Flagging Capabilities in CASA 3.4

- Modes : manual,unflag,quack,elevation,shadow,extend,clip,tfcrop,rflag,summary,display, list

- Design : C++ layout, python-tool, task tflagdata, task tflagcmd

- Performance : list mode, async-IO, data pre-selection, parallelization

- Other : flag-command subtable usage.

Team : Justo Gonzalez (ESO), Sandra Castro (ESO), Urvashi Rau (NRAO), [Jeff Kern (NRAO)]

Flagging Modes : manual, unflag, quack, elevation

mode='manual' or mode='unflag' : Standard MS-selection syntax :

field, spw, antenna, timerange, correlation, scan, intent, array, uvrange, observation

- Uses MS-selection to extract data indices and sets flags to True or False

---

mode = 'quack'   # Flag data at the beginning or end of a scan (operate on selected data).

```
quackinterval  =  Flag N seconds
quackmode      =  scan-beginning, scan-end, all but scan-beginning, all but scan-end
                     'beg'          'endb'         'tail'                  'end'
quackincrement = Calculate quack-interval from beginning of scan, or first unflagged point
```

---

mode = 'elevation'   # Flag data between specified elevation limits (operate on selected data)

lowerlimit , upperlimit = elevation limits in degrees.

- For each time, flag all baselines for antennas pointing at elevations outside the limits

Flagging Modes : clip

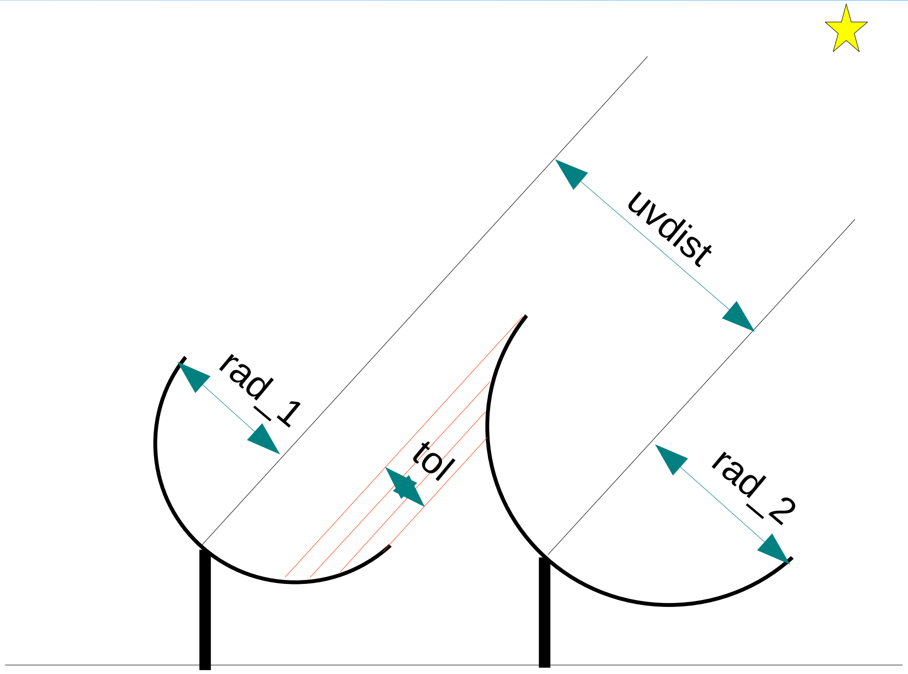
Threshold-based flagging on data-expressions (operates on selected data + expressions)

```
datacolumn = 'data', 'corrected', 'model', 'residual'

correlation = 'ABS_ALL'       # Correlation expression. Augmented MS-selection syntax.
                                 Expressions : 'ALL', 'RR','LL',  'I', 'Q','U','V', 'WVR'
                                 Functions : 'ABS', 'ARG', 'RE', 'IM', 'NORM'
                                ('WVR' refers to the water vapour radiometer of ALMA data.)

                                Ex : 'ABS_RR,LL,V'

clipzeros = True               # Flag exact zeros

clipminmax = [ 10.0, 50.0 ]    # Clip range.

clipoutside = True             # Clip outside this range

channelavg = False             # Average over all channels in each spw, or not
```

- 'clip' will always flag 'NaN' and '+/- Inf' if they exist, even if clipminmax=[ ]

- Protect your spectral-lines by de-selecting them in the spw-selection parameter for 'clip'.

Flagging Modes : shadow



Calculate/read U,V,W (in meters)

W > 0 : ant_1 is behind ant_2
W < 0 : ant_2 is behind ant_1

Flag 'behind' antenna if

uvdist < rad_1 + rad_2 - tolerance

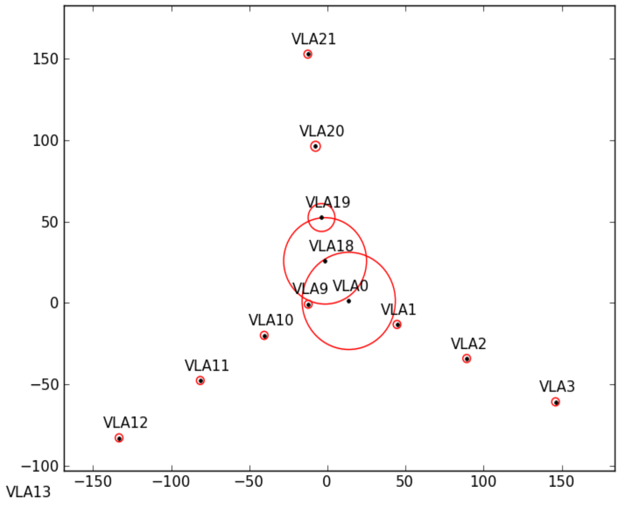
```
tolerance = 10.0        # Allow 10m of shadowing between antennas

antennafile = ' '       #  Optional text file with positions/diameters of extra antennas
                        #   Using these positions, uvw are calculated for all new
                        #   baselines that are not present in the measurements, and
                        #   used to calculate shadowing.
```

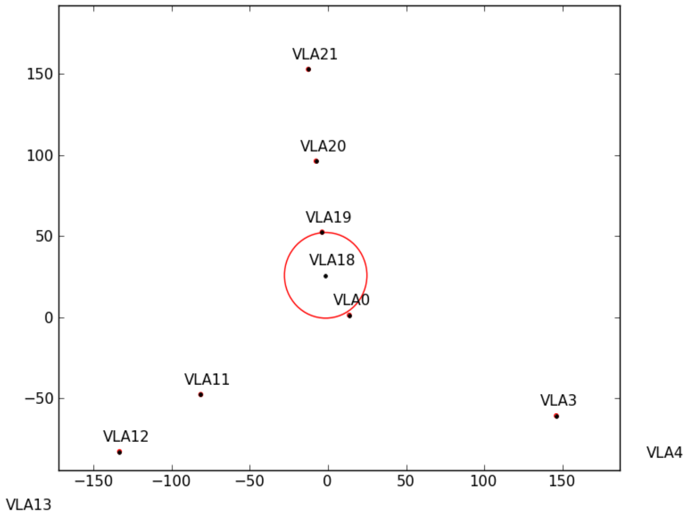
Flagging Modes : shadow

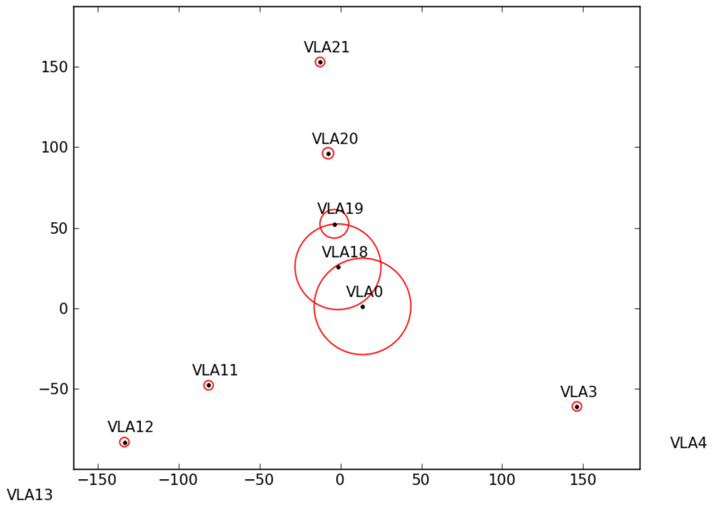
All antennas are in the MS.

Split out a subset of antennas into a new MS

Subset MS + external list of antennas.







Example text file :

```
name=VLA1
diameter=25.0
Position=[-1601144.96146, -5041998.01971, 3554864.76811]

name=VLA2
diameter=25.0
position=[-1601105.76646, -5042022.39178, 3554847.24515]

.... etc
```

Script to produce ant-list file :

```
import flaghelper;

antlist =
   flaghelper.extractAntennaInfo (
            msname='shadowtest.ms',
            antnamelist=['VLA1','VLA2','VLA9','VLA10'] );

flaghelper.writeAntennaList('antlist.txt',antlist);
```

Flagging Modes : tfcrop

Detect outliers on the 2D time-freq plane.

For each baseline,

(1) Average visibility amplitudes along time dimension to form an average spectrum

(2) Calculate a robust piece-wise polynomial fit for the band-shape at the base of RFI spikes

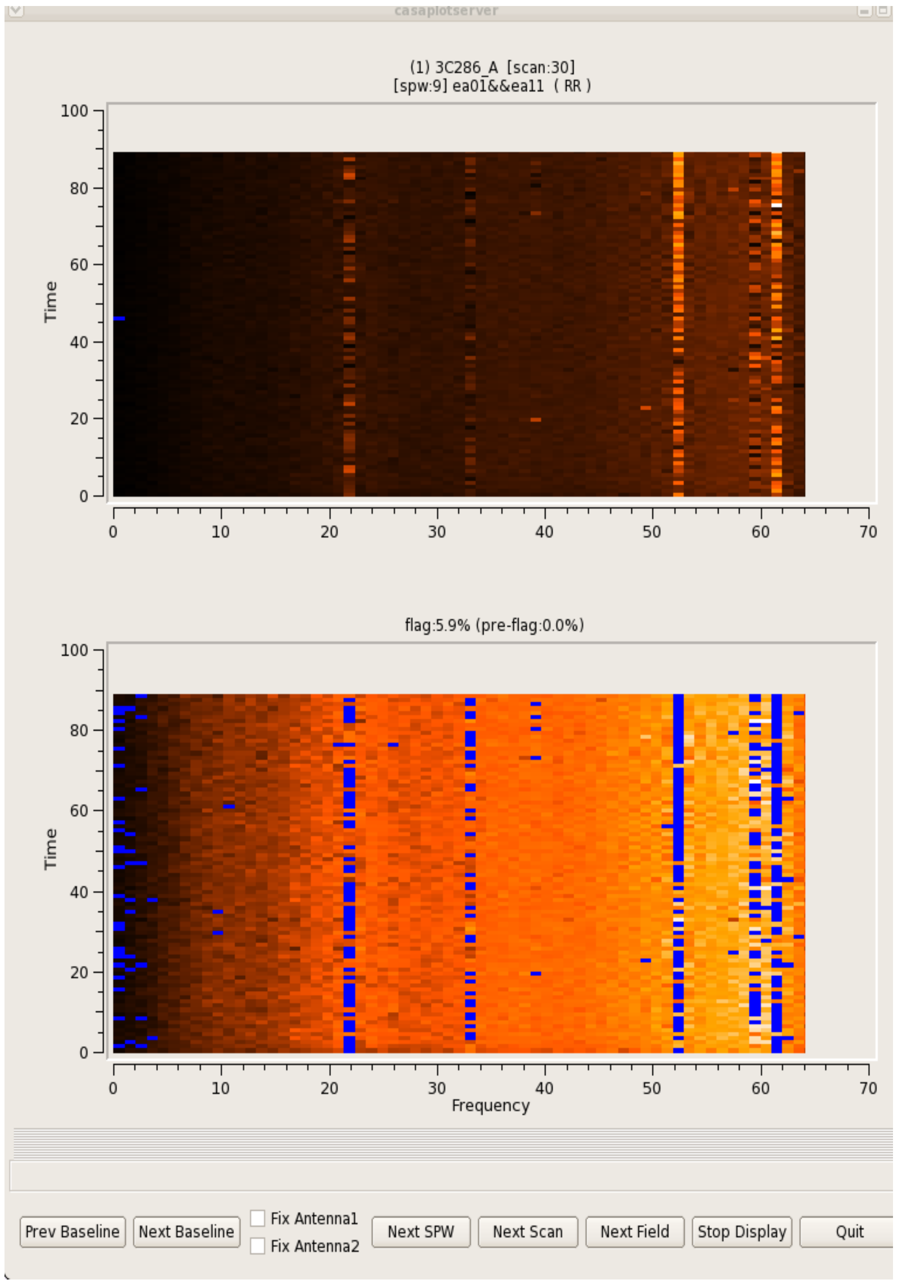
– Calculate 'sigma' of (data - fit).

(3) Flag points deviating from the fit by more than N-sigma

(4) Repeat (1-3) along the other dimension.

- Calculates and applies flags in one pass through the MS
- Can operate on un-calibrated data
- Flag extension can be done via mode='extend'

(Old documentation :
http://www.aoc.nrao.edu/~rurvashi/TFCrop)



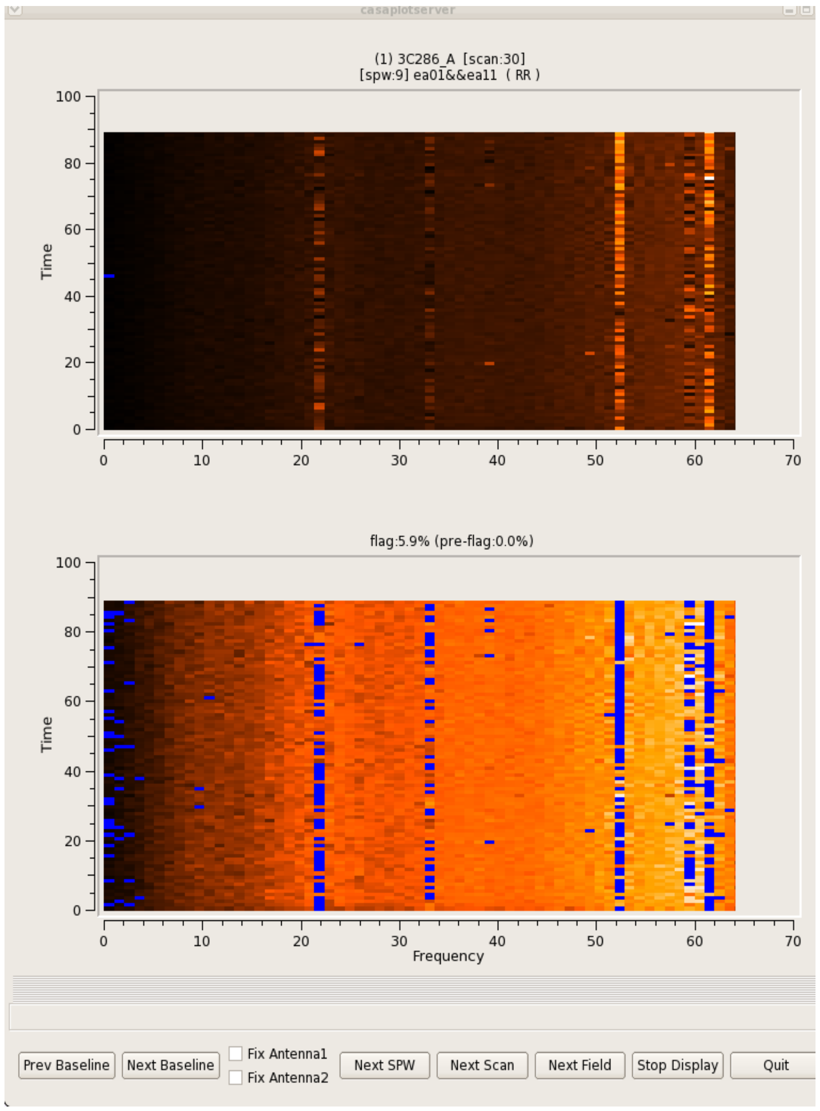
Flagging Modes : tfcrop

```
ntime  = 'scan' or '2.0min' or 120.0   #   Chunks of time across which to extend flags.

combinescans = True/False              #    if ntime > scan, join multiple scans

datacolumn = 'data', 'corrected', 'model', 'residual'

correlation = 'ABS_ALL'                # Correlation expression. Augmented MS-selection syntax.
                                          Expressions : 'ALL', 'RR','LL',  'I', 'Q','U','V', 'WVR'
                                          Functions : 'ABS', 'ARG', 'RE', 'IM', 'NORM'
                                          Ex : 'ABS_RR,LL,V'

timecutoff, freqcutoff  = 3.0          # Scale-factor for deviation from fit, to compute thresholds

timefit, freqfit  = 'line' or 'poly'   # Fit a straight line, or piecewise polynomial  (coming : only mean)

maxnpieces  = 7                        # maximum number of pieces in the polynomial fit (only for 'poly')

flagdimension  = 'freqtime'            # Directions in which to calculate stats 'timefreq','time','freq'


usewindowstats , halfwin               # Options to use sliding-window statistics as well, and windowsize.
```

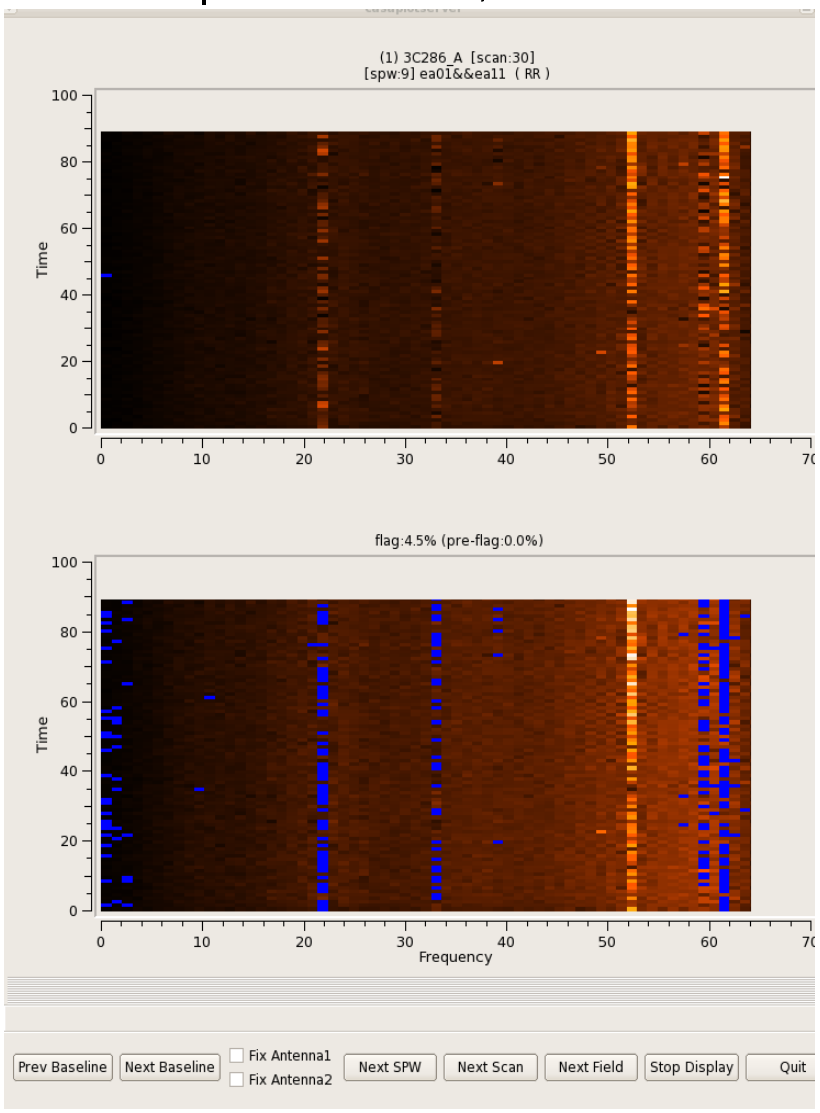
--- Protect your spectral-lines by de-selecting them in the spw-selection parameter.

Protecting spectral-lines, during automatic flagging

spw = ' 9 '



spw = ' 9 : 0~45; 53~63 '



This is relevant to 'clip' , 'extend' , 'tfcrop' , 'rflag'

Flagging Modes : rflag

(A close copy of RFLAG from AIPS (E.Greisen, 2011))

Detect outliers based on a sliding-window rms

(1) Time-Analysis (for each channel)

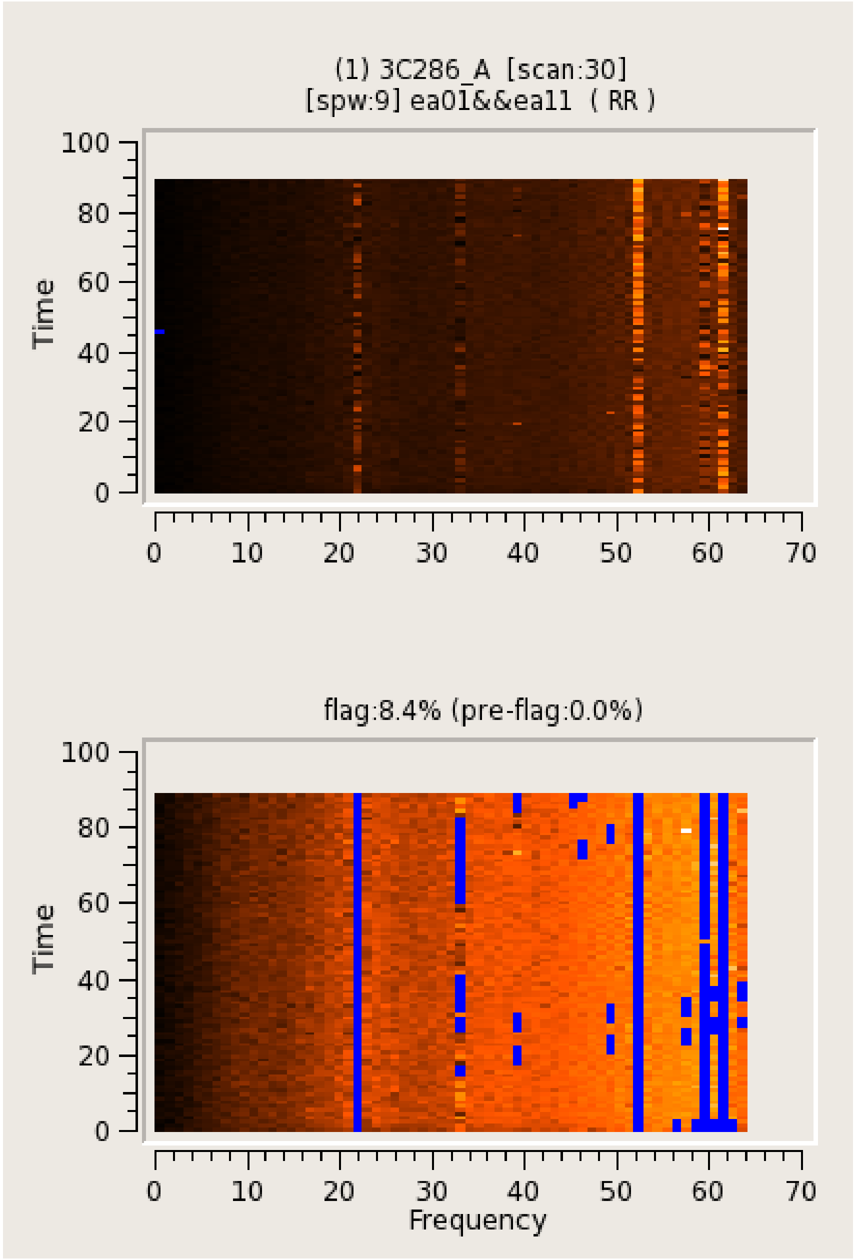
(1a) Calculate local RMS of real and imag parts of visibilities within a sliding time window.
(1b) Calculate the median RMS across time windows, deviations of lo cal RMS from this median, and the median deviation
(1c) Flag if local RMS > N x (medianRMS + medianDev)

(2) Spectral-Analysis (for each time)

(2a) Calculate Avg of real and imag pars of visibilities and their RMS across channels.
(2b) Calculate the deviation of each channel from this Avg, and the median-deviation.
(2d) Flag if deviation > N x medianDev

- Can be run with 1 or 2 passes through the data (calc + flag)
- Requires calibrated data.
- Flag extension can be done via mode='extend'

- CASA implementation currently....
  --- ignores weights (...coming soon.)
  --- does not have optype='medr', 'medi'

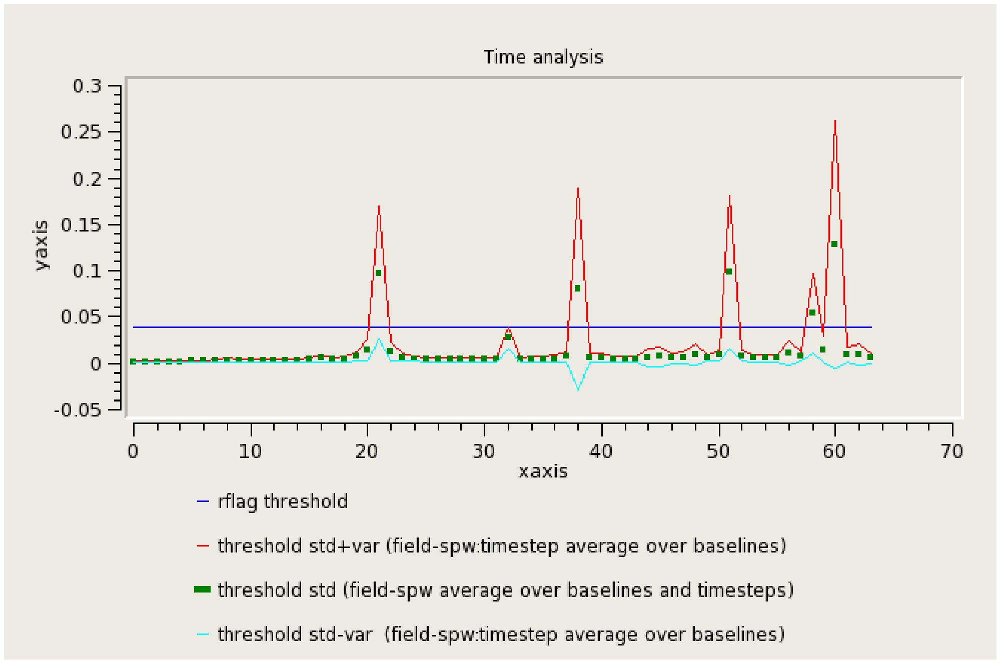


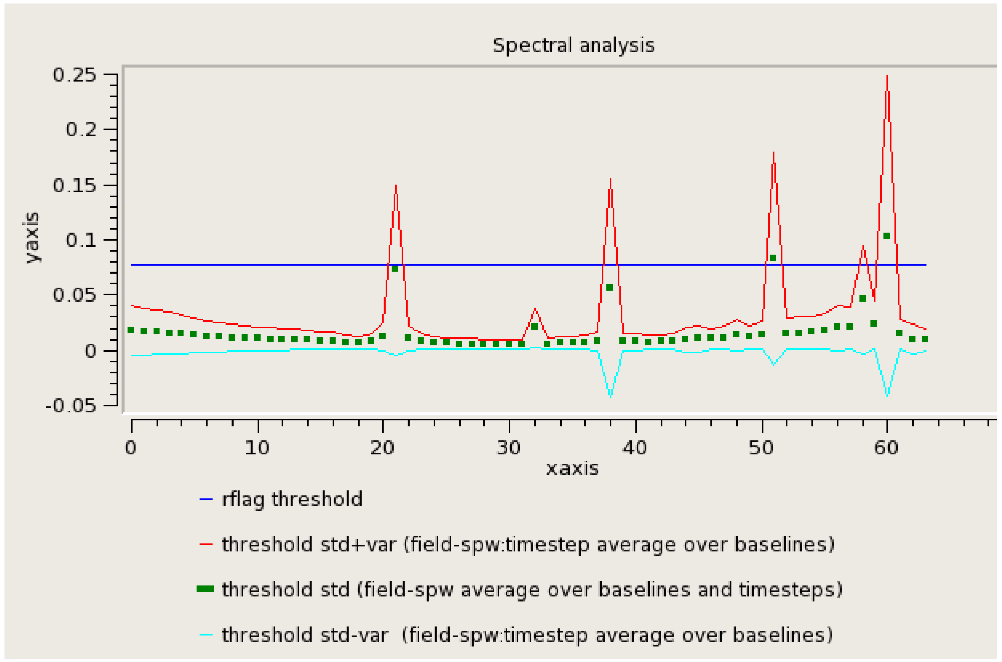
Flagging Modes : rflag

```
ntime  = 'scan' or '2.0min' or 120.0   #   Chunks of time across which to extend flags.

combinescans = True/False             #    if ntime > scan, join multiple scans

datacolumn = 'data', 'corrected', 'model', 'residual'

correlation = 'ABS_ALL'               # Correlation expression. Augmented MS-selection syntax.
                                         Expressions : 'ALL', 'RR','LL',  'I', 'Q','U','V', 'WVR'
                                         Functions : 'ABS', 'ARG', 'RE', 'IM', 'NORM'
                                         Ex : 'ABS_RR,LL,V'

winsize  = 3                          #   size of sliding time window [ fparm(1) ]

timedev = [ [ 0, 2, 0.0534 ] ]        #   time-series noise estimate [ noise ]
freqdev  = [ ]                        #   spectral noise estimate [ scutoff ]

timedevscale = 5                      #   threshold scaling for timedev [ fparm(9) ]
freqdevscale = 5                      # threshold scaling for freqdev [ fparm(10) ]

spectralmin = 1e+6                    # flag whole spectrum if freqdev > spectralmin [ fparm(6) ]
spectralmax = 0.0                     # flag whole spectrum if freqdev < spectralmax [ fparm(5)] ]

outfile  = 'thresholds.txt'           #  text file to hold output thresholds from rflag.
```

Current status : Available from the tool with different param names. Work in progress on task.

Flagging Modes : rflag





RFLAG in AIPS operates in two passes through the data : Calculate Thresholds + Calculate Flags

RFLAG in CASA :

(1) Calculate thresholds only (using statistics from the full time-range of the MS)
(2) User user-specified thresholds to calculate flags

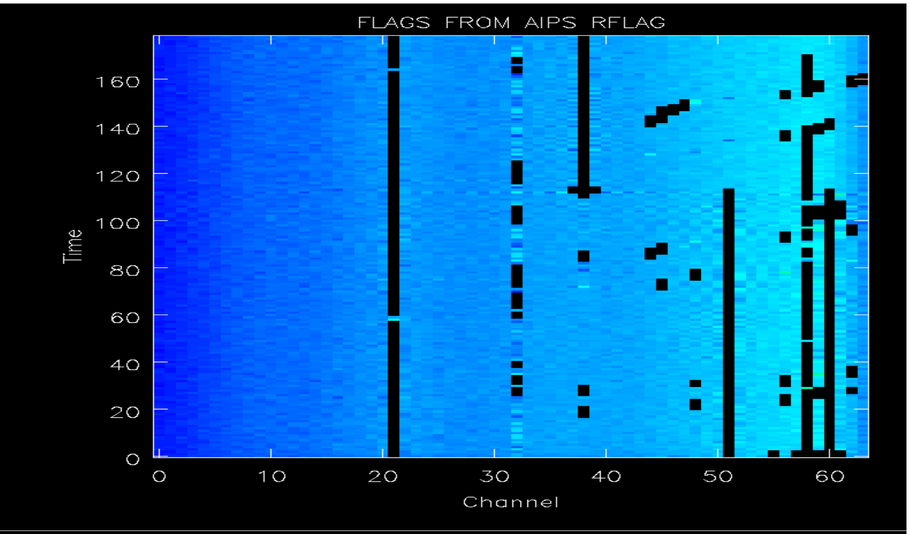
(3) Calculate thresholds per 'ntime' chunk and use them to calculate flags for the same chunk

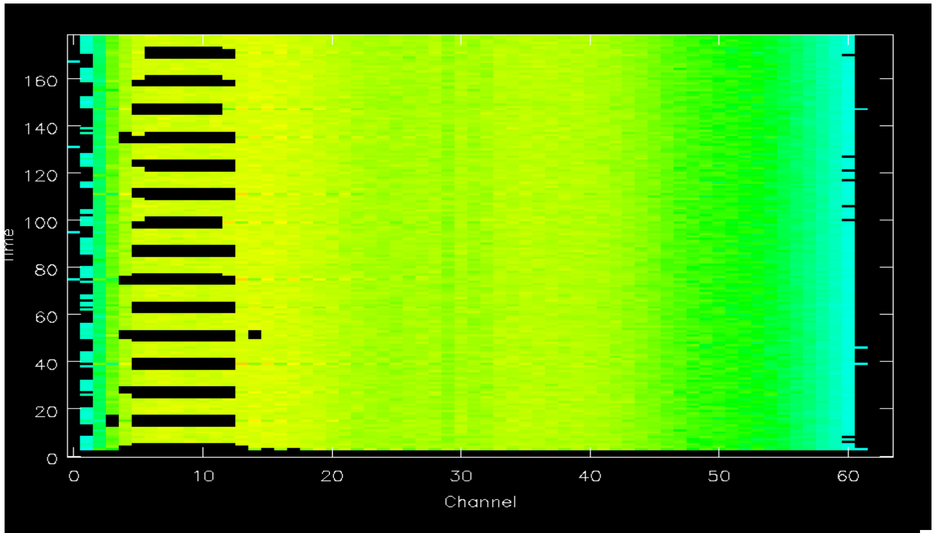
– this allows 'rflag' to operate in one pass through the MS.
– can choose a large-enough 'ntime' with combinescans=True to make the statistics work.
– this will give slightly different flags from (1) + (2), but could still be useful.

Flagging Modes : rflag (AIPS and CASA)

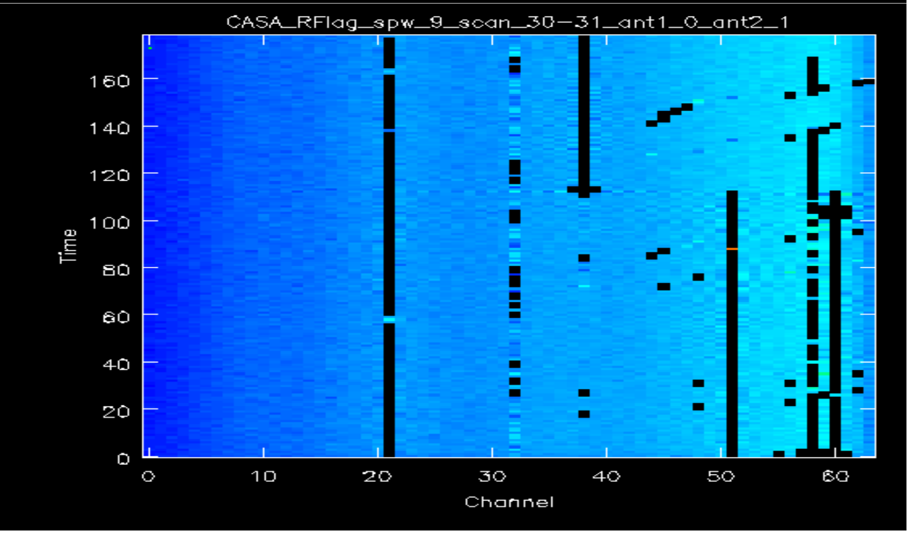
Time-analysis only (near 1 GHz)

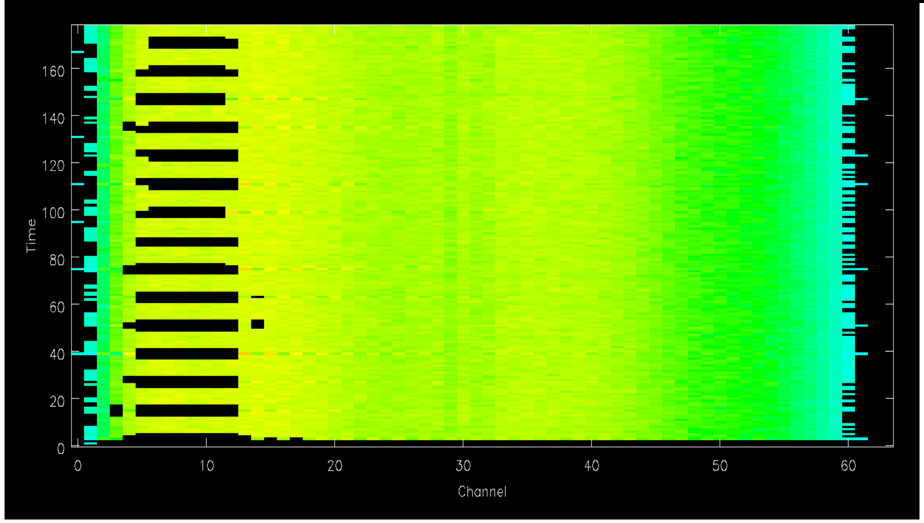
Time and spectral analysis (near 1.3 GHz)





AIPS

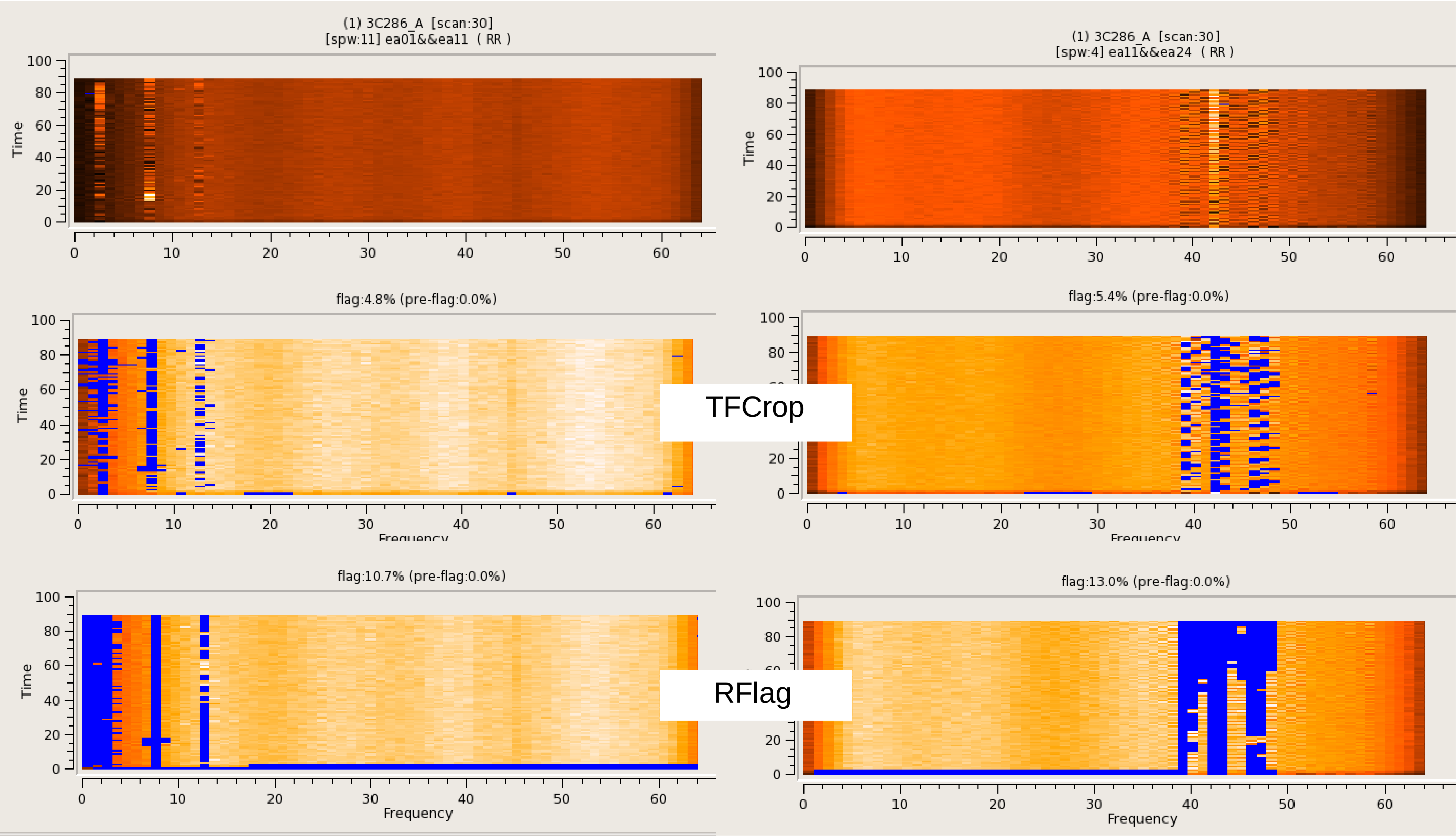




CASA

For this L-Band data-set, between AIPS and CASA RFLAG (auto thresholds, scale=3),
– Time-analysis thresholds agree within 1% , Spectral-analysis agree within 3%

TFCrop vs RFlag – narrow-band and broad-band RFI



Flagging Modes : extend

Extend flags along time, frequency, polarizations (and baselines). Operates on selected data

```
ntime  = 'scan' or '2.0min' or 120.0   #   Chunks of time across which to extend flags.

combinescans = True/False              #    if ntime > scan, join multiple scans

extendpols = True/False                #    Extend across all selected correlations

growtime = 80.0                        #   For each channel, flag entire timerange (ntime)
                                             if more than 80% of the data is flagged.

growfreq = 80.0                        #   For each time, flag selected channels (in current
                                            SPW)  if more than 80% of the data is flagged.

growaround = True/False                #   On the 2D time/freq plane, if more than 4
                                            surrounding points are flagged, flag it.

flagneartime = True/False              #   Flag points before after every flagged point in time
flagnearfreq  = True/False             #   Flag channels before/after every flagged one.

extendants = 80%  (coming soon)        #   For each channel/time, flag all baselines if more
                                             than 80% are flagged.
```

- Protect your spectral-lines by de-selecting them in the spw-selection parameters for 'extend'.

Flagging Modes : extend

Example :

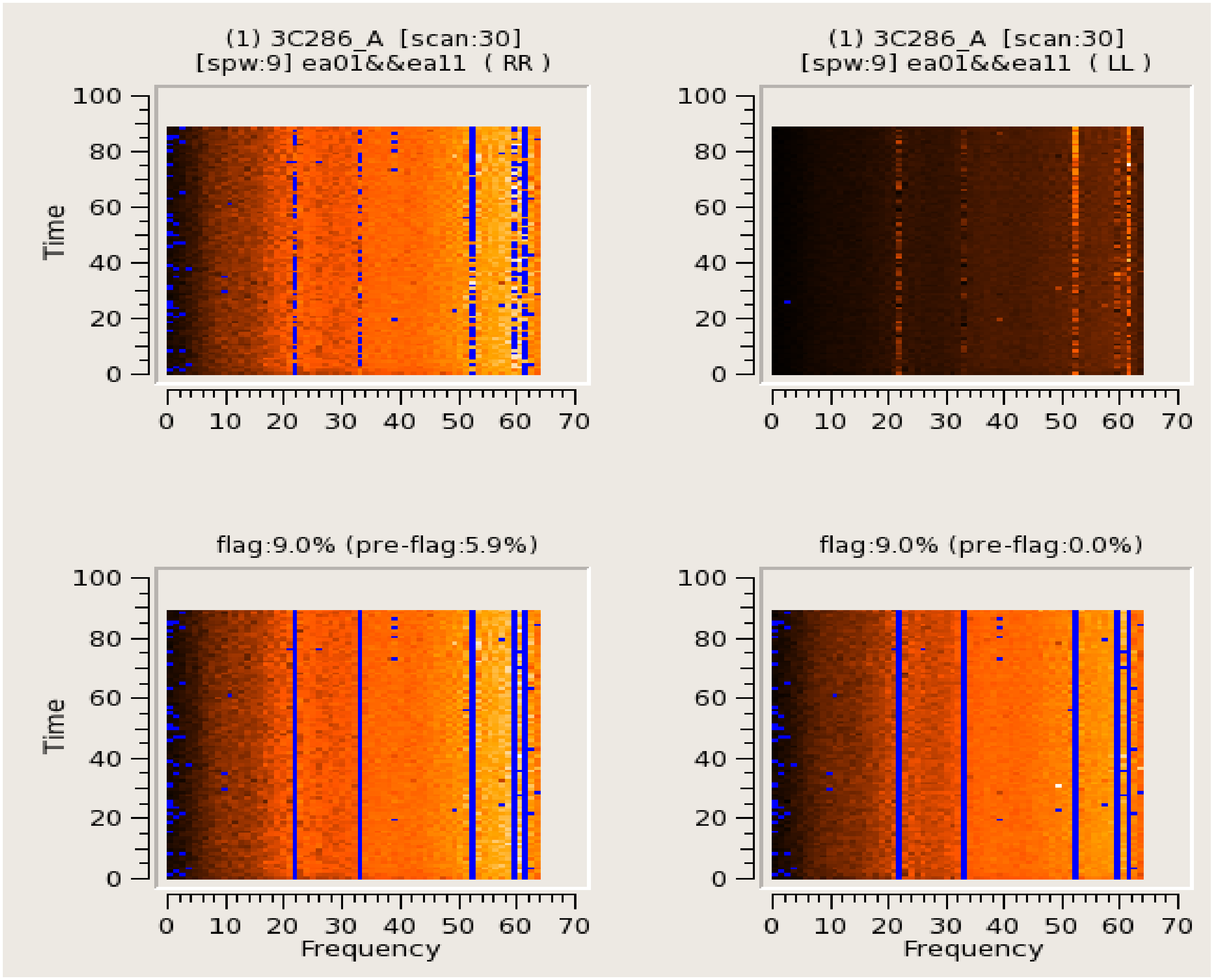
Flag only RR,

+

Growtime = 30.0

+

Extendpols = True



Flagging Modes : list

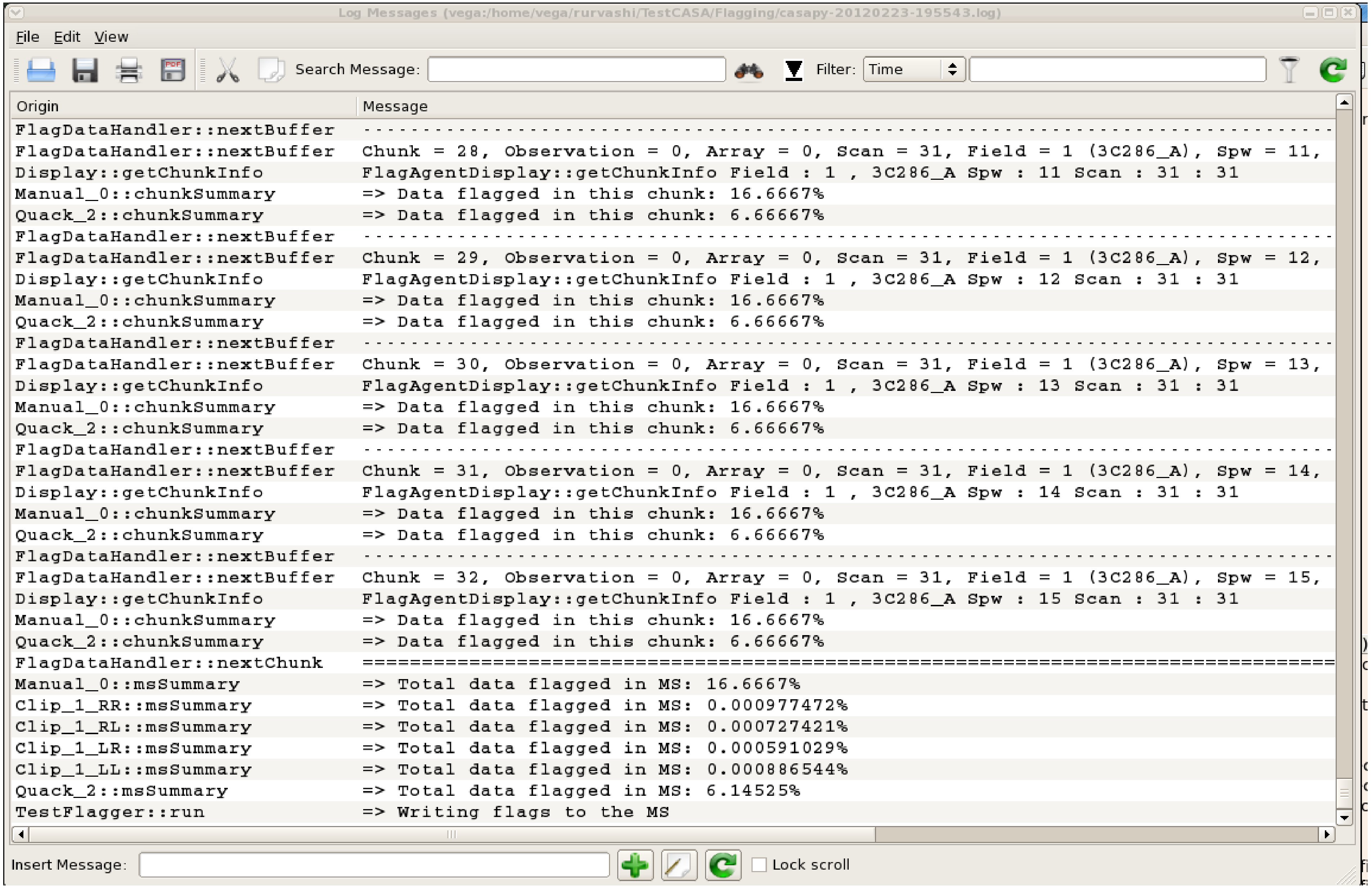
Supply a list of flag commands, to be run in one single pass through the data.

- Apply large numbers of online flags (all mode='manual') together.
- Combine modes that read the same data, to save on I/O

Example list of commands :

```
mode='manual'   antenna='1&&2'    spw='3'    timerange='12:23:02.3~12.24:45.2'
mode='manual'   antenna='ea05'    spw='5:50~60'   scan='2~10'
mode='manual'   antenna='ea01,ea24'    spw='9'    correlation='RR,RL'
mode='quack'    quackinterval=5.0
mode='shadow'  tolerance=5.0


mode='clip'    correlation='ABS_ALL'  clipminmax=[0.01,2.4]  clipoutside=True
mode='rflag'   timedev=[[1, 9, 0.02343]]    timedevscale=3   correlation='ABS_I,ABS_V'
mode='extend'   extendpols=True   growtime=30.0
```

Runtime flag summary...


Log Messages (vega:/home/vega/rurvashi/TestCASA/Flagging/casapy-20120223-195543.log)
File Edit View
Search Message:
Filter: Time
Insert Message:
Lock scroll

```
Origin                         Message
FlagDataHandler::nextBuffer    ------------------------------------------------------------------------------------
FlagDataHandler::nextBuffer    Chunk = 28, Observation = 0, Array = 0, Scan = 31, Field = 1 (3C286_A), Spw = 11,
Display::getChunkInfo          FlagAgentDisplay::getChunkInfo Field : 1 , 3C286_A Spw : 11 Scan : 31 : 31
Manual_0::chunkSummary         => Data flagged in this chunk: 16.6667%
Quack_2::chunkSummary          => Data flagged in this chunk: 6.66667%
FlagDataHandler::nextBuffer    ------------------------------------------------------------------------------------
FlagDataHandler::nextBuffer    Chunk = 29, Observation = 0, Array = 0, Scan = 31, Field = 1 (3C286_A), Spw = 12,
Display::getChunkInfo          FlagAgentDisplay::getChunkInfo Field : 1 , 3C286_A Spw : 12 Scan : 31 : 31
Manual_0::chunkSummary         => Data flagged in this chunk: 16.6667%
Quack_2::chunkSummary          => Data flagged in this chunk: 6.66667%
FlagDataHandler::nextBuffer    ------------------------------------------------------------------------------------
FlagDataHandler::nextBuffer    Chunk = 30, Observation = 0, Array = 0, Scan = 31, Field = 1 (3C286_A), Spw = 13,
Display::getChunkInfo          FlagAgentDisplay::getChunkInfo Field : 1 , 3C286_A Spw : 13 Scan : 31 : 31
Manual_0::chunkSummary         => Data flagged in this chunk: 16.6667%
Quack_2::chunkSummary          => Data flagged in this chunk: 6.66667%
FlagDataHandler::nextBuffer    ------------------------------------------------------------------------------------
FlagDataHandler::nextBuffer    Chunk = 31, Observation = 0, Array = 0, Scan = 31, Field = 1 (3C286_A), Spw = 14,
Display::getChunkInfo          FlagAgentDisplay::getChunkInfo Field : 1 , 3C286_A Spw : 14 Scan : 31 : 31
Manual_0::chunkSummary         => Data flagged in this chunk: 16.6667%
Quack_2::chunkSummary          => Data flagged in this chunk: 6.66667%
FlagDataHandler::nextBuffer    ------------------------------------------------------------------------------------
FlagDataHandler::nextBuffer    Chunk = 32, Observation = 0, Array = 0, Scan = 31, Field = 1 (3C286_A), Spw = 15,
Display::getChunkInfo          FlagAgentDisplay::getChunkInfo Field : 1 , 3C286_A Spw : 15 Scan : 31 : 31
Manual_0::chunkSummary         => Data flagged in this chunk: 16.6667%
Quack_2::chunkSummary          => Data flagged in this chunk: 6.66667%
FlagDataHandler::nextChunk     ====================================================================================
Manual_0::msSummary            => Total data flagged in MS: 16.6667%
Clip_1_RR::msSummary           => Total data flagged in MS: 0.000977472%
Clip_1_RL::msSummary           => Total data flagged in MS: 0.000727421%
Clip_1_LR::msSummary           => Total data flagged in MS: 0.000591029%
Clip_1_LL::msSummary           => Total data flagged in MS: 0.000886544%
Quack_2::msSummary             => Total data flagged in MS: 6.14525%
TestFlagger::run               => Writing flags to the MS
```

MS Flag Summary - output dictionary of flag counts

Count the number of flagged visibilities as a function of 'antenna', 'baseline', 'field', 'spw', 'correlation', 'array', 'observation'.

Spwchan = True	# Calculate counts per channel per SPW.
Spwcorr = True	# Calculate counts per correlation per spw
Minrel, maxrel = 0 – 1	# Summary counts contain entries only for those antennas, fields, spw, # correlations, etc.. which have flag-fractions between minrel and maxrel.
Minabs, maxabs = 0 – N	# Summary counts contain entries only for those antennas, fields, # spw, correlations, etc which have at-least/at-most minabs/maxabs flags

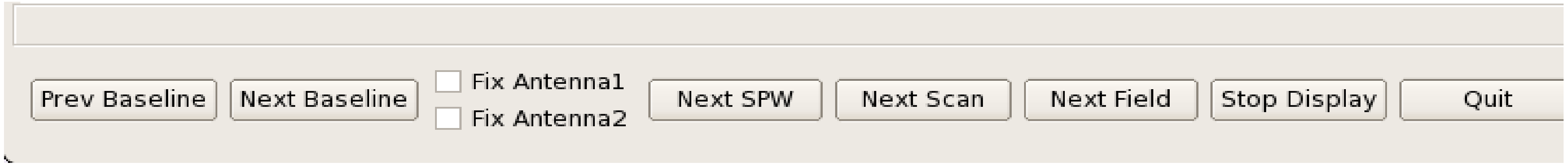
(coming soon) : Parameters to control summary views to be made and returned to the user (in python).
Example : 2D arrays (ant1 x ant2), one per spw,field, containing statistics from visibilities and flags, accumulated over time and channels.

Output : Dictionary of flag counts :

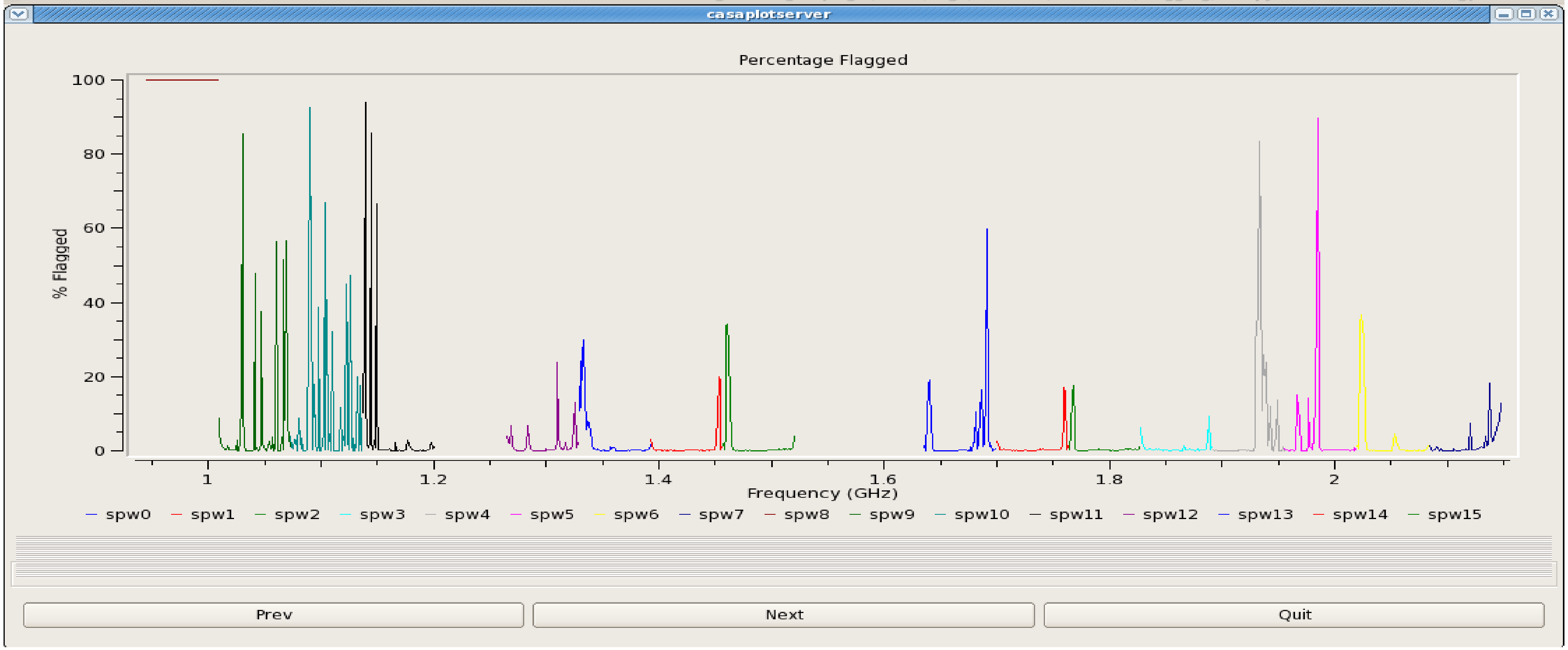
```
counts = tflagdata(vis='xxx.ms' , mode='summary');
print “Counts per spw : “, counts['spw'];
print “Total % flagged : “ , 100.0 * counts['flagged'] / counts['total']
print “Number of unflagged visibilities in spw 9 : “, counts['9']['total'] – counts['9']['flagged']
```

Report/Data/Flag Display

Display = 'data' : Visualize the 2D time-frequency plane per baseline, before and after flagging.



Display = 'report' : Gather reports from all agents that make then, make plots, return info to python.

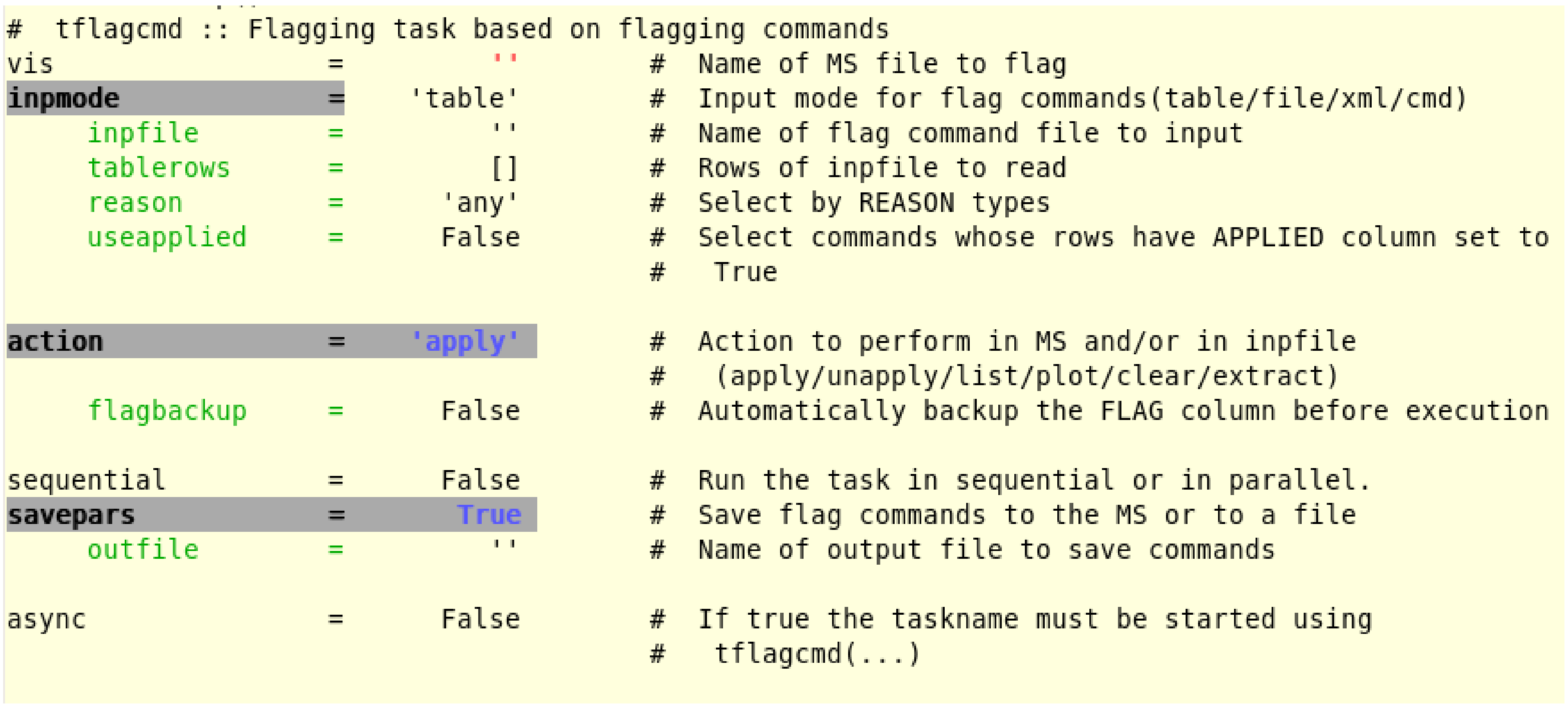


- known GUI problem : zoom/unzoom is unstable, and makes plots disappear.

Task : tflagdata

```
#  tflagdata ::  All purpose flagging task based on selections or a list of commands
vis                 =         ''          #  Name of MS file to flag
mode                =    'manual'         #  Flagging mode (list,manual,clip,shadow,quack,elevation
                                          #   ,tfcrop,extend,unflag,summary)
     field          =         ''          #  Field names or field index numbers: ''==>all,
                                          #   field='0~2,3C286'
     spw            =         ''          #  spectral-window/frequency/channel
     antenna        =         ''          #  antenna/baselines: ''==>all, antenna ='3,VA04'
     timerange      =         ''          #  time range: ''==>all,timerange='09:14:0~09:54:0'
     correlation    =         ''          #  Select data based on correlation
     scan           =         ''          #  scan numbers: ''==>all
     intent         =         ''          #  Select data based on observation intent: ''==>all
     feed           =         ''          #  multi-feed numbers: Not yet implemented
     array          =         ''          #  (sub)array numbers: ''==>all
     uvrange        =         ''          #  uv range: ''==>all; uvrange ='0~100klambda', default
                                          #   units=meters
     observation    =         ''          #  Select data based on observation ID: ''==>all

run                 =        True         #  Run and flag based on the parameters or the list given
                                          #   in inpfile
     writeflags     =        True         #  Write Flags to the MS
     sequential     =        True         #  Run the task in sequential or in parallel
     display        =         ''          #  Display data and/or end-of-MS reports at runtime
                                          #   (data, report, both).
     flagbackup     =       False         #  Back up the state of flags before the run

savepars            =       False         #  Save the current parameters to the MS or to a file
async               =       False         #  If true the taskname must be started using
                                          #   tflagdata(...)
```

Task : tflagcmd



```
#  tflagcmd :: Flagging task based on flagging commands
vis                 =         ''        #  Name of MS file to flag
inpmode             =    'table'        #  Input mode for flag commands(table/file/xml/cmd)
     inpfile        =         ''        #  Name of flag command file to input
     tablerows      =         []        #  Rows of inpfile to read
     reason         =      'any'        #  Select by REASON types
     useapplied     =      False        #  Select commands whose rows have APPLIED column set to
                                        #   True

action              =    'apply'        #  Action to perform in MS and/or in inpfile
                                        #   (apply/unapply/list/plot/clear/extract)
     flagbackup     =      False        #  Automatically backup the FLAG column before execution

sequential          =      False        #  Run the task in sequential or in parallel.
savepars            =       True        #  Save flag commands to the MS or to a file
     outfile        =         ''        #  Name of output file to save commands

async               =      False        #  If true the taskname must be started using
                                        #   tflagcmd(...)
```

Current implementation follows 'flagcmd', with as much backward-compatibility as possible.

- To be discussed : Stricter usage of the FLAG_CMD subtable in the MS.

  - 'summary' is not a flag command => currently not supported in this task
  - 'display=data' allows partial flag writes => incompatible with the 'applied' column of the flag_cmd subtable.

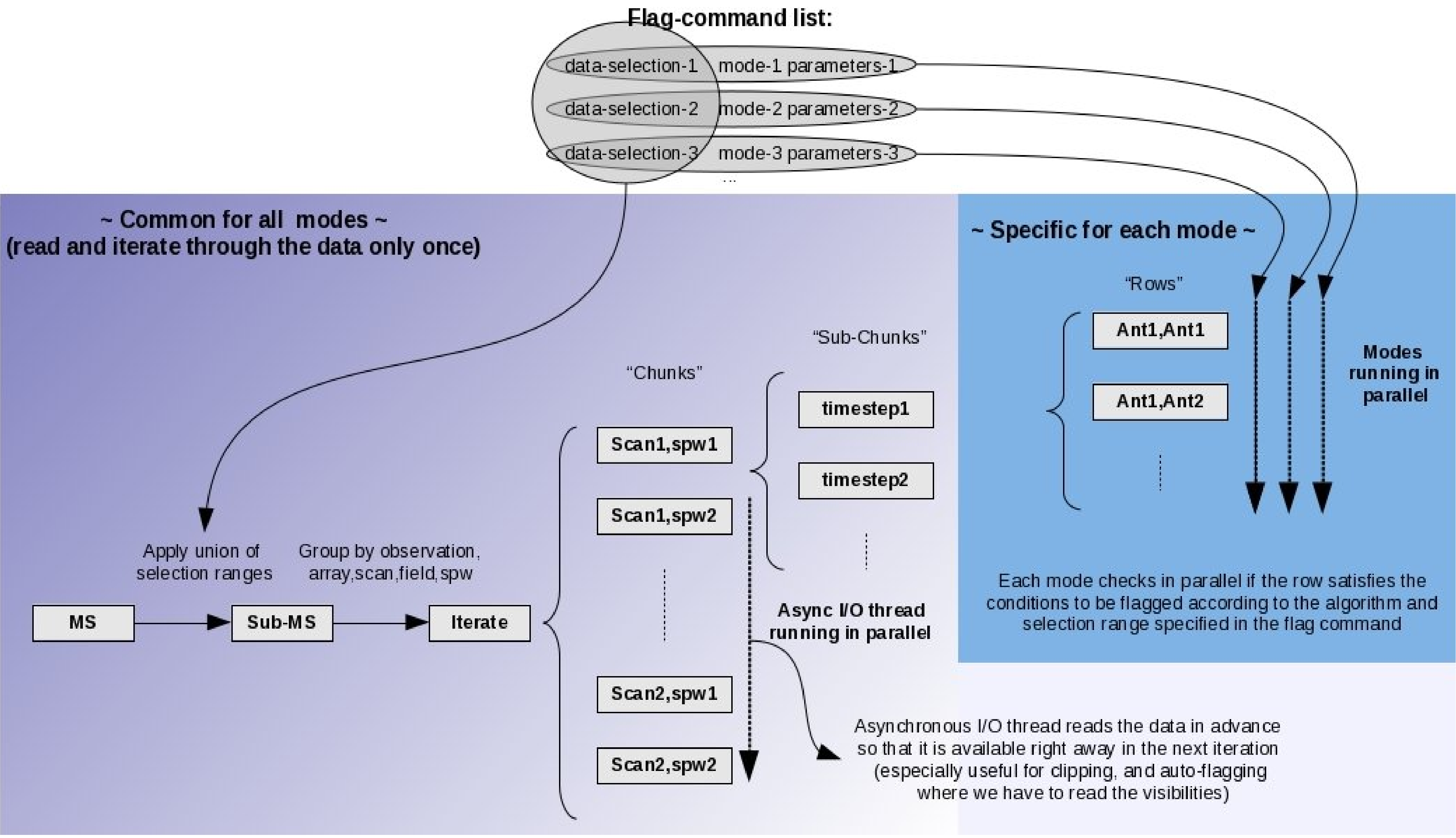
Design : python tool interface

All flagging modes, summary, and display operate as “agents” in a list.

Example script to flag shadowed antennas and exact zeros, and to count flags :

```
tf.open(msname='3C286.ms',ntime=100.0);

tf.selectdata();

tf.parseclipparameters(clipzeros=True);

ag2 = {'mode':'shadow', 'tolerance':10.0, 'apply':True}
tf.parseagentparameters(ag2);

ag3 = {'mode':'summary', spwchan=True, 'apply':True}
tf.parseagentparameters(ag3);

tf.init();

summary_reports = tf.run(writeflags=True, sequential=True);

tf.done();
```

Design : C++ infrastructure


Flag-command list:
data-selection-1 mode-1 parameters-1
data-selection-2 mode-2 parameters-2
data-selection-3 mode-3 parameters-3
...
~ Common for all modes ~
(read and iterate through the data only once)
~ Specific for each mode ~
"Rows"
Ant1,Ant1
Ant1,Ant2
Modes running in parallel
"Sub-Chunks"
timestep1
timestep2
"Chunks"
Scan1,spw1
Scan1,spw2
Scan2,spw1
Scan2,spw2
Apply union of selection ranges
Group by observation, array,scan,field,spw
MS
Sub-MS
Iterate
Async I/O thread running in parallel
Each mode checks in parallel if the row satisfies the conditions to be flagged according to the algorithm and selection range specified in the flag command
Asynchronous I/O thread reads the data in advance so that it is available right away in the next iteration (especially useful for clipping, and auto-flagging where we have to read the visibilities)

Performance-optimization choices

(1) List mode : Helps if multiple flag commands read the same data.

- Read data once, Apply multiple flag commands, Write flags once.
    - Manual-flag commands read only meta-data.
    - Shadow,elevation read meta-data + processing to calculate uvw, azel.
    - Clip reads visibilities.
    - tfcrop and rflag read visibilities and flags
    - Extend, summary read flags

(2) Pre-selection of data : Helps when operating on small section of a large dataset

- Select data based on a loose-union of input selection parameters from all commands

  (Implemented in python (currently) and available from the task. We will move this into C++)

(3) Asynchronous I/O : Helps when data I/O dominates cost (agents that read visibilities)

- While one chunk is being processed, pre-read data-columns required for the next.

  (Enabled only for agents that read visibility values.)

(4) Agent parallelization : Helps when processing dominates runtime.

- Run agents in parallel (helps only if agents are of similar type, and N_agents < N_cores)
- Different subsets of baselines are operated-upon in parallel.

  (useful only if all agents touch all baselines, and do not require communication across baselines)

(Disabled by default (for now). We will implement heuristics to decide what to choose.)

Performance Tests

Node : Casa-dev-08

(RHEL 5.7, 64bits,
8 cores at 2.5GHz,
8GB RAM, local disk)

Casapy mode :

casapy --nologger --noglogfile
--log2term –nogui -c “command”

Data Size : 49 GB

L-Band Wideband Continuum

Time to read visibilities
= 596 sec (10 min)

– We are still experimenting with async I/O and parallelization heuristics

More tests planned :
- spectral-line (>2k chan per spw)
- mosaics (non-contiguous scans per field)
- rflag performance

G55.7+3.4_10s (49GB) – Read-Only-Visibilities time: 596s

Mode		Old framework	New framework (w/o async I/O)		New framework (with async I/O)	
		Time	Time	Factor	Time	Factor
Read /Iterate VisCube	Tfcrop (ABS RR,ntime=100 freqfit='line')	1266s	898s	1.41	621s	2.04
	Tfcrop (ABS RR,ntime=100 freqfit='line', windowstats='both')	1699s	1089s	1.56	672s	2.53
	Clipping (ABS RR,LR,LR,LL, clipminmax=[0,0.02])	1124s	821s	1.37	625s	1.8
Iterate FlagCube	Extension (RR,ntime=100,growaround, growtime=50.0,growfreq=50)	1283s	197s	6.51	N/A	N/A
	Summary (with spw:channel breakdown)	290s	243s	1.19	N/A	N/A
Metadata-Based	Elevation (upperlimit=77.5)	206s	200s	~1	N/A	N/A
	Shadow (diameter=35m)	165s	38s	4.34	N/A	N/A
	Quack (quackinterval=5)	149s	42s	3.55	N/A	N/A
flagcmd	Manual (~900 online flags)	165s	36s	4.58	N/A	N/A

Performance Tests

Node : Casa-dev-08

(RHEL 5.7, 64bits,
8 cores at 2.5GHz,
8GB RAM, local disk)

Casapy mode :

```
casapy --nologger --noglogfile
--log2term –nogui -c “command”
```

Data Size : 168 GB

L-Band Wideband Continuum

Time to read visibilities
= 2497 sec (41 min)

– We are still experimenting with async I/O and parallelization heuristics

More tests planned :
- spectral-line (>2k chan per spw)
- mosaics (non-contiguous scans per field)
- rflag performance

G55.7+3.4_1ss (168GB) – Read-Only-Visibilities time: 2497s						
Mode		Old framework	New framework (w/o async I/O)		New framework (with async I/O)	
		Time	Time	Factor	Time	Factor
Read /Iterate VisCube	Tfcrop (ABS RR,ntime=100 freqfit='line')	14909s	5159s	2.89	2855s	5.22
	Tfcrop (ABS RR,ntime=100 freqfit='line', windowstats='both')	10787s	6807s	1.58	4113s	2.62
	Clipping (ABS RR,LR,LR,LL, clipminmax=[0,0.02])	4296s	3891s	1.1	3028s	1.42
Iterate FlagCube	Extension (RR,ntime=100,growaround, growtime=50.0,growfreq=50)	10686s	1694s	6.31	N/A	N/A
	Summary (with spw:channel breakdown)	2164s	1866s	1.16	N/A	N/A
Metadata-Based	Elevation (upperlimit=77.5)	1941s	1683s	1.15	N/A	N/A
	Shadow (diameter=35m)	846s	342s	2.47	N/A	N/A
	Quack (quackinterval=5)	1017s	341s	3.98	N/A	N/A
flagcmd	Manual (~900 online flags)	1207s	241s	5	N/A	N/A

Working with the FLAG_CMD subtable of the MS.

FLAG_CMD subtable of the MS :

APPLIED	COMMAND	INTERVAL	LEVEL	REASON	SEVERITY	TIME	TYPE
True/False	antenna='1', etc	0	0	“ why “	0	0	Flag/Unflag

Old “flagcmd” and “importevla” tasks.

- Defines a flag-command syntax (CAS-2405).

- data-selection parameters	: MS-Selection
- reason, flagtime, level, severity	: Other fields in the FLAG_CMD subtable
- clip, quack, unflag, shadow + subparams	: Based on task flagdata parameters

- Constructs flag commands from Flag.xml and saves them in the FLAG_CMD subtable of the MS.

- Runs flag commands in batch mode, with inputs from the FLAG_CMD subtable, or an external text file, or an xml file.

- New flag commands can be appended to the internal FLAG_CMD subtable, to serve as a history of (some) flagging actions.

- Allow selection of subsets of FLAG_CMD list for “un-apply” == “ unflag = True “

- Allow users to list, plot, edit and clear the subtable entries.

Problems with allowing task parameters in FLAG_CMD

(1) The FLAG_CMD subtable was designed for meta-data selections, but the flagcmd syntax includes task-parameter-based keywords.

- Commands that read visibilities, do not guarantee preservation of flag 'state' (clip,tfcrop,rflag)
- Encodes task-parameters inside the MS database (i.e. non-standard syntax).
- Task parameter names and their defaults change (as CASA evolves, and users request changes in defaults). We should not encode this info inside the MS.

(2) If it supports some non-manual flagdata modes, it should support all, including autoflag, unflag.

- When all modes are allowed, options such as re-apply/un-apply on operations such as ' tfcrop, rflag, unflag ' which read existing flags, can get complicated and difficult to interpret.
- Rflag has return-values. This cannot be handled via the subtable.

(3) Cannot preserve consistency between FLAG_CMD subtable and flagversions. (Not an AIPS flagtable)

- Users can edit the FLAG_CMD subtables, and 'un-apply' what was implemented as 'unflag' (incorrect)
- Not all flagging tasks save flag-commands (plotms, viewer, applycal), so except for pipeline scripts, FLAG_CMD will never represent the true flag-state of the MS.

(4) We now have end-user MS's with FLAG_CMD subtables using this format

- Since the flagcmd and importevla tasks were written in response to an immediate need to handle EVLA online flags, and implemented outside the CASA development process, it would have been best to avoid hard-coding these syntax decisions inside peoples' MS's.
- Only if users have saved their Flag.xml files, can we recover from this in a clean way.

Requirements + Solution

Functionality Requirements :

(1) Store online flags from Flag.xml in the MS, readable by flagging tasks.
(2) For every flagging action, save an equivalent command, to generate a history.
(3) Ability to extract flag commands from this history, and re-apply, un-apply subsets.

Our proposed solution :   FLAG_CMD subtable holds only meta-data selection strings from online flags.

(1) Store online flags from Flag.xml in the MS, using purely MS-Selection syntax, so that the 'FLAG' and 'APPLIED' fields of the subtable make sense. This is written at import-time, and optionally applied.

(2) For all other flagging actions, construct flag commands (task parameter lists), and save to an external text file.

(3) Flag commands can be applied from the FLAG_CMD subtable or external text files, or xml files, with reason-selections when appropriate. 'unapply' will work on the subtable. For others, revert to a flagversion, and re-apply.

(4) We will provide a convertor from the old to new formats (where possible).
---> manualflag : manual ,   cliprange = '0~1E-10' : clipminmax = [0,1E-10], diameter : tolerance (different meaning).

(5) Recent/future changes to the old format will be ignored.
---> Addition of 'poln' to the syntax is not required. An addition to MSSelection will solve it (CAS-3767).
In 'importevla2', we have implemented the same, approximate behavior of 'poln' without the new keyword.

(6) Until we sort this out, all users must continue to save their Flag.xml at import-time.
We will always be able to re-generate latest online flag commands from there, or use Flag.xml directly.

We need a decision on this from the CSSC/EVLA, so that we can make things consistent for the 3.4 release. Feature-freeze date is March 15.   Question : Is our solution acceptable ?

Some known problems + future additions

- CAS-3712 : If a flag-command contains a selection on something not present in the MS, an MSSelection exception appears. Solution-plan exists.

- CAS-3767 : Online flags “ pol 'R' for ea05 “ are interpreted as 'RR,RL,LR' for ea05, instead of 'RR,RL' for baselines where ea05 is ant_1, and 'RR,LR' for baselines where ea05 is ant_2. Solution-plan exists.

- CAS-3005 : Support for the new CalTable format has to wait until CalTable and Cal-Selection classes stabilize and add a few features (allow flagger to write to the CalTable, and iterate in time-chunks).

- CAS-3679 : All CASA tasks that modify flags (plotms, viewer, applycal) should generate equivalent flag-commands that can be saved in a text file, and re-used in the flagging tasks.

- Add internal heuristics to decide what type of agent-level parallelization to use, depending on number and type of input flag commands. For now, agent-level parallelization is turned off. Only async-IO is enabled.

- Add support for 2D data/flag views, as per pipeline-group requirements.
- Ability to apply flag commands via visbuffer-level function calls (integrate all flagging/display code)
- Additional agent options (rflag , extend, summary, quack)
- Ability to choose whether to write flags to the MS or not, per-chunk, from the flagger GUI.
- Ability to save plots from the GUI, and make/save plots without the GUI (for pipelines/scripts).