Chapter 1

Synthesis

package for processing synthesis data

include "synthesis.g"

Description
The synthesis package contains modules needed for processing synthesis data.

1.1 autoflag - Module

Module for automatic flagging of synthesis data
include "autoflag.g"

Description

The `autoflag` module provides automatic synthesis flagging capabilities within AIPS++. The primary purpose of this module is to flag data inside a MeasurementSet using a number of different algorithms and heuristics.
The facilities of the `autoflag` module are made available in Glish by including the following script:

```
- include 'autoflag.g'
T
```

where a hyphen precedes user input. The Glish response is indicated without the prompt.
An `autoflag` tool is created and attached to a specified measurement set as indicated in the following example:

```
- af:=autoflag('3C273XC1.MS')
```

A variety of algorithms (called *methods* in this context) can be applied for any given `autoflag` tool. A setdata method allows a user to apply set*methods* to the whole measurement set with default parameters or to a subset of the measurement set by specifying parameters. Individual methods are set up in advance by calling Glish functions of the tool, i.e.: `af.set`*`method`*`(`*`parameters`*`)`. Afterwards, a call to `af.run(`*`options`*`)` actually applies all the specified methods simultaneously. Here is an example:

```
- af:=autoflag('3C273XC1.MS')
- af.setdata()
- af.settimemed(thr=6,hw=5)
Added method 1: timemed (TimeMedian)
   *thr         = 6
   *hw          = 5
    rowthr      = 10
    rowhw       = 6
    column      = DATA
    expr        = ABS I
    debug       = F
    fignore     = F
T
- af.setuvbin(nbins=100,thr=.01)
Added method 2: uvbin (UVBinner)
   *thr         = 0.01
   *nbins       = 100
    plotchan    = F
    econoplot   = T
    column      = DATA
    expr        = ABS I
    fignore     = F
T
- af.settimemed(thr=5,hw=5,expr="- ABS XX YY")
Added method 3: timemed (TimeMedian)
    thr         = 5
   *hw          = 5
    rowthr      = 10
    rowhw       = 6
    column      = DATA
   *expr        = - ABS XX YY
    debug       = F
    fignore     = F
T
- af.run(trial=T)
   lots of interesting messages
-
```

Here, three methods -- a *UV binner* and two *time-median filters* -- are set up and executed. Note the following crucial points:

- The set*method*() functions respond with a summary of the parameters with which the method will be run. All parameter have reasonable defaults. Parameters for which you have specified an explicit non-default value are marked with an ``*'' symbol.

- You may set up multiple instances of a method, using different parameters for each instance. In the above example, two timemed methods are set up. The first one is applied to $|I|$, the second one -- to $|XX| - |YY|$.

- Most methods perform flagging based on some real value derived from a set of complex correlations, e.g. $|I|$, or $|XX| - |YY|$. You can specify how the value is derived by using the expr parameter. This parameter is either a string or an array of strings. A single string will be automatically split at whitespace, so "ABS XX" and 'ABS XX' are fully equivalent. Currently, the following types of expressions are recognized:

  func CC --- real function of a complex correlation. E.g. "ABS XX" for $|XX|$.

  +/- *func C1 C2* --- sum/difference of functions of two correlations. E.g. "- ABS RR LL" for $|RR| - |LL|$.

  func +/- *C1 C2* --- function of a sum/difference of two correlations. E.g. "ARG - XX YY" for $\arg(XX - YY)$.

  func I --- I is used to specify Stokes I. Depending on polarization frame, it is equivalent to either $XX + YY$ or $RR + LL$. $|I|$ ("ABS I") is in fact the default expression for all methods.

  The following functions are currently recognized: ABS, ARG, RE, IM, NORM. Any correlations present in the measurement set are recognized.

- The column parameter specifies which measurement set column to use. Possible values are DATA, MODEL or CORR for corrected data.

- All methods by default honor pre-existing flags (as determined by the FLAG and FLAGROW columns), in the sense that flagged values are omitted from all calculations. Methods may be asked to ignore existing flags via the fignore parameter. The reset option to autoflag.run clears all pre-existing flags prior to a flagging run.

- For each flagging run, autoflag will automatically produce a graphical *flagging report*, consisting of a summary and several plots showing the distribution of flags by baseline, antenna, frequency, time, etc. By default, this report is written in PostScript format to a file named flagreport.ps. However, the devfile parameter of autoflag.run() can be used to redirect this output to a different file or a different PGPlot device.
- If your measurement set contains different fields and/or spectral windows (or, more generally, different data description IDs), autoflag will automatically treat it as a set of disjoint chunks. Flagging is performed independently within each chunk (i.e. independently for each spectral window, field, etc.)
- autoflag will attempt to manage its memory consumption to stay roughly within the meximum physical memory specified in your .aipsrc file. If you find your system swapping to disk when running autoflag, you can improve performance by using a reduced memory setting. Conversely, if a lot of memory remains free during an autoflag run, performance may be improced by increasing the setting.

Tools

autoflag	Automatic synthesis flagging

1.1.1 autoflag - Tool

Package synthesis
Module autoflag

Automatic synthesis flagging

Description
An autoflag tool performs automatic synthesis flagging operations within AIPS++. A separate autoflag tool needs to be created for each separate measurement set for which processing is to be performed.

Constructors

autoflag	Construct an autoflag tool from a MeasurementSet

Functions

attach	Attaches tool to another measurement set
detach	Detaches from measurement set
done	Destroys the autoflag tool
help	Prints help on available flagging methods
reset	Disables specific flagging methods
resetall	Disables all flagging methods
run	Executes a flagging run
setdata	Set the data parameters selection for subsequent processing
setfreqmed	Sets up the freq-median flagging method
setnewtimemed	Sets up the new time-median flagging method
setselect	Sets up the Selector flagging method
setsprej	Sets up the spectral rejection flagging method
settimemed	Sets up the time-median flagging method
setuvbin	Sets up the UV Binner flagging method
summary	Prints a summary of active methods

autoflag - Constructor

Package	synthesis
Module	autoflag
Tool	autoflag

Construct an autoflag tool from a MeasurementSet

Synopsis
autoflag(msname)

Description
Create an autoflag tool, and attach it to a specified measurement set.

Arguments

msname	Measurement set to be processed
	AllowedTable name
	Defaultnone

Returns
autoflag tool

Example

```
af:=autoflag('3C273XC1.MS')
```

---

autoflag.settimemed - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Sets up the time-median flagging method

Synopsis

settimemed(thr, hw, rowthr, rowhw, norow, column, expr, fignore)

Description

The timemed method uses a median-in-time filter. It works by computing a sliding median of some value d_{cbt} (where c is frequency channel, b is baseline, and t is time slot), as specified by the expr parameter (see the general autoflag module description for details of how expr determines d.) The sliding median is computed in time, on a per-channel, per-baseline basis:

$$m_{cbt} = \text{median}\{d_{cbt'},\ \text{where}\ |t' - t| \leq \delta\}$$

(where δ is given by the hw parameter.) Then, a median absolute deviation (MAD) of the data with respect to the median is computed on a per-row basis:

$$M_{bt} = \text{median}\{|d_{cbt} - m_{cbt}|,\ \text{for all}\ c\}$$

(For single-channel data, $M_{bt} \equiv m_{0bt}$.) M_{bt} is then smoothed by applying a sliding median in time:

$$\tilde{M}_{bt} = \text{median}\{M_{bt'},\ \text{where}\ |t' - t| \leq \tilde{\delta}\}$$

(where $\tilde{\delta}$ is given by the rowhw parameter.) All points where

$$|d_{cbt} - m_{cbt}| > \theta \tilde{M}_{bt}$$

are then flagged. The threshold level θ is given by the thr parameter.

In addition, timemed can also flag entire rows based on their MAD. This is only performed for line data (specifically, when there are at least 10 channels per row), and may be explicitly disabled by specifying norow=T. To do this, it first computes a per-baseline MAD of the MAD:

$$\hat{M}_b = \mathrm{median}\{|M_{bt} - \tilde{M}_{bt}|, \text{ for all } t\}$$

and flags rows where $|M_{bt} - \tilde{M}_{bt}| > \hat{\theta}\hat{M}_b$, where $\hat{\theta}$ is given by the rowthr parameter.
Timemed will iterate over the measurement set -- and compute m_{cbt} -- only once. Existing flags are taken into account at this time (unless fignore is specified). MADs are then computed and recomputed iteratively, dropping flagged points and rows, until convergence is achieved (i.e. no more flags are raised). Since m_{cbt} is **not** recomputed as new flags are raised, running this method a second time may sometimes be of use.

Arguments

thr	Flagging threshold Allowednumeric (real) Default5
hw	Half-window for the sliding median, if time slots is less than 4 times half window, ignoring this chunk AllowedInt Default10
rowthr	Row flagging threshold Allowednumeric (real) Default10
rowhw	Half-window for the sliding median of the MAD AllowedInt Default10
norow	Disable row flagging AllowedBoolean DefaultF
column	Which column to base flagging on Allowed"DATA", "MODEL" or "CORR" Default"DATA"
expr	Expression to use Allowedsee text Default"ABS I"
fignore	Ignore existing flags AllowedBoolean DefaultF

Example

```
- af:=autoflag('3C273XC1.MS')
- af.setdata()
- af.settimemed(thr=5,hw=5,expr="- ABS RR LL")
```

---

autoflag.setnewtimemed - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Sets up the new time-median flagging method

Synopsis

```
setnewtimemed(thr, column, expr, fignore)
```

Description

The newtimemed method uses a median-in-time filter. It follows the procedure of timemed but considering a median value from all time slots in a chunk.

The median is computed on a per-channel, per-baseline basis. The thr parameter multiplying the standard deviation of the data in a chunk sets the threshold for flagging. If the thr parameter is specified (e.g. thr = 3), all points with greater than three standard deviations away from the median are flagged.

Arguments

thr	Flagging threshold Allowednumeric (real) Default3
column	Which column to base flagging on Allowed!"DATA", "MODEL" or "CORR" Default!"DATA"
expr	Expression to use Allowedsee text Default!"ABS I"
fignore	Ignore existing flags AllowedBoolean DefaultF

Example

```
- af:=autoflag('H031121.MS')
- af.setdata()
- af.setnewtimemed(thr=5, expr="ABS RR")
- af.run()
```

---

autoflag.setdata - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Set the data parameters selection for subsequent processing

Synopsis
setdata(mode, nchan, start, step, mstart, mstep, spwid, fieldid, msselect, async)

Description
This setup function selects which data are to be used subsequently. After invocation of setdata, only the selected subset of data are operated on.
Data can be selected by field ids, spectral window ids, channels, velocities. The selection is controlled by the mode argument:

fieldids Selection in field ids.

spwids Selection in spectral window ids.

channel Selection in channels using the nchan, start and step arguments.

velocity Selection in radio velocity, using the nchan, mstart and mstep arguments.

Spectral window ids must be specified for the channel and velocity modes.
For channel mode, the other fields have the following meaning:

nchan is the number of output channels. It defaults to 1, or the number of input channels.

start is the first channel from input dataset that is to be used. It defaults to 1 (i.e. first channel).

step gives the increment between selected input channels.

The channels are centered on velocities: start, start+step, start+2*step, etc.
Examples for choosing the parameters for setdata.

```
af.setdata(mode='fieldids',fieldid=1);
```

```
af.setdata(mode='spwids',spwid=1);
```

```
af.setdata(mode='spwids & fieldids',spwid=1, fieldid=1);
```

```
af.setdata(mode='channel', nchan=64, start=1, step=1, spwid=1)
```

For velocity modes, the mstart and mstep are the start and step velocities as strings.

```
af.setdata(mode='velocity', nchan=64, mstart='20km/s', mstep='-80m/s', spwid=1)
```

For multi-spectral window selection, the data can be selected as follows

```
af.setdata(mode='channel', nchan=[20,30], start=[1,1], step=[1,1],
            spwid=[1,2])
```

The above means that you would make a data selection of 20 channels (starting from 1 steping 1) from the first spectral window and 30 channels (starting from 1 steping 1) from the second spectral window.

Note: Current autoflag setdata implementation does not support cumulative set data. For example, instead of doing

```
af.setdata(mode='spwids', spwid=1)
af.setdata(mode='fieldids', fieldid=1)

The input should be

af.setdata(mode='spwids & fieldids', spwid=1, fieldid=1)
```

Arguments

mode	Type of processing: fieldids, spwids, channel or velocity Allowed:'fieldids'|'spwids'|'channel'|'velocity'
nchan	Number of channels to select AllowedVector of Ints Default:1
start	Start channels (1-relative) AllowedVector of Ints
step	Step in channel number AllowedVector of Int
mstart	Start velocity (e.g. '20Km/s') AllowedQuantity
mstep	Step in velocity (e.g. '100m/s' AllowedQuantity
spwid	Spectral Window Ids (1 relative) to select AllowedVector of Ints Default:[]
fieldid	Field Ids (1 relative) to select AllowedVector of Ints Default:[]
msselect	TQL select string applied as a logical "and" with the other selections AllowedString
async	Run asynchronously in the background? AllowedBool Default:dowait

Example

```
af:=autoflag('testdata.ms');
af.setdata(mode='channel',nchan=64,start=1,step=1,spwid=1,msselect=
'SCAN_NUMBER < 5')
af.setselect(quack=[13,2],unflag=T);
af.run();
```

```
af.resetall();
af.summary();
af.setuvbin(thr=.01);
af.run();
```

---

autoflag.setfreqmed - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Sets up the freq-median flagging method

Synopsis

```
setfreqmed(thr, hw, rowthr, rowhw, column, expr, fignore)
```

Description

The freqmed method uses a median-in-frequency filter. It is very similar to timemed, except that it is based on a sliding median in frequency:

$$m_{cbt} = \text{median}\{d_{c\prime bt}, \text{ where } |c\prime - c| \leq \delta\}$$

In all other respects this method is identical to timemed, hence we refer you you to the settimemed description for all details.

Arguments

thr	Flagging threshold Allowednumeric (real) Default5
hw	Half-window for the sliding median AllowedInt Default10
rowthr	Row flagging threshold Allowednumeric (real) Default10
rowhw	Half-window for the sliding median of the MAD AllowedInt Default10
column	Which column to base flagging on Allowed"DATA", "MODEL" or "CORR" DefaultDATA
expr	Expression to use Allowedsee text Default"ABS I"
fignore	Ignore existing flags AllowedBool DefaultF

Example

```
- af:=autoflag('3C273XC1.MS')
- af.setdata()
- af.setfreqmed(thr=5,hw=5,expr="- ABS RR LL")
```

---

autoflag.setsprej - Function

```
Package  synthesis
 Module  autoflag
   Tool  autoflag
```

Sets up the spectral rejection flagging method

Synopsis

```
setsprej(ndeg, rowthr, rowhw, norow, spwid, fq, chan, region,
column, expr, fignore)
```

Description

The sprej method works by fitting a user-specified segment of the spectrum (presumably, the continuum) by a polynomial. It then flags entire rows based on the chi-square statistic of the fit, in a manner similar to timemed. Please refer to setimemed for detailed math, substituting χ^2_{bt} for M_{bt}. The value that is actually fitted is specified via the expr parameter (see the general autoflag module description for details).

The segement to be fitted can be specified as a range of channel indices (1-based) or frequencies. Frequencies may be given as string measures ("1.4GHz") or float values (units of MHz assumed). A vector of two values [C1,C2] specifies the segment from C1 to C2, inclusive. Several disjoint segments may be specified with a 2N vector or a (2,N) matrix.

For a measurement set containing multiple spectral windows, you can set the region argument to a record of records, with each sub-record containing spwid and fq and/or chan fields, specifying the segments of a specific spectral window. See examples below.

Arguments

ndeg	Number of degrees for polynomial AllowedInt Default2
rowthr	Flagging threshold, in MADs of χ^2 Allowednumeric (real) Default5
rowhw	Half-window for the sliding median AllowedInt Default6
norow	Disable row flagging AllowedBoolean DefaultF
spwid	Spectral window number (1-based), or [] or F for all windows AllowedF or [] or Int DefaultF
fq	Range of frequencies Allowed2N or (2,N) array of Strings w/units or Floats DefaultF
chan	Range of channels (1-based). Either ‘‘fq’’ or ‘‘chan’’ must be specified. Allowed2N or (2,N) array of Ints DefaultF
region	A compound spectral segment specification (see example) AllowedRecord of records DefaultF
column	Which column to base flagging on Allowed"DATA", "MODEL" or "CORR" Default"DATA"
expr	Expression to use Allowedsee text Default"ABS I"
fignore	Ignore existing flags AllowedBool DefaultF

Example

```
- af:=autoflag('3C273XC1.MS')
- af.setdata()
# use segments 807-807.5 and 809-809.5 MHz for fitting
- af.setsprej(fq=[807,807.5,809,809.5))
```

```
...
# use channels 10-20 and 30-40
- af.setsprej(chan=[[10,20],[30,40]])
...
# use different segements in different spectral windows
- region := [=]
- region[1] := [ spwid=1,fq="1.4GHz 1.401GHz" ]
- region[2] := [ spwid=2,chan=[10,20] ]
- region[3] := [ spwid=4,chan=[50,60] ]
- af.setsprej(region=region)
```

---

autoflag.setuvbin - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Sets up the UV Binner flagging method

Synopsis

setuvbin(thr, minpop, nbins, plotchan, econoplot, column, expr, fignore)

Description

The uvbin method works by binning the data on a *value* vs. UV-distance plane (*value* is determined by the expr parameter -- see the general autoflag module description for a description of this mechanism.) Data in underpopulated bins is then flagged. The cut-off point for flagging may be expressed in one of two ways:

- If the maxpop parameter is specified (e.g. maxpop=4), all bins with less than maxpop points are flagged.
- If the thr parameter is specified, a probability cut-off is selected. This is done as follows. The cumulative probability $p(N)$, defined as the probability for a given point to reside in a bin of a population $\leq N$, is first computed. All bins of population N, where $p(N)$ <thr, are then flagged.

In simpler terms, using thr=0.01 will flag up to 1% of your data points, beginning with the least populous bins. I.e., first the single-point bins are flagged, then the 2-point bins, and so on, up to the $N-1$-point bins, where N is such that flagging the N-point bins would exceed 1% of all data points.

- When both maxpop and thr are specified, uvbin will use the higher of the two resulting cut-offs.

Uvbin can be asked to produce a value vs. UV-distance plot, and a plot of $p(N)$. These plots are attached to the flagging report. Uuse of the econoplot mode is recommended when dealing with medium-to-large data sets, for significant reduction in plot file size and printing time.

Arguments

thr	Population cut-off, as $p(N)$ value. AllowedFloat, 0 for none Default0
minpop	Population cut-off, as minimum bin count. Either thr or maxpop must be specified, or both. AllowedInt, 0 for none Default0
nbins	Number of bins (one number for both axes, or separate for UV, value) AllowedInt or 2 Ints Default50
plotchan	Channel number (1-based) to plot, T for middle channel, or F for no plot AllowedInt or Bool DefaultF
econoplot	Produce an economical plot (crowded bins are not resolved) AllowedBool DefaultF
column	Which column to base flagging on Allowed:"DATA", "MODEL" or "CORR" Default:"DATA"
expr	Expression to use Allowedsee text Default:"ABS I"
fignore	Ignore existing flags AllowedBool DefaultF

Example

```
- af:=autoflag('3C273XC1.MS')
- af.setdata()
- af.setuvbin(nbins=[50,100],expr="- ABS RR LL",plotchan=40,econoplot=T)
```

---

autoflag.setselect - Function

```
Package  synthesis
 Module  autoflag
   Tool  autoflag
```

Sets up the Selector flagging method

Synopsis
setselect(spwid, field, fq, chan, corr, ant, baseline, timerng, autocorr, timeslot, dtime, clip, flagrange, quack, unflag)

Description

The select method allows you to directly flag or unflag a subset of your data. Subsets can be specified via any combination of field, spectral window, frequency or channel number, correlations (i.e. polarizations), antennas, baselines, time ranges. The method will flag/unflag a whole subset, or, optionally, only autocorrelations, or only specific time slots within the subset. Select also includes the option to flag VLA quacks (specified by scan interval and scan start time).
The spwid, field, fq, chan, corr, ant, baseline, timerng parameters select a subset of the measurement set, on a logical-AND basis.

- An empty array ([]) or a boolean F (which is the default for any parameter), means ‘‘do not select on this’’.
- fq specifies ranges of frequencies. You can use strings with units (i.e. "800MHz"), or floats, in which case MHz is assumed.
- chan specifies ranges of channel numbers (1-based). Negative numbers can be used to specify channels starting from the end of the window, e.g., [-10,-1] means the last 10 channels.
- Where ranges are specified, the method expects either a $2N$ vector:

  $$[S_1, E_1, S_2, E_2, ..., S_N, E_N]$$

  or a (2,N) array of start/end points.

  For example, if you want to indicate 7 channels (1,2,3,4,5,6,7), either chan = [1,7] or chan = [[1,2],[2,3],...,[6,7]] does the work.

  NOTE: The usual means of indicating a range with the ":" delimiter does not apply for the channel subexpression. For example, the equivalent to [1:3] to select channels 1 through 3 would be [1,3].

- Antennas can be specified by name or 1-based number (which is sometimes the same thing). Names must correspond to the ANTENNA subtable of the measurement set.
- Baselines may be specified either as an array of strings, using antenna names or numbers delimited by ``-'' (e.g. "1-2 4-7", or "RT1-RTA"), or as a (2,N) array of antenna numbers (e.g. $[[1,2],[4,7]]$).

The autocorr, timeslot+dtime, clip, flagrange and quack parameters specify what to flag (or unflag) within the selected subset, on a logical-OR basis. If none are specified, then the entire subset is flagged (or unflagged).

Clipping. The clip and flagrange parameters enable flagging based on the value of an expression. Clip implies flagging of all values outside a specific range, flagrange implies flagging within a specific range. Like other flagging methods, you can specify values via expressions. Two types of expressions are recognized: correlation-derived expresisons (see the general autoflag module description for details), and UVW expressions (see below). To use a single expression and range, set clip or flagrange to a record as folows:

```
- af:=autoflag('3C273XC1.MS')
- af.setdata();
- af.setselect(clip=[expr='+ ABS XX YY',min=1e-6,max=.5])
# Above the feeds are assumed to be linear; use RR and LL for circular.
- af.run();
```

The expr field specifies the expression to clip on, and min and max specify the range. Either the min or max field may be omitted, in which case $\pm\infty$ is assumed. To flag several ranges and/or expressions, set clip or flagrange to a record of records:

```
- af:=autoflag('3C273XC1.MS')
- af.setdata();
- cliprec:=[=]
- cliprec[1] := [expr='+ ABS XX YY',min=1e-6,max=.5]
- cliprec[3] := [expr='- ABS XX YY',max=.5]
- cliprec[2] := [expr="UVD",min=1]
- af.setselect(clip=cliprec)
```

A shorthand way is also available: use expression as the field name, and set the field to (a) a record of min, max, (b) a 2-element array of (min,max), or (c) a single scalar value (assumed to be max). Thus, the record above may be equivalently formed as:

```
- cliprec['+ ABS XX YY'] := [1e-6,.5]         # equivalent to [min=1e-6,max=.5]
- cliprec['- ABS XX YY'] := .5                # equivalent to [max=.5]
- cliprec['UVD']         := [min=1]
- af.setselect(clip=cliprec)
```

When specifying several ranges for the same expression, you must naturally use the longhand form.

In addition to the regular correlation expressions, select also recognizes a few UVW-based expressions. The simplest ones are: U, V, W, ABS U, ABS V, ABS W. UVD stands for uv-distance, and corresponds to $\sqrt{u^2+v^2}$. Two more expressions represent angles in the uv-plane:

UVA stands for uv *position angle*, specified in degrees clockwise from the v axis.

HA stands for hour angle (East-West array assumed). It is specified in degrees *counterclockwise* from the u axis, using the following definition: $u = B\cos\alpha_h,\ v = B\sin\alpha_h\sin\delta$, where B is baseline, α_h is hour angle, and δ is declination.

When using the UVA and HA expressions, you are actually defining *sectors* in the uv-plane, from min. to max. angle, e.g.:

```
- af.setselect(flagrange=[expr="uva",min=47,max=49])
```

...flags everything in the sector between 47 and 49 degrees, in terms of positional angle.

Quack is a special option for VLA datasets, to flag the ''quacks'' at the beginning of each scan. It is specified by two numbers, $[t_{si}, \tau]$. t_{si} is the scan interval, used to determine the beginning of a new scan. τ is the number of seconds to flag at the start of a scan.

Arguments

spwid	Spectral window numbers (1-based), or [] or F for all Allowed Array of ints Default F
field	Field names or numbers (1-based), or [] or F for all Allowed Array of ints/strings Default F
fq	Range(s) of frequencies, or [] or F for all Allowed 2N or (2,N) array of Strings w/units or floats Default F
chan	Range(s) of channels (1-based), or [] or F for all Allowed 2N or (2,N) array of ints Default F
corr	Correlations (by name, e.g. ‘‘XX YY’’), or [] or F for all Allowed array of strings Default F
ant	Antennas (by name or number), or [] or F for all Allowed array of strings/ints Default F
baseline	Baselines, or [] or F for all Allowed array of strings/ints Default F
timerng	Time ranges, as MJDs or date strings, or [] or F for all Allowed array of strings/floats Default F
autocorr	Flag autocorrelations Allowed Boolean Default F
timeslot	Flag specific time slots. Specifies center time as MJDs or date strings Allowed array of strings/floats Default F
dtime	Tolerance of timeslot parameter, in seconds Allowed numeric Default 10
clip	Flag values outside a specific range Allowed Record, see above Default F
flagrange	Flag values within a specific range Allowed Record, see above Default F
quack	Flag VLA quacks -- scan interval and quack length, in seconds Allowed 2-element numeric array Default F
unflag	Clear flags instead of setting them Allowed Boolean Default F

Example

```
- af:=autoflag('3C273XC1.MS')
- af.setdata();
# flag all autocorrelations and VLA quacks
- af.setselect(autocorr=T,quack=[120,60])
# flag some frequencies on specific antennas, during a certain period of time
- af.setselect(ant=[1,2,4],timerng="1998/01/20/20:00 1998/01/20/22:00",
               fq=[807.51,807.56])
# flag RR and RL on a few bad time slots on a specific baseline
- af.setselect(corr="RR RL",baseline="3-7",
               timeslot="1998/01/20/20:00:30 1998/01/20/20:20:30")
```

---

autoflag.attach - Function

```
                          Package  synthesis
                           Module  autoflag
                             Tool  autoflag
```

Attaches tool to another measurement set

Synopsis

attach(msname)

Description

This function will detach the tool from a measurement set, and attach it to a another one. All flagging methods specified via set*method*() calls are preserved.

Arguments

msname	Measurement set to be processed
	AllowedTable name
	Defaultnone

Returns

Boolean T, or a fail

Example

```
af.attach('3C273XC1.MS)
```

---

autoflag.run - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Executes a flagging run

Synopsis

```
run(globparm, plotscr, plotdev, devfile, reset, trial)
```

Description

This function will perform a flagging run utilizing methods configured by prior set*method*() calls. It has a number of optional parameters affecting various aspects of the flagging process:

- Run will optionaly produce a graphical flagging report (plots of flag density per baseline, antenna, time, etc.), both on-screen and in a file. Either option may be turned on or off. To redirect hardcopy output, the devfile argument may be set to a filename for PostScript output, or to a full PGPlot device specification (''file/device'').
- The globparm argument allows you to set parameters that are ''global'' to all methods. For example, using globparm=[hw=10,fignore=T] will cause **all** methods to use the given values for the hw and fignore parameters, unless another value was explicitly set in the set*method*() call. In other words, globparm takes precedence over default values, but not over set*method*() arguments.
- When using trial, flagging is performed and flagging reports are generated without writing the results back into the measurement set.
- The reset option clears all pre-existing flags from the FLAG and FLAGROW columns of the measurement set prior to running

the methods. If trial is specified as well, the MS is not modified, but all methods run as if all flags were clear to begin with (in fact, reset and trial together is equivalent to globparm=[fignore=T].)

Arguments

globparm	Record of global method parameters AllowedRecord Defaultempty
plotscr	Number of plots per window for on-screen flagging report, or default for no report Allowed$[nx, ny]$ DefaultNo report
plotdev	Number of plots per page in hardcopy flagging report, or [] or F for no report AllowedBoolean F or $[nx, ny]$ DefaultF
devfile	File or PGPlot device for hardcopy report AllowedString Defaultflagreport.ps/ps
reset	Reset pre-existing flags before running AllowedBoolean DefaultF
trial	Do a trial run only AllowedBoolean DefaultF

Returns
Boolean T, or a fail

Example

```
af:=autoflag('3C273XC1.MS)
af.setdata();
...
af.setselect(autocorr=T,quack=[240,120])
...
af.setuvbin(thr=.01)
...
af.run()
```

---

autoflag.summary - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Prints a summary of active methods

Synopsis
summary()

Description

This function prints a summary of the methods that have been enabled via set*method*(), along with their parameters. Non-default values are marked with an ‘‘*’’ symbol.

Returns
Boolean T

Example

```
- af.summary()
Measurement set: test.MS2
Methods set up:
1: uvbin (UVBinner):
    thr         = 0.001
    nbins       = 50
    plotchan    = F
    econoplot   = T
    column      = DATA
    expr        = + ABS XX YY
    fignore     = F
...
```

---

autoflag.help - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Prints help on available flagging methods

Synopsis
help(names)

Description

This function prints a summary of the available flagging methods and their parameters. Use help() with no arguments to get help of all methods, or help("method names") for specific ones.

Arguments

names	Names of methods (all methods, if omitted)	
	AllowedString	
	DefaultF	

Returns
Boolean T

Example

```
af.help()
af.help("timemed freqmed")
```

---

autoflag.reset - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Disables specific flagging methods

Synopsis
reset(methods)

Description

This function disables methods previously set up by calls to set*method*(). You can specify methods by name, in which case all instances of that method are disabled, or by number (call summary() first to look up numbers).
The function takes any number of arguments. Each argument may be a string, a method number, or an array of such. Calling reset() with no arguments produces a fail; you should use resetall() if you want to reset all methods.

Arguments

methods	Names or numbers of methods
	AllowedString or int
	Defaultnone

Returns
Boolean T

Example

```
- af.reset("timemed freqmed")
Reset 3 method(s), 3 remaining
T
- af.reset(1,3)
Reset 2 method(s), 3 remaining
T
-
```

---

autoflag.resetall - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Disables all flagging methods

Synopsis
resetall()

Description

This function disables all methods previously set up by calls to set*method*().

Returns
Boolean T

Example

```
- af.resetall()
All methods reset
T
-
```

---

autoflag.detach - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Detaches from measurement set

Synopsis
detach()

Description

This function detaches the flagger tool from a measurement set and closes the MS.

Returns
Boolean T

Example

```
- af.detach()
T
-
```

---

autoflag.done - Function

Package	synthesis
Module	autoflag
Tool	autoflag

Destroys the autoflag tool

Synopsis
done()

Description

This function destroys the autoflag tool.

Returns
Boolean T

Example

```
- af.done()
T
-
```