Radio Frequency Interference (RFI): Identification and Excision

Anna D. Kapińska (NRAO)

This talk...

In this talk we are using **CASA 5.7.2** (`casa -r 5.7.2-4.el7` on lustre)

Interactive steps can be done later, after the talk. All relevant task parameters are given on the slides.

Data set used in this talk is an excerpt (0.23GB) of the 3C75 data, and can be accessed here:
http://www.aoc.nrao.edu/~akapinsk/drw/drwRFI.ms.tar

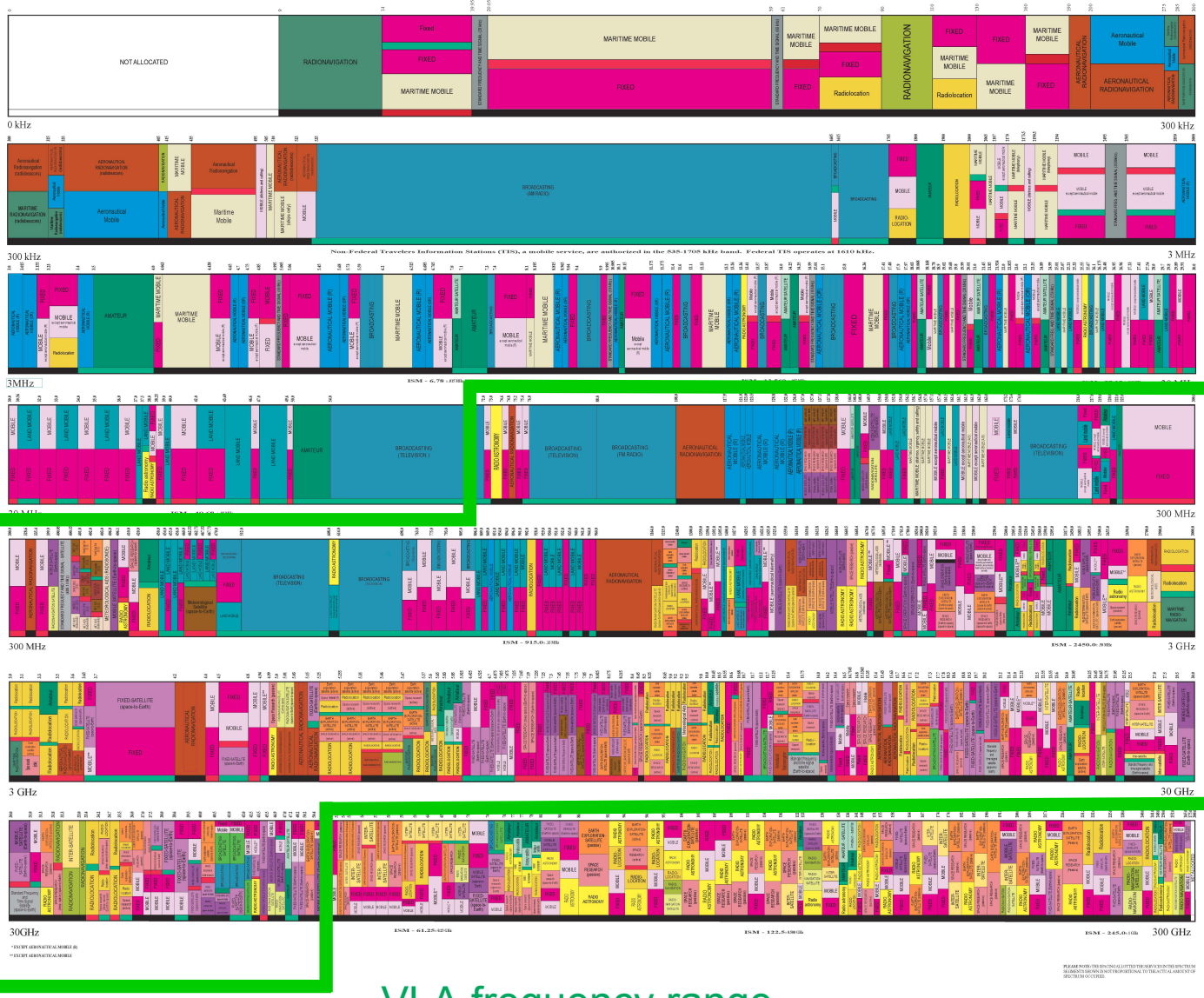
What is RFI?

RFI – Radio Frequency Interference

A disturbance caused by various sources emitting around our targeted frequencies that affect our data, introducing noise. Often RFI is stronger than the science data, and hinders signal we are after.

UNITED STATES FREQUENCY ALLOCATIONS

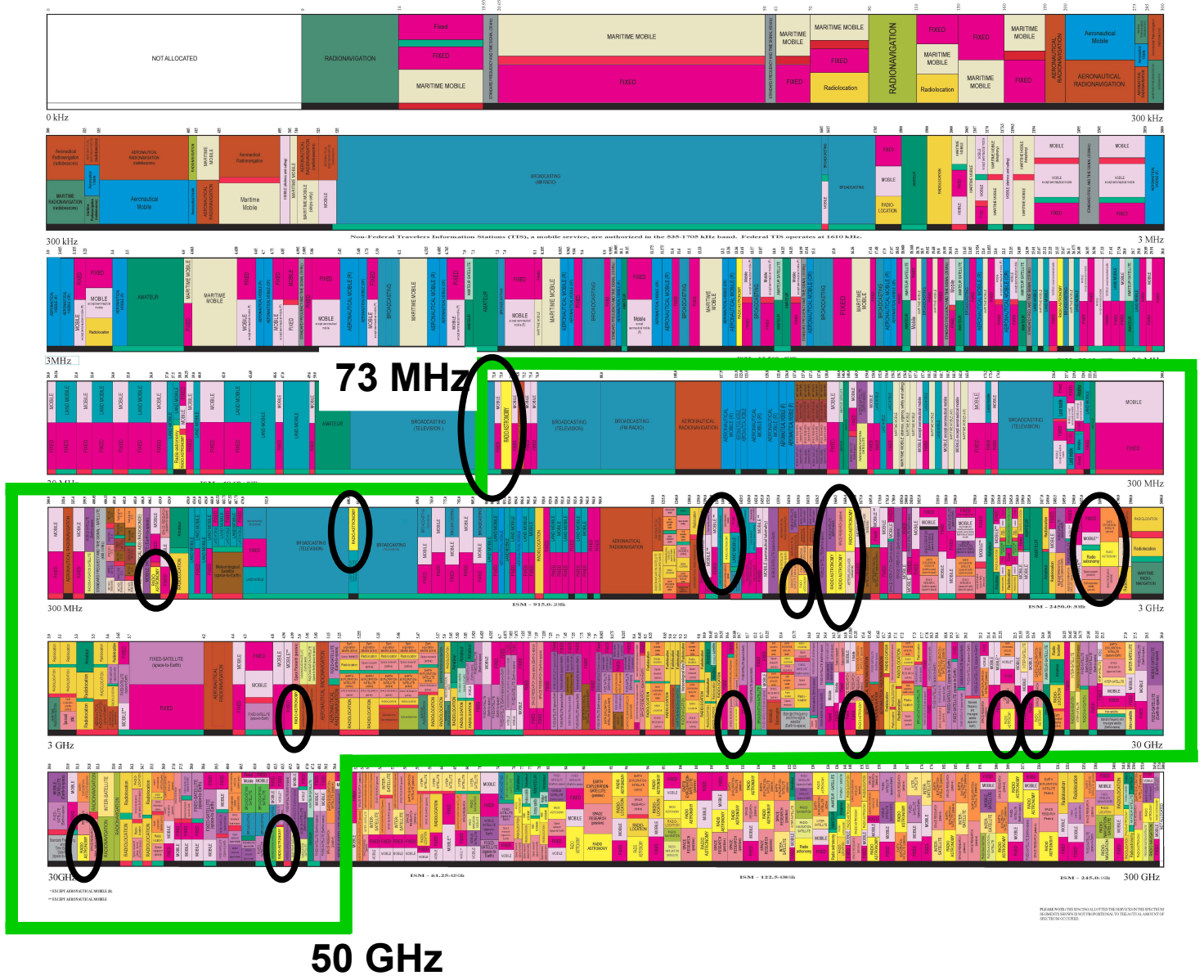
THE RADIO SPECTRUM



VLA frequency range

UNITED STATES FREQUENCY ALLOCATIONS

THE RADIO SPECTRUM



RADIO SERVICES COLOR LEGEND

AERONAUTICAL MOBILE
AERONAUTICAL MOBILE SATELLITE
AERONAUTICAL RADIONAVIGATION
AMATEUR
AMATEUR SATELLITE
BROADCASTING
BROADCASTING SATELLITE
EARTH EXPLORATION SATELLITE
FIXED
FIXED SATELLITE
INTER-SATELLITE
LAND MOBILE
LAND MOBILE SATELLITE
MARITIME MOBILE
MARITIME MOBILE SATELLITE
MARITIME RADIONAVIGATION
METEOROLOGICAL
METEOROLOGICAL SATELLITE
MOBILE
MOBILE SATELLITE
RADIO ASTRONOMY
RADIODETERMINATION SATELLITE
RADIOLOCATION
RADIOLOCATION SATELLITE
RADIONAVIGATION
RADIONAVIGATION SATELLITE
SPACE OPERATION
SPACE RESEARCH
STANDARD FREQUENCY AND TIME SIGNAL
STANDARD FREQUENCY AND TIME SIGNAL SATELLITE

ACTIVITY CODE

FEDERAL EXCLUSIVE
FEDERAL/NON-FEDERAL SHARED
NON-FEDERAL EXCLUSIVE

ALLOCATION USAGE DESIGNATION

SERVICE	EXAMPLE	DESCRIPTION
Primary	FIXED	Capital Letters
Secondary	Mobile	1st Capital with lower case letters

This chart is a graphic single-point-in-time portrayal of the Table of Frequency Allocations used by the FCC and NTIA. As such, it may not completely reflect all aspects, i.e. footnotes and recent changes made in the Table of Frequency Allocations. Therefore, for complete information, users should consult the Table to determine the current status of U.S. allocations.

U.S. DEPARTMENT OF COMMERCE
National Telecommunications and Information Administration
Office of Spectrum Management

JANUARY 2016

RFI at the VLA

https://science.nrao.edu/facilities/vla/docs/manuals/obsguide/rfi

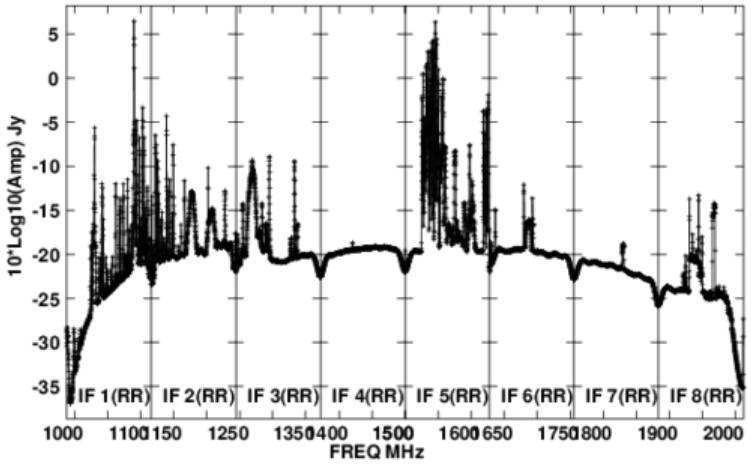
Biggest issue at lower frequency bands: 4, P, L, S, C, and D-configuration.

But it does not mean it doesn't exist at higher frequency bands.

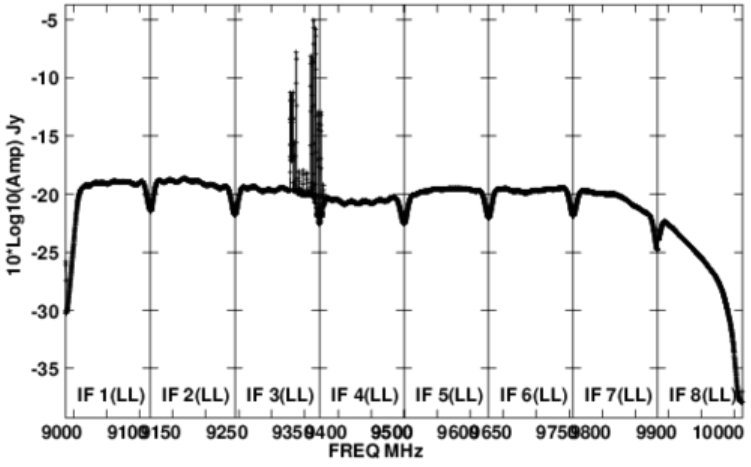
Can be internal or external. Internally-generated RFI is minimised, your NRAO staff working hard to eliminate these!

Examples of RFI in the VLA wide band observations (January 2021, A configuration)

L band



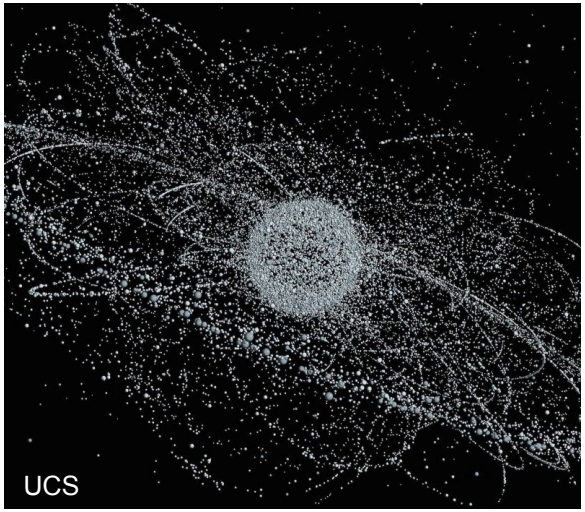
X band (in 9-10 GHz)



Satellite Transmissions & Clarke Belt

Earth is now heavily surrounded by the satellites.

2000+ satellites orbiting Earth, hundreds along the *Clarke Belt* (zone of geosynchronous / geostationary satellites).

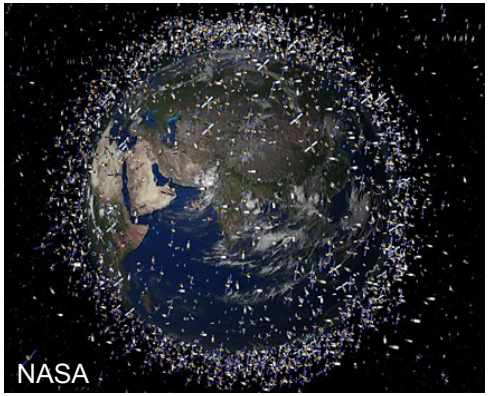


How does it affect VLA data?

→significant degradation of data can occur if VLA antenna observes within 10° of the satellite, @VLA Clarke Belt at declinations –15° to +5°

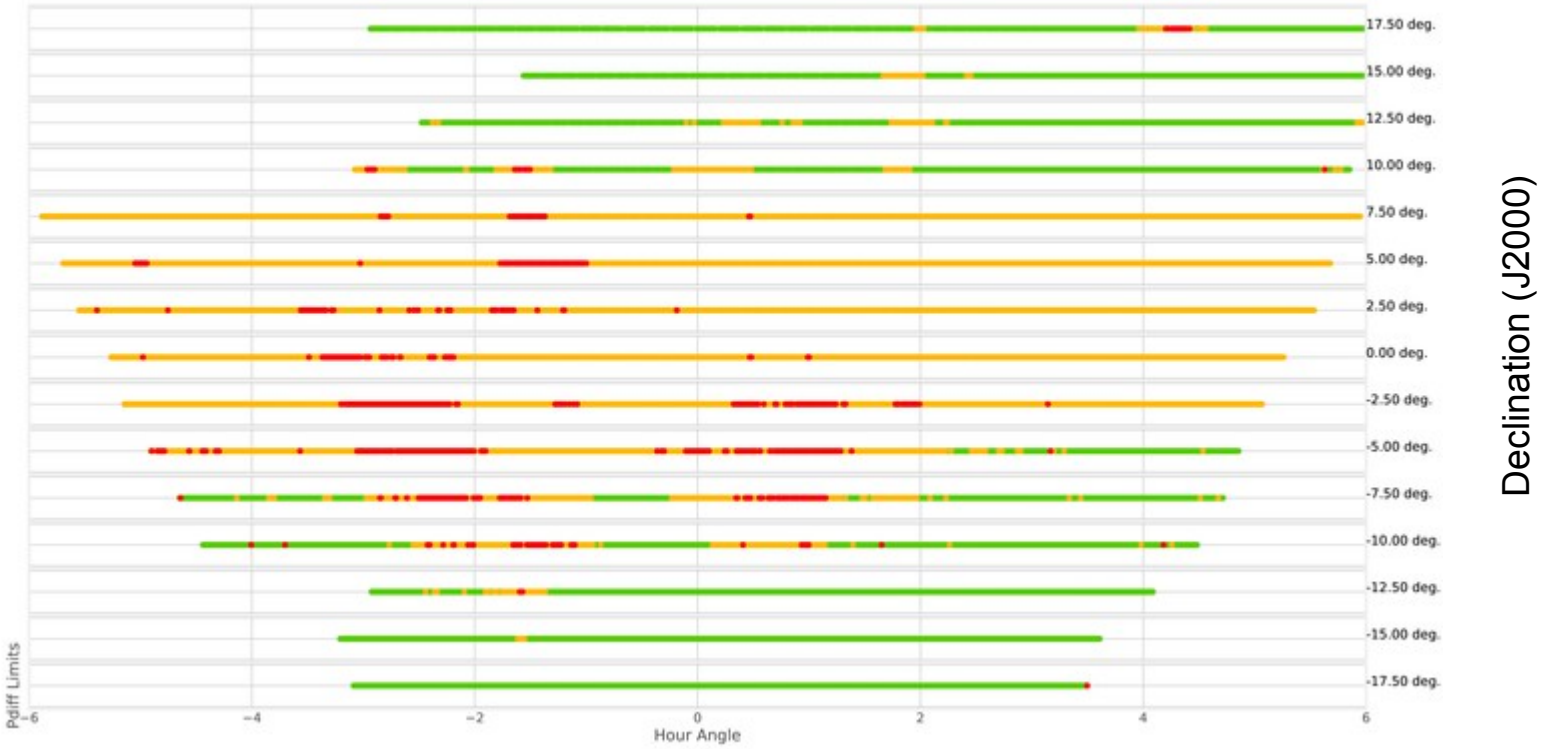
→mainly S, C, Ku, K and Ka bands

→in C, X and Ku bands satellites can also saturate the 3-bit samplers (special 8/3-bit set up required)



Satellite Transmissions & Clarke Belt

S band (2 – 4 GHz) survey of satellite interference at VLA (conducted in 2016/2017)



Hour angle

>20% compression
5–20% compression
<5% compression

Finding RFI in your data

CASA task that allow you to visually inspect the data: `plotms()`

→ Lorant's talk yesterday

Remember: if you leave off strong RFI in your data,
the images will have issues such as 'ripples', high noise etc.

But this is also how you can check for the remaining
offending RFI in your data: **image!**

CASA Flagging tasks

`flagcmd()` → apply flags info on which stored in external file

Example: Online Flags
→ issues recorded by the operators such as *slew, subreflector, focus errors*

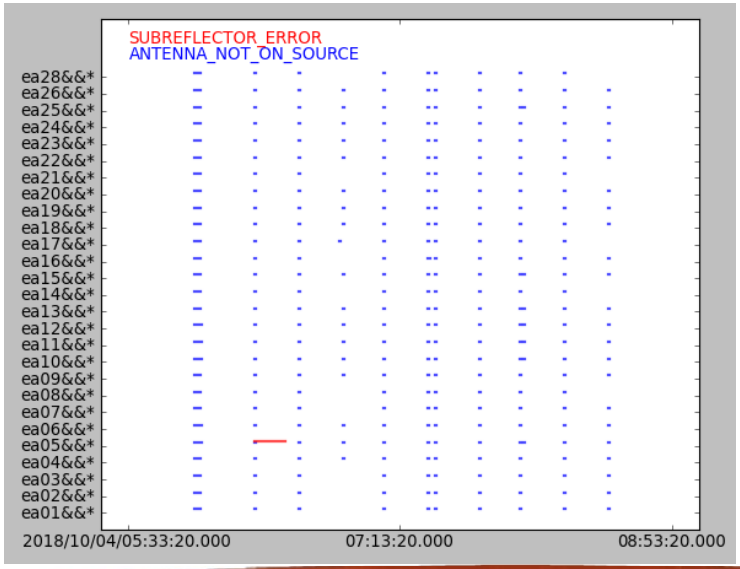
During downloading data from NRAO archive → apply online flags

Apply telescope flags: ☑ Apply flags generated during observing

But if you choose not to do that before getting archive data, these flags can be applied afterwards e.g. with `flagcmd()`.

Or you may want to plot the online flags to inspect them →

`flagcmd(vis='drwRFI.ms', action='plot')`



CASA Flagging tasks

`flagdata()` → most often used task to flag data

Deterministic

```
mode = 'manual'      #Flag subset of data based on MS-selection syntax
mode = 'unflag'      #Unflag data with MS-selection syntax (careful!)
mode = 'quack'       #Flag data at the beginning/end of a scan
mode = 'shadow'      #Flag baselines with shadowed antennas
mode = 'elevation'   #Flag data between specified elevation limits
mode = 'clip'        #Threshold-based flagging on data expressions (incl.
                         flagging zero-amplitude data)
```

Autoflag

```
mode = 'tfcrop'   #Find and flag outliers on the 2D time-frequency plane
mode = 'rflag'    #Find and flag outliers via sliding-window RMS filters
mode = 'extend'   #Extend flags around existing ones
```

Operational

```
mode = 'summary'  #Count existing flags and return statistics
mode = 'list'     #Apply flags from external list
```

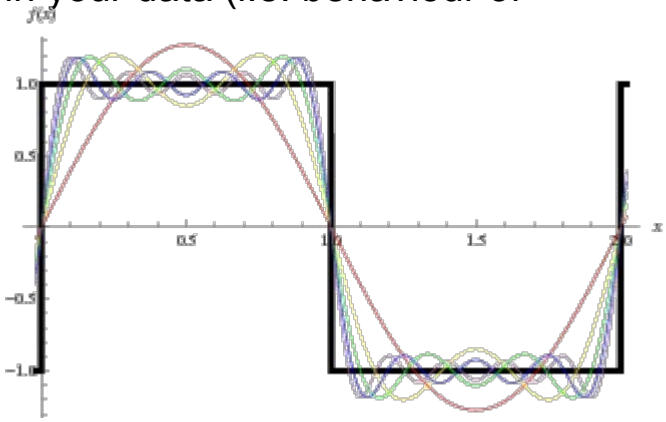
Little help first: Hanning smoothing

Do not use this on spectral line data!

Strong RFI can give rise to the Gibbs phenomenon occurring in your data (i.e. behaviour of Fourier series at a discontinuity).

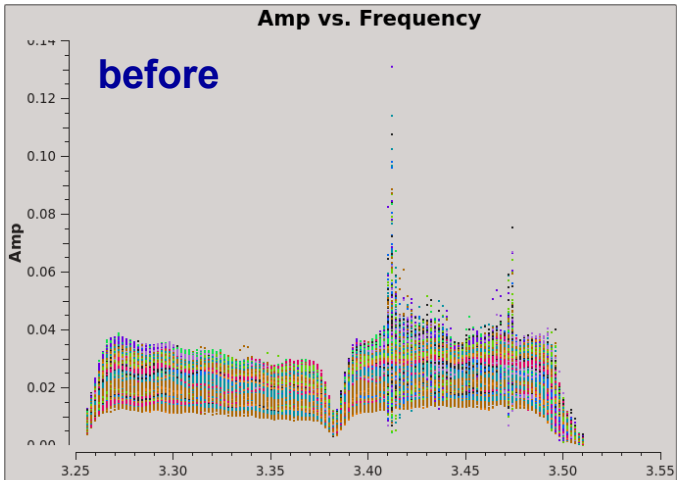
This is seen as ‘ringing’ → a pattern spreading to channels neighbouring the strong narrow RFI spike.

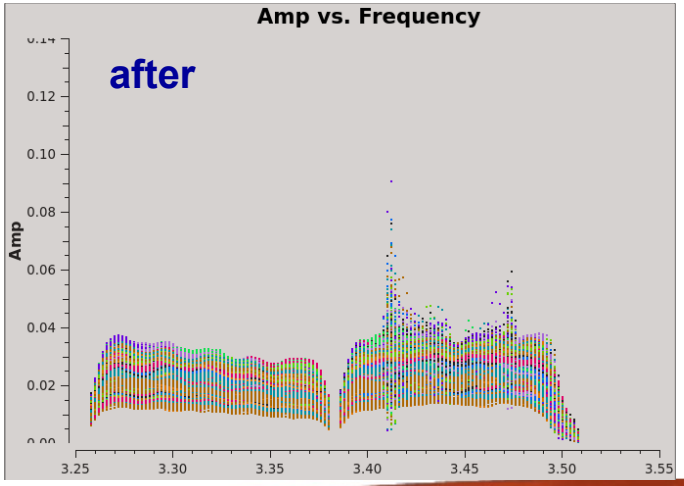
Hanning smoothing algorithm, `hanningsmooth()`, will:
→ remove amplitude spikes and reduce the ringing, reducing number of affected channels
→ but also reduce spectral resolution by a factor of 2



Wolfram

Our data →

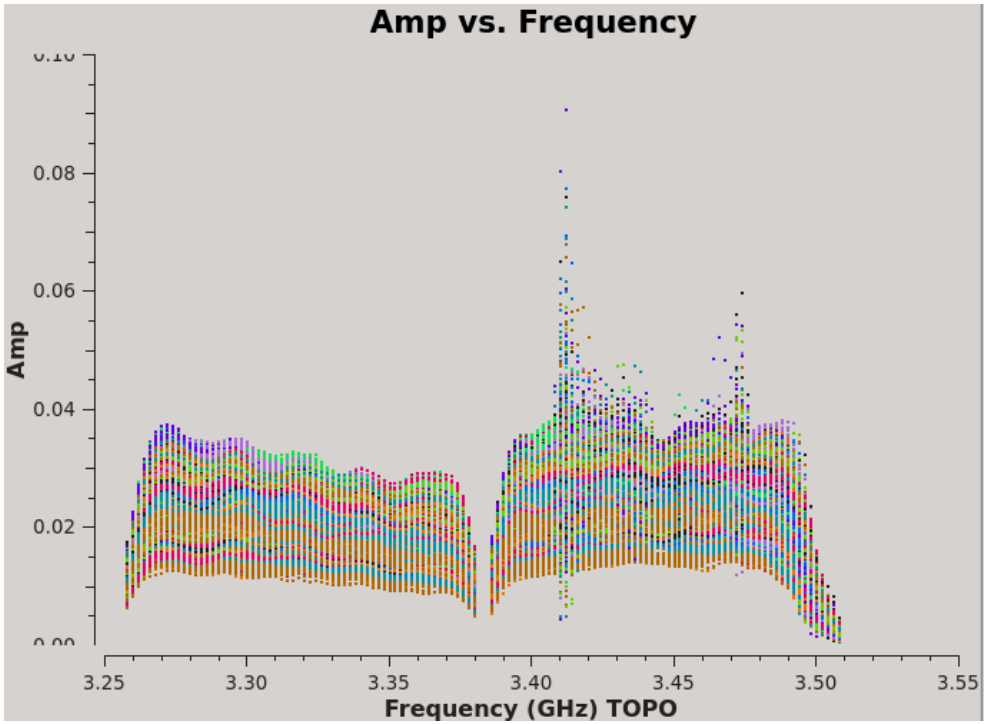




Manual flagging in action

Let us first use `plotms()` to see where the RFI is in our dataset we are using here.

```
# In CASA
default plotms
vis='drwRFI.ms'
xaxis='frequency'
yaxis='amp'
avgtime='1e4'
coloraxis='Antenna1'
customsymbol=True
symbolshape='circle'
symbolsize=2
inp

go
```



INTERACTIVE

Manual flagging in action

Let us first use `plotms()` to see where the RFI is in our dataset we are using here.

Let's remove this spike

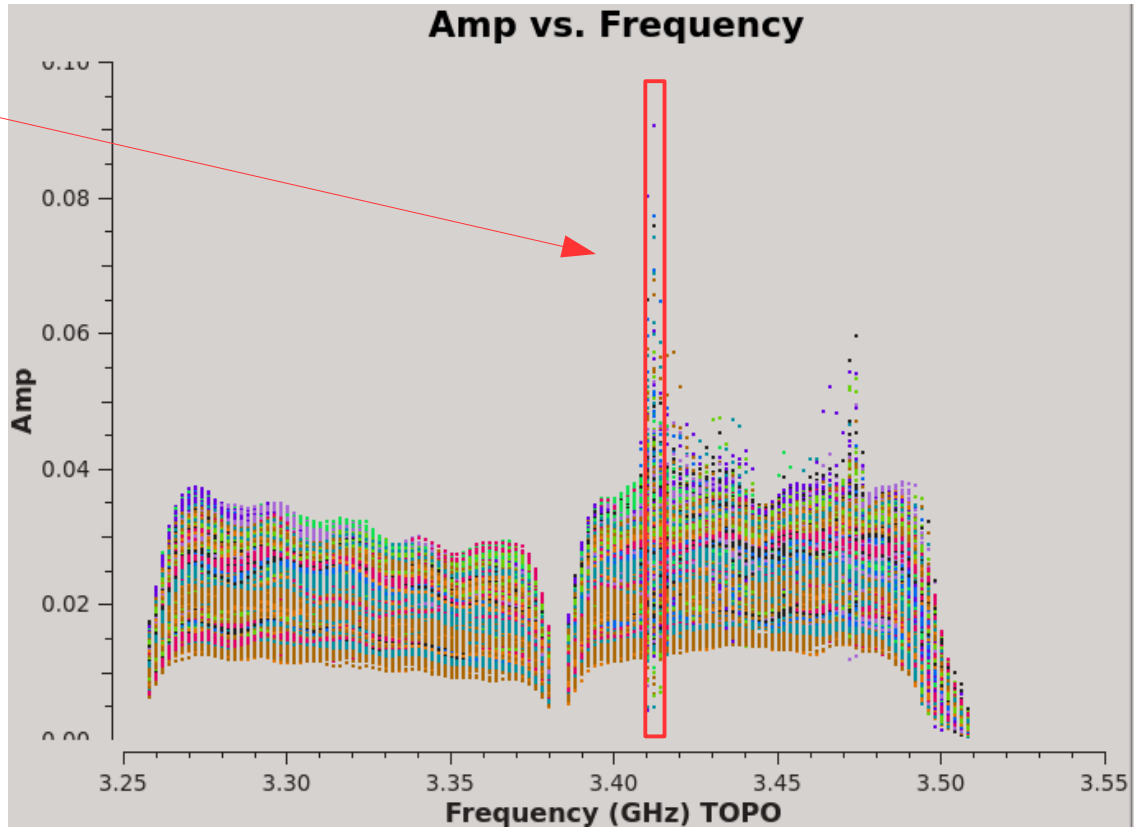
It's in:

spw=1, channels=13~15

After inspecting (paging through) all the antennas one by one it is clear almost half of them are affected, so we will fully remove those channels.

To page through antennas run `plotms()` with parameter `iteraxis='antenna'` and use these buttons

[**Note:** See Lorant's talk for how to *Locate* the data you may want to flag]



Manual flagging in action

Once the RFI is located, the best is to use flagging tasks outside the **plotms(**. This way we can backup the flags, and revert steps if needed.

flagdata() task with **mode = 'manual'**

```
# In CASA
default flagdata
vis='drwRFI.ms'
mode='manual'
spw='1:13~15'
flagbackup=True        # required if to restore previous flagging versions
inp

go
```

INTERACTIVE

Manual flagging in action

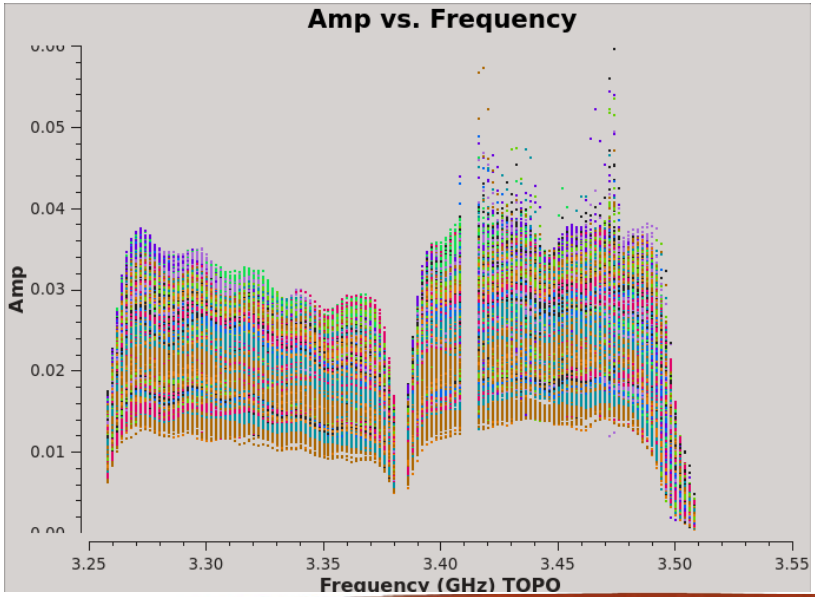
To see the effect of our flagging get back to `plotms()` and inspect the data again.

→ if you still have the `plotms()` running, check the *Reload* box and click *Plot*

→ if you previously closed the task, then just type

```
tget plotms()
go
```

This is the output you should see now.
Notice that the largest RFI spike is gone.



INTERACTIVE

Flagging Manager

You might have noticed that we used parameter **`flagbackup=True`** in the execution of the **`flagdata()`** task a few slides before. But this is not the only way of saving the flagging commands, you can save the state of your flags manually at any time.

Here is an example of how you could do it with task **`flagmanager()`**:

```
# In CASA
default flagmanager
vis='drwRFI.ms'
mode='save'
versionname='after_manual_1'
comment='after manual flagging'
inp

go
```

INTERACTIVE

Flagging Manager

If you run both the `flagmanager()` task to save the flags, and the `flagdata()` task with `flagbackup=True`, you should have two files now within the ms that contain the flagging done so far.

This is where the files live

`ls -l drwRFI.ms.flagversions`

Terminal output:

```
drwxr-x--- 2 akapinsk nmstaff 4096 Mar 10 15:24 flags.after_manual_1/
drwxr-x--- 2 akapinsk nmstaff 4096 Mar 10 15:24 flags.flagdata_1/
-rw-r----- 1 akapinsk nmstaff   51 Mar 10 15:24 FLAG_VERSION_LIST
```

To access information in these files, and revert flagging, we will use again **`flagmanager()`**

Flagging Manager

Oh, oops, I made a mistake! How to restore data before the `flagdata()` run? Help!!

(By the way, if you used `plotms()` to do the flagging, the answer is “You can't restore your data”)

Yes, `flagdata()` has a mode you can use, but this can unleash a dragon!

```
mode = 'unflag'
```

will unflag everything, not just your last `flagdata()` execution

Better to use → **`flagmanager()`**

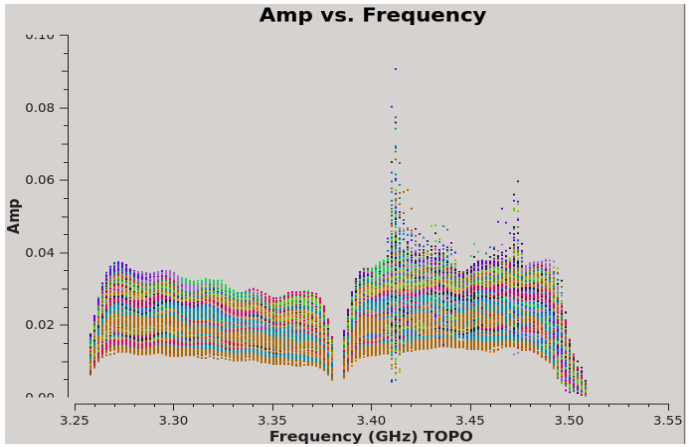
Flagging Manager

Oh, oops, I made a mistake! How to restore data before the `flagdata()` run? Help!!

```
# In CASA
default flagmanager
vis='drwRFI.ms'
mode='restore'
versionname='flagdata_1'
inp

go
```

To see what this resulted in, reload your `plotms()`.
Again, if you closed your `plotms()` just run:

```
tget plotms
inp

go
```



Flagging Manager

Let's again have a look at all the flag versions we have now for our ms file, this time properly using CASA tasks

```
# In CASA
tget flagmanager
mode='list'
inp

go
```

Logger output:

```
…ager::::+ ##########################################
…ager::::+ ##### Begin Task: flagmanager        #####
…nager::::  flagmanager(vis="drwRFI.ms",mode="list",versionname="",oldname="",comment="",
…ager::::+          merge="replace")
…ger::open  Table type is Measurement Set
…lagger::
…agger:: +  MS : /lustre/aoc/sciops/akapinsk/DRW/presentation/drwRFI.ms
…lagger::   main : working copy in main table
…lagger::   flagdata_1 : Flags autosave on 2019-10-07 11:21:55
…lagger::   after_manual_1 : after manual flagging
…nager::::  ##### End Task: flagmanager          #####
…ager::::+ ##########################################
```

Note: Restoring will not remove previous flag tables

Keeping track of the amount of flagging

It's always good to keep track of how much you have flagged. Sometime you may even need that info for the publication (depending on the reviewer…)

```
# In CASA
default flagdata
vis='drwRFI.ms'
mode='summary'
inp

go
```

Logger output:

```
…etResult   antenna ea09 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea28 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea02 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea19 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea01 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea04 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea08 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea20 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea06 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea03 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea11 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea07 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea13 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea14 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea16 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea18 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea26 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea21 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea05 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea12 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea15 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   antenna ea22 flagged: 107648 total: 1.60358e+06 (6.71%)
…etResult   field J0259+0747 flagged: 2.25504e+06 total: 2.24502e+07 (10%)
…etResult   array 0 flagged: 2.25504e+06 total: 2.24502e+07 (10%)
…etResult   Total Flagged: 2.25504e+06 Total Counts: 2.24502e+07 (10%)
…gdata:::: Flags are not written to the MS. (action='calculate')
…gdata:::: ##### End Task: flagdata             #####
…data::::+ ########################################
```

Flagdata quack mode

It's always a good idea to remove the first few seconds of each of your scans.

Why?

→ antennas have just been slewing to new source, the slewing is flagged (online flags), but often they need 'settling' time, the 'setting' time may result is some bad data

Easy to do, there is a special mode in the **`flagdata()`** task:

```
mode = 'quack'
quackinterval = 5.0        # in seconds
quackmode = 'beg'          # this one means beginning of each scan,
                           #     but there are also other modes: endb, end, etc.
```

Flagdata quack mode

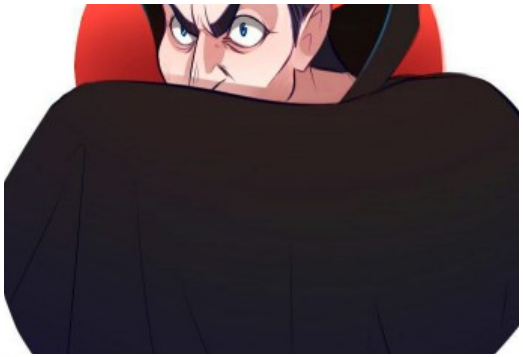
It's always a good idea to remove the first few seconds of each of your scans.

Why?

→ antennas have just been slewing to new source, the slewing is flagged (online flags), but often they need 'settling' time, the 'setting' time may result is some bad data

Easy to do, there is a special mode in the `flagdata()` task:

```
mode = 'quack'
quackinterval = 5.0        # in seconds
quackmode = 'beg'          # this one means beginning of each scan,
                           #   but there are also other modes: endb, end, etc.
```



.... and shadow mode

Very important for observations at low elevations in compact configurations

```
mode = 'shadow'
```

Auto-flagging: TFCrop

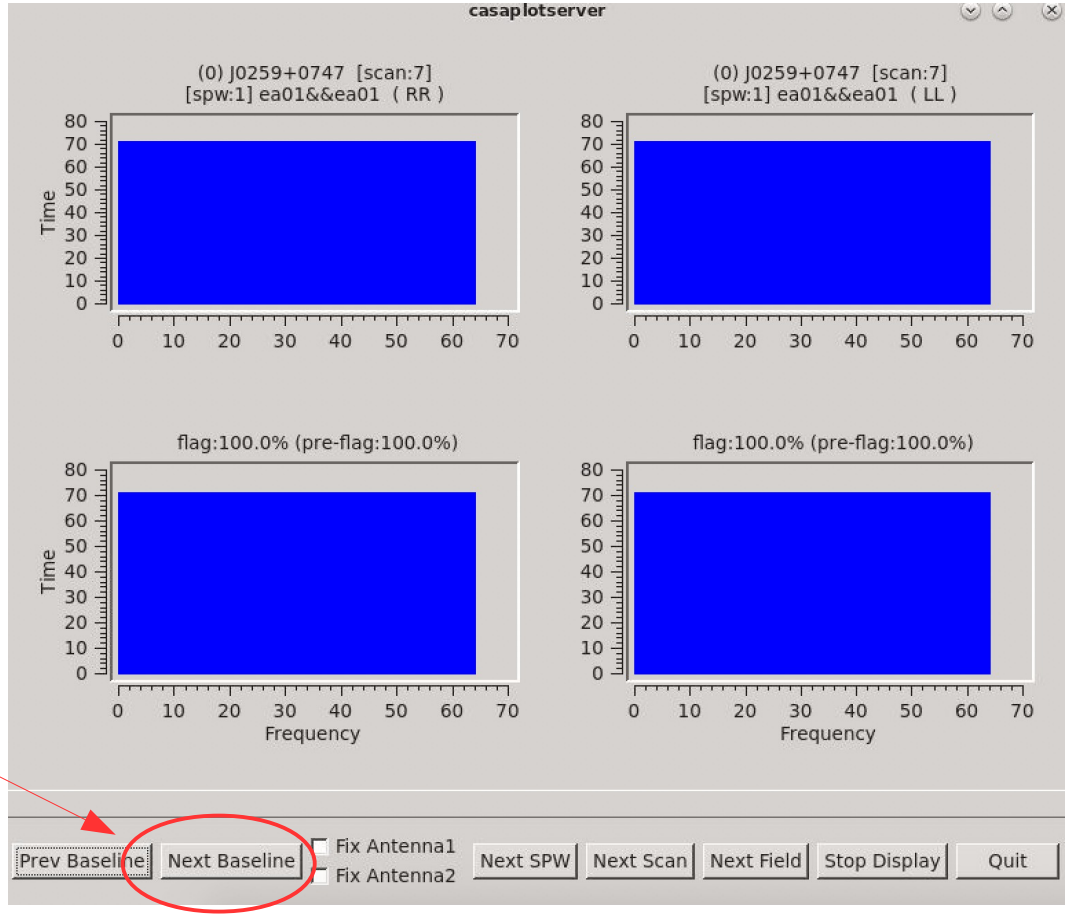
The auto-flagging algorithm `tfcrop` detects outliers in your data on the 2D time-frequency plane. Its statistics is based on each baseline independently.

```
# In CASA
tget flagdata
mode='tfcrop'
spw='1'
timecutoff=3.0
action='calculate'
display='both'
flagbackup=True       # required if to restore previous flagging versions
inp          # stop here for a moment to see what other options you can set

go
```

INTERACTIVE

Auto-flagging: TFCrop

It's BLUE because all is flagged

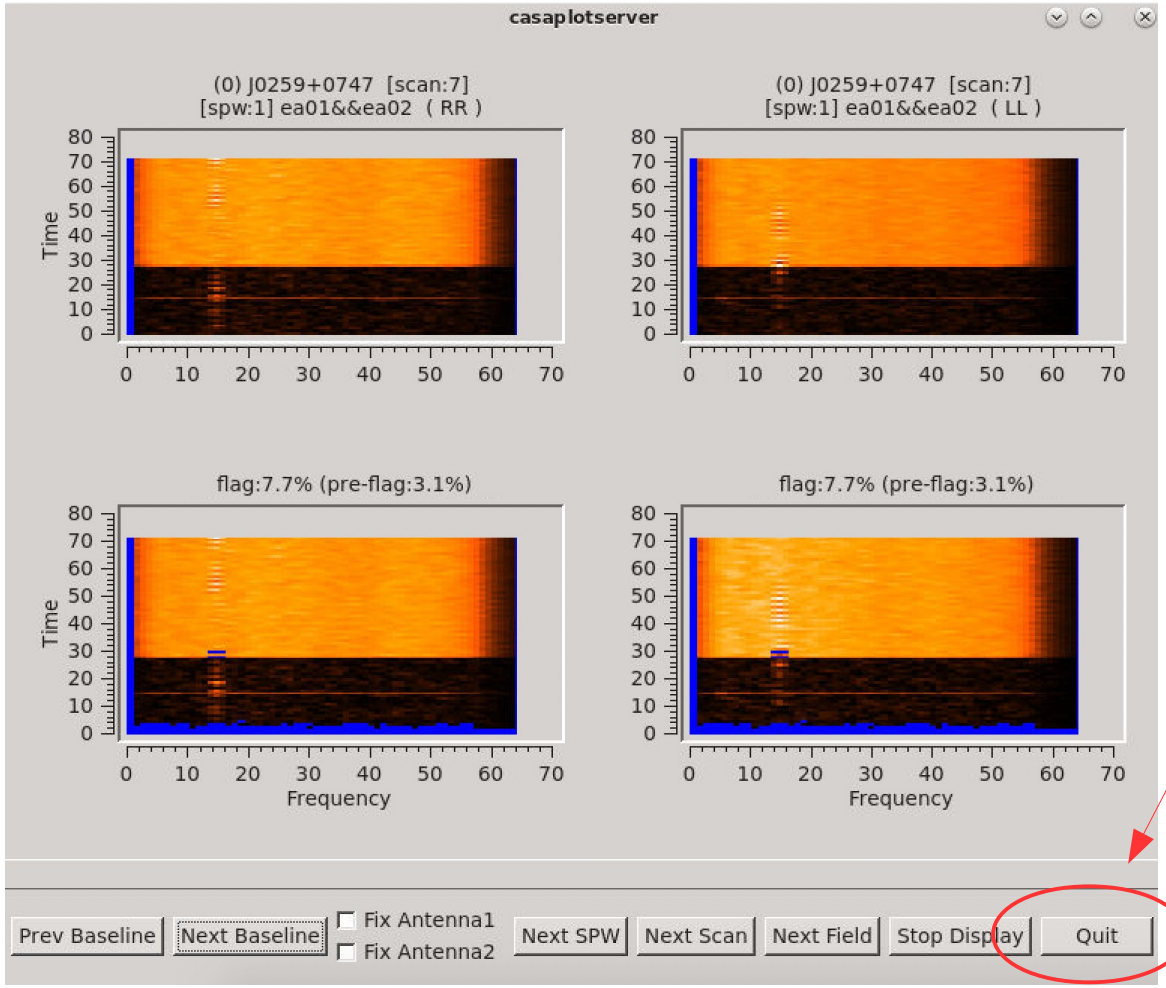


Navigate to the next baseline

Auto-flagging: TFCrop

Current flagging status

After TFCrop flagging



Careful! *Quit* will kill the task!

Stop Display will just turn off display and let `flagdata()` to finish.

Since here we are just calculating flags so far, you can click either *Stop Display* or *Quit* to exit.

Auto-flagging: TFCrop *cont.*

For best results you need to tune the `tfcrop` parameters, e.g.

```
timecutoff, freqcutoff      # threshold for finding outliers, in units of fit st.dev.
flagdimension = 'freqtime'  # direction(s) in which to calculate statistics
channelavg, timeavg         # pre-average the data
timefit = 'line'    # fitting function along time axis, line is default (ok: poly/line)
freqfit = 'poly'    # fitting function along freq axis, poly is default (ok: poly/line)
maxnpieces = n      # n order of polynomial in fitting functions above
```

Sometime you may also need to vary the parameters for e.g. spws or bands within the same data set.

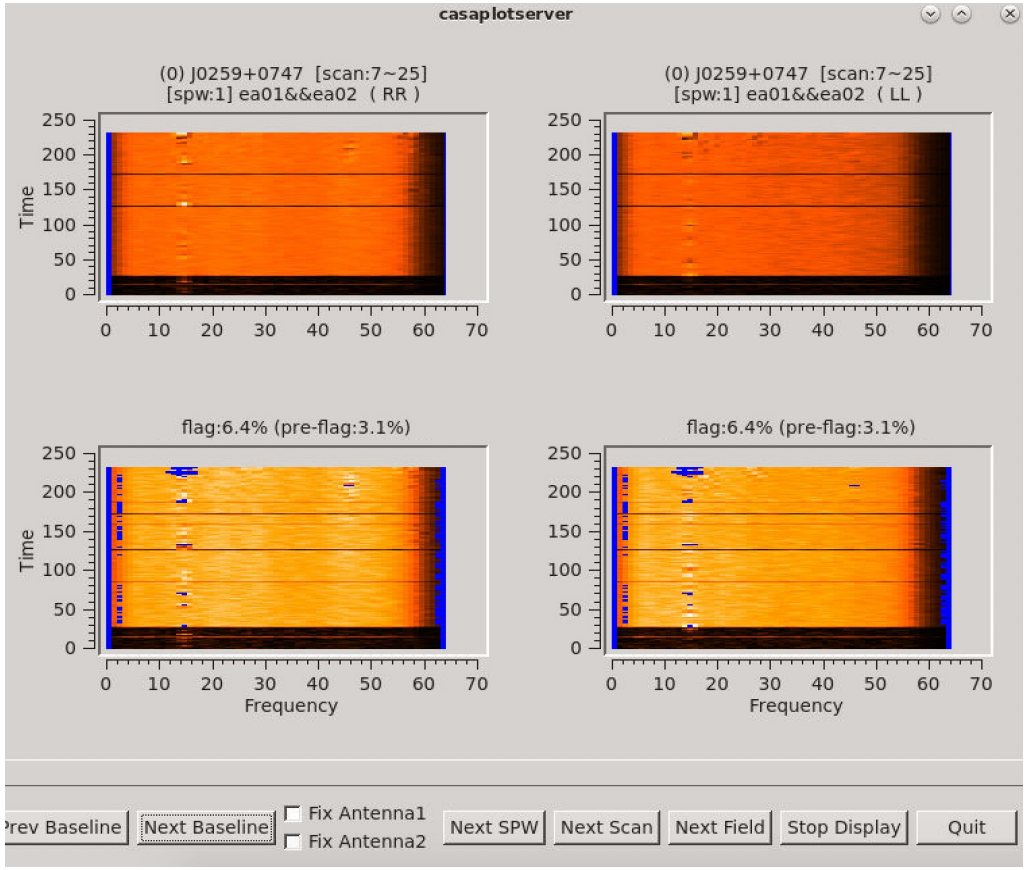
Each data set is different and may need different parameter set up for best results
→ make sure you inspect your data well
→ know what you are dealing with, and
→ choose the parameters accordingly.

And yes, you can run `tfcrop` on your data multiple times !

Auto-flagging: TFCrop

Let's try some tuning to see if we can remove the most offending RFI

```
# In CASA
tget flagdata
maxnpieces = 4
usewindowstats = 'std'
halfwin = 2
combinescans=True
inp

go
```



INTERACTIVE

Auto-flagging: TFCrop

If you are happy with the amount of flagging, then let us now apply it

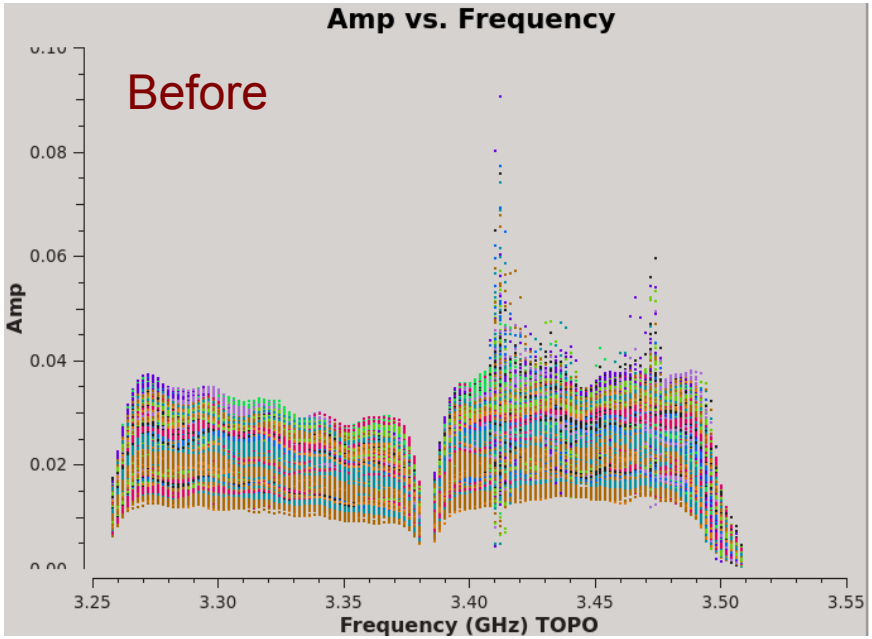
```
# In CASA
tget flagdata
action='apply'
display=''
inp

go
```

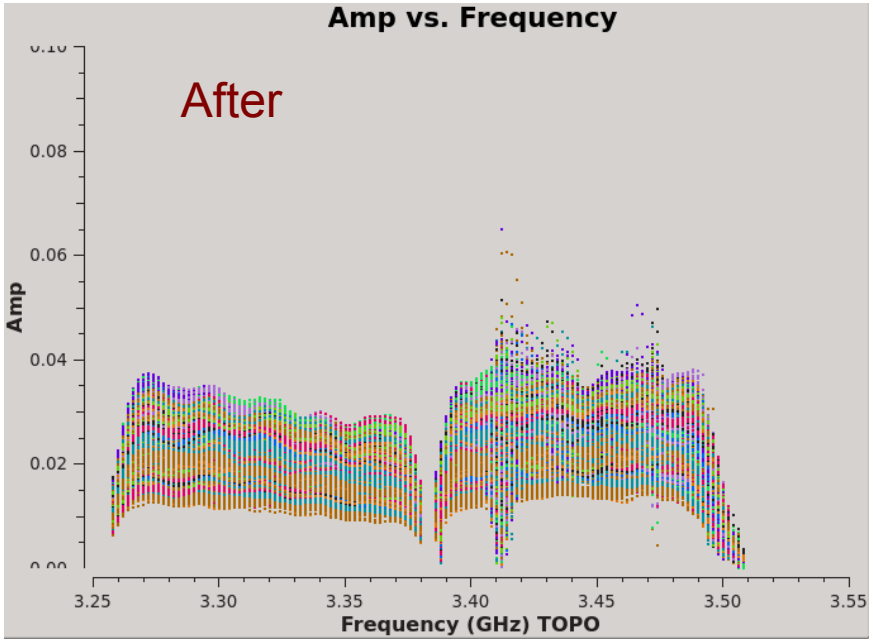
INTERACTIVE

Auto-flagging: Inspect results

```
# In CASA
tget plotms
inp

go
```

Or just check *Reload* box and click *Plot* if you still have `plotms()` open





Extending your flags

Available as a standalone mode in `flagdata() → mode='extend'`
or as an `extendflags` parameter within `tfcrop` and `rflag` modes

→ will extend or grow flags accumulated in the MS file along time, frequency, polarisation, baseline etc

Flag extension:
→ e.g. if you applied flag only to RR product, you can extend that afterwards also to LL.

Flag growing, example parameters:

```
growtime=80.0     # for each channel flag entire timerange if >80% data flagged
growfreq=92.0     # for each time flag selected chans if >92% data flagged
growaround        # flag a data point if >4 neighbouring points are flagged
flagneartime      # flag a data point before and after a flagged one
```

Note: It is recommended flag extension is used when executing auto-flagging modes.

Extending your flags

There is still yucky RFI there so let's try more

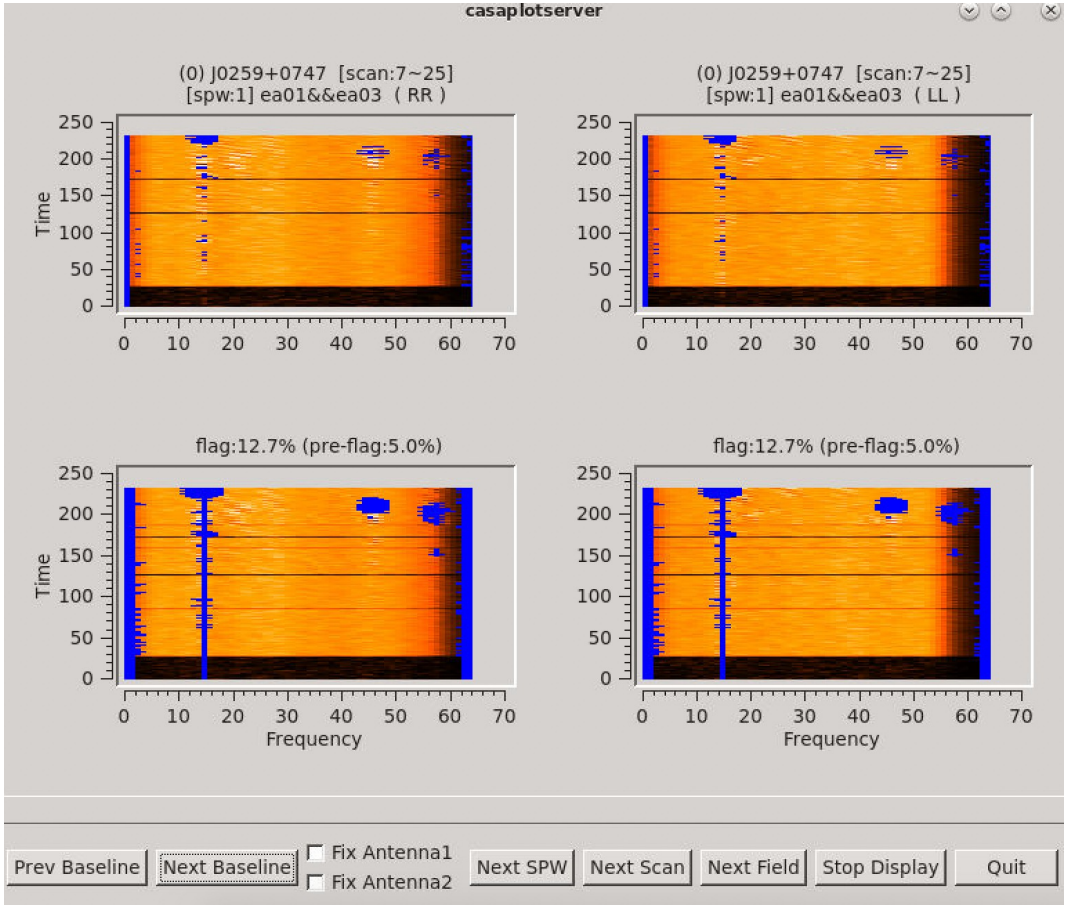
```
# In CASA
default flagdata
Vis='drwRFI.ms'
mode = 'extend'
combinescans = True
growtime = 30.0
growfreq = 30.0
growaround = True
flagneartime = True
flagnearfreq = True
action = 'calculate'
display = 'both'
inp

go
```

INTERACTIVE

Extending your flags

Woohoo, finally getting the remaining RFI removed!

3rd baseline shown in 2nd spw

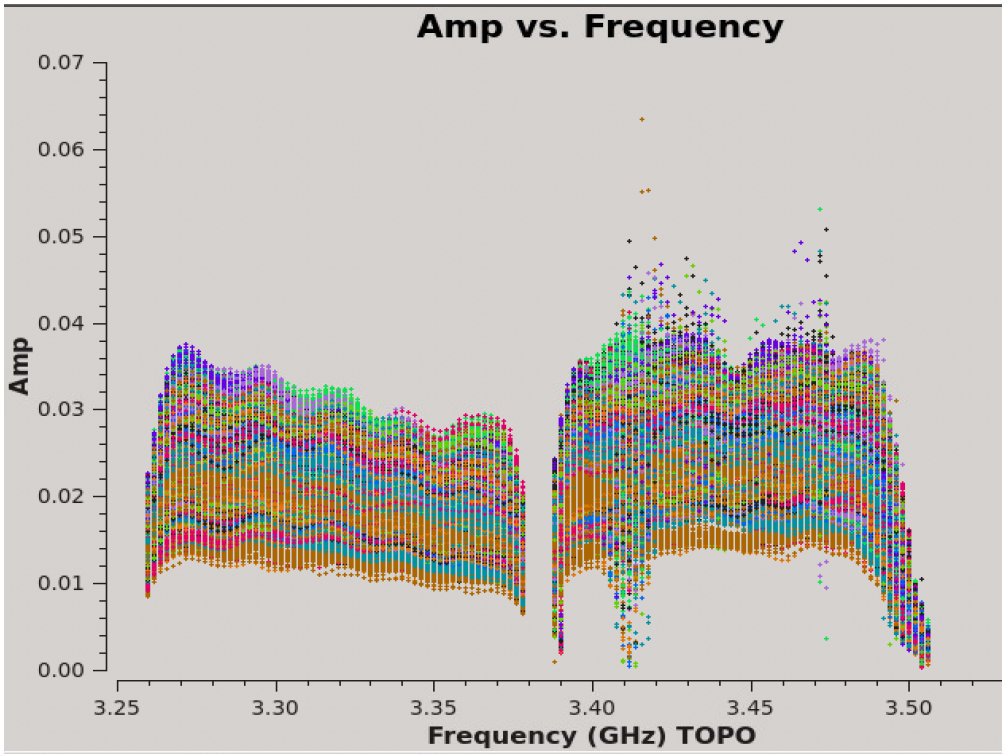


Extending your flags

So, if you are now happy with the extra flagging, click *Quit* on the display, and let's apply these flags

```
# In CASA
tget flagdata
action = 'apply'
display = ''
inp

go


# In CASA
tget plotms
inp

go
```

Current result:



INTERACTIVE

Auto-flagging: Rflag

The `rflag` algorithm detects outliers in your data based on local rms statistics.

Requires calibrated data!

The calibration pipeline uses Rflag

Step 1. `rflag` iterates through time chunks calculating local rms of *imag* and *real* visibilities within a sliding time window, and deriving a median rms across given time window

Step 2. `rflag` iterates through frequency chunks (channels) for each time chunk calculating rms of avg of *imag* and *real* visibilities

---

Again, for best results you need to tune the `rflag` parameters, e.g.

```
winsize                          # number of timesteps in the sliding time window
timedevscale, freqdevscale       # st.dev. threshold for outlier flagging
channelavg, timeavg              # pre-average the data
```

Again, can be run multiple times on the same data set.

TFCrop vs Rflag: which to use when?

Due to differences in the algorithms, worth executing both on the same data set.

	TFCrop	Rflag
How does it work?	→ search for RFI spikes above smooth base, per baseline	→ use local vs global stats to find outliers
Strong, spiky RFI	Great!	Good, but continuous RFI in time/freq needs tuning
Noisy RFI	Good only for bright RFI, won't work well for low noisy RFI spikes	Great!
Broadband RFI	Not robust, but possible with some tuning	Good for noisy RFI. Continuous RFI needs tuning.
Extended emission	Great! [each baseline treated separately]	Not great, biased by high flux density on short baselines
Raw, uncalibrated data	**Yes**	**No**
Calibrated data	**Yes**	**Yes**

Auto-flagging: Spectral lines

Be thoughtful of your precious spectral lines! If you just run `tfcrop` or `rflag` without adjusting for the location of your spectral line(s), it/they will likely be removed during auto-flagging.

Solution:

1. You know exactly where your line is, note its location in spw and channel(s)
2. In the **`flagdata()`** task exclude that location with the `spw` parameter

Example:

– you have 2 spectral windows (0,1), each with 64 channels
– your line is in spw=1, channels=21~22
– exclude that location with the following format of the `spw` parameter

```
spw = '0, 1:0~20; 23~63'
```

Pipeline: not optimised for dealing with spectral lines

If you do not know where your lines are (e.g. you are searching for some) **DO NOT** run pipeline or auto-flagging on your data!

A note on statwt()

DATA	MODEL	CORRECTED

WEIGHT	SIGMA

Typically the WEIGHT and SIGMA columns are set to some arbitrary values (e.g. 1), or are theoretically estimated from poorly known antenna and receiver properties.

`statwt()` will empirically measure the visibility scatter (e.g. as a function of time, antenna, and/or baseline) and use it to set WEIGHT and SIGMA

→ it may be beneficial sometimes to weight down any remaining RFI in your data with `statwt()` prior to imaging

Requires calibrated data!

Final points to remember

Online CASA flagging guide
http://casaguides.nrao.edy/index.php?title=VLA_CASA_Flagging-CASA5.7.0

→ All data have some level of RFI, and with progressing technological evolution it will only get worse for us

→ Inspect your data thoroughly, and remove the most obstructive RFI as this is what will affect the quality and noise of your images the most.

→ Use combination of auto-flagging algorithms and manual flagging if required (but careful with spectral lines!!). Execute auto-flagging multiple times → statistics these algorithms rely on will change each time!

→ You can flag all types of data:
* the visibilities,
* the weights, and
* solutions in calibration tables

www.nrao.edu
science.nrao.edu
public.nrao.edu

The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.