Reynolds 2016' first appeared in CASA 4.7 and has the same scale below 11 GHz but is more accurate above that

frequency, make sure to use this for 15 or 7mm data. If you haven't managed to observe 1934-638 in your observation, you may be able to 'borrow' an observation from an observation directly before or after yours if it happened to use the same setup, otherwise your best bet is to check the ATCA Calibrator database for a recent observation of your calibrator and use the flux from that.

```
# In CASA
setjy(vis=msname,field=pri,scalebychan=True,standard='Perley-Butler 2010',usescratch=True)

Out[30]:
{'0': {'0': {'fluxd': array([ 9.18244858,  0.          ,  0.          ,  0.          ])}},
 '1': {'fluxd': array([ 9.92884018,  0.          ,  0.          ,  0.          ])}},
 '2': {'fluxd': array([ 10.75012454,  0.          ,  0.          ,  0.          ])}},
 '3': {'fluxd': array([ 11.6425861,  0.          ,  0.          ,  0.          ])}},
 '4': {'fluxd': array([ 12.58972782,  0.          ,  0.          ,  0.          ])}},
 '5': {'fluxd': array([ 13.54956179,  0.          ,  0.          ,  0.          ])}},
 '6': {'fluxd': array([ 14.42993683,  0.          ,  0.          ,  0.          ])}},
 '7': {'fluxd': array([ 15.04066761,  0.          ,  0.          ,  0.          ])}},
 'fieldName': '1934-638'},
 'format': "{field Id: {spw Id: {fluxd: [I,Q,U,V] in Jy}, 'fieldName':field name }}"
```

Setting `scalebychan=True` scales the flux density on a per channel basis or else on a per spw basis. Setting `usescratch=True` will populate the MODEL_DATA column with model visibilities. We can then see what this spectrum would look like with **plotms** (Figure 5).

```
# In CASA
plotms(vis=msname,,field=pri,xaxis='frequency',yaxis='amp',
       correlation='xx,yy',ydatacolumn='model',coloraxis='spw')
```

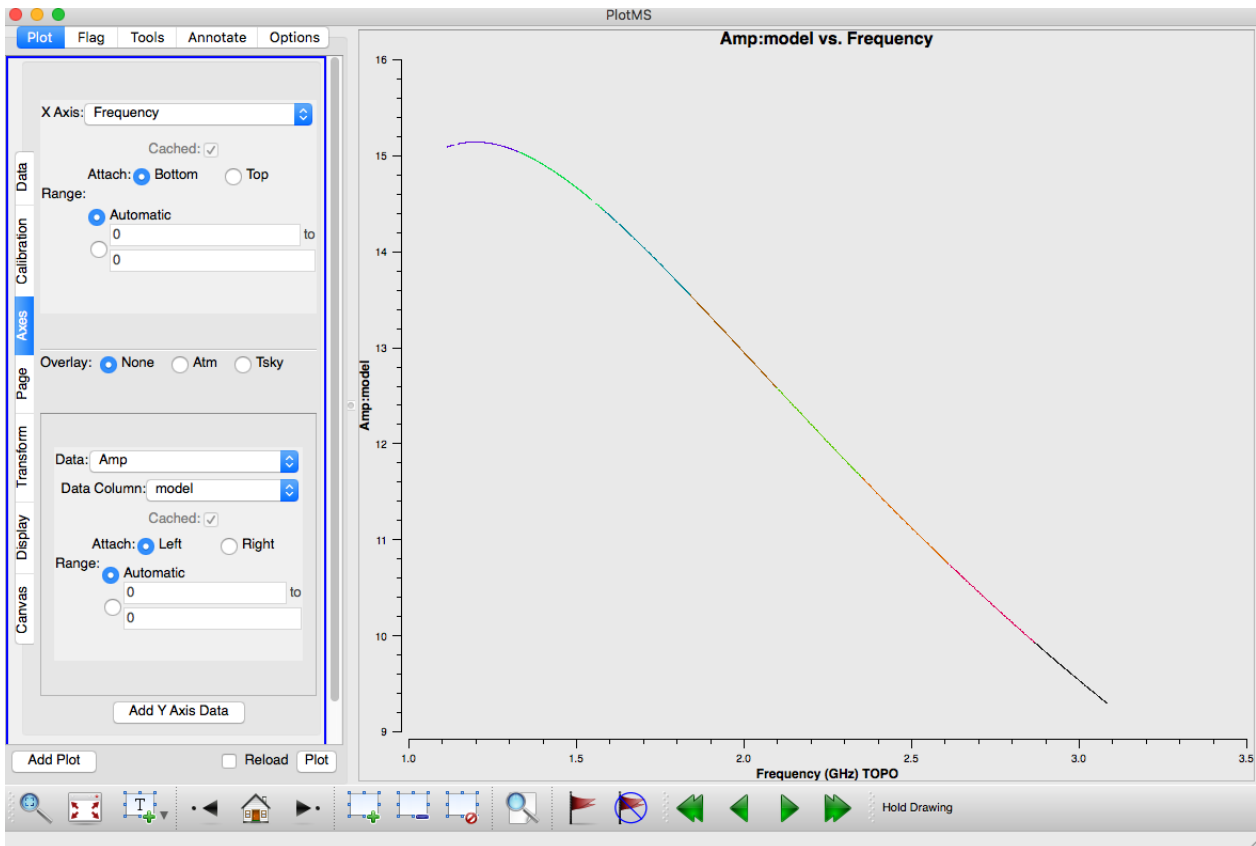


Figure 5: plotms of amp vs frequency for the model of 1934.

The next thing we want to do is derive a bandpass solution for our data. Before we can do that though, we need to phase calibrate the data so we can integrate over a long solution interval to

do the bandpass solution. For the gain calibration we use the task **gaincal** and for the bandpass calibration the task **bandpass**. Because we're processing polarization data from an Az/El telescope we switch on parallactic angle rotation using `parang=True` in all the steps from here.

```
# In CASA
gaincal(vis=msname, caltable='cal.G0', field=pri, refant=ref, gaintype='G',
        calmode='p', parang=True, solint='60s')

bandpass(vis=msname, caltable='cal.B0', field=pri, spw='', refant=ref, solnorm=True,
         solint='inf', bandtype='B', gaintable=['cal.G0'], parang=True)
```

After the bandpass solution we're ready to determine the gains using the secondary calibrator. We also solve for the gains on the primary. We're using a 60s solution interval and apply the bandpass table we've just created using the `gaintable` parameter.

```
# In CASA
gaincal(vis=msname, caltable='cal.G1', field=pri+' '+sec, refant=ref, spw='*',
        gaintype='G', calmode='ap', parang=True, solint='60s', gaintable=['cal.B0'])
```

The remaining calibration to solve for is the polarisation leakage or D-terms. For the ATCA we can often use the unpolarised primary calibrator 1934-638 for this, but you can also use the secondary calibrator. In both cases we need to determine the polarisation of the secondary so we can correct the gains for source polarisation. Here we use **polcal** with the primary calibrator and apply the previously determined bandpass and gain calibration.

```
# In CASA
polcal(vis=msname, caltable='cal.D0', field=pri, refant=ref,
       gaintable=['cal.B0', 'cal.G1'], poltype='Df', solint='inf')
```

```
# In CASA
plotcal(caltable='cal.G1', xaxis='time', yaxis='amp', poln='/', iteration='antenna, spw')
```

4.3 Final Calibration & Applying

We now want to take our initial calibration solutions and determine better, or final, solutions.

We use the `qufromgain` routine to extract polarised source model for the secondary. The `qufromgain` routine works out the Q and U of the calibrator for each spectral window and returns the mean values. We capture those and use them as a first order correction for the next round of calibration. Since we don't know the flux of the secondary yet, it is assumed to be 1 by **gaincal** and our source model is relative to this. The `qufromgain` routine assumes the X feed is offset by 45 degrees for ATCA (except for 7mm observations when it uses 135 degrees, if you have analyzed polarised 7mm data let us know if you think this is incorrect). We also need to select the secondary calibrator with the field id, which is 1. You are advised to use `plotcal` to inspect the gain table used in this step. The `qufromgain` routine looks at the ratio between the X and Y gains, so we specify `poln='/'` to inspect that.

```
# In CASA
from recipes.atcapolhelpers import qufromgain
qu = qufromgain('cal.G1', fieldids=[1])
smodel=[1, qu[1][0], qu[1][1], 0]
```

As you can see (print `smodel`) the secondary calibrator is only very weakly polarised ($\sim 1\%$) so this step is not as important here as it might be for a strongly polarised calibrator. Now we redo the

whole calibration using our best estimates for gains, leakages and secondary polarisation. We have to split the gain calibration into two steps so we can specify the source model for the secondary. We use the parameter `append=True` to get the solutions for the primary and secondary into the same table as that is what we need for the fluxscale step that comes next.

```
# In CASA
bandpass(vis=msname,caltable='cal.B1',field=pri,spw='',refant=ref,solnorm=True,
         solint='inf',bandtype='B',gaintable=['cal.G1','cal.D0'],parang=True)

gaincal(vis=msname,caltable='cal.G2',field=pri,refant=ref,spw='*',
        gaintype='G',calmode='ap',parang=True,solint='60s',gaintable=['cal.B1','cal.D0'])

gaincal(vis=msname,caltable='cal.G2',field=sec,refant=ref,spw='*',
        gaintype='G',calmode='ap',parang=True,solint='60s',
        gaintable=['cal.B1','cal.D0'],smodel=smodel,append=True)

polcal(vis=msname,caltable='cal.D1',field=pri,refant=ref,gaintable=['cal.B1','cal.G2'],
       poltype='Df',solint='inf')
```

Now is a good time to check your solutions with plots. This can be done with **plotms** or **plotcal**. There are different ways of plotting the solutions, and both tasks allow for locating or flagging of bad solutions (see Figures 6, 7, 8, and 9).

```
# In CASA
#can use plotms to inspect the solutions tables.

#iterate over the antenna
plotms(vis='cal.B1', xaxis='freq', yaxis='amp', iteraxis='antenna',coloraxis='spw')
#all antennas on one plot
plotms(vis='cal.B1', xaxis='freq', yaxis='amp',coloraxis='spw')
#can also use plotcal
plotcal(caltable='cal.B1',xaxis='freq',yaxis='amp',iteration='antenna,spw',subplot=331)

#look at gain solutions for primary
plotms(vis='cal.G2', xaxis='freq', yaxis='amp',field=pri, iteraxis='antenna',coloraxis='spw')
plotms(vis='cal.G2', xaxis='time', yaxis='phase',field=pri, iteraxis='antenna',
       plotrange=[0,0,-180.,180. ],coloraxis='spw')

#look at gain solutions for secondary
plotms(vis='cal.G2', xaxis='freq', yaxis='amp',field=sec, iteraxis='antenna',coloraxis='spw')
plotms(vis='cal.G2', xaxis='time', yaxis='phase',field=sec, iteraxis='antenna',
       plotrange=[0,0,-180.,180. ],coloraxis='spw')
```

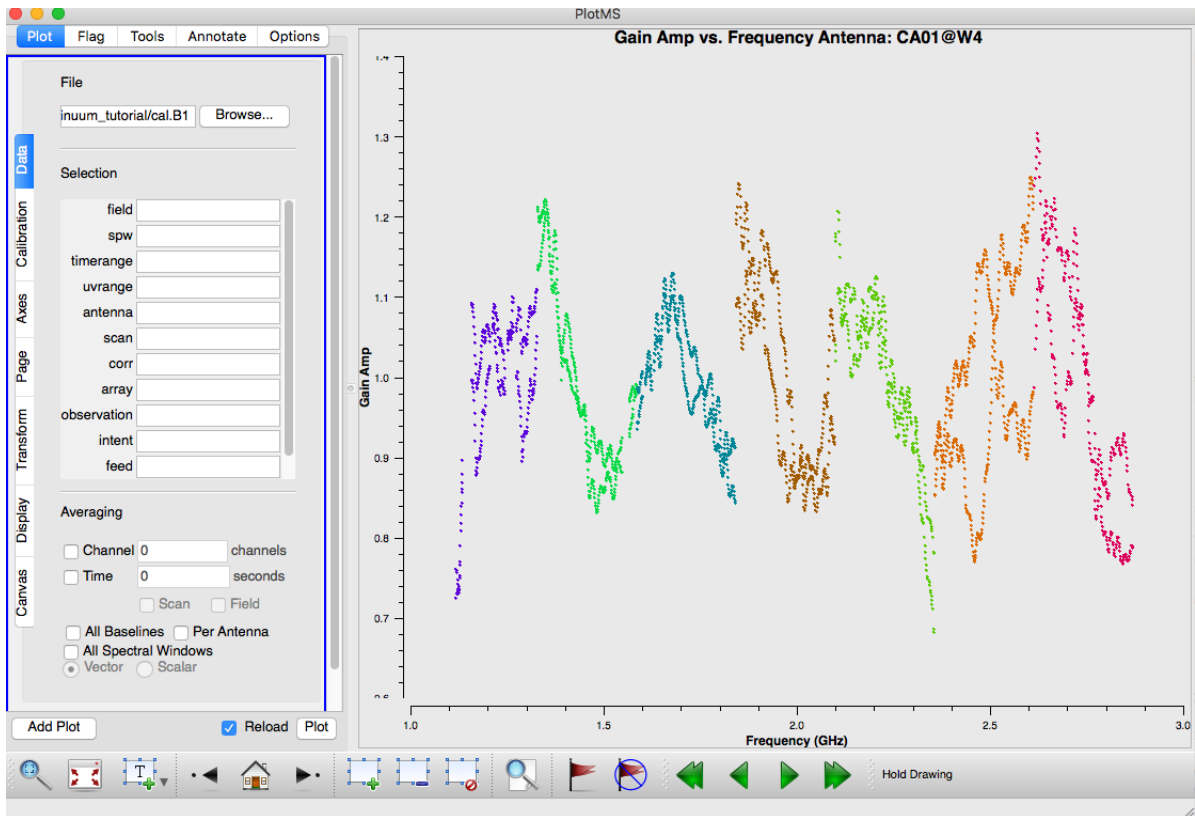


Figure 6: plotms of bandpass solutions caltable cal.B1 iterating over antenna.

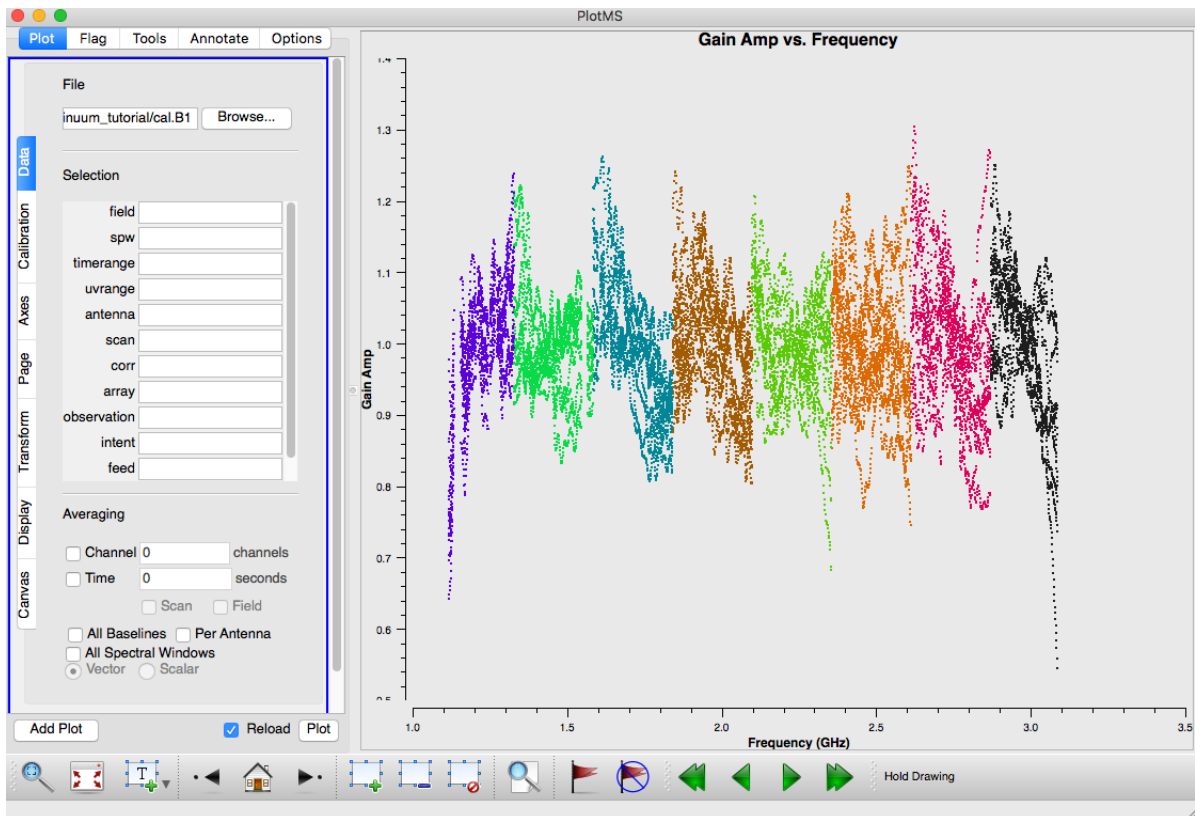


Figure 7: plotms of bandpass solutions caltable cal.B1, all antenna on one plot.

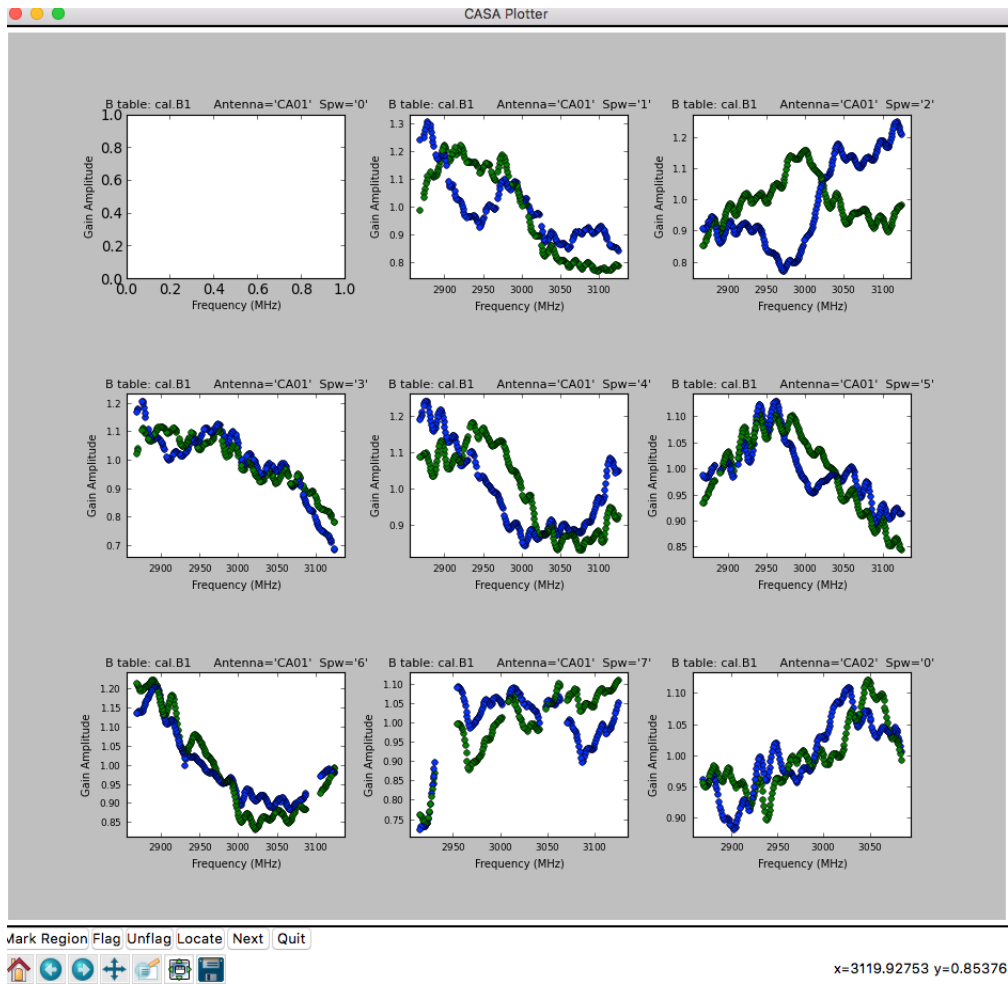


Figure 8: plotcal of bandpass solutions cal.B1 iterating over antenna and spw, using subplots.

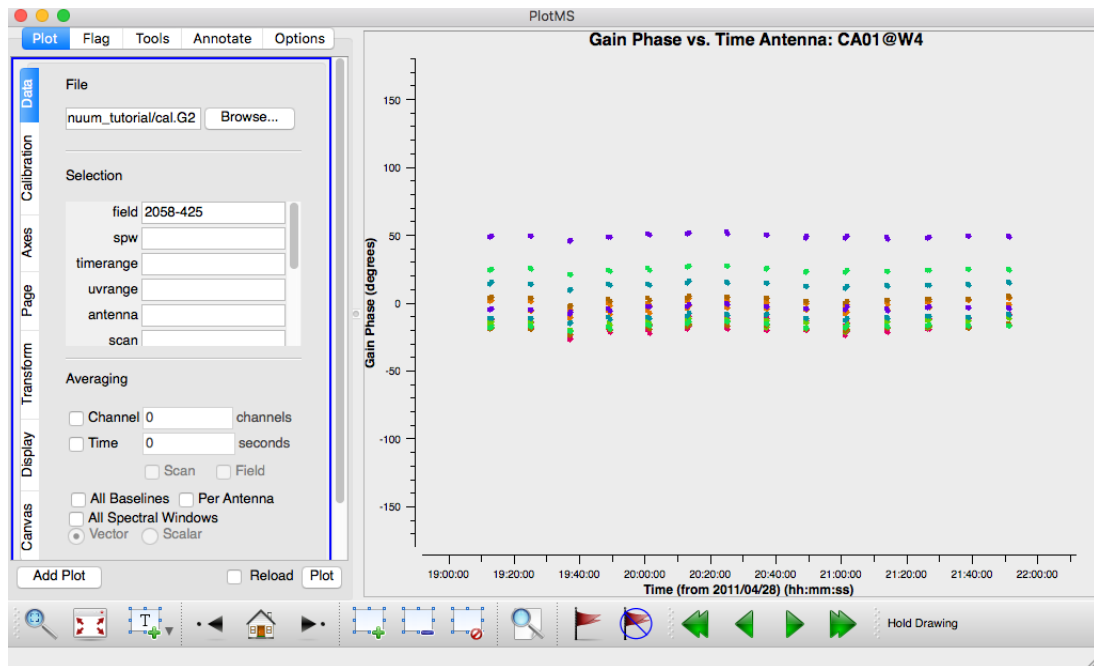


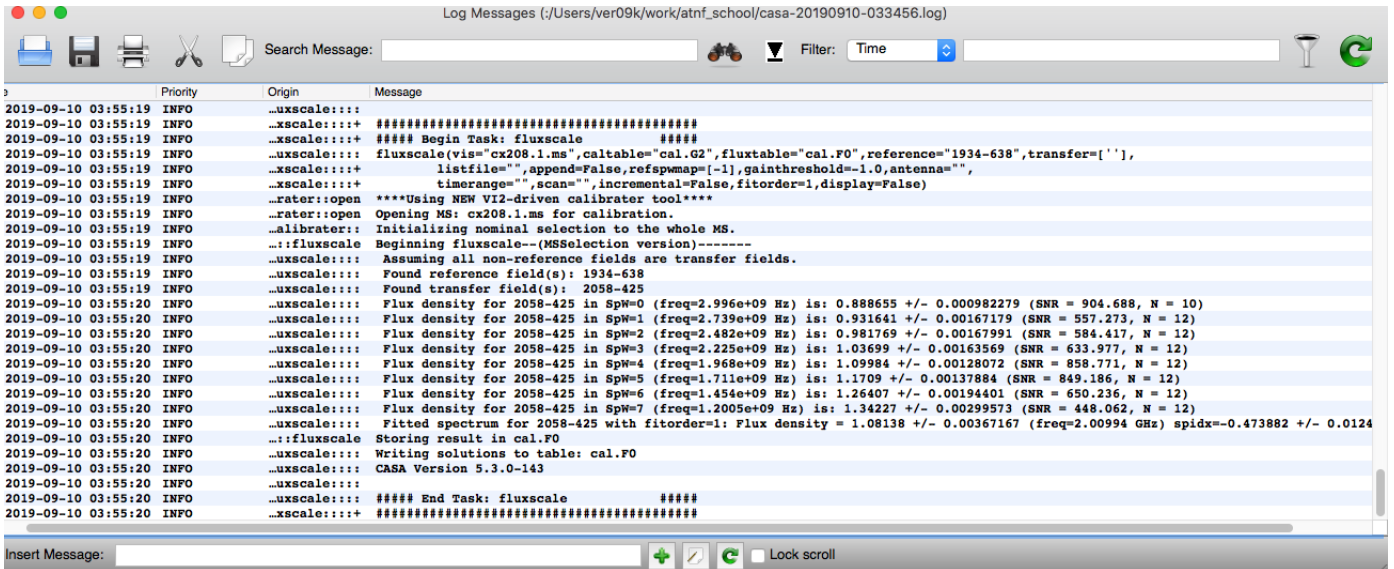
Figure 9: plotms of gain solutions cal.G2 for secondary calibrator, phase vs time, iterating over antenna.

All that remains now is to correct the flux scale using comparison between the primary and secondary calibrator and then apply all the corrections.

```
# In CASA
```

```
flux_transfer=fluxscale(vis=msname,caltable='cal.G2',fluxtable='cal.F0',reference=pri)
```

If you check your logger output you should see the amplitude values for the secondary and a solved for spectral index, something like this (Figure 10):



```
2019-09-10 03:55:19 INFO ..uxscale:::
2019-09-10 03:55:19 INFO ..uxscale::: + #####
2019-09-10 03:55:19 INFO ..uxscale::: + ##### Begin Task: fluxscale #####
2019-09-10 03:55:19 INFO ..uxscale::: fluxscale(vis="cx208.1.ms",caltable="cal.G2",fluxtable="cal.F0",reference="1934-638",transfer=[''],
2019-09-10 03:55:19 INFO ..uxscale::: listfile="",append=False,refspmap=[-1],gainthreshold=-1.0,antenna="",
2019-09-10 03:55:19 INFO ..uxscale::: timerange="",scan="",incremental=False,fitorder=1,display=False)
2019-09-10 03:55:19 INFO ..rater::open ****Using NEW VI2-driven calibrator tool****
2019-09-10 03:55:19 INFO ..rater::open Opening MS: cx208.1.ms for calibration.
2019-09-10 03:55:19 INFO ..alibrator:: Initializing nominal selection to the whole MS.
2019-09-10 03:55:19 INFO ..:fluxscale Beginning fluxscale--(MSSelection version)-----
2019-09-10 03:55:19 INFO ..uxscale::: Assuming all non-reference fields are transfer fields.
2019-09-10 03:55:19 INFO ..uxscale::: Found reference field(s): 1934-638
2019-09-10 03:55:19 INFO ..uxscale::: Found transfer field(s): 2058-425
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=0 (freq=2.996e+09 Hz) is: 0.888655 +/- 0.000982279 (SNR = 904.688, N = 10)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=1 (freq=2.739e+09 Hz) is: 0.931541 +/- 0.00167179 (SNR = 557.273, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=2 (freq=2.482e+09 Hz) is: 0.981769 +/- 0.00167991 (SNR = 584.417, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=3 (freq=2.225e+09 Hz) is: 1.03699 +/- 0.00163569 (SNR = 633.977, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=4 (freq=1.968e+09 Hz) is: 1.09984 +/- 0.00128072 (SNR = 858.771, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=5 (freq=1.711e+09 Hz) is: 1.1709 +/- 0.00137884 (SNR = 849.186, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=6 (freq=1.454e+09 Hz) is: 1.26407 +/- 0.00194401 (SNR = 650.236, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Flux density for 2058-425 in SpW=7 (freq=1.2005e+09 Hz) is: 1.34227 +/- 0.00299573 (SNR = 448.062, N = 12)
2019-09-10 03:55:20 INFO ..uxscale::: Fitted spectrum for 2058-425 with fitorder=1: Flux density = 1.08138 +/- 0.00367167 (freq=2.0094 GHz) spidx=-0.473882 +/- 0.0124
2019-09-10 03:55:20 INFO ..:fluxscale Storing result in cal.F0
2019-09-10 03:55:20 INFO ..uxscale::: Writing solutions to table: cal.F0
2019-09-10 03:55:20 INFO ..uxscale::: CASA Version 5.3.0-143
2019-09-10 03:55:20 INFO ..uxscale:::
2019-09-10 03:55:20 INFO ..uxscale::: ##### End Task: fluxscale #####
2019-09-10 03:55:20 INFO ..uxscale::: + #####
```

Figure 10: Result of running fluxscale.

Once that is done its time to apply our calibration tables using the task **applycal**. We use the gainfield parameter in **applycal** to indicate which solutions to apply from each calibration table.

```
# In CASA
```

```
#save flags first
```

```
flagmanager(vis=msname,mode='save',versionname='before_applycal')
```

```
applycal(vis=msname,gaintable=['cal.B1','cal.D1','cal.F0'],gainfield=[pri,pri,pri],
         field=pri,parang=True,flagbackup=False)
```

```
applycal(vis=msname,gaintable=['cal.B1','cal.D1','cal.F0'],gainfield=[pri,pri,sec],
         field=sec+', '+tar,parang=True,flagbackup=False)
```

5 Inspection & Flagging

Once the calibration has been applied we want to inspect our output and see how well we did. Using **plotms** we can look at the data before (Figure 11) and after calibration (Figure 12).

```
# In CASA
```

```
plotms(vis=msname,field=pri,xaxis='frequency',yaxis='amp',correlation='xx,yy',
       ydatacolumn='data',coloraxis='spw')
```

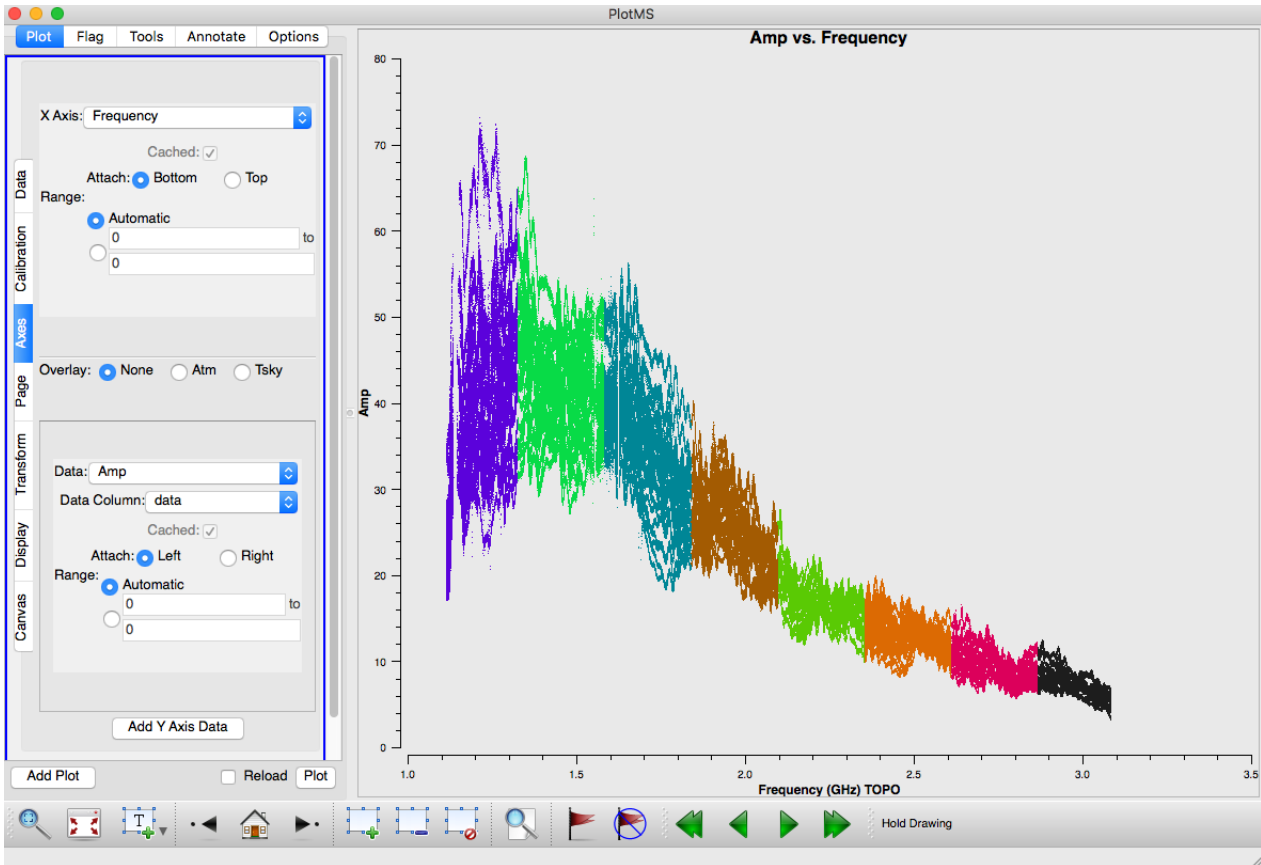


Figure 11: plotms of 1934 amplitude vs frequency **before** applying calibration.

Now make the plot using the "corrected" data axis for the ydata (Figure 12).

```
# In CASA
plotms(vis=msname,field=pri,xaxis='frequency',yaxis='amp',correlation='xx,yy',
       ,ydatacolumn='corrected',coloraxis='spw')
```

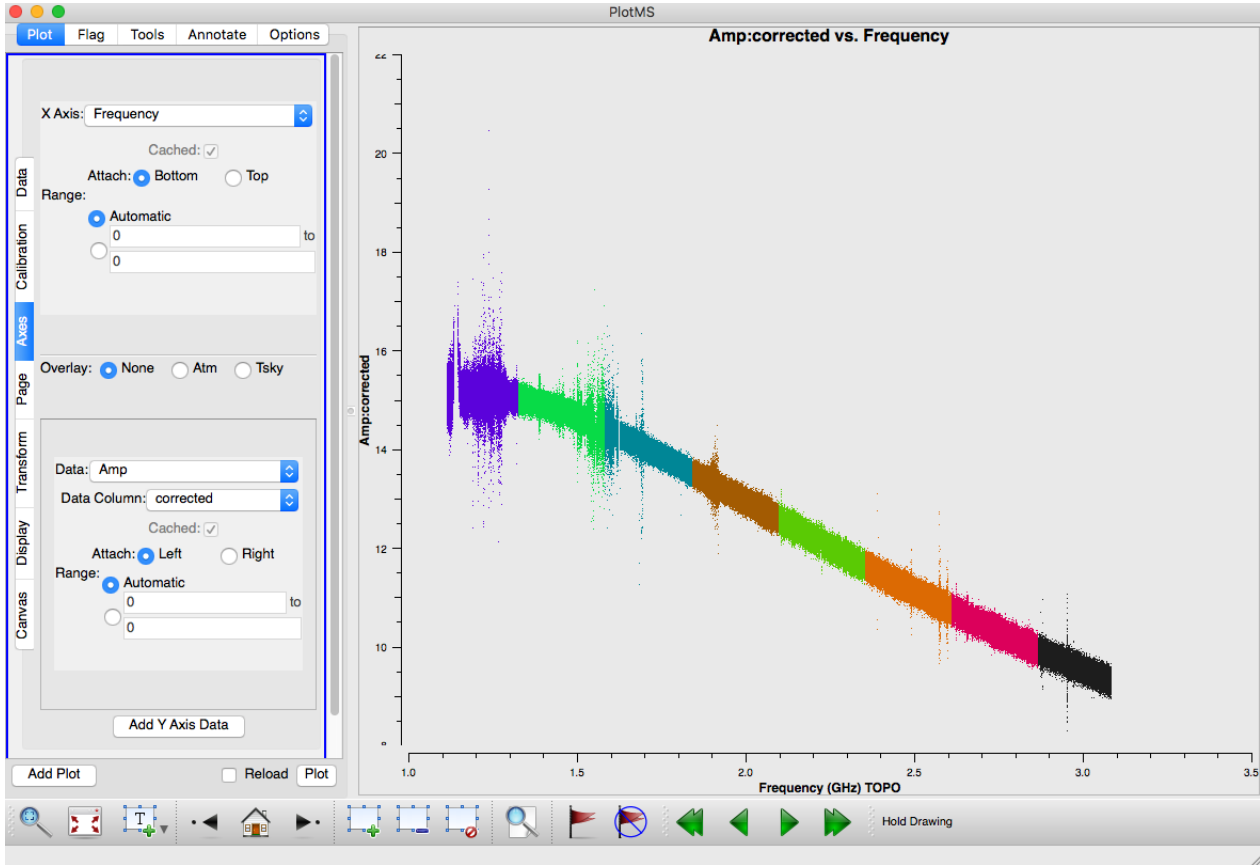


Figure 12: plots of 1934 amplitude vs frequency **after** applying calibration.

We can see it looks much better. And we can do similar for the secondary calibrator (Figure 13).

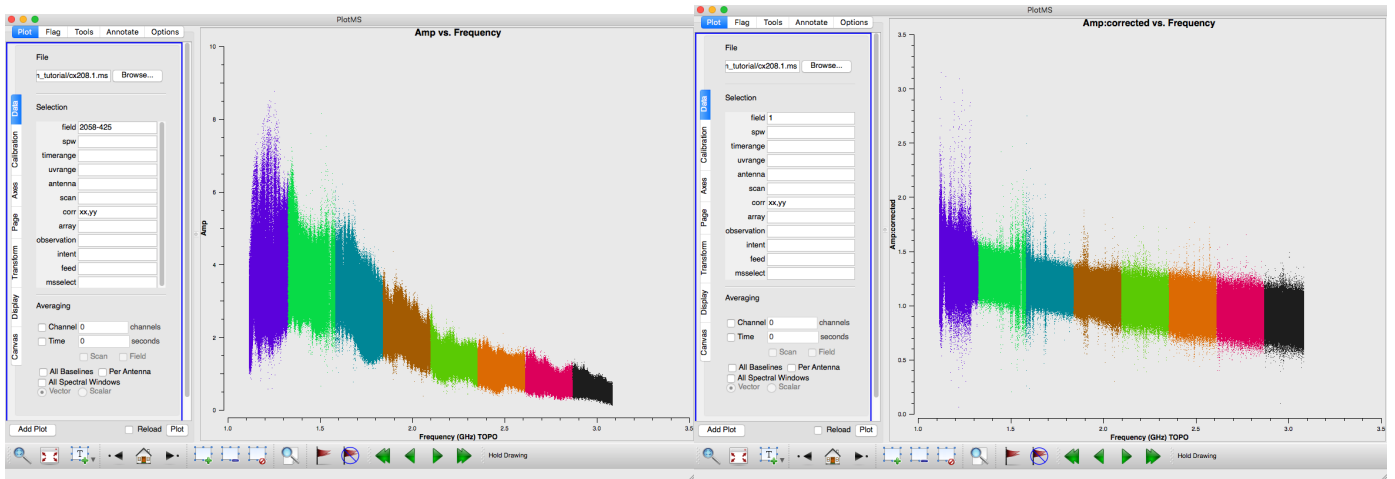


Figure 13: plots of secondary amplitude vs frequency **before** (left) and **after** (right) applying calibration.

Other plots that can be useful to look at are real vs imag, phase vs amp, amp vs uvdistance, amp/phase vs time. In Figure 14 we see the real vs imag for the primary and secondary calibrators. For the primary calibrator we see clump of points that has a spread along the real axis and is tightly constrained around 0 along the imaginary axis. For a well calibrated point source that has a varying amplitude over the frequency range you're plotting, this is what we expect. For the secondary we see a more tightly constrained clump around 0 on the imaginary axis and ~ 1.2 on the real axis, this is what you'd expect for a point source that had a roughly constant amplitude across the band.

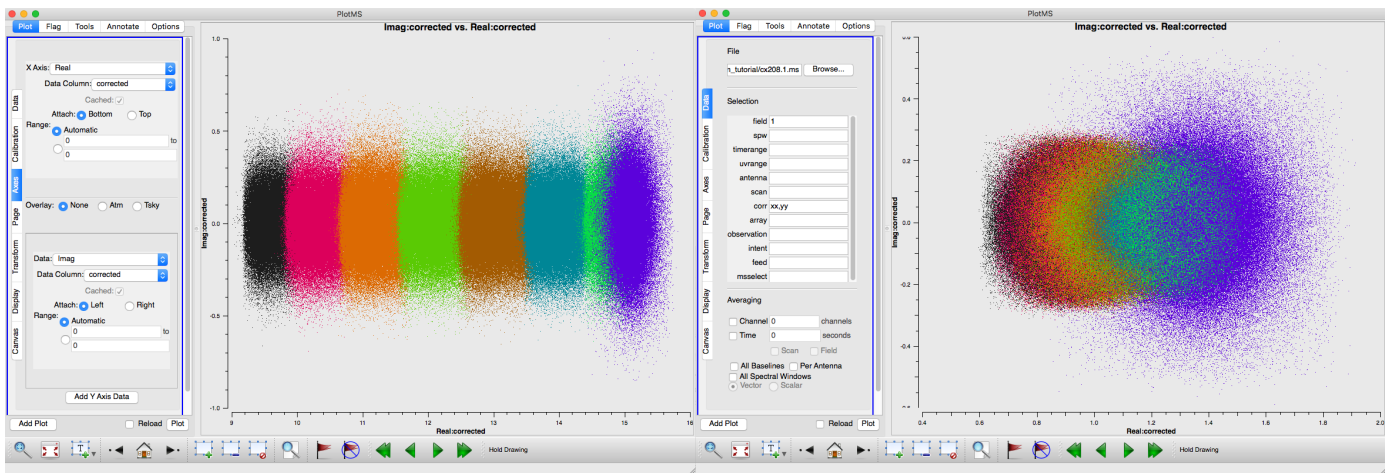


Figure 14: plotms of secondary real vs imag values for the primary calibrator (left) and secondary calibrator (right) after applying calibration.

The plots look pretty good, though some RFI remains. Applying calibration can bring out more RFI. So we can do some more flagging. The other task for automatic RFI flagging in CASA is with `flagdata mode='rflag'`. The difference here is that `rflag` needs to be run on calibrated data. So we can use the round of calibration we just did to run `rflag` on the calibrators and the target corrected data.

```
# In CASA
#first save the flags
flagmanager(vis=msname,mode='save',versionname='before_rflag')

#do rflag for primary calibrator
flagdata(vis=msname, mode='rflag', field=pri, spw='0~7', datacolumn='corrected',
         action='apply',display='report', correlation='ABS_ALL', timedevscale=3.0,
         freqdevscale=3.0, winsize=3, combinescans=True, ntime='999999min',
         extendflags=False,flagbackup=False)

#do rflag for secondary calibrator
flagdata(vis=msname, mode='rflag', field=sec, spw='0~7', datacolumn='corrected',
         action='apply',display='report', correlation='ABS_ALL', timedevscale=3.0,
         freqdevscale=3.0, winsize=3, combinescans=True, ntime='999999min',
         extendflags=False,flagbackup=False)

#do rflag for target
flagdata(vis=msname, mode='rflag', field=tar, spw='0~7', datacolumn='corrected',
         action='apply',display='report', correlation='ABS_ALL', timedevscale=3.0,
         freqdevscale=3.0, winsize=3, combinescans=True, ntime='999999min',
         extendflags=False,flagbackup=False)

#extend flags for both
flagdata(vis=msname, mode='extend', field=pri+' '+sec, spw='0~7', action='apply',
         display='report',flagbackup=False, extendpols=True, correlation='', growtime=95.0,
         growfreq=95.0, growaround=True, flagneartime=False, flagnearfreq=False,
         combinescans=True, ntime='999999min')
```

Now if we make the same plot for the primary calibrator again (Figure 15 compared to Figure 12) we see the majority of RFI is gone.

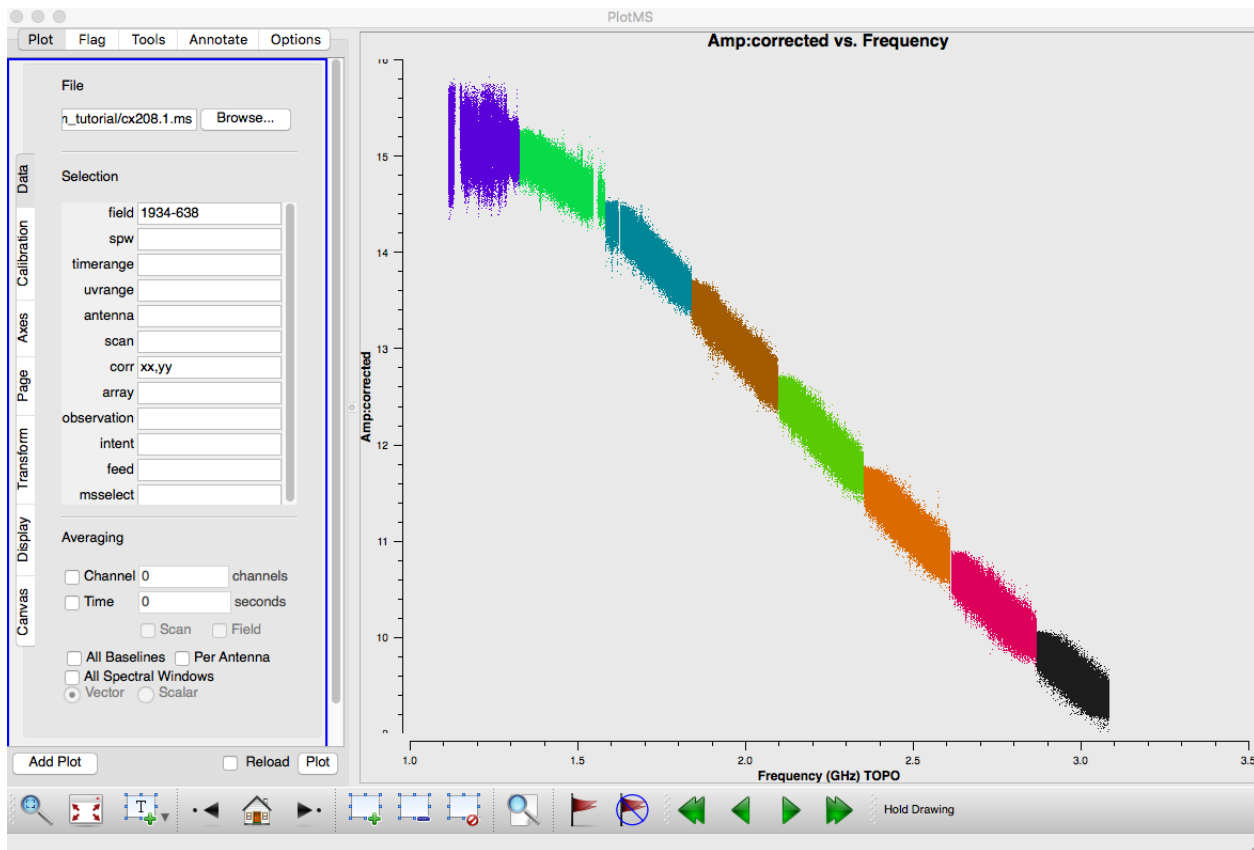


Figure 15: plotms of 1934 amplitude vs frequency **after** applying calibration and after running rflag.

We could now go back and generate a new round of calibration solutions from the newly flagged data (rerun steps from Section 4.3), but for now we will proceed.

6 Splitting

Once the target data has been calibrated and flagged you may want to split it off into its own measurement set. This is achieved with the task **split**. We select the 'corrected' data column to get the calibrated visibilities.

```
# In CASA
split(vis=msname, outputvis='cx208_target.ms',datacolumn='corrected')

targetms='cx208_target.ms',

#save the flags for the new MS
flagmanager(vis=targetms,mode='save',versionname='after_split')
```

7 Imaging

7.1 Cleaning

Now we are finally ready to make some images. The imaging task in CASA is called **clean** (in **clean** or **help(clean)** for more information). The full 1.1-3.1 GHz band is too wide to image properly in one go (even with multi-frequency synthesis, MFS), so we will break it up into two chunks. We use `mode='mfs'` and `nterms=2` since its still 1GHz of bandwidth we need to perform multi-frequency synthesis.

We use the `spw` parameter to select the first 4 spws (`spw='0~3'`), or the higher frequency half of the band. At an average frequency of about 2.7 GHz the primary beam is about 17' and the resolution is about $\sim 20''$, but because of our *uv* coverage the beam is highly elliptical with a much smaller minor axis. Based on this we'll choose a cell size of 1 arcsecond (we could go to 0.5 arcseconds) and an image size of 2048 pixels. We will make a test image to see what the field looks like and use the interactive option so we can set the area to clean. If you wanted to perform self-calibration you would set `usescratch=True`, this would populate the MODEL column with the model clean components to use for self calibration.

```
# In CASA
clean(vis=targetms, imagename='2051-377.2700.0', spw='0~3', mode='mfs',
      nterms=2, niter=3000, threshold='5e-5Jy', imsize=2048, cell='1.0arcsec', stokes='I',
      weighting='briggs', robust=0.5, interactive=True)
```

This will open an interactive cleaning window. You can zoom in and out of different regions. And you can set clean regions. In the example we've used a simple box, but you can use polygons or other shapes. Once you have set the clean area you can save it to a file and then enter this as the mask for deeper cleaning later. Once you put your clean boxes around the bright sources selecting the green arrow will continue deconvolution, but bring back the interactive viewer after completing one cycle. Then you can set more boxes, as more (fainter) point sources should show up now. Once you're happy with your boxes you can hit the blue arrow to continue cleaning non-interactively until the number of iterations is met or the threshold is reached (see Figures 16, 17, 18, and 19 for examples).

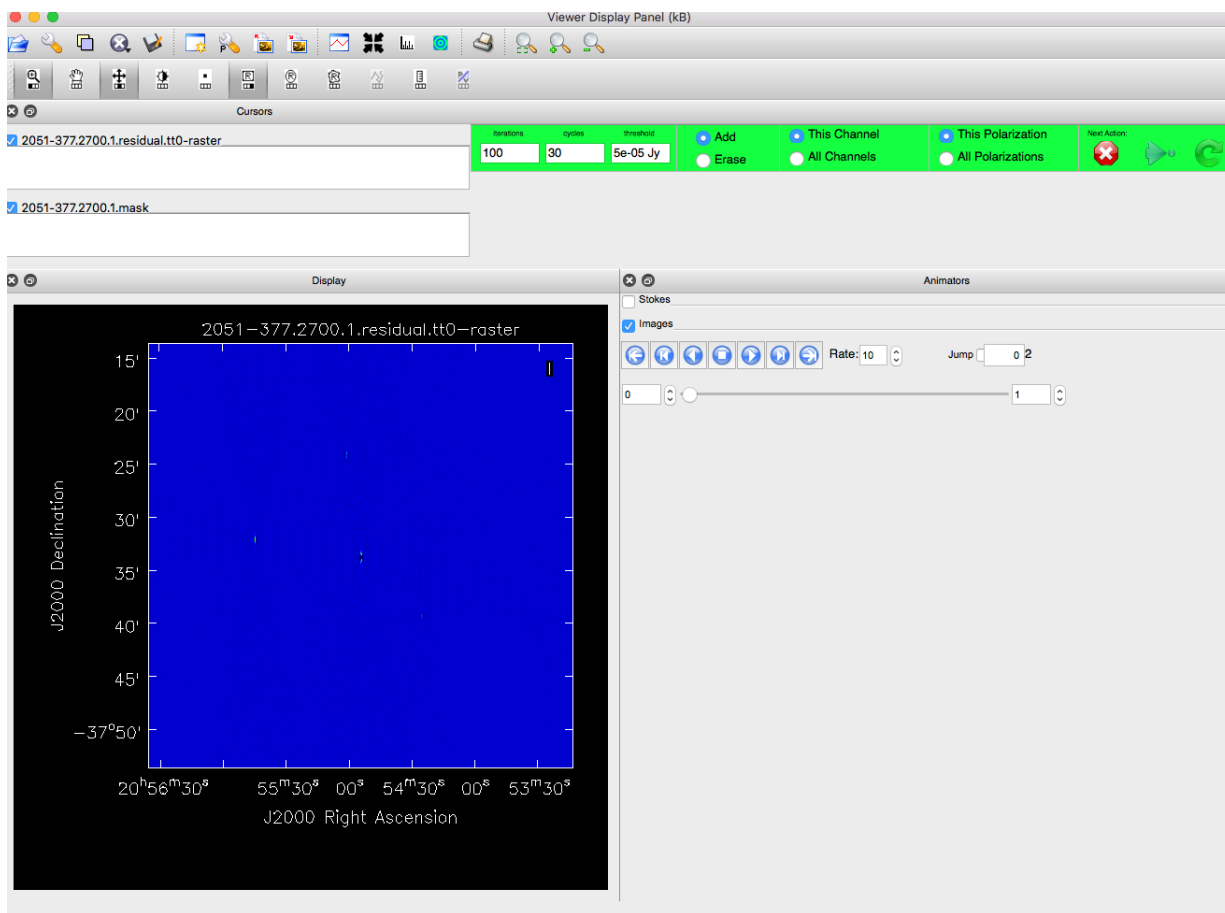


Figure 16: Target cleaning interactive window.

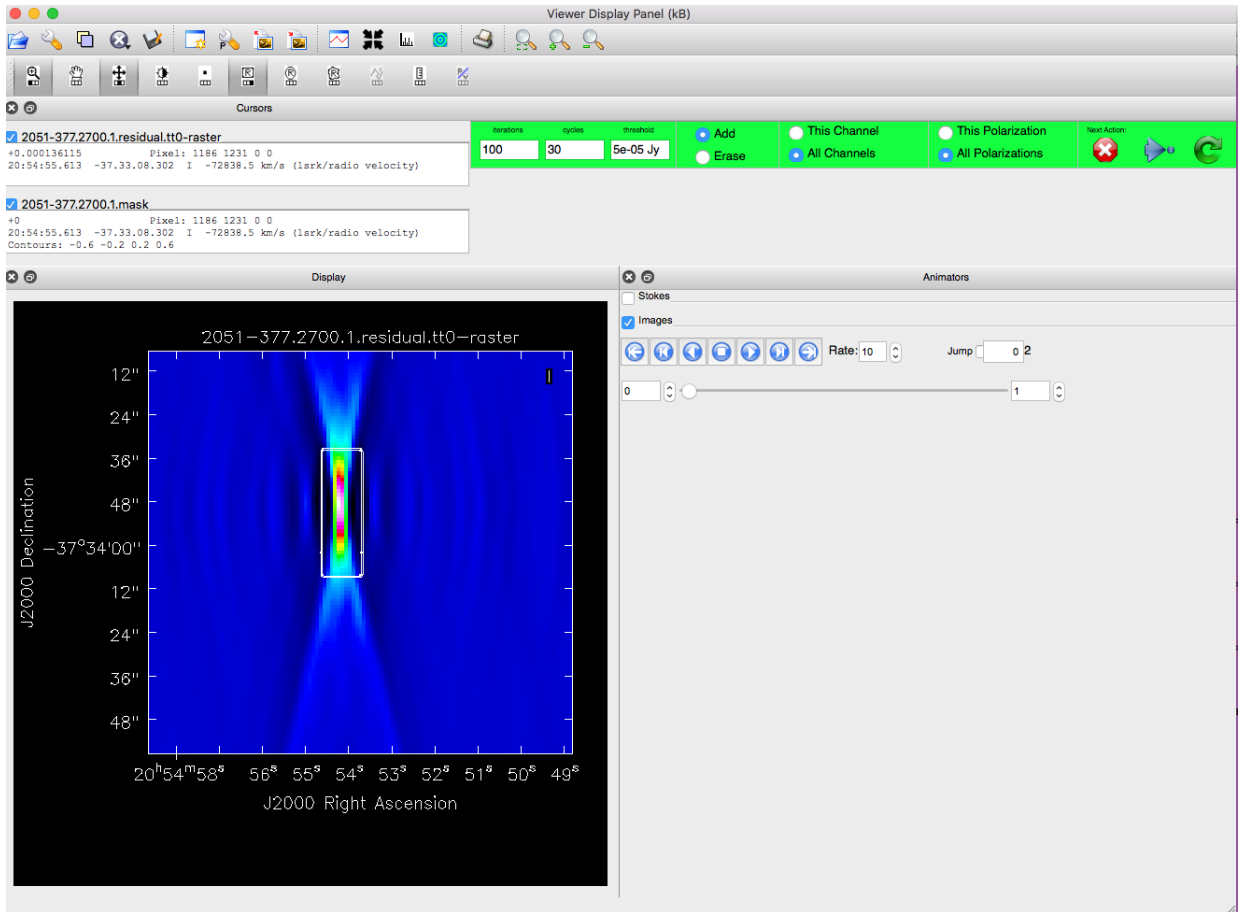


Figure 17: Target cleaning interactive window zoomed in on central source, with clean box placed.

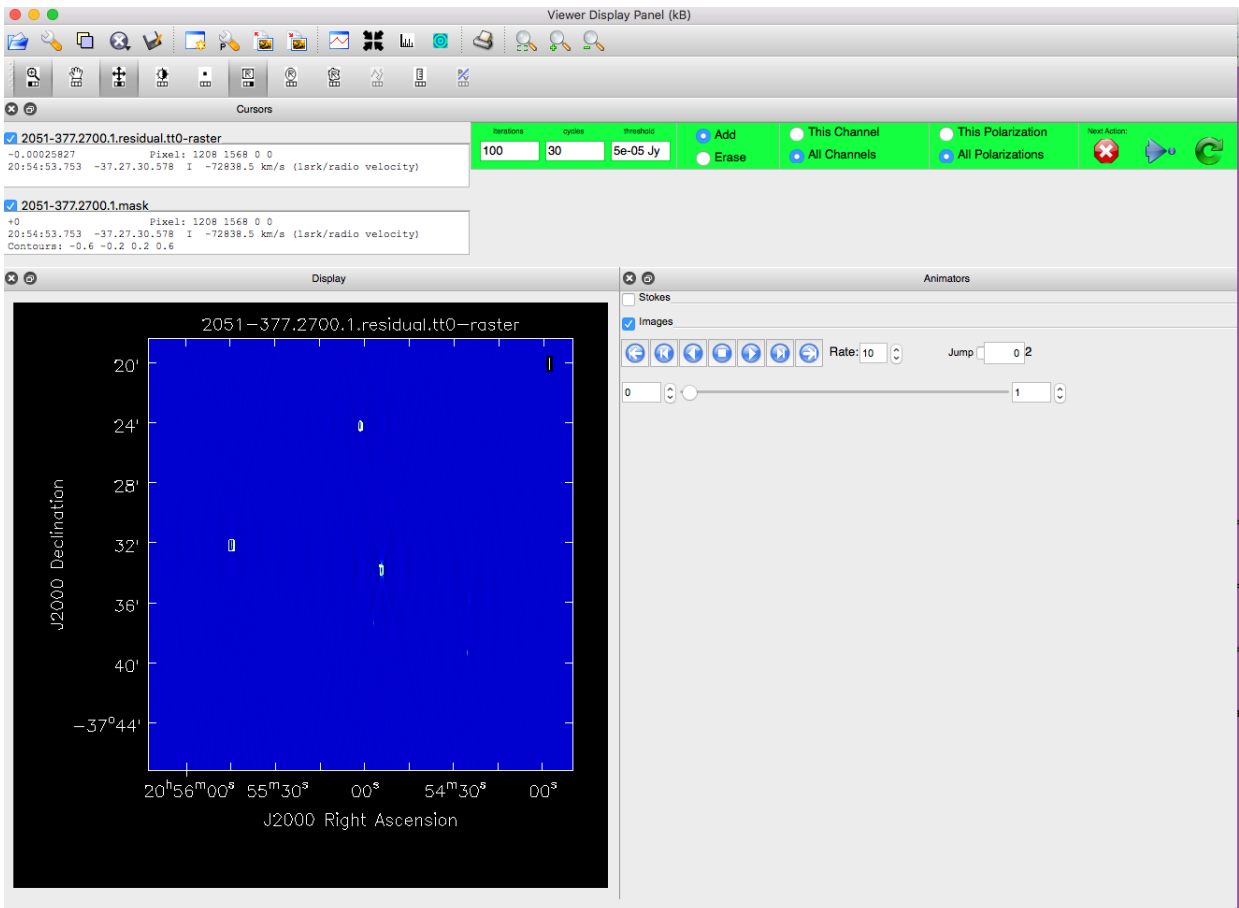


Figure 18: Target cleaning interactive window, zoomed out, several boxes placed.

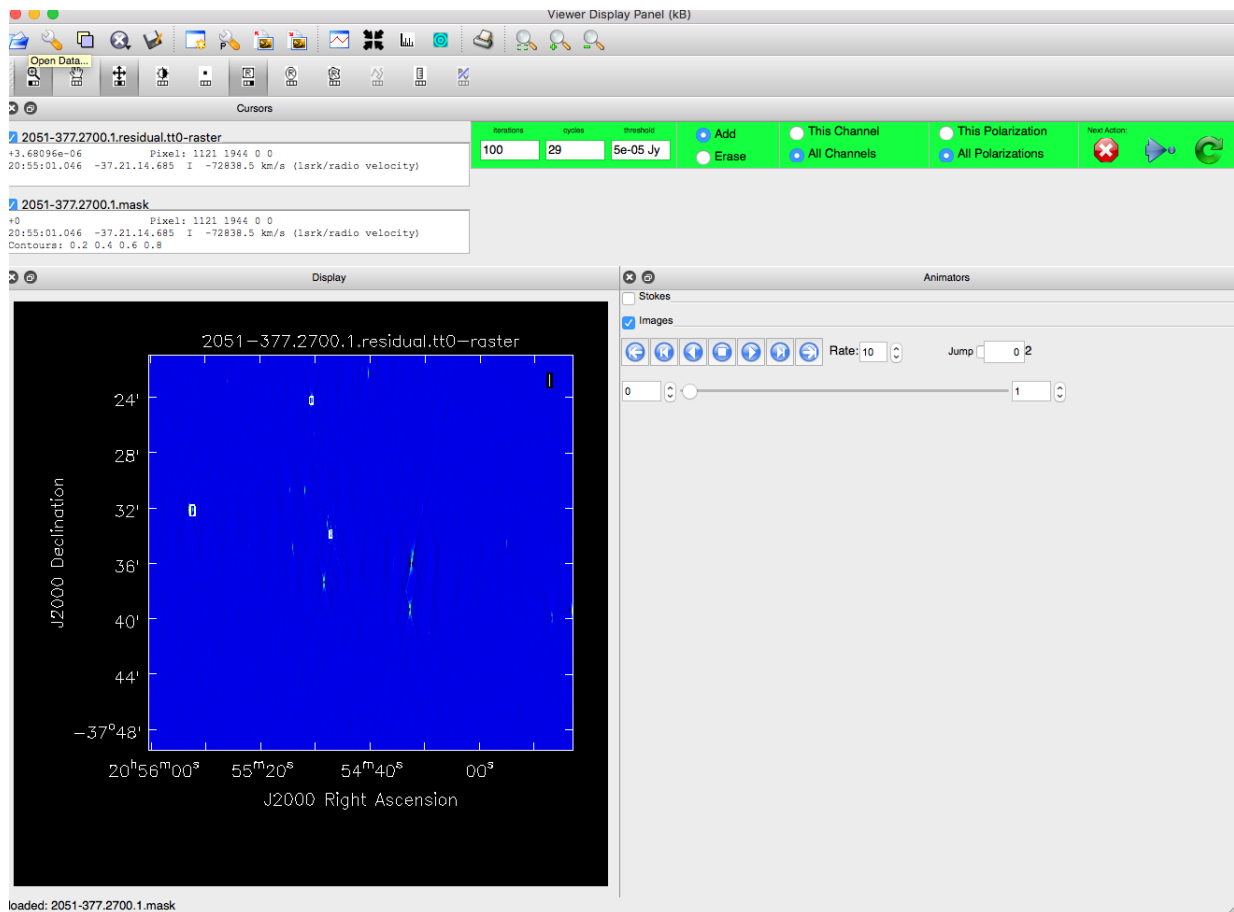


Figure 19: Target cleaning interactive window, zoomed out, several boxes placed **after** first cycle of cleaning, more point sources now visible.

Once the cleaning is done, there will be several files or images created. There will be

- .mask : mask file containing the locations of the boxes you created
- .flux : the primary beam attenuation pattern (used to create a primary beam corrected image)
- .model.tt# : model images, or clean component images for the first two taylor terms (nterms=2)
- .residual.tt# : residual images
- .psf.tt# : beam or point spread function images, showing the synthesised dirty beams
- .image.tt# : the final cleaned and restored images
- .image.alpha (and .image.alpha.error) : spectral index images created from the taylor terms.

7.2 Image Analysis

To view any of this images you can open the CASA viewer by simply type `viewer()` at the CASA prompt, which will open a viewer window (similar to the clean window) and a menu should appear asking which image(s) to load (select the image you want and hit "raster image"). You can open more than one at a time, and the CASA viewer can do things like overlay contours, measure/display a spectrum, as well as other features.

If you prefer to view the images in a different fits viewer you can use the CASA task `exportfits`:

```
# In CASA
exportfits(imagename='2051-377.2700.0.image.tt0',fitsimage='2051-377.2700.0.image.tt0.fits')
```

You can then go on and clean the other part of the band (spw='4~7').

```
# In CASA
clean(vis=targetms, imagename='2051-377.1600.0', spw='4~7', mode='mfs',
      nterms=2, niter=3000, threshold='5e-5Jy', imsize=2400, cell='1.0arcsec', stokes='I',
      weighting='briggs', robust=0.5, interactive=True)
```

You can use the task **imhead** to get header information, such as the center frequency or the beam shape.

```
# In CASA
head1600=imhead('2051-377.1600.0.image.tt0', mode='list')
head2700=imhead('2051-377.2700.0.image.tt0', mode='list')

CASA <24>: head2700
Out[24]:
{'beammajor': {'unit': 'arcsec', 'value': 21.770307540893555},
 'beamminor': {'unit': 'arcsec', 'value': 2.556743860244751},
 'beampa': {'unit': 'deg', 'value': 0.52105712890625},
 'bunit': 'Jy/beam',
 'cdelt1': -4.84813681109536e-06,
 'cdelt2': 4.84813681109536e-06,
 'cdelt3': 1.0,
 'cdelt4': 1027892614.7435131,
 'crpix1': 1200.0,
 'crpix2': 1200.0,
 'crpix3': 0.0,
 'crpix4': 0.0,
 'crval1': 5.475563285,
 'crval2': -0.655560211,
 'crval3': 1.0,
 'crval4': 2610222502.292313,
 'ctype1': 'Right Ascension',
 'ctype2': 'Declination',
 'ctype3': 'Stokes',
 'ctype4': 'Frequency',
 'cunit1': 'rad',
 'cunit2': 'rad',
 'cunit3': '',
 'cunit4': 'Hz',
 'datamax': 0.026629474014043808,
 'datamin': -0.0006516462890431285,
 'date-obs': '2011/04/28/18:59:15',
 'equinox': 'J2000',
 'imtype': 'Intensity',
 'masks': array([],
                 dtype='|S1'),
 'maxpixpos': array([1202, 1190, 0, 0], dtype=int32),
 'maxpos': '20:54:54.232 -37.33.49.000 I 2610501343.58Hz',
 'minpixpos': array([1205, 1168, 0, 0], dtype=int32),
 'minpos': '20:54:53.979 -37.34.11.000 I 2610501343.58Hz',
 'object': '2051-377',
 'observer': 'obs',
 'projection': 'SIN',
 'reffreqtype': 'LSRK',
 'restfreq': array([ 2.10000000e+09]),
 'shape': array([2400, 2400, 1, 1], dtype=int32),
 'telescope': 'ATCA'}
```

There is the task **imstat** which can give statistics about the images.

```
# In CASA
stats1600=imstat(imagename='2051-377.1600.0.image.tt0')
stats2700=imstat(imagename='2051-377.2700.0.image.tt0')

CASA <27>: stats2700
Out[27]:
{'blc': array([0, 0, 0, 0], dtype=int32),
 'blcf': '20:56:35.775, -37.53.36.310, I, 2610501343.58Hz',
 'flux': array([ 0.08951914]),
 'max': array([ 0.02662947]),
 'maxpos': array([1202, 1190,    0,    0], dtype=int32),
 'maxposf': '20:54:54.232, -37.33.49.000, I, 2610501343.58Hz',
 'mean': array([ 9.80188502e-07]),
 'medabsdevmed': array([ 1.67812486e-05]),
 'median': array([ -2.50002543e-08]),
 'min': array([-0.00065165]),
 'minpos': array([1205, 1168,    0,    0], dtype=int32),
 'minposf': '20:54:53.979, -37.34.11.000, I, 2610501343.58Hz',
 'npts': array([ 5760000.]),
 'q1': array([ -1.67777380e-05]),
 'q3': array([ 1.67850194e-05]),
 'quartile': array([ 3.35627574e-05]),
 'rms': array([ 8.68720098e-05]),
 'sigma': array([ 8.68664874e-05]),
 'sum': array([ 5.64588577]),
 'sumsq': array([ 0.04346926]),
 'trc': array([2399, 2399,    0,    0], dtype=int32),
 'trcf': '20:53:14.011, -37.13.37.325, I, 2610501343.58Hz'}
```

From here you could do some self calibration, but that is not covered in this tutorial (or you could go back and try this data with the miriad tutorial and see how the results compare).