

CASA Task Reference Manual

CASA Group, eds

March 30, 2016

Copyright ©2016 AUI
CASA Task Reference Manual.

The CASA Task Reference Manual contains the documentation on the task-based functionality within the system. There are three broad packages:

- General- modules that are of general use for astronomical processing
- Synthesis - modules needed for processing synthesis data
- Utility - non-astronomy specific functionality
- Third Party - modules that interface to 3rd party packages

Contents

0.1	Tasks - Module	5
0.1.1	accum	6
0.1.2	applycal	11
0.1.3	asdmsummary	18
0.1.4	bandpass	20
0.1.5	blcal	27
0.1.6	boxit	33
0.1.7	browsetable	36
0.1.8	calstat	39
0.1.9	caltabconvert	41
0.1.10	clean	43
0.1.11	clearcal	61
0.1.12	clearplot	63
0.1.13	clearstat	64
0.1.14	concat	65
0.1.15	conjugatevis	69
0.1.16	csvclean	71
0.1.17	cvel	76
0.1.18	cvel2	83
0.1.19	deconvolve	90
0.1.20	delmod	93
0.1.21	exportasdm	95
0.1.22	exportfits	98
0.1.23	exportuvfits	101
0.1.24	feather	105
0.1.25	find	108
0.1.26	fixplanets	109
0.1.27	fixvis	112
0.1.28	flagcmd	115
0.1.29	flagdata	131
0.1.30	flagmanager	159
0.1.31	fluxscale	161
0.1.32	ft	167
0.1.33	gaincal	171
0.1.34	gencal	178

0.1.35	hanningsmooth	186
0.1.36	imcollapse	191
0.1.37	imcontsub	195
0.1.38	imfit	197
0.1.39	imhead	206
0.1.40	imhistory	214
0.1.41	immath	216
0.1.42	immoments	224
0.1.43	impbcor	229
0.1.44	importatca	233
0.1.45	importasap	235
0.1.46	importasdm	237
0.1.47	importevla	244
0.1.48	importfits	251
0.1.49	importfitsidi	253
0.1.50	importgmrt	255
0.1.51	importmiriad	257
0.1.52	importuvfits	259
0.1.53	importvla	261
0.1.54	imrebin	265
0.1.55	imreframe	269
0.1.56	imregrid	271
0.1.57	imsmooth	280
0.1.58	imstat	286
0.1.59	imsubimage	294
0.1.60	imtrans	298
0.1.61	imval	301
0.1.62	imview	306
0.1.63	initweights	311
0.1.64	listcal	314
0.1.65	listhistory	319
0.1.66	listfits	320
0.1.67	listobs	321
0.1.68	listpartition	326
0.1.69	listsdm	329
0.1.70	listvis	331
0.1.71	makemask	337
0.1.72	mosaic	344
0.1.73	msview	352
0.1.74	mstransform	355
0.1.75	msuvbin	368
0.1.76	oldhanningsmooth	372
0.1.77	oldsplit	374
0.1.78	plotants	380
0.1.79	plotbandpass	382
0.1.80	plotcal	386

0.1.81	plotms	391
0.1.82	plotuv	402
0.1.83	plotweather	406
0.1.84	partition	408
0.1.85	polcal	415
0.1.86	predictcomp	422
0.1.87	impv	426
0.1.88	rmfit	431
0.1.89	rmtables	436
0.1.90	sdaverage	437
0.1.91	sdbaseline	443
0.1.92	sdbaseline2	453
0.1.93	sdcal	459
0.1.94	sdcal2	467
0.1.95	sdcoadd	477
0.1.96	sdfit	480
0.1.97	sdfalg	488
0.1.98	sdfalgmanager	495
0.1.99	sdgrid	497
0.1.100	sdimaging	503
0.1.101	sdimprocess	510
0.1.102	sdlist	514
0.1.103	sdmath	516
0.1.104	sdplot	522
0.1.105	sdreduce	532
0.1.106	sdsave	543
0.1.107	sdscale	548
0.1.108	sdsstat	551
0.1.109	setjy	559
0.1.110	simalma	570
0.1.111	simobserve	576
0.1.112	simanalyze	583
0.1.113	slsearch	588
0.1.114	smoothcal	591
0.1.115	specfit	593
0.1.116	specsmooth	603
0.1.117	splattotable	609
0.1.118	split	611
0.1.119	spxfit	617
0.1.120	statwt	623
0.1.121	tclean	629
0.1.122	testconcat	632
0.1.123	tsdbaseline	634
0.1.124	tsdcal	643
0.1.125	tsdfit	651
0.1.126	tsdsmooth	657

0.1.127 uvcontsub	661
0.1.128 uvcontsub3	666
0.1.129 uvmodelfit	670
0.1.130 uvsub	675
0.1.131 viewer	677
0.1.132 wvrgcal	680
0.1.133 virtualconcat	684
0.1.134 vishead	687
0.1.135 visstat	690
0.1.136 visstat2	695
0.1.137 widebandpbcor	701
0.1.138 widefield	706
Tasks-Module.html	

0.1 Tasks - Module

CASA Tasks

accum-task.html

0.1.1 accum

Requires:

Synopsis

Accumulate incremental calibration solutions into a calibration table

Description

Accum will interpolate and extrapolate a calibration table onto a new table that has a regularly-space time grid.

The first run of accum defines the time grid and fills this table with the results from the input table.

Subsequent use of accum will combine additional calibration tables onto the same grid of the initial accum table to obtain an output accum table. See below for concrete examples.

Accum tables are similar to CL tables in AIPS Incremental tables are similar to SN tables in AIPS

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
tablein	Input cumulative calibration table; use " on first run allowed: string Default:
incrtable	Input incremental calibration table to add allowed: string Default:
caltable	Output (cumulative) calibration table allowed: string Default:
field	List of field names to process from tablein allowed: stringArray Default:
calfield	List of field names to use from incrtable. allowed: stringArray Default:
interp	Interpolation mode to use for resampling incrtable solutions allowed: string Default: linear
accumtime	Time-interval when create cumulative table allowed: any Default: variant 1.0
spwmap	Spectral window combinations to apply allowed: intArray Default: -1

Returns

void

Example

Accum will interpolate and extrapolate a temporal calibration table onto a new table that has a regularly-space time grid.

The first run of accum defines the time grid and fills this table with the results from the input table.

Subsequent use of accum will combine additional calibration tables onto the same grid of the initial accum table to obtain an output accum table. See below for a concrete example.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none.  example: vis='ngc5921.ms'
tablein -- Input cumulative calibration table.
      default: '' means none
      On first execution of accum, tablein=''
      and accumtime is used to generate tablein with
      the specified time gridding.
accumtime -- The time separation when making tablein.
      default: 1.0 (1 second). This time should not be
      less than the visibility sampling time, but should
      be less than about 30% of a typical scan length.
incrtable -- The calibration data to be interpolated onto the
      tablein file.
      default: ''. Must be specified
caltable -- The output cumulated calibration file.
      default: '' means use tablein as the output file

field -- Select field(s) from tablein to process.
      ['go listobs' to obtain the list id's or names]
      default: '' = all fields
      If field string is a non-negative integer, it is assumed to
      be a field index otherwise, it is assumed to be a field name
      field='0~2'; field ids 0,1,2
      field='0,4,5~7'; field ids 0,4,5,6,7
      field='3C286,3C295'; field named 3C286 and 3C295
      field = '3,4C*'; field id 3, all names starting with 4C
calfield -- Select field(s) from incrtable to process.
      default: '' = all fields
interp -- Interpolation type (in time[,freq]) to use for each gaintable.
      When frequency interpolation is relevant (B, Df, Xf),
      separate time-dependent and freq-dependent interp
      types with a comma (freq_after_ the comma).
      Specifications for frequency are ignored when the
      calibration table has no channel-dependence.
      Time-dependent interp options ending in 'PD' enable a
      "phase delay" correction per spw for non-channel-dependent
      calibration types.
      For multi-obsId datasets, 'perobs' can be appended to
```

```

the time-dependent interpolation specification to
enforce obsId boundaries when interpolating in time.
default: '' --> 'linear,linear' for all gaintable(s)
example: interp='nearest' (in time, freq-dep will be
                           linear, if relevant)
        interp='linear,cubic' (linear in time, cubic
                              in freq)
        interp='linearperobs,spline' (linear in time
                                      per obsId,
                                      spline in freq)
        interp=',spline' (spline in freq; linear in
                          time by default)
        interp=['nearest,spline','linear'] (for multiple gaintables)
Options: Time: 'nearest', 'linear'
        Freq: 'nearest', 'linear', 'cubic', 'spline'
spwmap -- Spectral windows combinations to form for gaintable(s)
default: [] (apply solutions from each spw to that spw only)
Example: spwmap=[0,0,1,1] means apply the caltable solutions
        from spw = 0 to the spw 0,1 and spw 1 to spw 2,3.
        spwmap=[[0,0,1,1],[0,1,0,1]] (for multiple gaintables)
async -- Run task in a separate process
        default: False; example: async=True

```

Examples:

Create an accum table with 10-sec sampling, filling it with the calibration in 'first_cal' with the desired interpolation.

```

taskname = 'accum'
default()
vis = 'mydata.ms'
tablein = ''
accumtime = 10
incrtable = 'first_cal'
caltable = 'accum1_cal'
accum()

```

If you plot 'accum1_cal' with plotcal, you can see how the incrtable was interpolated.

Continue accumulating calibrations in accum1_cal from 'second_cal'

```

taskname = 'accum'
default()
vis = 'mydata.ms'
tablein = 'accum1_cal'

```

```
incrtable = 'second_cal'  
caltable = 'accum1_cal'  
accum()
```

[applycal-task.html](#)

0.1.2 applycal

Requires:

Synopsis

Apply calibrations solutions(s) to data

Description

Applycal reads the specified gain calibration tables, applies them to the (raw) data column (with the specified selection), and writes the calibrated results into the corrected column. This is done in one step, so all available calibration must be specified. Applycal will overwrite existing corrected data.

Standard data selection is supported. See `help par.selectdata` for more information.

One or more calibration tables (both temporal, frequency, polarization calibrations) can be specified in the `gaintable` parameter. The calibration values associated with a restricted list of fields can also be selected for each table.

See task `accum` for instructions on forming calibration incrementally. See task `split` for saving corrected data in another visibility file.

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
field	Select field using field id(s) or field name(s) allowed: string Default:
spw	Select spectral window/channels allowed: string Default:
intent	Select observing intent allowed: string Default:
selectdata	Other data selection parameters allowed: bool Default: True
timerange	Select data based on time range allowed: string Default:
uvrange	Select data within uvrange (default units meters) allowed: any Default: variant
antenna	Select data based on antenna/baseline allowed: string Default:
scan	Scan number range allowed: string Default:
observation	Select by observation ID(s) allowed: any Default: variant
msselect	Optional complex data selection (ignore for now) allowed: string Default:
docallib	Use callib or traditional cal apply parameters allowed: bool Default: False
callib	Cal Library filename allowed: string Default:
gaintable	Gain calibration table(s) to apply on the fly allowed: stringArray Default:
gainfield	Select a subset of calibrators from gaintable(s) allowed: stringArray Default: 12
interp	Interp type in time[,freq], per gaintable. default==linear,linear allowed: stringArray Default:
spwmap	Spectral windows combinations to form for gaintables(s) allowed: intArray Default:
calwt	Calibrate data weights per gaintable.

Example

Applycal reads the specified gain calibration tables or cal library, applies them to the (raw) data column (with the specified selection), and writes the calibrated results into the corrected column. This is done in one step, so all available calibration tables must be specified.

Applycal will overwrite existing corrected data, and will flag data for which there is no calibration available.

In the traditional interface (docallib=False), all calibration tables (both temporal, frequency, polarization calibrations) are specified in the gaintable parameter. The calibration values associated with a restricted list of fields can also be selected for each table in gainfield.

As of CASA v4.2, docallib=True provides specification of an ensemble of calibration tables and directives via a cal library file.

See task accum for instructions on forming calibration incrementally. See task split for copying out any portion of the data and selected columns to a new visibility file.

Keyword arguments:

vis -- Name of input visibility file
 default: < none>; example: vis='ngc5921.ms'

--- Data Selection: the data to which the calibration will be applied
 (see help par.selectdata for more detailed information)

field -- Select field id(s) or field name(s) to apply calibration.
 [run listobs to obtain the list id's or names]
 default: ''=all fields
 If field's string is an integer >=0, it is assumed to be an index
 otherwise, it is assumed to be a field name
 field='0~2'; field ids 0,1,2
 field='0,4,5~7'; field ids 0,4,5,6,7
 field='3C286,3C295'; fields named 3C286 and 3C295
 field = '3,4C*'; field id 3, all names starting with 4C
spw -- Select spectral window/channels

```

    type 'help par.selection' for more examples.
    spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
    spw='<2'; spectral windows less than 2 (i.e. 0,1)
    spw='0:5~61'; spw 0, channels 5 to 61, INCLUSIVE
    spw='*:5~61'; all spw with channels 5 to 62
    spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.
    spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
    spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60
        NOTE ';' to separate channel selections
    spw='0:0~10^2,1:20~30^5'; spw 0, channels 0,2,4,6,8,10,
        spw 1, channels 20,25,30
intent -- Select observing intent
    default: '' (no selection by intent)
    intent='*BANDPASS*' (selects data labelled with
        BANDPASS intent)
selectdata -- Other data selection parameters
    default: True
timerange -- Select data based on time range:
    default = '' (all); examples,
    timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
    Note: if YYYY/MM/DD is missing, date defaults to first day in
        data set
    timerange='09:14:0~09:54:0' picks 40 min on first day
    timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min on next day
    timerange='09:44:00' data within one integration of time
    timerange='>10:24:00' data after this time
uvrange -- Select data within uvrange (default units meters)
    default: '' (all); example:
    uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
    uvrange='>4klambda'; uvranges greater than 4 kilolambda
antenna -- Select data based on antenna/baseline
    default: '' (all)
    If antenna's string is an integer >=0, it is taken to be an index
        otherwise, it is assumed to be an antenna name
    antenna='5&6'; baseline between antenna index 5 and index 6.
    antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
    antenna='5&6;7&8'; baseline 5-6 and 7-8
    antenna='5'; all baselines with antenna index 5
    antenna='05'; all baselines with antenna name 05--vla antenna 5.
    antenna='5,6,10'; all baselines with antennas 5,6 and 10
scan -- Scan number range
observation -- Select by observation ID(s).
    default: '' = all
    example: '0~3,6'
mselect -- Optional complex data selection (ignore for now)

```

```

--- Calibration files to apply
docallib -- Control means of specifying the caltables:
    default: False ==> Use gaintable,gainfield,interp,spwmap,calwt
        If True, specify a file containing cal library in callib
callib -- If docallib=True, specify a file containing cal
    library directives
gaintable -- Gain calibration table(s) to apply
    default: '' (none);
    examples: gaintable='ngc5921.gcal'
        gaintable=['n5921.ampcal','n5921.phcal','n5921.bpass']
    All gain table types: 'G', GSPLINE, 'T', 'B', 'BPOLY', 'D's'
    can be applied.
gainfield -- Select a subset of calibrators from each gaintable
    default:'' ==> all sources in table;
    'nearest' ==> nearest (on sky) available field in table
    otherwise, same syntax as field
    example: gainfield='0~3'
        gainfield=['0~3','4~6'] (for multiple gaintables)
interp -- Interpolation type (in time[,freq]) to use for each gaintable.
    When frequency interpolation is relevant (bandpass solutions,
    frequency-dependent polcal solutions, ALMA Tsys)
    separate time-dependent and freq-dependent interp
    types with a comma (freq_after_ the comma).
    Specifications for frequency are ignored when the
    calibration table has no channel-dependence.
    Time-dependent interp options ending in 'PD' enable a
    "phase delay" correction per spw for non-channel-dependent
    calibration types.
    For multi-obsId datasets, 'perobs' can be appended to
    the time-dependent interpolation specification to
    enforce obsId boundaries when interpolating in time.
    Add 'flag' to the freq-dependent interpolation options
    to enforce channel-dependent flagging (rather than
    interpolation/extrapolation).
    default: '' --> 'linear,linear' for all gaintable(s)
    example: interp='nearest' (in time, freq-dep will be
        linear, if relevant)
        interp='linear,cubic' (linear in time, cubic
            in freq)
        interp='linearperobs,splineflag' (linear in time
            per obsId,
            spline in
            freq with
            channelized
            flagging)
        interp=',spline' (spline in freq; linear in

```



```

                                time by default)
                                interp=['nearest,spline','linear'] (for multiple gaintables)
Options: Time: 'nearest', 'linear', 'nearestPD', 'linearPD'
          Freq: 'nearest', 'linear', 'cubic', 'spline',
                'nearestflag', 'linearflag', 'cubicflag', 'splineflag',

spwmap -- Spectral windows combinations to form for gaintable(s)
default: [] (apply solutions from each spw to that spw only)
Example: spwmap=[0,0,1,1] means apply the caltable solutions
          from spw = 0 to the spw 0,1 and spw 1 to spw 2,3.
          spwmap=[[0,0,1,1],[0,1,0,1]] (for multiple gaintables)

```

Complicated example:

```

gaintable=['tab1','tab2','tab3']
gainfield='3C286'
interp=['linear','nearest']
spwmap=[[ ],[0,0,2]]

```

This means: apply 3 cal tables, selecting only solutions for 3C286 from tab1 (but all fields from tab2 and tab3, indicated by no gainfield entry for these files). Linear interpolation (in time) will be used for 'tab1' and 'tab3' (default); 'tab2' will use nearest. For the 'tab2', the calibration spws map will be mapped to the data spws according to 0->0, 0->1, 2->2. (I.e., for data spw=0 and 2, the spw mapping is one to one, but data spw 1 will be calibrated by solutions from spw 0.)

```

parang -- If True, apply the parallactic angle correction. FOR ANY
          POLARIZATION CALIBRATION AND IMAGING, parang = True
          default: False
calwt -- Calibrate weights along with data for each gaintable
default: True (for all specified gaintables)
examples: calwt=False (for all specified gaintables)
          calwt=[True,False,True] (specified per gaintable)
applymode -- Calibration apply mode:
            '='calflag' (default) calibrate data and apply flags from solutions
            'trial' report on flags from solutions, dataset entirely unchanged
            'flagonly' apply flags from solutions only, data not calibrated
            'calonly' calibrate data only, flags from solutions NOT
                      applied (use with extreme caution!)
            'calflagstrict' or 'flagonlystrict' same as above
                      except flag spws for which calibration is
                      unavailable in one or more tables (instead of
                      allowing them to pass uncalibrated and
                      unflagged)

```

```
flagbackup -- Back up the state of the flags before applying calibration
              default: True
async -- Run task in a separate process
              default: False; example: async=True
```

asdmsummary-task.html

0.1.3 asdmsummary

Requires:

Synopsis

Summarized description of an ASDM dataset.

Description

Given an ASDM directory, this task will print informations about the content of the dataset contained in that directory (down to the level of a subscan).

Arguments

Inputs	
asdm	Name of input ASDM directory
	allowed: string
	Default:

Returns

void

Example

The asdmsummary task prints to the CASA log a description of the content of an SDM dataset.

Example:

```
asdmsummary(sdm='10C-119_sb3070258_1.55628.42186299768')
```

Prints information about the requested ASDM dataset to the CASA logger.

Keyword argument:

```
asdm -- Name of input ASDM directory.  
example: sdm='10C-119_sb3070258_1.55628.42186299768'
```

bandpass-task.html

0.1.4 bandpass

Requires:

Synopsis

Calculates a bandpass calibration solution

Description

Determines the amplitude and phase as a function of frequency for each spectral window containing more than one channel. Strong sources (or many observations of moderately strong sources) are needed to obtain accurate bandpass functions. The two solution choices are: Individual antenna/channel solutions 'B'; and a polynomial fit over the channels 'BPOLY'. The 'B' solutions can be determined at any specified time interval, and is recommended in most applications.

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
caltable	Name of output gain calibration table allowed: string Default:
field	Select field using field id(s) or field name(s) allowed: string Default:
spw	Select spectral window/channels allowed: string Default:
intent	Select observing intent allowed: string Default:
selectdata	Other data selection parameters allowed: bool Default: True
timerange	Select data based on time range allowed: string Default:
uvrange	Select data within uvrange (default units meters) allowed: any Default: variant
antenna	Select data based on antenna/baseline allowed: string Default:
scan	Scan number range allowed: string Default:
observation	Select by observation ID(s) allowed: any Default: variant
msselect	Optional complex data selection (ignore for now) allowed: string Default:
solint	Solution interval in time[,freq] allowed: any Default: variant inf
combine	Data axes which to combine for solve (obs, scan, spw, and/or field) allowed: string Default: scan
refant	Reference antenna name(s) allowed: string Default:
minblperant	Minimum baselines _per antenna_ required for solve allowed: int Default: 4
minsnr	Reject solutions below this SNR (only applies for band-type = B) allowed: double Default: 3.0

Example

Determines the amplitude and phase as a function of frequency for each spectral window containing more than one channel. Strong sources (or many observations of moderately strong sources) are needed to obtain accurate bandpass functions. The two solution choices are: Individual antenna/channel solutions 'B'; and a polynomial fit over the channels 'BPOLY'. The 'B' solutions can be determined at any specified time interval, and is recommended if each channel has good signal-to-noise. Other, 'BPOLY' is recommended.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'
caltable -- Name of output bandpass calibration table
           default: none; example: caltable='ngc5921.bcal'
```

--- Data Selection (see help par.selectdata for more detailed information)

```
field -- Select field using field id(s) or field name(s).
        [run listobs to obtain the list id's or names]
        default: ''=all fields
        If field string is a non-negative integer, it is assumed a field index
        otherwise, it is assumed a field name
        field='0~2'; field ids 0,1,2
        field='0,4,5~7'; field ids 0,4,5,6,7
        field='3C286,3C295'; field named 3C286 and 3C295
        field = '3,4C*'; field id 3, all names starting with 4C
spw -- Select spectral window/channels
      default: ''=all spectral windows and channels
      spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
      spw='<2'; spectral windows less than 2 (i.e. 0,1)
      spw='0:5~61'; spw 0, channels 5 to 61
      spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.
      spw='0~2:2:6'; spw 0,1,2 with channels 2 through 6 in each.
      spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60
      NOTE: ';' to separate channel selections
      spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,
      spw 1, channels 20-30, and spw 2, channels, 1,2 and 3
intent -- Select observing intent
         default: '' (no selection by intent)
         intent='*BANDPASS*' (selects data labelled with
                              BANDPASS intent)
```

```

selectdata -- Other data selection parameters
    default: True
timerange -- Select data based on time range:
    default = '' (all); examples,
    timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
    Note: if YYYY/MM/DD is missing defaults to first day in data set
    timerange='09:14:0~09:54:0' picks 40 min on first day
    timerange= '25:00:00~27:30:00' picks 1 hr to 3 hr 30min on next day
    timerange='09:44:00' data within one integration of time
    timerange='>10:24:00' data after this time
uvrange -- Select data within uvrange (default meters)
    default: '' (all); example:
    uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
    uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
antenna -- Select data based on antenna/baseline
    default: '' (all)
    If antenna string is a non-negative integer, it is assumed an antenna index
    otherwise, it is assumed as an antenna name
    antenna='5&6'; baseline between antenna index 5 and index 6.
    antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
    antenna='5&6;7&8'; baseline 5-6 and 7-8
    antenna='5'; all baselines with antenna 5
    antenna='5,6,10'; all baselines with antennas 5, 6 and 10
scan -- Select data based on scan number - New, under developement
    default: '' (all); example: scan='>3'
observation -- Observation ID(s).
    default: '' = all
    example: '0~2,4'
msselect -- Optional complex data selection (ignore for now)

--- Solution parameters
solint -- Solution interval in time (units optional), and (optionally)
    in frequency. Frequency pre-averaging can be
    specified after a comma in units of channels or Hz.
    If nothing is specified for frequency, no freq pre-averaging
    will be done.
default: 'inf' (~infinite, up to boundaries controlled by combine,
    with no pre-averaging in frequency)
Options for time: 'inf' (~infinite), 'int' (per integration), any float
    or integer value with or without units
Options for freq: an integer with 'ch' suffix will enforce
    pre-averaging by the specified number
    of channels.
    a numeric value suffixed with frequency
    units (e.g., 'Hz','kHz','MHz') will enforce
    pre-averaging by an integral number of

```


channels amounting to no more than the specified bandwidth

examples: solint='1min'; solint='60s', solint=60 --> 1 minute
solint='0s'; solint=0; solint='int' --> per integration
solint='-1s'; solint='inf' --> ~infinite, up to boundaries enforced by combine
solint='inf,8Mhz' --> ~infinite in time, with 8MHz pre-average in freq
solint='int,32ch' --> per-integration in time, with 32-channel pre-average in freq

combine -- Data axes to combine for solving
default: 'scan' --> solutions will break at obs, field, and spw boundaries but may extend over multiple scans (per obs, field and spw) up to solint.
Options: '', 'obs', 'scan', 'spw', 'field', or any comma-separated combination in a single string
example: combine='scan,spw' --> extend solutions over scan boundaries (up to the solint), and combine spws for solving

refant -- Reference antenna name(s); a prioritized list may be specified
default: '' (no reference antenna)
example: refant='13' (antenna with index 13)
refant='VA04' (VLA antenna #4)
refant='EA02,EA23,EA13' (EVLA antenna EA02, use EA23 and EA13 as alternates if/when EA02 drops out)

Use 'go listobs' for antenna listing

minblperant -- Minimum number of baselines required per antenna for each solve
Antennas with fewer baselines are excluded from solutions. Amplitude solutions with fewer than 4 baselines, and phase solutions with fewer than 3 baselines are only trivially constrained, and are no better than baseline-based solutions.
default: 4
example: minblperant=10 => Antennas participating on 10 or more baselines are included in the solve

minsnr -- Reject solutions below this SNR (only applies for bandtype = B)
default: 3.0

solnorm -- Normalize bandpass amplitudes and phase for each spw, pol, ant, and timestamp
default: False (no normalization)

bandtype -- Type of bandpass solution (B or BPOLY)
default: 'B'; example: bandtype='BPOLY'
'B' does a channel by channel solution for each specified spw.
'BPOLY' is somewhat experimental. It will fit an nth order polynomial for the amplitude and phase

as a function of frequency. Only one fit is made for all specified spw, and edge channels should be omitted.

Use taskname=plotcal in order to compare the results from B and BPOLY.

fillgaps -- Fill flagged solution channels by interpolation
 default: 0 (don't interpolate)
 example: fillgaps=3 (interpolate gaps 3 channels wide and narrower)

degamp -- Polynomial degree for BPOLY amplitude solution
 default: 3; example: degamp=2

degphase -- Polynomial degree for BPOLY phase solution
 default: 3; example: degphase=2

visnorm -- Normalize data prior to BPOLY solution
 default: False; example: visnorm=True

maskcenter -- Number of channels to avoid in center of each band
 default: 0; example: maskcenter=5 (BPOLY only)

maskededge -- Fraction of channels to avoid at each band edge (in %)
 default: 5; example: maskededge=3 (BPOLY only)

append -- Append solutions to the (existing) table. Appended solutions must be derived from the same MS as the existing caltable, and solution spws must have the same meta-info (according to spw selection and solint) or be non-overlapping.
 default: False; overwrite existing table or make new table

--- Other calibrations to apply on the fly before determining bandpass solution

docallib -- Control means of specifying the caltables:
 default: False ==> Use gaintable, gainfield, interp, spwmap, calwt
 If True, specify a file containing cal library in callib

callib -- If docallib=True, specify a file containing cal library directives

gaintable -- Gain calibration table(s) to apply
 default: '' (none);
 examples: gaintable='ngc5921.gcal'
 gaintable=['ngc5921.ampcal', 'ngc5921.phcal']

gainfield -- Select a subset of calibrators from gaintable(s)
 default: '' ==> all sources in table;
 'nearest' ==> nearest (on sky) available field in table
 otherwise, same syntax as field
 example: gainfield='0~3'
 gainfield=['0~3', '4~6']

interp -- Interpolation type (in time[,freq]) to use for each gaintable.
 When frequency interpolation is relevant (B, Df, Xf),
 separate time-dependent and freq-dependent interp

```

types with a comma (freq_after_ the comma).
Specifications for frequency are ignored when the
calibration table has no channel-dependence.
Time-dependent interp options ending in 'PD' enable a
"phase delay" correction per spw for non-channel-dependent
calibration types.
For multi-obsId datasets, 'perobs' can be appended to
the time-dependent interpolation specification to
enforce obsId boundaries when interpolating in time.
default: '' --> 'linear,linear' for all gaintable(s)
example: interp='nearest'      (in time, freq-dep will be
                                linear, if relevant)
        interp='linear,cubic'  (linear in time, cubic
                                in freq)
        interp='linearperobs,spline' (linear in time
                                per obsId,
                                spline in freq)
        interp=',spline'      (spline in freq; linear in
                                time by default)
        interp=['nearest,spline','linear'] (for multiple gaintables)
Options: Time: 'nearest', 'linear'
        Freq: 'nearest', 'linear', 'cubic', 'spline'
spwmap -- Spectral windows combinations to form for gaintable(s)
default: [] (apply solutions from each spw to that spw only)
Example: spwmap=[0,0,1,1] means apply the caltable solutions
        from spw = 0 to the spw 0,1 and spw 1 to spw 2,3.
        spwmap=[[0,0,1,1],[0,1,0,1]]
parang -- If True, apply the parallactic angle correction (required
for polarization calibration)
default: False

```

`blcal-task.html`

0.1.5 blcal

Requires:

Synopsis

Calculate a baseline-based calibration solution (gain or bandpass)

Description

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
caltable	Name of output gain calibration table allowed: string Default:
field	Select field using field id(s) or field name(s) allowed: string Default:
spw	Select spectral window/channels allowed: string Default:
intent	Select observing intent allowed: string Default:
selectdata	Other data selection parameters allowed: bool Default: True
timerange	Select data based on time range allowed: string Default:
uvrange	Select data within uvrange (default units meters) allowed: any Default: variant
antenna	Select data based on antenna/baseline allowed: string Default:
scan	Scan number range allowed: string Default:
observation	Select by observation ID(s) allowed: any Default: variant
msselect	Optional complex data selection (ignore for now) allowed: string Default:
solint	Solution interval allowed: any Default: variant inf
combine	Data axes which to combine for solve (obs, scan, spw, and/or field) allowed: string Default: scan
freqdep	Solve for frequency dependent solutions allowed: bool Default: False
calmode	Type of solution" ('ap', 'p', 'a') allowed: string Default: ap
solnorm	Normalize average solution amplitudes to 1.0 allowed: bool Default: False
gaintable	Gain calibration table(s) to apply on the fly

Example

This task determines a baseline by baseline gain (time) or bandpass (freq) for all baseline pairs in the data set. For the usual antenna-based calibration of interferometric data, this task gaincal is recommended, even with only one to three baselines. For arrays with closure errors, use blcal

Keyword arguments:

vis -- Name of input visibility file

default: none; example: vis='ngc5921.ms'

caltable -- Name of output Gain calibration table

default: none; example: caltable='ngc5921.gcal'

--- Data Selection (see help par.selectdata for more detailed information)

field -- Select field using field id(s) or field name(s).

[run listobs to obtain the list id's or names]

default: ''=all fields

If field string is a non-negative integer, it is assumed a field index otherwise, it is assumed a field name

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

field='3C286,3C295'; field named 3C286 and 3C295

field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels

default: ''=all spectral windows and channels

spw='0~2,4'; spectral windows 0,1,2,4 (all channels)

spw='<2'; spectral windows less than 2 (i.e. 0,1)

spw='0:5~61'; spw 0, channels 5 to 61

spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.

spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.

spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60

spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,

spw 1, channels 20-30, and spw 2, channels, 1,2 and 3

intent -- Select observing intent

default: '' (no selection by intent)

intent='*BANDPASS*' (selects data labelled with
BANDPASS intent)

selectdata -- Other data selection parameters

default: True

timerange -- Select data based on time range:

default = '' (all); examples,

timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'

Note: if YYYY/MM/DD is missing dat defaults to first day in data set
timerange='09:14:0~09:54:0' picks 40 min on first day
timerange= '25:00:00~27:30:00' picks 1 hr to 3 hr 30min on next day
timerange='09:44:00' data within one integration of time
timerange='>10:24:00' data after this time

uvrange -- Select data within uvrange (default units kilo-lambda)
default: '' (all); example:
uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
uvrange='>4klambda'; uvranges greater than 4 kilo lambda
uvrange='0~1000km'; uvrange in kilometers

antenna -- Select data based on antenna/baseline
default: '' (all)
If antenna string is a non-negative integer, it is assumed an antenna index
otherwise, it is assumed as an antenna name
antenna='5&6'; baseline between antenna index 5 and index 6.
antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
antenna='5&6;7&8'; baseline 5-6 and 7-8
antenna='5'; all baselines with antenna 5
antenna='5,6,10'; all baselines with antennas 5 and 6

scan -- Scan number range - New, under development

observation -- Observation ID(s).
default: '' = all
example: '0~2,4'

msselect -- Optional complex data selection (ignore for now)

solint -- Solution interval (units optional)
default: 'inf' (~infinite, up to boundaries controlled by combine);
Options: 'inf' (~infinite), 'int' (per integration), any float
or integer value with or without units
examples: solint='1min'; solint='60s', solint=60 --> 1 minute
solint='0s'; solint=0; solint='int' --> per integration
solint='-1s'; solint='inf' --> ~infinite, up to boundaries
enforced by combine

combine -- Data axes to combine for solving
default: 'scan' --> solutions will break at obs, field, and spw boundaries,
but may extend over multiple scans (per obs, field, and spw) up
to solint.
Options: '', 'obs', 'scan', 'spw', 'field', or any comma-separated
combination in a single string
example: combine='scan,spw' --> extend solutions over scan boundaries
(up to the solint), and combine spws for solving

freqdep -- Solve for frequency dependent solutions
default: False (gain; True=bandpass); example: freqdep=True

calmode -- Type of solution
default: 'ap' (amp and phase); example: calmode='p'
Options: 'p', 'a', 'ap'

```

solnorm -- Normalize solutions. For freqdep=F, this is a global (per-spw)
         normalization of amplitudes (only). For freqdep=T, each baseline
         solution spectrum is separately normalized by its (complex) mean.
         default: False (no normalization)

gaintable -- Gain calibration table(s) to apply
            default: '' (none);
            examples: gaintable='ngc5921.gcal'
                     gaintable=['ngc5921.ampcal','ngc5921.phcal']
gainfield -- Select a subset of calibrators from gaintable(s)
            default:'' ==> all sources in table;
            'nearest' ==> nearest (on sky) available field in table
            otherwise, same syntax as field
            example: gainfield='0~3'
                     gainfield=['0~3','4~6']
interp -- Interpolation type (in time[,freq]) to use for each gaintable.
         When frequency interpolation is relevant (B, Df, Xf),
         separate time-dependent and freq-dependent interp
         types with a comma (freq_after_ the comma).
         Specifications for frequency are ignored when the
         calibration table has no channel-dependence.
         Time-dependent interp options ending in 'PD' enable a
         "phase delay" correction per spw for non-channel-dependent
         calibration types.
         For multi-obsId datasets, 'perobs' can be appended to
         the time-dependent interpolation specification to
         enforce obsId boundaries when interpolating in time.
         default: '' --> 'linear,linear' for all gaintable(s)
         example: interp='nearest' (in time, freq-dep will be
                                   linear, if relevant)
                 interp='linear,cubic' (linear in time, cubic
                                   in freq)
                 interp='linearperobs,spline' (linear in time
                                   per obsId,
                                   spline in freq)
                 interp=',spline' (spline in freq; linear in
                                   time by default)
                 interp=['nearest,spline','linear'] (for multiple gaintables)
         Options: Time: 'nearest', 'linear'
                  Freq: 'nearest', 'linear', 'cubic', 'spline'
spwmap -- Spectral windows combinations to form for gaintable(s)
        default: [] (apply solutions from each spw to that spw only)
        Example: spwmap=[0,0,1,1] means apply the caltable solutions
                  from spw = 0 to the spw 0,1 and spw 1 to spw 2,3.
                  spwmap=[[0,0,1,1],[0,1,0,1]]
parang -- If True, apply the parallactic angle correction (required

```



```
for polarization calibration)  
default: False
```

boxit-task.html

0.1.6 boxit

Requires:

Synopsis

Box regions in image above given threshold value.

Description

Returns a list of boxes, one for each contiguous set of pixels above the threshold value. If given "regionfile", outputs boxes in regionfile+'.rgn'

Arguments

Inputs		
imagename		Name of image to threshold allowed: string Default:
regionfile		Output region file allowed: string Default:
threshold	mJy	Threshold value. Must include units. allowed: doublemJy Default: 0.0
maskname		Output mask name (optional). allowed: string Default:
chanrange		Range of channel ids allowed: string Default:
polrange		Range of polarization ids allowed: string Default:
minsize		Minimum number of pixels for a boxable island allowed: int Default: 2
diag		Count diagonal connections? allowed: bool Default: False
boxstretch		Increase box sizes by this many pixels beyond thresholded pixels. allowed: int Default: 1
overwrite		Overwrite existing region file? allowed: bool Default: False

Returns

void

Example

This tool finds all 2-dimensional (RA/dec) regions in the given

4D (only!) image which are contiguous sets of pixels (islands) above the given] threshold. It creates a box for each island, a rectangular "cutout". The boxes are stored as regions in the output regionfile. Works on multi-plane images, but only boxes 2-D regions in each plane. (Doesn't create cubes/3D boxes.)

NOTE: THIS TASK WILL NOT WORK ON IMAGES THAT DO NOT HAVE 4 DIMENSIONS WHICH INCLUDE A DIRECTION COORDINATE, A SPECTRAL COORDINATE, AND A STOKES COORDINATE. If your image has, eg just a direction coordinate, you can add the required axes using `ia.adddegaxes` and remove them post-run with `imsubimage` with `dropdeg=T`.

```

imagenam -- Name of input images:
           default: none; example: imagenam='myimage.image'
regionfile -- Name of output region file (adds extension .rgn).
             default: none; if not given uses imagenam+'.rgn'
threshold -- value (with units) to use for island threshold.
             default: 0.0.
maskname -- Optional output mask name.
            default: '' (do not write mask image)
chanrange -- Range of channel ids
             default: '' (find boxes for all channels)
             example: '5~7' (find boxes for channel 5,6,7)
polrange -- Range of polarization ids
            default: '' (find boxes for all polarizations)
            example: '0~1' (find boxes for polarization 0,1)
minsize -- minimum size of island to get a box (in number of pixels)
           default: 2
diag -- count diagonal connections as same island or not
        default: False
boxstretch -- number of pixels to increase outward size of each box; can
              range from -1 to 5.
              default: 1
overwrite -- Overwrite existing region file and/or mask?
             default: False. If False, gives warning if file exists.

```

browsetable-task.html

0.1.7 browsetable

Requires:

Synopsis

Browse a table (MS, calibration table, image)

Description

Arguments

Inputs	
tablename	Name of input table allowed: string Default:
mightedit	Warning: the GUI seems to ignore whether the table tool is opened read-only - just be careful, esp. if filtering. allowed: bool Default: False
sortlist	Columns to sort by (ascending) allowed: any Default: variant
taql	TaQL query string for prefiltering the table. allowed: string Default:
skipcols	Columns to omit allowed: any Default: variant

Returns

void

Example

This task brings up a browser that can open and display any CASA table. The tablename can be specified at startup, or any table can be loaded after the browser comes up.

Parameters:

```
tablename -- Name of table file on disk (vis, calibration table, image)
            default: none; example: tablename='ngc5921.ms'
mightedit -- If True disable the filtering options (below) and allow
            editing the table. Warning: the GUI appears to ignore
            whether the table tool is opened read-only - just be
            aware that you should not edit filtered tables unless
            you know what you are doing.
sortlist  -- List of columns to sort by.
            default: [] (none)
taql      -- TaQL query string for prefiltering the table.
            default: "" (none); example: taql="ANTENNA2 < 6"
skipcols  -- Columns to NOT display.
            default: [] (none); example: skipcols='feed1, feed2'
```

Some comments on using browsetable (see cookbook also):

Most often you will browse a measurement set. Either specify the vis name as the tablename, or when the browser comes up,

click on <file> (upper left), then click on <open table>

If you want to look at sub-tables, use the tab table keywords along the left side to bring up a panel with the sub-tables listed (Fig 3.8), then choose (left-click) a table and View.

Note that one useful feature is that you can Edit any table and its contents. Use the Edit tab (to the right of the file tab). Be careful with this, and make a backup copy of the table before editing!

Use the Close Tables and Exit option from the Files menu to quit the casabrowser.

To get a plot of two table values, click on tools, then click on plot 2D. For example, to get a u-v plot, in the Plotter Option Gui,

```
set Rows: 0    to <Large Number>
X Axis:  UVW    Slice (set 0)
Y Axis:  UVW    Slice (set 1)
```

click 'Clear and Plot' on right.

For visibility plots

X Axis: TIME

Y Axis: DATA Slice Amplitude

click 'Clear and Plot' on right.

calstat-task.html

0.1.8 calstat

Requires:

Synopsis

Displays statistical information on a calibration table

Arguments

Outputs	
xstat	Statistical information for the calibration table allowed: any Default: variant
Inputs	
caltable	Name of input calibration table allowed: string Default:
axis	Which values to use allowed: string Default: amplitude
datacolumn	Which data column to use allowed: string Default: gain
useflags	Take flagging into account? (not implemented) allowed: bool Default: True

Returns

void

Example

This task returns statistical information about a column in a calibration table.

The following values are computed: mean value, sum of values, sum of squared values, median, median absolute deviation, quartile, minimum, maximum,

variance, standard deviation, root mean square.

Keyword arguments:

caltable -- Name of input calibration table
 default: '', example: vis='ggtau.1mm.amp.gcal'

axis -- Which data to analyze. The possible values are 'amp', 'amplitude', 'phase',
 'real', 'imag', 'imaginary'. Also, the name of any real valued MS column can
 be given, e.g. TIME, POLY_COEFF_AMP, REF_ANT, ANTENNA1, FLAG, ...

 default: 'amplitude'
 axis='gain'

 The phase of a complex number is in radians in the range $[-\pi; \pi]$.

datacolumn -- Which data column to use if axis is 'amp', 'amplitude',
 'phase', 'real', 'imag' or 'imaginary'.
 default: 'gain'
 datacolumn='gain'

useflags -- Take MS flags into account (not implemented, this parameter
 has no effect!)
 default: False
 useflag=False
 useflag=True

If useflags=False, flagged values are included in the statistics.

If useflags=True, any flagged values are not used in the statistics.

caltabconvert-task.html

0.1.9 caltabconvert

Requires:

Synopsis

Convert old-style caltables into new-style caltables.

Description

This task converts old-style (up to CASA 3.3.0) caltables into new-style (CASA 3.4.0 and later) caltables. It is provided as a convenience and is strictly temporary. The information transferred should be enough for most calibration purposes. BPOLY and GSPLINE versions are not supported. Only simple bugs will be fixed. If there are other issues, it is suggested that a new-style caltable be created directly.

Arguments

Inputs	
caltabold	Name of the old-style caltable. allowed: string Default:
vis	Name of the visibility file (MS) associated with the old-style caltable. allowed: string Default:
ptype	Type of data in the new-format caltable ("complex" or "float"; default is "complex"). allowed: string Default: complex
caltabnew	Name of the new-style caltable. If not specified, the suffix ".new" is appended to the name of old-style caltable. allowed: string Default:

Returns

boolean

Example

This task converts old-style (up to CASA 3.3.0) caltables into new-style (CASA 3.4.0 and later) caltables. It is provided as a convenience and is strictly temporary. The information transferred should be enough for most calibration purposes. BPOLY and GSPLINE versions are not supported. Only simple bugs will be fixed. If there are other issues, it is suggested that a new-style caltable be created directly.

Arguments:

caltabold -- Name of the old-style caltable.
default: none
example: caltabold='gronk.g0'

vis -- Name of the visibility file (MS) associated with the old-style caltable.
default: none
example: vis='blurp.ms'

ptype -- Type of data in the new-format caltable.
default: "complex"; allowed values: "complex" or "float"
example: ptype="complex"

NB: The old-style caltables do not have this information, so it is imperative that users get it correct. "complex" refers to caltables that have complex gains (e.g., produced by gaincal, bpcal, etc.). "float" refers to caltables that real numbers such as delays (e.g., produced by gencal).

caltabnew -- Name of the new-style caltable.
default: "" --> the suffix ".new" is appended to the name of the old-style caltable
example: caltabold='gronk_new.g0'

`clean-task.html`

0.1.10 clean

Requires:

Synopsis

Invert and deconvolve images with selected algorithm

Description

Form images from visibilities. Handles continuum and spectral line cubes.

Arguments

Inputs		
vis	Name of input visibility file	
	allowed:	any
	Default:	variant
imagename	Pre-name of output images	
	allowed:	any
	Default:	variant
outlierfile	Text file with image names, sizes, centers for outliers	
	allowed:	string
	Default:	
field	Field Name or id	
	allowed:	any
	Default:	variant
spw	Spectral windows e.g. '0~3', " is all	
	allowed:	any
	Default:	variant
selectdata	Other data selection parameters	
	allowed:	bool
	Default:	True
timerange	Range of time to select from data	
	allowed:	any
	Default:	variant
uvrange	Select data within uvrange	
	allowed:	any
	Default:	variant
antenna	Select data based on antenna/baseline	
	allowed:	any
	Default:	variant
scan	Scan number range	
	allowed:	any
	Default:	variant
observation	Observation ID range	
	allowed:	any
	Default:	variant
intent	Scan Intent(s)	
	allowed:	any
	Default:	variant
mode	Spectral gridding type (mfs, channel, velocity, frequency)	
	allowed:	string
	Default:	mfs
resmooth	Re-restore the cube image to a common beam when True	
	allowed:	bool
	Default:	False
gridmode	Gridding kernel for FFT-based transforms, default="None"	

Returns

void

Example

The clean task has many options:

- 1) Make 'dirty' image and 'dirty' beam (psf)
- 2) Multi-frequency-continuum images or spectral channel imaging
- 3) Full Stokes imaging
- 4) Mosaicking of several pointings
- 5) Multi-scale cleaning
- 6) Widefield cleaning
- 7) Interactive clean boxing
- 8) Use starting model (eg from single dish)

```
vis -- Name(s) of input visibility file(s)
      default: none;
      example: vis='ngc5921.ms'
              vis=['ngc5921a.ms','ngc5921b.ms']; multiple MSes
imagenam -- Pre-name of output images:
      default: none; example: imagenam='m2'
      output images are:
          m2.image; cleaned and restored image
              With or without primary beam correction
          m2.psf; point-spread function (dirty beam)
          m2.flux; relative sky sensitivity over field
m2.flux.pbcoverage; relative pb coverage over field
                      (gets created only for ft='mosaic')
          m2.model; image of clean components
          m2.residual; image of residuals
          m2.interactive.mask; image containing clean regions
      To include outlier fields:
          imagenam=['n5921','outlier1','outlier2']
outlierfile --- Text file name which contains image names, sizes, field
                  centers (See 'HINTS ON CLEAN WITH FLANKING FIELDS' below
                  for the format of this outlier file.)
field -- Select fields to image or mosaic. Use field id(s) or name(s).
      ['go listobs' to obtain the list id's or names]
      default: ''= all fields
      If field string is a non-negative integer, it is assumed to
```

```

be a field index otherwise, it is assumed to be a
field name
    field='0~2'; field ids 0,1,2
    field='0,4,5~7'; field ids 0,4,5,6,7
    field='3C286,3C295'; field named 3C286 and 3C295
    field = '3,4C*'; field id 3, all names starting with 4C
    For multiple MS input, a list of field strings can be used:
    field = ['0~2','0~4']; field ids 0-2 for the first MS and 0-4
        for the second
    field = '0~2'; field ids 0-2 for all input MSes
spw -- Select spectral window/channels
    NOTE: channels de-selected here will contain all zeros if
        selected by the parameter mode subparameters.
    default: ''=all spectral windows and channels
    spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
    spw='0:5~61'; spw 0, channels 5 to 61
    spw='<2'; spectral windows less than 2 (i.e. 0,1)
    spw='0,10,3:3~45'; spw 0,10 all channels, spw 3,
channels 3 to 45.
    spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
    For multiple MS input, a list of spw strings can be used:
    spw=['0','0~3']; spw ids 0 for the first MS and 0-3 for the second
    spw='0~3' spw ids 0-3 for all input MS
    spw='3:10~20;50~60' for multiple channel ranges within spw id 3
    spw='3:10~20;50~60,4:0~30' for different channel ranges for spw ids 3 and 4
    spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,
        spw 1, channels 20-30, and spw 2, channels, 1,2 and 3
    spw='1~4;6:15~48' for channels 15 through 48 for spw ids 1,2,3,4 and 6

selectdata -- Other data selection parameters
    default: True

>>> selectdata=True expandable parameters
    See help par.selectdata for more on these

timerange -- Select data based on time range:
    default: '' (all); examples,
    timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
    Note: if YYYY/MM/DD is missing date defaults to first
day in data set
    timerange='09:14:0~09:54:0' picks 40 min on first day
    timerange='25:00:00~27:30:00' picks 1 hr to 3 hr
30min on NEXT day
    timerange='09:44:00' pick data within one integration
of time
    timerange='>10:24:00' data after this time

```

For multiple MS input, a list of timerange strings can be used:

```
timerange=['09:14:0~09:54:0','>10:24:00']
timerange='09:14:0~09:54:0'; apply the same timerange for
    all input MSes
```

uvrange -- Select data within uvrange (default units meters)
 default: '' (all); example:

```
uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
uvrange='>4klambda'; uvranges greater than 4 kilo lambda
```

For multiple MS input, a list of uvrange strings can be used:

```
uvrange=['0~1000klambda','100~1000klamda']
uvrange='0~1000klambda'; apply 0-1000 kilo-lambda for all
    input MSes
```

antenna -- Select data based on antenna/baseline
 default: '' (all)
 If antenna string is a non-negative integer, it is
 assumed to be an antenna index, otherwise, it is
 considered an antenna name.

antenna='5&6'; baseline between antenna index 5 and
 index 6.

antenna='VA05&VA06'; baseline between VLA antenna 5
 and 6.

antenna='5&6;7&8'; baselines 5-6 and 7-8
 antenna='5'; all baselines with antenna index 5
 antenna='05'; all baselines with antenna number 05
 (VLA old name)

antenna='5,6,9'; all baselines with antennas 5,6,9
 index number

For multiple MS input, a list of antenna strings can be used:

```
antenna=['5','5&6'];
antenna='5'; antenna index 5 for all input MSes
```

scan -- Scan number range.
 default: '' (all)
 example: scan='1~5'

For multiple MS input, a list of scan strings can be used:

```
scan=['0~100','10~200']
scan='0~100; scan ids 0-100 for all input MSes
```

Check 'go listobs' to insure the scan numbers are in order.

observation -- Observation ID range.
 default: '' (all)
 example: observation='1~5'

intent -- Scan intent (case sensitive)
 default: '' (all)
 example: intent='TARGET_SOURCE'


```
example: intent='TARGET_SOURCE1,TARGET_SOURCE2'
example: intent='TARGET_POINTING*'
```

mode -- Frequency Specification:

NOTE: Channels deselected with spw parameter will contain all zeros.

See examples below.

default: 'mfs'

mode = 'mfs' means produce one image from all specified data.

mode = 'channel'; Use with nchan, start, width to specify output image cube.

mode = 'velocity', channels are specified in velocity.

mode = 'frequency', channels are specified in frequency.

>>> mode='mfs' expandable parameters

Make a continuum image from the selected frequency channels/range using Multi-frequency synthesis algorithm for wide-band narrow field imaging.

mode='mfs' examples:

spw = '0,1'; mode = 'mfs'

will produce one image made from all channels in spw 0 and 1

spw='0:5~28~2'; mode = 'mfs'

will produce one image made with channels (5,7,9,...,25,27)

nterms -- Number of Taylor terms to be used to model the frequency dependence of the sky emission. nterms=1 is equivalent to assuming no frequency dependence.

nterms>1 runs the MS-MFS algorithm, and the choice of nterms should depend on the expected shape and SNR of the spectral structure, across the chosen bandwidth. Output images represent taylor-coefficients of the sky spectrum (images with file-name extensions of tt0,tt1,etc).

A spectral index map is also computed as the ratio of the first two terms (following the convention of $I(\nu) = I(\text{ref_}\nu) \times (\nu/\nu_0)^{\alpha}$).

Additionally, a spectral-index error image is made

by treating taylor-coefficient residuals as errors, and propagating them through the division used to compute spectral-index.

It is meant to be a guide to which parts of the spectral-index image to trust, and the values may not always represent a statistically-correct error.

For more details about this algorithm, please refer to

"A multi-scale multi-frequency deconvolution algorithm for synthesis

imaging in radio interferometry", Rau and Cornwell, AA, Volume 532, 2011

**** Note that the software implementation of the MS-MFS algorithm
for nterms>1 currently does not allow combination with
mosaics, and pbcor.****

reffreq -- The reference frequency (for nterms>1) about which
the Taylor expansion is done. reffreq='' defaults to the
middle frequency of the selected range.

>>> mode='channel', 'velocity', and 'frequency' expandable parameters

nchan -- Total number of channels in the output image.

Example: nchan=100.

Default: -1; Automatically selects enough channels to cover
data selected by 'spw' consistent with 'start' and 'width'.

It is often easiest to leave nchan at the default value.

start -- First channel, velocity, or frequency.

For mode='channel'; This selects the channel index number
from the MS (0 based) that you want to correspond to the
first channel of the output cube. The output cube will be
in frequency space with the first channel having the
frequency of the MS channel selected by start. start=0

refers to the first channel in the first selected spw, even
if that channel is de-selected in the spw parameter.

Channels de-selected by the spw parameter will be filled with
zeros if included by the start parameter. For example,
spw=3~8:3~100 and start=2 will produce a cube that starts on
the third channel (recall 0 based) of spw index 3, and the
first channel will be blank.

example:start=5

For mode='velocity' or 'frequency': default='';

starts at first input channel of first input spw

examples: start='5.0km/s', or start='22.3GHz'.

width -- Output channel width

For mode='channel', default=1; >1 indicates channel averaging

example: width=4.

For mode= 'velocity' or 'frequency', default=''; width of
first input channel, or more precisely, the difference
in frequencies between the first two selected channels.

-- For example if channels 1 and 3 are selected with spw,

then the default width will be the difference between their
frequencies, and not the width of channel 1.

-- Similarly, if the selected data has uneven channel-spacing,

the default width will be picked from the first two selected channels. In this case, please specify the desired width. When specifying the width, one must give units examples: width='1.0km/s', or width='24.2kHz'. Setting width>0 gives channels of increasing frequency for mode='frequency', and increasing velocity for mode='velocity'.

interpolation -- Interpolation type for spectral gridding onto the uv-plane. Options: 'nearest', 'linear', or 'cubic'. default = 'linear'

Note : 'linear' and 'cubic' interpolation requires data points on both sides of each image frequency. Errors are therefore possible at edge channels, or near flagged data channels.

When image channel width is much larger than the data channel width there is nothing much to be gained using linear or cubic thus not worth the extra computation involved.

resmooth -- if the cube has a different restoring beam/channel. Restore image to a common beam or leave as is (default) options: True or False default = False

chaniter -- specify how spectral CLEAN is performed, default: chaniter=False; example: chaniter=True; step through channels

outframe -- For mode='velocity', 'frequency', or 'channel': default spectral reference frame of output image Options: '', 'LSRK', 'LSRD', 'BARY', 'GEO', 'TOPO', 'GALACTO', 'LGROUP', 'CMB' default: ''; same as input data example: frame='bary' for Barycentric frame

veltype -- for mode='velocity' gives the velocity definition Options: 'radio', 'optical' default: 'radio' NOTE: the viewer always defaults to displaying the 'radio' frame, but that can be changed in the position tracking pull down.

mode='channel' examples:

spw = '0'; mode = 'channel': nchan=3; start=5; width=4 will produce an image with 3 output planes plane 1 contains data from channels (5+6+7+8) plane 2 contains data from channels (9+10+11+12)

```

        plane 3 contains data from channels (13+14+15+16)
        spw = '0:0~63^3'; mode='channel'; nchan=21; start = 0;
width = 1
        will produce an image with 20 output planes
        Plane 1 contains data from channel 0
        Plane 2 contains data from channel 2
        Plane 21 contains data from channel 61
        spw = '0:0~40^2'; mode = 'channel'; nchan = 3; start =
5; width = 4
        will produce an image with three output planes
        plane 1 contains channels (5,7)
        plane 2 contains channels (13,15)
        plane 3 contains channels (21,23)

psfmode -- method of PSF calculation to use during minor cycles:
default: 'clark': Options: 'clark','clarkstokes', 'hogbom'
'clark' use smaller beam (faster, usually good enough);
for stokes images clean components peaks are searched
in the  $I^2+Q^2+U^2+V^2$  domain
'clarkstokes' locate clean components independently in
each stokes image
'hogbom' full-width of image (slower, better for poor
uv-coverage)
Note: psfmode will also be used to clean if imagermode = ''
imagermode -- Advanced imaging e.g. mosaic or Cotton-Schwab clean
default: imagermode='csclean': Options: '', 'csclean', 'mosaic'
'' => psfmode cleaning algorithm used
NOTE: imagermode 'mosaic' (and/or) any gridmode not blank
      (and/or) nterms>1 : will always use CS style clean.

>>> gridmode='' expandable parameters
      The default value of '' has no effect.

>>> gridmode='widefield' expandable parameters
      Apply corrections for non-coplanar effects during imaging
      using the W-Projection algorithm (Cornwell et al. IEEE JSTSP
      (2008)) or faceting or a combination of the two.

wprojplanes is the number of pre-computed w-planes used for
the W-Projection algorithm. wprojplanes=1 disables
correction for non-coplanar effects. default value wprojplanes=-1
means clean will determine the number to use.
facets is the number of facets on each side of the image
(i.e. the total number of facets is 'facets x facets').
If wprojplanes>1, W-Projection is done for each facet.
Usually when many wprojection convolution functions

```

sizes are above ~400 pixels ,
it might be faster to use a few facets with wprojection.

```
>>> gridmode='aprojection' expandable parameters
Corrects for the (E)VLA time-varying PB effects
including polarization squint using the A-Projection
algorithm (Bhatnagar et al., AandA, 487, 419 (2008)).
This can optionally include w-projection also.
```

wprojplanes is the number of pre-computed w-planes used
for W-Projection algorithm. wprojplanes=1 disables
correction for non-coplanar effects.

cfcache is the name of the directory to store the
convolution functions and weighted sensitivity pattern
function.

rotpainc (in degrees) is the Parallaxic Angle increment
used for OTF rotation of the convolution function.

painc (in degrees) is the Parallaxic Angle increment
used to compute the convolution functions.

```
>>> imagermode='mosaic' expandable parameter(s):
Make a mosaic of the different pointings (uses csclean style
too)
mosweight -- Individually weight the fields of the mosaic
default: False; example: mosweight=True
This can be useful if some of your fields are more
sensitive than others (i.e. due to time spent
on-source); this parameter will give more weight to
higher sensitivity fields in the overlap regions.
ftmachine -- Gridding method for the mosaic;
Options: 'ft' or 'mosaic'
default: 'mosaic';
'ft' implies standard interferometric gridding. The residual visibilities are
imaged for each pointing and combined in the image plane with the
appropriate PB to make the mosaic.
```

'mosaic' (grid using the Fourier transform of PB as convolution function
and mosaic combination is done in visibilities).

ONLY if imagermode='mosaic' is chosen and
ftmachine='mosaic', is heterogeneous imaging (CARMA, ALMA) or

wideband beam accounting
possible using the right convolution derived from primary beams for
each baseline and for different frequencies
CAVEAT: ftmachine='mosaic' uses Fourier transforms of the primary beams/
for mosaicing. Making an image which is too small for the pointing coverage
aliasing due to standard Fourier transform wrap around.

scaletype -- Controls scaling of pixels in the image plane.
(controls what is seen if interactive=True)
It does *not* affect the scaling of the *final* image -
that is done by pbcor.
default='SAULT'; example: scaletype='PBCOR'
Options: 'PBCOR', 'SAULT'
'SAULT' when interactive=True shows the residual
with constant noise across the mosaic.
Can also be achieved by setting pbcor=False.
'PBCOR' uses the SAULT scaling scheme for
deconvolution, but if interactive=True shows the
primary beam corrected image during interactive.

cyclefactor -- Controls the threshold at which the
deconvolution cycle will pause to degrid and subtract the
model from the visibilities.
With poor PSFs, reconcile often (cyclefactor=4 or 5) for
reliability.
With good PSFs, use cyclefactor = 1.5 to 2.0 for speed.
Note: threshold = cyclefactor * max sidelobe * max residual
default: 1.5; example: cyclefactor=4
cyclespeedup -- The major cycle threshold doubles in this
number of iterations.
Default: -1 (no doubling)
Example: cyclespeedup=3
Try cyclespeedup = 50 to speed up cleaning.

flatnoise -- Controls whether searching for clean components
is done in a constant noise residual image (True) or in an
optimal signal-to-noise residual image (False) when
ftmosaic='mosaic' is chosen.
default=True

>>> imagermode='csclean' expandable parameter(s):
Image using the Cotton-Schwab algorithm in between major cycles

cyclefactor -- See above, under imagermode='mosaic'.
cyclespeedup -- See above, under imagermode='mosaic'.

multiscale -- set of scales to use in deconvolution. If set, cleans with several resolutions using Hogbom clean. The scale sizes are in units of cellsize. So if cell='2arcsec', a multiscale scale=10 => 20arcsec. The first scale is recommended to be 0 (point), we suggest the second be on the order of synthesized beam, the third 3-5 times the synthesized beam, etc.. Avoid making the largest scale too large relative to the image width or the scale of the lowest measured spatial frequency. For example, if the synthesized beam is 10" FWHM and cell=2", try multiscale = [0,5,15].

default: multiscale=[] (standard CLEAN with psfmode algorithm, no multi-scale).

Example: multiscale = [0,5,15]

>>> multiscale expandable parameter(s):

negcomponent -- Stop component search when the largest scale has found this number of negative components;

-1 means continue component search even if the largest component is negative. default: -1; example: negcomponent=50

smallscalebias -- A bias toward smaller scales.

The peak flux found at each scale is weighted by a factor = 1 - smallscalebias*scale/max_scale, so that Fw = F*factor.

Typically the values range from 0.2 to 1.0.

default: 0.6

imsize -- Image size in pixels (x, y). DOES NOT HAVE TO BE A POWER OF 2 (but has to be even and factorizable to 2,3,5,7 only).

default = [256,256]; example: imsize=[350,350]

imsize = 500 is equivalent to [500,500]

If include outlier fields, e.g., [[400,400],[100,100]] or use outlierfile.

Avoid odd-numbered imsize.

cell -- Cell size (x,y)

default= '1.0arcsec';

example: cell=['0.5arcsec','0.5arcsec'] or

cell=['1arcmin', '1arcmin']

cell = '1arcsec' is equivalent to ['1arcsec','1arcsec']

NOTE:cell = 2.0 => ['2arcsec', '2arcsec']

phasecenter -- direction measure or fieldid for the mosaic center

default: '' => first field selected ;

example: phasecenter=6

phasecenter='J2000 19h30m00 -40d00m00'

phasecenter='J2000 292.5deg -40.0deg'

```

        phasecenter='J2000 5.105rad -0.698rad'
    If include outlier fields,
        e.g. ['J2000 19h30m00 -40d00m00',J2000 19h25m00 -38d40m00']
    or use outlierfile.
restfreq -- Specify rest frequency to use for output image
    default='' Occasionally it is necessary to set this (for
    example some VLA spectral line data). For example for
    NH_3 (1,1) put restfreq='23.694496GHz'
stokes -- Stokes parameters to image
    default='I'; example: stokes='IQUV';
    Options: 'I','Q','U','V','IV','QU','IQ','UV','IQU','IUV','IQUV','RR','LL','XX'
niter -- Maximum number iterations,
    if niter=0, then no CLEANing is done ("invert" only).
    (niter=0 can be used instead of the 'ft' task to predict/save a model)
For cube or multi field images niter is the maximum number of iteration
    clean will use for each image plane.
    The number of iterations used may be less than niter if threshold value
    is reached
    default: 500; example: niter=5000
gain -- Loop gain for CLEANing
    default: 0.1; example: gain=0.5
threshold -- Flux level at which to stop CLEANing
    default: '0.0mJy';
    example: threshold='2.3mJy' (always include units)
            threshold = '0.0023Jy'
            threshold = '0.0023Jy/beam' (okay also)
interactive -- use interactive clean (with GUI viewer)
    default: interactive=False
    example: interactive=True
    interactive clean allows the user to build the cleaning
    mask interactively using the viewer. The viewer will
    appear every npercycle iteration, but modify as needed
    The final interactive mask is saved in the file
    imagename_interactive.mask. The initial masks use the
    union of mask and cleanbox (see below).

>>> interactive=True expandable parameters
    npercycle -- this is the number of iterations between each
    interactive update of the mask. It is important to modify
    this number interactively during the cleaning, starting with
    a low number like 20, but then increasing as more extended
    emission is encountered.

mask -- Specification of cleanbox(es), mask image(s), primary beam
    coverage level, and/or region(s) to be used for CLEANing.
    CLEAN tends to perform better, and is less likely to diverge,

```


if the CLEAN component placement is limited by a mask to where real emission is expected to be. As long as the image has the same shape (size), mask images (e.g. from a previous interactive session) can be used for a new execution. NOTE: the initial clean mask actually used is the union of what is specified in mask and <imagenam>.mask

default: [] or '' : no masking; Possible specification types:

(a) Cleanboxes, specified using the CASA region format

(http://casaguides.nrao.edu/index.php?title=CASA_Region_Format)

Example : mask='box [[100pix , 130pix] , [120pix, 150pix]]'

mask='circle [[120pix , 40pix] ,6pix]'

mask='circle[[19h58m52.7s,+40d42m06.04s] , 30.0arcsec]'

If used with a spectral cube, it will apply to all channels.

Multiple regions may be specified as a list of pixel ranges.

Example : mask= ['circle [[120pix , 40pix] ,6pix]',

'box [[100pix , 130pix] , [120pix, 150pix]]'

(b) Filename with cleanbox shapes defined using the CASA region format.

Example: mask='mycleanbox.txt'

The file 'mycleanbox.txt' contains :

box [[100pix , 130pix] , [120pix, 150pix]]

circle [[150pix , 150pix] ,10pix]

rotbox [[60pix , 50pix] , [30pix , 30pix] , 30deg]

(c) Filename for image mask. Example: mask='myimage.mask'

Multiple mask files may be specified.

example : mask=['mask1.mask', 'mask2.mask']

(d) Filename for region specification (e.g. from viewer).

Example: mask='myregion.rgn'

(e) Combinations of the above options.

Example: mask=['mycleanbox.txt', 'myimage.mask',

'myregion.rgn', 'circle [[120pix , 40pix] ,6pix]']

(f) Threshold on primary-beam.

A number between 0 and 1, used as a threshold of primary beam coverage. The primary beam coverage map (imagenam + '.flux(.pbcoverage)') will be made and the CLEAN component placement will be limited to where it is > the number.

(g) True or False.

True: like (f), but use minpb as the number.

False: go maskless (and expect trouble).

(For masks for multiple fields, please see 'HINTS ON CLEAN WITH FLANKING FIELDS')

uvtaper -- Apply additional uv tapering of the visibilities.

default: uvtaper=False; example: uvtaper=True

>>> uvtaper=True expandable parameters

outertaper -- uv-taper on outer baselines in uv-plane

[bmaj, bmin, bpa] taper Gaussian scale in uv or

angular units. NOTE: the on-sky FWHM in arcsec is roughly
the uv taper/200 (klambda).
default: outertaper=[]; no outer taper applied
example: outertaper=['5klambda'] circular taper
FWHM=5 kilo-lambda
outertaper=['5klambda','3klambda','45.0deg']
outertaper=['10arcsec'] on-sky FWHM 10 arcseconds
outertaper=['300.0'] default units are lambda
in aperture plane
innertaper -- uv-taper in center of uv-plane
[bmaj,bmin,bpa] Gaussian scale at which taper falls to
zero at uv=0
default: innertaper=[]; no inner taper applied
NOT YET IMPLEMENTED
modelimage -- Name of model image(s) to initialize cleaning. If
multiple images, then these will be added together to
form initial staring model NOTE: these are in addition
to any initial model in the <imagename>.model image file
default: '' (none); example: modelimage='orion.model'
modelimage=['orion.model','sdorion.image'] Note: if the
units in the image are Jy/beam as in a single-dish
image, then it will be converted to Jy/pixel as in a
model image, using the restoring beam in the image
header and zeroing negatives. If the image is in Jy/pixel then it is taken
as is.

When nterms>1, a one-to-one mapping is done between images
in this list and Taylor-coefficients. If more than nterms
images are specified, only the first nterms are used.
It is valid to supply fewer than nterms model images.
Example : Supply an estimate of the continuum flux from a
previous imaging run.

weighting -- Weighting to apply to visibilities:
default='natural'; example: weighting='uniform';
Options: 'natural','uniform','briggs',
'superuniform','briggsabs','radial'

>>> Weighting expandable parameters

For details on weighting please see Chapter3
of late Dr. Brigg's thesis (<http://www.aoc.nrao.edu/dissertations/dbriggs>)
For weighting='briggs' and 'briggsabs'
robust -- Brigg's robustness parameter
default=0.0; example: robust=0.5;
Options: -2.0 to 2.0; -2 (uniform)/+2 (natural)
For weighting='briggsabs'
noise -- noise parameter to use for Briggs "abs"

weighting

```
example noise='1.0mJy'
npixels -- uv-box used for weight calculation
           a box going from -npixel/2 to +npixel/2 on each side
           around a point is used to calculate weight density.
           0 means box is pixel size
example npixels=2
Default = 0
```

Exception: when choosing superuniform it does not make sense to use npixels=0 as it is uniform thus if npixels is 0 it will be forced to 6 or a box of -3pixels to 3pixels

```
restoringbeam -- Output Gaussian restoring beam for CLEAN image
[bmaj, bmin, bpa] elliptical Gaussian restoring beam
default units are in arc-seconds for bmaj,bmin, degrees
for bpa default: restoringbeam=[]; Use PSF calculated
from dirty beam.
example: restoringbeam=['10arcsec'] circular Gaussian
FWHM 10 arcseconds example:
restoringbeam=['10.0','5.0','45.0deg'] 10"x5"
at 45 degrees
```

```
pbcor -- Output primary beam-corrected image
If pbcor=False, the final output image is NOT corrected for
the PB pattern (particularly important for mosaics), and
therefore is not "flux correct". Correction can also be
done after the fact using immath to divide
<imagename>.image by the <imagename>.flux image.
default: pbcor=False; output un-corrected image
example: pbcor=True; output pb-corrected image (masked outside
minpb)
```

```
minpb -- Minimum PB level to use for pb-correction and pb-based masking.
default=0.2;
example: minpb=0.01
When imagermode is *not* 'mosaic' :
minpb is applied to the flux image (sensitivity-weighted pb).
minpb is used to create a mask, only when pbcor=True
When imagermode='mosaic' :
minpb is applied to the flux.pbcoverage image
(mosaic pb with equal weight per pointing)
minpb is always used to create a mask (regardless of
pbcor=True/False)
```

```
usescratch -- if True will create scratch columns if they are
not there. And after clean completes the predicted model
```

visibility is from the clean components are written to the ms. This increases the ms size by the data volume. if False then the model is saved in the ms header and the calculation of the visibilities is done on the fly when using calibration or plotms. Use True if you want to access the moedl visibilities in python, say.

```
allowchunk -- Partition the image cube by channel-chunks.
              default=False;
              False: Major cycle grids all channels. Minor cycle steps
                    through all channels before the next major cycle.
              True: Major and minor cycles are performed one chunk
                    at a time, and output images cubes are concatenated.
async -- Run asynchronously
              default = False; do not run asynchronously
```

=====

HINTS ON CLEAN WITH FLANKING FIELDS

There are two ways of specifying multi-field images for clean.

- (a) Task parameters are used to define the first(main) field.
A text file containing definitions of all additional fields is supplied to the 'outlierfile' task parameter.

This outlier file must contain the following parameters per field

Required : imagename, imsize, phasecenter

Optional : mask, modelimage

The parameter set for each field must begin with 'imagename'.

Parameters can be listed in a single line or span multiple lines.

Example : Three fields.

```
- Task Inputs :
  imagename = 'M1_0'
  outlierfile='outlier.txt'
  imsize = [1024,1024]
  phasecenter = 'J2000 13h27m20.98 43d26m28.0'

- Contents of outlier file 'outlier.txt':
  imagename = 'M1_1'
  imsize = [128,128]
  phasecenter = 'J2000 13h30m52.159 43d23m08.02'
  mask = ['out1.mask', 'circle[[40pix,40pix],5pix]' ]
  modelimage = 'out1.model'
```

```

imagenname = 'M1_2'
imsize = [128,128]
phasecenter = 'J2000 13h24m08.16 43d09m48.0'

```

In this example, the first field 'M1_0' is defined using main task parameters. The next two 'M1_1' and 'M1_2' are listed in the file 'outlier.txt'. A mask and modelimage has been supplied only for the second field (M1_1). Fields with unspecified masks will use the full field for cleaning.

(b) Specify all fields as lists for each task parameter :

Parameters that support lists for multi-field specification :
 'imagenname', 'imsize', 'phasecenter', 'mask', 'modelimage'

Example : Three fields (same as above)

```

imagenname = ['M1_0','M1_1','M1_2']
imsize = [[1024,1024],[128,128],[128,128]]
phasecenter = ['J2000 13h27m20.98 43d26m28.0',
               'J2000 13h30m52.159 43d23m08.02',
               'J2000 13h24m08.16 43d09m48.0']
mask=[[''], ['out1.mask','circle[[40pix,40pix],5pix]'],['']]
modelimage=[''], ['out1.model'],['']]

```

Note : All lists must have the same length.

In the examples for both (a) and (b), the following images will be made:
 M1_0.image, M1_1.image, M1_2.image cleaned images
 M1_0.model, M1_1.model, M1_2.model model images
 M1_0.residual, M1_1.residual, M1_2.residual residual images

Note : The old AIPS-style outlier-file and boxfile formats have been deprecated. However, due to user-requests, they will continue be supported in CASA 3.4. Note that the old outlier file format does not support the specification of modelimage and mask for each field. The new format is more complete, and less ambiguous, so please consider updating your scripts.

clearcal-task.html

0.1.11 clearcal

Requires:

Synopsis

Re-initializes the calibration for a visibility data set

Arguments

Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
field	Select field using field id(s) or field name(s) allowed: string Default:
spw	Select spectral window/channel. allowed: string Default:
intent	Select observing intent allowed: string Default:
addmodel	Add MODEL_DATA scratch column allowed: bool Default: False

Returns

void

Example

```
Clearcal reinitializes the calibration columns in a measurement set.
Specificially, it will set the MODEL_DATA column (if present) to
unity in total intensity and zero in polarization, and it will
set the CORRECTED_DATA column to the original (observed) DATA
```

in the DATA column. Use the field and spw parameters to select which data to initialize. If the dataset does not yet have the scratch columns, they will be created (MODEL_DATA only if addmodel=True) and initialized for the whole dataset (field, spw, and intent will be ignored in this case).

Keyword arguments:

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'

field -- Select field using field id(s) or field name(s).
        [run listobs to obtain the list id's or names]
        default: ''=all fields
        If field string is a non-negative integer, it is assumed a field index
        otherwise, it is assumed a field name
        field='0~2'; field ids 0,1,2
        field='0,4,5~7'; field ids 0,4,5,6,7
        field='3C286,3C295'; field named 3C286 and 3C295
        field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window
      default: ''=all spectral windows and channels
      spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
      spw='<2'; spectral windows less than 2 (i.e. 0,1)
      spw='0:5~61'; spw 0, channels 5 to 61
      spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.
      spw='0~2:2:6'; spw 0,1,2 with channels 2 through 6 in each.
      NB: Multiple channel ranges per spw are not supported in clearcal.

intent -- Select observing intent
        default: '' (no selection by intent)
        intent='*BANDPASS*' (selects data labelled with
                             BANDPASS intent)

addmodel -- add MODEL_DATA along with CORRECTED_DATA if True;
           otherwise it will add/reset only CORRECTED_DATA, model visibilities
           will then be evaluated when needed.
           default: False (model will not be added)
```

clearplot-task.html

0.1.12 clearplot

Requires:

Synopsis

Clear the matplotlib plotter and all layers

Arguments

Inputs

Returns

void

Example

Run the task clearplot when you want to clear completely the matplotlib, but keep it available for additional plotting.

Typing 'go clearplot()'	will not change the current task being scrutinized
Typing 'clearplot()'	will change the current task assignment to clearplot which is generally not what is desired.

clearstat-task.html

0.1.13 clearstat

Requires:

Synopsis

Clear all autolock locks

Arguments

Inputs

Returns

void

Example

This task is useful if another task that is running indicates that it is trying to obtain a lock on a file.

Typing 'go clearstat()'	will not change the current task being scrutinized
Typing 'clearstat()'	will change the current task assignment to clearpstat which is generally not what is desired.

concat-task.html

0.1.14 concat

Requires:

Synopsis

Concatenate several visibility data sets.

Description

The list of data sets given in the vis argument are chronologically concatenated into an output data set in concatvis, i.e. the data sets in vis are first ordered by the time of their earliest integration and then concatenated.

If there are fields whose direction agrees within the direction tolerance (parameter dirtol), the actual direction in the resulting, merged output field will be the one from the chronologically first input MS.

If concatvis already exists (e.g., it is the same as the first input data set), then the other input data sets will be appended to the concatvis data set. There is no limit to the number of input data sets.

If none of the input data sets have any scratch columns (model and corrected columns), none are created in the concatvis. Otherwise these columns are created on output and initialized to their default value (1 in model column, data in corrected column) for those data with no input columns.

Spectral windows for each data set with the same channelization, and within a specified frequency tolerance of another data set will be combined into one spectral window.

A field position in one data set that is within a specified direction tolerance of another field position in any other data set will be combined into one field.

The field names need not be the same—only their position is used.

Each appended dataset is assigned a new observation id (provided the entries in the observation table are indeed different).

Keyword arguments: vis – Name of input visibility files to be combined
default: none; example: vis = ['src2.ms', 'ngc5921.ms', 'ngc315.ms'] concatvis –
Name of visibility file that will contain the concatenated data note: if this file
exists on disk then the input files are added to this file. Otherwise the new file
contains the concatenated data. Be careful here when concatenating to an
existing file. default: none; example: concatvis='src2.ms' example:
concatvis='outvis.ms'

freqtol – Frequency shift tolerance for considering data to be in the same
spwid. The number of channels must also be the same. default: " == 1 Hz
example: freqtol='10MHz' will not combine spwid unless they are within 10

MHz. Note: This option is useful to combine spectral windows with very slight frequency differences caused by Doppler tracking, for example.

dirtol – Direction shift tolerance for considering data as the same field default: " == 1 mas (milliarcsec) example: dirtol='1arcsec' will not combine data for a field unless their phase center differ by less than 1 arcsec. If the field names are different in the input data sets, the name in the output data set will be the first relevant data set in the list.

respectname – If true, fields with a different name are not merged even if their direction agrees (within dirtol) default: False

timesort – If true, the output visibility table will be sorted in time. default: false. Data in order as read in. example: timesort=true Note: There is no constraint on data that is simultaneously observed for more than one field; for example multi-source correlation of VLBA data.

copypointing – Make a proper copy of the POINTING subtable (can be time consuming). If False, the result is an empty POINTING table. default: True

visweightscale – The weights of the individual MSs will be scaled in the concatenated output MS by the factors in this list. SIGMA will be scaled by 1/sqrt(factor). Useful for handling heterogeneous arrays. Use plotms to inspect the "Wt" column as a reference for determining the scaling factors.

See the cookbook for more details. example: [1.,3.,3.] - scale the weights of the second and third MS by a factor 3 and the SIGMA column of these MS by a factor 1/sqrt(3). default: [] (empty list) - no scaling

Arguments

Inputs	
vis	Name of input visibility files to be concatenated allowed: stringArray Default:
concatvis	Name of output visibility file allowed: string Default:
freqtol	Frequency shift tolerance for considering data as the same spwid allowed: any Default: variant
dirtol	Direction shift tolerance for considering data as the same field allowed: any Default: variant
respectname	If true, fields with a different name are not merged even if their direction agrees allowed: bool Default: False
timesort	If true, sort by TIME in ascending order allowed: bool Default: False
copypointing	Copy all rows of the POINTING table. allowed: bool Default: True
visweightscale	List of the weight scaling factors to be applied to the individual MSs allowed: doubleArray Default:

Example

```
concat(vis=['src2.ms','ngc5921.ms'], concatvis='src2.ms')
    will concatenate 'ngc5921.ms' into 'src2.ms', and the original
src2.ms is lost

concat(vis=['src2.ms','ngc5921.ms'], concatvis='out.ms')
    will concatenate 'ngc5921.ms' and 'src2.ms' into a file named
    'out.ms'; the original 'ngc5921.ms' and 'src2.ms' are untouched.

concat(vis=['src2.ms','ngc5921.ms'], concatvis='out.ms', dirtol='0.5arcsec')
```

like the previous example but using a direction tolerance increased to 0.5 arcsec. Fields whose directions differ by less than this limit are merged into one field with the name and direction from the chronologically first input MS.

```
concat(vis=['v1.ms','v2.ms'], concatvis = 'vall.ms')
  then
concat(vis=['v3.ms','v4.ms'], concatvis = 'vall.ms')
  vall.ms will contains v1.ms+v2.ms+v3.ms+v4.ms
```

Note: run flagmanager to save flags in the concatvis

conjugatevis-task.html

0.1.15 conjugatevis

Requires:

Synopsis

Change the sign of the phases in all visibility columns.

Arguments

Inputs	
vis	Name of input visibility file. allowed: string Default:
spwlist	Spectral window selection allowed: any Default: variant ""
outputvis	Name of output visibility file allowed: string Default:
overwrite	Overwrite the outputvis if it exists. allowed: bool Default: False

Example

Change the sign of the phases in all visibility columns

Keyword arguments:

```
vis -- Name of input visibility file
default: none; example='3C273XC1.ms'
spwlist -- Select spectral window
default: [] all spws will be conjugated; example: spw=[1,2]
outputvis -- name of output visibility file
            default: 'conjugated_'+vis; example= 'conjugated.ms'
overwrite -- Overwrite the outputvis if it exists
default=False; example: overwrite=True
```

Example:

```
conjugatevis(vis='NGC253.ms', spwlist=[0,1], outputvis='NGC253-conj.ms')
```

Will conjugate all visibilities for spectral windows 0 and 1 and store the modified data in NGC253-conj.ms.

csvclean-task.html

0.1.16 csvclean

Requires:

Synopsis

This task does an invert of the visibilities and deconvolve in the image plane.

Description

This task does an invert of the visibilities and deconvolve in the image plane. It does not do a uvdata subtraction (aka Cotton-Schwab major cycle) of model visibility as in clean. - For ALMA Commissioning

Arguments

Inputs		
vis	Name of input visibility file	
	allowed:	string
	Default:	
imagename	Name of image	
	allowed:	string
	Default:	
field	Select field using field id(s) or field name(s)	
	allowed:	string
	Default:	
spw	Select spectral window/channels	
	allowed:	any
	Default:	variant
advise	Boolean to determine if advice on image cell is requested	
	allowed:	bool
	Default:	False
mode	define the mode to operate csvclean: option continuum, cube	
	allowed:	string
	Default:	continuum
nchan	Number of channels (planes) in output image; -1 = all	
	allowed:	int
	Default:	-1
width	width of output spectral channels	
	allowed:	variant
	Default:	variant 1
imsize	Image size in pixels (nx,ny), symmetric for single value	
	allowed:	intArray
	Default:	256256
cell	arcsec The image cell size in arcseconds [x,y].	
	allowed:	doubleArrayarcsec
	Default:	1.01.0
phasecenter	Image center: direction or field index	
	allowed:	any
	Default:	variant
niter	Maximum number of iterations	
	allowed:	int
	Default:	500
weighting	Type of weighting	
	allowed:	string
	Default:	natural
restoringbeam	Output Gaussian restoring beam for CLEAN image	
	allowed:	stringArray
	Default:	
interactive	Create a mask interactively or not.	
	allowed:	bool
	Default:	False

Example

This task does not do a uvdata subtraction (aka Cotton-Schwab major cycle) of model visibility as in clean. - For ALMA Commissioning

Keyword arguments:

vis -- Name of input visibility file

default: none; example: vis='ngc5921.ms'

imagename -- Name of output CASA image. (only the prefix)

default: none; example: imagename='m2'

output images are:

m2.image; cleaned and restored image

With or without primary beam correction

m2dirty.image; dirty image

m2psf.image; point-spread function (dirty beam)

m2.model; image of clean components

m2.mask; image containing clean regions, when interactive=True

field -- Select fields in mosaic. Use field id(s) or field name(s).

['go listobs' to obtain the list id's or names]

default: ''= all fields

If field string is a non-negative integer, it is assumed to be a field index otherwise, it is assumed to be a

field name

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

field='3C286,3C295'; field named 3C286 and 3C295

field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels

NOTE: This selects the data passed as the INPUT to mode

default: ''= all spectral windows and channels

spw='0~2,4'; spectral windows 0,1,2,4 (all channels)

spw='0:5~61'; spw 0, channels 5 to 61

spw='<2'; spectral windows less than 2 (i.e. 0,1)

spw='0,10,3:3~45'; spw 0,10 all channels, spw 3,

channels 3 to 45.

spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.

spw='0:0~10;15~60'; spectral window 0 with channels

0-10,15-60

spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,

spw 1, channels 20-30, and spw 2, channels, 1,2 and 3

```

    advise -- This determines whether advice for imsize and cell is
              requested. If set to True. It won't run clean but return
              values for imsize and cell estimated for the longest
              baseline in the data

imsize -- Image pixel size (x,y). DOES NOT HAVE TO BE A POWER OF 2
        default = [256,256]; example: imsize=[350,350]
        imsize = 500 is equivalent to [500,500].
        Avoid odd-numbered imsize.

cell -- Cell size (x,y)
       default= '1.0arcsec';
       example: cell=['0.5arcsec','0.5arcsec'] or
       cell=['1arcmin', '1arcmin']
       cell = '1arcsec' is equivalent to ['1arcsec','1arcsec']
       NOTE:cell = 2.0 => ['2arcsec', '2arcsec']

phasecenter -- direction measure or fieldid for the mosaic center
              default: '' => first field selected ; example: phasecenter=6
              or phasecenter='J2000 19h30m00 -40d00m00'

mode -- this determines what kind of image to make
       continuum or cube. In continuum all the selected data
       channels are combined in a 1 channel image using
       multifrequency synthesis.
       options are 'cube' and 'continuum'
       default: 'continuum'
>>> mode='cube' expandable parameters
nchan -- sets the number of channel in the output
        image. e.g nchan=10

width -- image channel width in terms of the number of
        channel of the first spw of the data selected
        e.g width=2

niter -- Maximum number of iterations,
        if niter=0, then no CLEANing is done ("invert" only)
        default: 500; example: niter=5000

weighting -- Weighting to apply to visibilities:
            default='natural'; example: weighting='uniform';
            Options: 'natural','uniform','briggs',
                    'superuniform','briggsabs','radial'

```

```

restoringbeam -- Output Gaussian restoring beam for CLEAN image
                [bmaj, bmin, bpa] elliptical Gaussian restoring beam.
                Default units are in arc-seconds for bmaj and bmin, and in degrees
                for bpa. Default: restoringbeam=[]; Use PSF calculated
                from dirty beam.
                example: restoringbeam=['10arcsec'] or restorinbeam='10arcsec', circular Gauss
                FWHM 10 arcseconds example:
                restoringbeam=['10.0','5.0','45.0deg'] 10"x5"
                at 45 degrees

interactive -- Create a mask interactively or not.
                default=False; example: interactive=True
                The viewer will open with the image displayed. Select the
                region for the mask and double click in the middle of it.

```

cvel-task.html

0.1.17 cvel

Requires:

Synopsis

regrid an MS to a new spectral window / channel structure or frame

Description

The intent of cvel is to transform channel labels and the visibilities to a spectral reference frame which is appropriate for the science analysis, e.g. from TOPO to LSRK to correct for Doppler shifts throughout the time of the observation. Naturally, this will change the shape of the spectral feature to some extent. According to the Nyquist theorem you should oversample a spectrum with twice the numbers of channels to retain the shape. Based on some tests, however, we recommend to observe with at least 3-4 times the number of channels for each significant spectral feature (like 3-4 times the linewidth). This will minimize regridding artifacts in cvel.

If cvel has already established the grid that is desired for the imaging, clean should be run with exactly the same frequency/velocity parameters as used in cvel in order to avoid additional regridding in clean.

Hanning smoothing is optionally offered in cvel, but tests have shown that already the regridding process itself, if it involved a transformation from TOPO to a non-terrestrial reference frame, implies some smoothing (due to channel interpolation) such that Hanning smoothing may not be necessary.

Arguments

Inputs	
vis	Name of input measurement set allowed: string Default:
outputvis	Name of output measurement set allowed: string Default:
passall	Pass through (write to output MS) non-selected data with no change allowed: bool Default: False
field	Select field using field id(s) or field name(s) allowed: any Default: variant
spw	Select spectral window/channels allowed: any Default: variant
selectdata	Other data selection parameters allowed: bool Default: True
antenna	Select data based on antenna/baseline allowed: string Default:
timerange	Range of time to select from data allowed: string Default:
scan	scan number range allowed: string Default:
array	(sub)array indices allowed: string Default:
mode	Regridding mode allowed: string Default: channel
nchan	Number of channels in output spw (-1=all) allowed: int Default: -1
start	First channel in input to use allowed: any Default: variant 0
width	Number of input channels to average allowed: any Default: variant 1
interpolation	Spectral interpolation method allowed: string Default: linear
phasecenter	Phase center direction to be used for the spectral coordinate transformation: position or field index allowed: any Default: variant
restfreq	rest frequency (see help) allowed: string

Returns

void

Example

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'

outputvis -- Name of output measurement set (required)
            default: none; example: vis='ngc5921-regridded.ms'

passall -- if False, data not meeting the selection is omitted/deleted
           or flagged (if in-row); if True, data not meeting the selection
           on field and spw is passed through without modification
default: False; example:
      field='NGC5921'
      passall=False : only data from NGC5921 is included in output MS,
                      no data from other fields (e.g. 1331+305) is included
      passall=True  : data from NGC5921 is transformed by cvel, all other
                      fields are passed through unchanged

field -- Select fields in mosaic. Use field id(s) or field name(s).
        ['go listobs' to obtain the list id's or names]
default: ''= all fields
        If field string is a non-negative integer, it is assumed to
        be a field index otherwise, it is assumed to be a
field name
      field='0~2'; field ids 0,1,2
      field='0,4,5~7'; field ids 0,4,5,6,7
      field='3C286,3C295'; field named 3C286 and 3C295
      field = '3,4C*'; field id 3, all names starting with 4C

spw --Select spectral window/channels
      NOTE: This selects the data passed as the INPUT to mode
default: ''=all spectral windows and channels
      spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
      spw='0:5~61'; spw 0, channels 5 to 61
      spw='<2'; spectral windows less than 2 (i.e. 0,1)
      spw='0,10,3:3~45'; spw 0,10 all channels, spw 3,
channels 3 to 45.
      spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
```

```

        spw='0:0~10;15~60'; spectral window 0 with channels
0-10,15-60
        spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,
        spw 1, channels 20-30, and spw 2, channels, 1,2 and 3

        selectdata -- Other data selection parameters
        default: True

>>> selectdata=True expandable parameters

        antenna -- Select data based on antenna/baseline
        default: '' (all)
        If antenna string is a non-negative integer, it is
        assumed to be an antenna index, otherwise, it is
        considered an antenna name.
        antenna='5&6'; baseline between antenna index 5 and
index 6.
        antenna='VA05&VA06'; baseline between VLA antenna 5
        and 6.
        antenna='5&6;7&8'; baselines 5-6 and 7-8
        antenna='5'; all baselines with antenna index 5
        antenna='05'; all baselines with antenna number 05
(VLA old name)
        antenna='5,6,9'; all baselines with antennas 5,6,9
        index numbers

        timerange -- Select data based on time range:
        default = '' (all); examples,
        timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
        Note: if YYYY/MM/DD is missing date defaults to first
day in data set
        timerange='09:14:0~09:54:0' picks 40 min on first day
        timerange= '25:00:00~27:30:00' picks 1 hr to 3 hr
30min on NEXT day
        timerange='09:44:00' pick data within one integration
of time
        timerange='>10:24:00' data after this time

        scan -- Scan number range.
        default: '' (all)
        example: scan='1~5'
        Check 'go listobs' to insure the scan numbers are in
order.

        array -- Select data by (sub)array indices
        default: '' (all); example:

```



```

array='0~2'; arrays 0 to 2

mode -- Frequency Specification:
      NOTE: See examples below:
      default: 'channel'
           mode = 'channel'; Use with nchan, start, width to specify
           output spw. Produces equidistant grid based on first
           selected channel. See examples below.
           mode = 'velocity', means channels are specified in
velocity.
           mode = 'frequency', means channels are specified in
frequency.
           mode = 'channel_b', alternative 'channel' mode.
           Does not force an equidistant grid. Faster.

>>> mode expandable parameters
      Start, width are given in units of channels, frequency
or velocity as indicated by mode
      nchan -- Number of channels in output spw
           default: -1 = all channels; example: nchan=3
      start -- Start or end input channel (zero-based) depending on the sign of the
           default=0; example: start=5
      width -- Output channel width in units of the input
channel width (sign indicates whether the start parameter is lower(+) or upper(-) end)
           default=1; example: width=4
      interpolation -- Interpolation method (linear, nearest, cubic, spline, fftshift)
           default = 'linear'
      examples:
           spw = '0,1'; mode = 'channel'
           will produce a single spw containing all channels in spw
0 and 1
           spw='0:5~28~2'; mode = 'channel'
           will produce a single spw made with channels
(5,7,9,...,25,27)
           spw = '0'; mode = 'channel': nchan=3; start=5; width=4
           will produce an spw with 3 output channels
           new channel 1 contains data from channels (5+6+7+8)
           new channel 2 contains data from channels (9+10+11+12)
           new channel 3 contains data from channels (13+14+15+16)
           spw = '0:0~63~3'; mode='channel'; nchan=21; start = 0;
width = 1
           will produce an spw with 21 channels
           new channel 1 contains data from channel 0
           new channel 2 contains data from channel 2
           new channel 21 contains data from channel 61
           spw = '0:0~40~2'; mode = 'channel'; nchan = 3; start =

```

5; width = 4

will produce an spw with three output channels
new channel 1 contains channels (5,7)
new channel 2 contains channels (13,15)
new channel 3 contains channels (21,23)

phasecenter -- Direction measure or fieldid. To be used in mosaics to indicate
the center direction to be used in the spectral coordinate transformation.
default: '' => first field selected ; example: phasecenter=6
or phasecenter='J2000 19h30m00 -40d00m00'

restfreq -- Specify rest frequency to use for output image
default='' Occasionally it is necessary to set this (for
example some VLA spectral line data). For example for
NH₃ (1,1) put restfreq='23.694496GHz'

outframe -- output reference frame (not case-sensitive)
possible values: LSRK, LSRD, BARY, GALACTO, LGROUP, CMB, GEO, TOPO, or SOURCE
(SOURCE is meant for solar system work and corresponds to GEO + radial velocity
correction for ephemeris objects).
default='' (keep original reference frame) ; example: outframe='BARY'

veltype -- definition of velocity (in mode)
default = 'radio'

hanning -- if true, Hanning smooth frequency channel data to remove Gibbs ringing

=====

The intent of cvel is to transform channel labels and the
visibilities to a spectral reference frame which is appropriate
for the science analysis, e.g. from TOPO to LSRK to correct for
Doppler shifts throughout the time of the observation. Naturally,
this will change the shape of the spectral feature to some extent.
According to the Nyquist theorem you should oversample a spectrum
with twice the numbers of channels to retain the shape. Based on
some tests, however, we recommend to observe with at least
3-4 times the number of channels for each significant spectral
feature (like 3-4 times the linewidth). This will minimize
regridding artifacts in cvel.

If cvel has already established the grid that is desired for the
imaging, clean should be run with exactly the same frequency/velocity
parameters as used in cvel in order to avoid additional regridding in
clean.

Hanning smoothing is optionally offered in cvel, but tests have shown that already the regridding process itself, if it involved a transformation from TOPO to a non-terrestrial reference frame, implies some smoothing (due to channel interpolation) such that Hanning smoothing may not be necessary.

cvel2-task.html

0.1.18 cvel2

Requires:

Synopsis

Regrid an MS or MMS to a new spectral window, channel structure or frame

Description

The intent of cvel2 is to transform channel labels and the visibilities to a spectral reference frame which is appropriate for the science analysis, e.g. from TOPO to LSRK to correct for Doppler shifts throughout the time of the observation. Naturally, this will change the shape of the spectral feature to some extent. According to the Nyquist theorem you should oversample a spectrum with twice the numbers of channels to retain the shape. Based on some tests, however, we recommend to observe with at least 3-4 times the number of channels for each significant spectral feature (like 3-4 times the linewidth). This will minimize regridding artifacts in cvel2.

If cvel2 has already established the grid that is desired for the imaging, clean should be run with exactly the same frequency/velocity parameters as used in cvel2 in order to avoid additional regridding in clean.

Hanning smoothing is optionally offered in cvel2, but tests have shown that already the regridding process itself, if it involved a transformation from TOPO to a non-terrestrial reference frame, implies some smoothing (due to channel interpolation) such that Hanning smoothing may not be necessary. This version of cvel2 also supports Multi-MS input, in which case it will create an output Multi-MS too.

NOTE: The parameter passall is not supported in cvel2. The user may achieve the same results of passall=True by splitting out the data that will not be regridded with cvel2 and concatenate regridded and non-regridded sets at the end. In the case of Multi-MS input, the user should use virtualconcat to achieve a concatenated MMS.

Arguments

Inputs	
vis	Name of input Measurement set or Multi-MS. allowed: string Default:
outputvis	Name of output Measurement Set or Multi-MS. allowed: string Default:
keepmms	If the input is a Multi-MS the output will also be a Multi-MS. allowed: bool Default: True
passall	HIDDEN parameter. Pass through (write to output MS) non-selected data with no change allowed: bool Default: False
field	Select field using ID(s) or name(s). allowed: any Default: variant
spw	Select spectral window/channels. allowed: any Default: variant
scan	Select data by scan numbers. allowed: any Default: variant
antenna	Select data based on antenna/baseline. allowed: any Default: variant
correlation	Correlation: " ==> all, correlation='XX,YY'. allowed: any Default: variant
timerange	Select data by time range. allowed: any Default: variant
intent	Select data by scan intent. allowed: any Default: variant
array	Select (sub)array(s) by array ID number. allowed: any Default: variant
uvrange	Select data by baseline length. allowed: any Default: variant
observation	Select by observation ID(s). allowed: any Default: variant
feed	Multi-feed numbers: Not yet implemented.

Returns

void

Example

Detailed description of keyword arguments:

--- Input/Output parameters ---

vis -- Name of input visibility file
default: ''; example: vis='ngc5921.ms'

outputvis -- Name of output visibility file or Multi-MS
default: ''; example: outputvis='ngc5921.mms'

keepmms -- Create a Multi-MS as the output if the input is a Multi-MS.
default: True

By default it will create a Multi-MS when the input is a Multi-MS.
The output Multi-MS will have the same partition axis of the input MMS.
See 'help partition' for more information on the MMS format.

NOTE: It is not possible to combine the spws if the input MMS was partitioned with
separationaxis='spw'. In this case, the task will abort with an error.

--- Data selection parameters ---

field -- Select field using field id(s) or field name(s).
[run listobs to obtain the list of d's or names]
default: ''=all fields If field string is a non-negative
integer, it is assumed to be a field index
otherwise, it is assumed to be a field name
field='0~2'; field ids 0,1,2
field='0,4,5~7'; field ids 0,4,5,6,7
field='3C286,3C295'; fields named 3C286 and 3C295
field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels
default: ''=all spectral windows and channels
spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
spw='<2'; spectral windows less than 2 (i.e. 0,1)

```

spw='0:5~61'; spw 0, channels 5 to 61
spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.
spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
spw = '*:3~64' channels 3 through 64 for all sp id's
      spw = ' :3~64' will NOT work.

```

NOTE: mstransform does not support multiple channel ranges per spectral window (';').

```

scan -- Scan number range
      default: ''=all

```

```

antenna -- Select data based on antenna/baseline
          default: '' (all)
          Non-negative integers are assumed to be antenna indices, and
          anything else is taken as an antenna name.

```

examples:

```

antenna='5&6': baseline between antenna index 5 and index 6.
antenna='VA05&VA06': baseline between VLA antenna 5 and 6.
antenna='5&6;7&8': baselines 5-6 and 7-8
antenna='5': all baselines with antenna 5
antenna='5,6,10': all baselines including antennas 5, 6, or 10
antenna='5,6,10&': all baselines with *only* antennas 5, 6, or
                  10. (cross-correlations only. Use &&
                  to include autocorrelations, and &&&
                  to get only autocorrelations.)
antenna='!ea03,ea12,ea17': all baselines except those that
                          include EVLA antennas ea03, ea12, or
                          ea17.

```

```

correlation -- Correlation types or expression.
             default: '' (all correlations)
             example: correlation='XX,YY'

```

```

timerange -- Select data based on time range:
            default: '' (all); examples,
            timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
            Note: if YYYY/MM/DD is missing date, timerange defaults to the
            first day in the dataset
            timerange='09:14:0~09:54:0' picks 40 min on first day
            timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min
            on next day
            timerange='09:44:00' data within one integration of time
            timerange='>10:24:00' data after this time

```

```

array -- (Sub)array number range
        default: ''=all

uvrange -- Select data within uvrange (default units meters)
        default: ''=all; example:
            uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
            uvrange='>4klambda';uvranges greater than 4 kilo-lambda
            uvrange='0~1000km'; uvrange in kilometers

observation -- Select by observation ID(s)
        default: ''=all

feed -- Selection based on the feed - NOT IMPLEMENTED YET
        default: ''=all

datacolumn -- Which data column to use for processing (case-insensitive).
        default: 'all';
        options: 'data', 'model', 'corrected', 'all','float_data', 'lag_data',
            'float_data,data', 'lag_data,data'.
        example: datacolumn='data'

        NOTE: 'all' = whichever of the above that are present. If the requested
            column does not exist, the task will exit with an error.

mode -- Frequency Specification:
        NOTE: See examples below:
        default: 'channel'
            mode = 'channel'; Use with nchan, start, width to specify
                output spw. Produces equidistant grid based on first
                selected channel. See examples below.
            mode = 'velocity', means channels are specified in
velocity.
            mode = 'frequency', means channels are specified in
frequency.
            mode = 'channel_b', alternative 'channel' mode.
        Does not force an equidistant grid. Faster.

>>> mode expandable parameters
        Start, width are given in units of channels, frequency
or velocity as indicated by mode
        nchan -- Number of channels in output spw
            default: -1 = all channels; example: nchan=3
        start -- Start or end input channel (zero-based) depending on the sign of the
            default=0; example: start=5
        width -- Output channel width in units of the input
channel width (sign indicates whether the start parameter is lower(+) or upper(-) end

```



```

        default=1; example: width=4
interpolation -- Interpolation method (linear, nearest, cubic, spline, fftsh
        default = 'linear'
examples:
    spw = '0,1'; mode = 'channel'
        will produce a single spw containing all channels in spw
0 and 1
    spw='0:5~28^2'; mode = 'channel'
        will produce a single spw made with channels
(5,7,9,...,25,27)
    spw = '0'; mode = 'channel': nchan=3; start=5; width=4
        will produce an spw with 3 output channels
        new channel 1 contains data from channels (5+6+7+8)
        new channel 2 contains data from channels (9+10+11+12)
        new channel 3 contains data from channels (13+14+15+16)
    spw = '0:0~63^3'; mode='channel'; nchan=21; start = 0;
width = 1
        will produce an spw with 21 channels
        new channel 1 contains data from channel 0
        new channel 2 contains data from channel 2
        new channel 21 contains data from channel 61
    spw = '0:0~40^2'; mode = 'channel'; nchan = 3; start =
5; width = 4
        will produce an spw with three output channels
        new channel 1 contains channels (5,7)
        new channel 2 contains channels (13,15)
        new channel 3 contains channels (21,23)

phasecenter -- Direction measure or fieldid. To be used in mosaics to indicate
        the center direction to be used in the spectral coordinate transformation.
        default: '' (first selected field)
        options: FIELD_ID (int) or center coordinate measure (str).
        example: phasecenter=6 or phasecenter='J2000 19h30m00 -40d00m00'

restfreq -- Specify rest frequency to use for output image
        default='' Occasionally it is necessary to set this (for
        example some VLA spectral line data). For example for
        NH_3 (1,1) put restfreq='23.694496GHz'

outframe -- output reference frame (not case-sensitive)
        possible values: LSRK, LSRD, BARY, GALACTO, LGROUP, CMB, GEO, TOPO, or SOURCE
        (SOURCE is meant for solar system work and corresponds to GEO + radial velocity
        correction for ephemeris objects).
        default='' (keep original reference frame) ; example: outframe='BARY'

veltype -- definition of velocity (in mode)

```

```
default = 'radio'
```

```
hanning -- if true, Hanning smooth frequency channel data to remove Gibbs ringing
```

```
=====
```

The intent of cvel2 is to transform channel labels and the visibilities to a spectral reference frame which is appropriate for the science analysis, e.g. from TOPO to LSRK to correct for Doppler shifts throughout the time of the observation. Naturally, this will change the shape of the spectral feature to some extent. According to the Nyquist theorem you should oversample a spectrum with twice the numbers of channels to retain the shape. Based on some tests, however, we recommend to observe with at least 3-4 times the number of channels for each significant spectral feature (like 3-4 times the linewidth). This will minimize regridding artifacts in cvel2.

If cvel2 has already established the grid that is desired for the imaging, clean should be run with exactly the same frequency/velocity parameters as used in cvel2 in order to avoid additional regridding in clean.

Hanning smoothing is optionally offered in cvel2, but tests have shown that already the regridding process itself, if it involved a transformation from TOPO to a non-terrestrial reference frame, implies some smoothing (due to channel interpolation) such that Hanning smoothing may not be necessary.

deconvolve-task.html

0.1.19 deconvolve

Requires:

Synopsis

Image based deconvolver

Description

Several algorithms are available to deconvolve an image with a known psf (dirty beam), or a Gaussian beam. The algorithms available are clark and hogbom clean, a multiscale clean and a mem clean.

NOTE: Recommend using taskname=clean if psf is a dirty beam

Arguments

Inputs		
imagename		Input image to deconvolve allowed: string Default:
model		Output image containing deconvolved point model allowed: string Default:
psf		Point spread function (dirty beam) allowed: stringArray Default:
alg		Algorithm to use (clark, hogbom, multiscale, mem) allowed: string Default: clark
niter		number of iteration in deconvolution process allowed: int Default: 10
gain		CLEAN gain parameter allowed: double Default: 0.1
threshold	mJy	level below which sources will not be deconvolved allowed: doublemJy Default: 0.0
mask		image mask to limit region of deconvolution allowed: string Default:
scales		scale sizes (pixels) to deconvolve allowed: intArray Default: 0310
sigma	mJy	mem parameter: Expected noise in image allowed: doublemJy Default: 0.0
targetflux	Jy	mem parameter: Estimated total flux in image allowed: doubleJy Default: 1.0
prior		mem parameter: prior image for mem search allowed: string Default:

Returns

void

Example

Several algorithms are available to deconvolve an image with a known psf (dirty beam), or a Gaussian beam. The algorithms available are clark and hogbom clean, a multiscale clean and a mem clean. For more deconvolution control, use clean.

Keyword arguments:

```

imagenam -- Name of input image to be deconvolved
model    -- Name of output image containing the clean components
psf      -- Name of psf image (dirty beam) to use
           example: psf='casaxmlf.image' .
           If the psf has 3 parameter, then a Gaussian
           psf is assumed with the values representing
           the major , minor and position angle  values
           e.g psf=['3arcsec', '2.5arcsec', '10deg']
alg      -- algorithm to use: default = 'clark'
           options: clark, hogbom, multiscale or mem.
niter    -- Maximum number of iterations
gain     -- CLEAN gain parameter; fraction to remove from peak
threshold -- Halt deconvolution if the maximum residual image is
           below this threshold.
           default = '0.0Jy'
mask     -- mask image (same shape as image and psf) to limit region
           where deconvoltion is to occur

-----parameters useful for multiscale only
scales   -- in pixel numbers; the size of component to deconvolve.
           default value [0,3,10]
           recommended sizes are 0 (point), 3 (points per clean beam), and
           10 (about a factor of three lower resolution)

-----parameters useful for mem only
sigma    -- Estimated noise for image
targetflux -- Target total flux in image
prior    -- Prior image to guide mem

```

delmod-task.html

0.1.20 delmod

Requires:

Synopsis

Deletes model representations in the MS

Arguments

Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
otf	Delete the on-the-fly model data keywords allowed: bool Default: True
field	Select field using field id(s) or field name(s) allowed: string Default:
scr	Delete the MODEL_DATA scr col (if it exists) allowed: bool Default: False

Returns

void

Example

This utility task is to be used to delete the model visibility data representations in the MS. The 'otf' representation is the new (as of v3.4) 'scratch-less' model data, stored as keywords in the MS header containing model data formation instructions. It is generated by the setjy, ft, and clean tasks (usescratch=F), and if present, overrides the old-fashioned MODEL_DATA column (if present). If a user

wishes to use the MODEL_DATA column _after_ having operated with the 'otf' representation, this task can be used to delete the 'otf' representation to make the MODEL_DATA column visible. (Create the MODEL_DATA column by using usescratch=T in setjy, ft, or clean; or by running the clearcal task with addmodel=T.)

If otf=T, specific fields can be selected for deletion using standard field selection semantics. If field='', all fields' models will be deleted.

For convenience, this method also provides a means for deleting the MODEL_DATA column by setting scr=T. Note that it is not possible to delete the MODEL_DATA column per field.

If otf=F and scr=F, delmod will provide a listing of the header field records.

[exportasdm-task.html](#)

0.1.21 exportasdm

Requires:

Synopsis

Convert a CASA visibility file (MS) into an ALMA or EVLA Science Data Model

Arguments

Inputs	
vis	MS name allowed: string Default:
asdm	Name of output ASDM directory (on disk) allowed: string Default:
datacolumn	specifies which MS data column is used to fill the visibilities in the ASDM allowed: string Default: data
archiveid	the X0 in uid://X0/X1/X<running> allowed: string Default: S0
rangeid	the X1 in uid://X0/X1/X<running> allowed: string Default: X1
subscanduration	maximum duration of a subscan in the output ASDM allowed: string Default: 24h
sbduration	maximum duration of a scheduling block (and therefore exec block) in the output ASDM allowed: string Default: 2700s
apcorrected	data to be marked as having atmospheric phase correction allowed: bool Default: False
verbose	produce log output allowed: bool Default: True
showversion	Report the version of ASDM class set being used allowed: bool Default: True
useversion	Selects the version of MS2asdm to be used ('v3' (default and only option presently)) allowed: string Default: v3

Returns

bool

Example

```
exportasdm(vis='ngc4826.ms', asdm='uid__S021_X1418_X1',
           datacolumn='corrected', archiveid='S021', rangeid='X1418',
           verbose=False)
```

will produce an ASDM named 'uid__S021_X1418_X1' using the datacolumn 'corrected' in the MS 'ngc4826.ms' with minimal log output.

The sbduration parameter controls the number of execution blocks (EBs) into which exportasdm subdivides the visibilities from your input MS. If the total observation time in the MS is shorter than what is given in sbduration, a single EB will be created.

Note concerning ALMA data: exportasdm presently is not able to export from MSs containing WVR data. If you attempt to export such an MS, you will receive an error message saying that you can only export data of processor type "CORRELATOR". It will also give you the list of SPWs which contain CORRELATOR data. You will then have to split out these SPWs using the task "split" and run exportasdm on the resulting MS.

Also EVLA data can be exported. Note here that exportasdm does not produce online flags and that a subsequent reimport of the data must be done with online=False. Also, importevla will only work on your ASDM if you have exported it with apcorrected=False (the default).

```
importevla('xosrosdm', vis = 'xosro.ms')
exportasdm(vis='xosro.ms', asdm='xosrosdm', apcorrected=False)
importevla(asdm='xosro2asdm', vis='xosro2-reimp.ms', online=False)
```

[exportfits-task.html](#)

0.1.22 exportfits

Requires:

Synopsis

Convert a CASA image to a FITS file

Description

CASA-produced images can be exported as FITS files for transporting to other software packages or publication. No subimaging of the fits image can be made with this task. The spectral reference frame can be changed prior to export using the task `imreframe`.

Arguments

Inputs	
imagename	Name of input CASA image allowed: string Default:
fitsimage	Name of output image FITS file allowed: string Default:
velocity	Use velocity (rather than frequency) as spectral axis allowed: bool Default: False
optical	Use the optical (rather than radio) velocity convention allowed: bool Default: False
bitpix	Bits per pixel allowed: int Default: -32
minpix	Minimum pixel value (if minpix > maxpix, value is automatically determined) allowed: any Default: variant 0
maxpix	Maximum pixel value (if minpix > maxpix, value is automatically determined) allowed: any Default: variant -1
overwrite	Overwrite pre-existing imagename allowed: bool Default: False
dropstokes	Drop the Stokes axis? allowed: bool Default: False
stokeslast	Put Stokes axis last in header? allowed: bool Default: True
history	Write history to the FITS image? allowed: bool Default: True
dropdeg	Drop all degenerate axes (e.g. Stokes and/or Frequency)? allowed: bool Default: False

Example

```
exportfits(imagename='NGC3256-continuum.image', fitsimage='NGC3256cont.fits', history=False)
```

[exportuvfits-task.html](#)

0.1.23 exportuvfits

Requires:

Synopsis

Convert a CASA visibility data set to a UVFITS file:

Description

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
fitsfile	Name of output UV FITS file allowed: string Default:
datacolumn	Visibility file data column allowed: string Default: corrected
field	Select field using field id(s) or field name(s) allowed: any Default: variant
spw	Select spectral window/channels allowed: string Default:
antenna	Select data based on antenna/baseline allowed: string Default:
timerange	Select data based on time range allowed: string Default:
avgchan	Channel averaging width (value > 1 indicates averaging) allowed: int Default: 1
writesyscal	Write GC and TY tables, (Not yet available) allowed: bool Default: False
multisource	Write in multi-source format allowed: bool Default: True
combinespw	Export the spectral windows as IFs allowed: bool Default: True
writestation	Write station name instead of antenna name allowed: bool Default: True
padwithflags	Fill in missing data with flags to fit IFs allowed: bool Default: False
overwrite	Overwrite output file if it exists? allowed: bool Default: False

Example

This task writes a UVFITS file, a general format data set used to transfer data between different software systems. It is written in floating point format. Different programs have different restrictions on what forms of UVFITS files they will use, especially whether they will accept multiple sources and/or spectral windows in the same file. See the spw, multisource, and combinespw descriptions below.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'
fitsfile -- Name of output UV FITS file
      default: none; example='3C273XC1.fits'
datacolumn -- Visibility file data column
      default: => 'corrected'; example: datacolumn='model'
      Options: 'data' (raw), 'corrected', 'model', 'weight'
field -- Select field using field id(s) or field name(s).
      [run listobs to obtain the list id's or names]
      default: ''=all fields
      If field string is a non-negative integer, it is assumed a field index
      otherwise, it is assumed a field name
      field='0~2'; field ids 0,1,2
      field='0,4,5~7'; field ids 0,4,5,6,7
      field='3C286,3C295'; field named 3C286 adn 3C295
      field = '3,4C*'; field id 3, all names starting with 4C
spw -- Select spectral window/channels
      type 'help par.selection' for more examples.
      spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
      spw='<2'; spectral windows less than 2 (i.e. 0,1)
      spw='0:5~61'; spw 0, channels 5 to 61, INCLUSIVE
      spw='*:5~61'; all spw with channels 5 to 62
      spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.
      spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
      spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60
      NOTE ';' to separate channel selections
      spw='0:0~10~2,1:20~30~5'; spw 0, channels 0,2,4,6,8,10,
      spw 1, channels 20,25,30
antenna -- Select data based on antenna/baseline
      default: '' (all)
      If antenna string is a non-negative integer, it is assumed
      an antenna index
```



```

        otherwise, it is assumed as an antenna name
        antenna='5&6'; baseline between antenna index 5 and index 6.
        antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
        antenna='5&6;7&8'; baseline 5-6 and 7-8
        antenna='5'; all baselines with antenna index 5
        antenna='05'; all baselines with antenna name '05', vla antenna
        antenna='5,6,10'; all baselines with antennas 5, 6 and 10
timerange -- Select data based on time range:
    default = '' (all); examples,
    timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
    Note: if YYYY/MM/DD is missing dat defaults to first day in data set
    timerange='09:14:0~09:54:0' picks 40 min on first day
    timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min on next day
    timerange='09:44:00' data within one integration of time
    timerange='>10:24:00' data after this time
avgchan -- Channel averaging width (value > 1 indicates averaging)
    default =>1; example: avgchan=3
    output data will average channels in groups of three.
multisource -- Write in multi-source format
    default: => True;
    false if one source is selected
    True works with AIPS, but not difmap.
combinespw -- If True, export the spectral windows as IFs.
    Otherwise multiple windows will use multiple FREQIDs.
    default: => True;
    all spectral windows must have same shape.
    True is recommended for AIPS, and mandatory for difmap.
padwithflags -- If True, and combinespw is True, fill in missing
    data as needed to fit the IF structure. This is
    appropriate if the MS had a few frequency-dependent
    flags applied, and was then time-averaged by split, or
    when exporting for use by difmap. If the spectral
    windows were observed at different times,
    padwithflags=True will add a large number of flags,
    making the output file significantly longer. It does
    not yet support spectral windows with different widths.
writestation -- Write station name instead of antenna name
    default: True;
writesyscal -- Write GC and TY tables
    default: => False; Not yet available
overwrite -- Overwrite specified output file if it exists?

```

feather-task.html

0.1.24 feather

Requires:

Synopsis

Combine two images using their Fourier transforms

Description

The algorithm converts each image to the gridded visibility plane, combines them, and reconverts them into an combined image. Each image must include a well-defined beam shape (clean beam) in order for feathering to work well. The two images must have the same flux density normalization scale.

Arguments

Inputs	
imagename	Name of output feathered image allowed: string Default:
highres	Name of high resolution (interferometer) image allowed: string Default:
lowres	Name of low resolution (single dish) image allowed: string Default:
sdfactor	Scale factor to apply to Single Dish image allowed: double Default: 1.0
effdishdiam	New effective SingleDish diameter to use in m allowed: double Default: -1.0
lowpassfiltersd	Filter out the high spatial frequencies of the SD image allowed: bool Default: False

Example

Feathering is a simple method for combining two images with different spatial resolution. The processing steps are:

1. Regrid the low-resolution image to a temporary copy matching the resolution of the high-resolution image,
2. Transform each image to the spatial-frequency plane (gridded).
3. Scale the low-resolution image (uv-grid) by the ratio of the volumes of the two 'clean beams' (high-res/low-res).
4. Add to this, the uv-grid of the high-resolution image, scaled by $(1-wt)$ where 'wt' is the Fourier transform of the 'clean beam' defined in the low-resolution image.
5. Transform back to the image plane.

Both input images must have a well-defined beam shape for this task to work. This could be a 'clean beam' for interferometric images, and a 'primary-beam' for a single-dish image.

The two images must also have the same flux density normalization scale.

Keyword arguments:

imagename -- Name of output feathered image
default: none; example: imagename='orion_combined.im'
highres -- Name of high resolution (interferometer) image
default: none; example: highres='orion_vla.im'

This image is often a clean image obtained from synthesis observations.

lowres -- Name of low resolution (single dish) image
default: none; example: lowres='orion_gbt.im'

This image is often a image from a single-dish observations or a clean image obtained from lower resolution synthesis observations.

sdfactor -- value by which to scale the Single Dish image. Default is 1.0
Basically modifying the flux scale of the SD image

effdishdiam -- New effective SingleDish diameter to use in m. Obviously one can only reduce

lowpassfiltersd -- If True the high spatial frequency in the SD image is rejected.

Any data outside the maximum uv distance that the SD has illuminated is filtered out.

Comments:

This task can be used as one method of combining single-dish and interferometric images after they have been separately made.

The clean task allows another method of combining single-dish and interferometric data. The single-dish image can be used as a starting model for the interferometric image-reconstruction. If there is some overlap between the spatial-frequencies contained in the single-dish image and the interferometer sampling function, then, such a starting model will help constrain the solutions on the short-baselines of the interferometric data.

find-task.html

0.1.25 find

Requires:

Synopsis

Find string in tasks, task names, parameter names:

Description

Lists the following: 1) All of the task names that have the string 2) All of the tasks whose contents (e.g., documentation, parameters, etc) have the string 3) All of the parameter names that have the string

Arguments

Inputs	
matchstring	String to match in the documentation
	allowed: string
	Default:

Example

```
Find string in tasks, task names, parameter names:
```

```
Lists the following:
```

- 1) All of the task names that have the string
- 2) All of the tasks whose contents (e.g., documentation, parameters, etc) have the string
- 3) All of the parameter names that have the string

```
Keyword arguments:
```

```
matchstring -- String to match in the documentation  
              default: ''; example: matchstring='vis'
```

fixplanets-task.html

0.1.26 fixplanets

Requires:

Synopsis

Changes FIELD and SOURCE table entries based on user-provided direction or POINTING table, optionally fixes the UVW coordinates

Description

This task's main purpose is to correct observations which were performed with correct pointing and correlation but for which incorrect direction information was entered in the FIELD and SOURCE table of the MS. If you actually want to change the phase center of the visibilities in an MS, you should use task fixvis.

Input Parameters vis – Name of the input visibility set

field – field selection string

fixuvw – recalc uvw coordinates? (default: False)

direction – if set, don't use pointing table but set direction to this value. The direction can either be given explicitly or as the path to a JPL Horizons ephemeris (for an example of the format, see directory data/ephemerides/JPL-Horizons/). Alternatively, the ephemeris table can also be provided as mime format file, i.e. a saved email as obtained via the commands (for example): import recipes.ephemerides.request as jplreq
jplreq.request_from_JPL(objnam='Mars',startdate='2012-01-01',enddate='2013-12-31', date_incr='0.1 d', get_axis_orientation=False, get_axis_ang_orientation=True, get_sub_long=True, use_apparent=False, get_sep=False, return_address='YOUR_EMAIL_ADDESS', mailserver='YOUR_MAIL_SERVER_ADDRESS') Note: some mail clients may not save the JPL mail properly. Confirmed to work is Thunderbird.

default= " (use pointing table)

example: 'J2000 19h30m00 -40d00m00'

refant – if using pointing table information, use it from this antenna default: 0 (antenna id 0) examples: 'DV06' (antenna with name DV06) 3 (antenna id 3)

reftime – if using pointing table information, use it from this timestamp default: 'first' examples: 'median' will use the median timestamp for the given field using only the unflagged maintable rows '2012/07/11/08:41:32' will use the given timestamp (must be within the observaton time)

Arguments

Inputs	
vis	Name of the input visibility set. allowed: string Default:
field	Fields to operate on. Blank = all. allowed: any Default: variant ""
fixuvw	recalc uvw? allowed: bool Default: False
direction	if set, don't use pointing table but set direction to this value allowed: any Default: variant
refant	if using pointing table information, use it from this antenna allowed: any Default: variant 0
reftime	if using pointing table information, use it from this timestamp ('first', 'median', or YYYY/MM/DD/hh:mm:ss) allowed: string Default: first

Example

Examples:

```
fixplanets('uid__A002_X1c6e54_X223.ms', 'Titan', True)
    will look up the pointing direction from antenna 0 for field 'Titan' in
    the POINTING table based on the first timestamp in the main table rows for
    this field, enter this direction in the FIELD and SOURCE tables, and then
    recalculate the UVW coordinates for this field.

fixplanets('uid__A002_X1c6e54_X223.ms', 'Titan', True, 'Titan_55438-56292dUTC.tab')
    will attach the ephemeris table 'Titan_55438-56292dUTC.tab' to field 'Titan'
    and then recalculate the UVW coordinates for this field.

fixplanets('uid__A002_X1c6e54_X223.ms', 'Titan', False, 'J2000 12h30m15 -02d12m00')
    will set the directions for field 'Titan' in the FIELD and SOURCE table to the
```

given direction and not recalculate the UVW coordinates.
(This can be useful for several purposes, among them preparing a concatenation of datasets. Only fields with the same direction will be recognised as identical. fixplanets can then be run again after the concatenation using parameters as in the first example above.)

fixvis-task.html

0.1.27 fixvis

Requires:

Synopsis

Recalculates (u, v, w) and/or changes Phase Center

Description

Recalculates (u, v, w) and/or changes Phase Center

Arguments

Inputs	
vis	Name of the input visibility set. allowed: string Default:
outputvis	Name of the output visibility set. (Can be the same as vis.) allowed: string Default:
field	Fields to operate on. " = all. allowed: any Default: variant ""
refcode	reference frame to convert UVW coordinates to allowed: string Default:
reuse	base UVW calculation on the old values? allowed: bool Default: True
phasecenter	use this direction as phase center allowed: string Default:
distances	(experimental) List of the distances (as quanta) of the fields selected by field. allowed: any Default: variant ""
datacolumn	when applying a phase center shift, modify visibilities only in this/these column(s) allowed: string Default: all

Example

If the phase center is changed, the corresponding modifications are applied to the visibility columns given by the parameter "datacolumn" which is by default set to "all" (DATA, CORRECTED, and MODEL).

Input Parameters

```
vis          -- Name of the input visibility set

outputvis    -- Name of the output visibility set, default: same as vis

field        -- field selection string
```

```

refcode    -- Reference frame to convert to, default: the refcode of PHASE_DIR in the
             FIELD table
             example: 'B1950'

reuse      -- base recalculation on existing UVW coordinates? default=True
             ignored if parameter 'phasecenter' is set

phasecenter -- if set to a valid direction: change the phase center for the given
               field to this value
             example: 'J2000 9h25m00s -05d12m00s'
               If given without the equinox, e.g. '0h01m00s +00d12m00s', the parameter
               is interpreted as a pair of offsets in RA and DEC to the present
               phasecenter.
               NOTE: The RA offset can be given in units of time or angle. If given
               as a time (i.e. as a single number with a time unit as in, e.g., 12s
               or in the XXhXXmXXs or XX:XX:XX.XXX formats), it is applied as is.
               If given as an angle (e.g., 0.01deg), it is divided by the cos(DEC)
               before it is applied.

distances -- (experimental) List of the distances (as quanta) of the fields selected by
             to be used for refocussing.
             If empty, the distances of all fields are assumed to be infinity.
             If not a list but just a single value is given, this is applied to
             all fields.
             default: []
             examples: ['2E6km', '3E6km']    '15au'

datacolumn -- when applying a phase center shift, modify visibilities only in
             this/these column(s)
             default: 'all' (DATA, CORRECTED, and MODEL)
             example: 'DATA,CORRECTED' (will not modify MODEL)

Examples:

fixvis('NGC3256.ms','NGC3256-fixed.ms')
    will recalculate the UVW coordinates for all fields based on the existing
    phase center information in the FIELD table.

fixvis('Moon.ms','Moon-fixed.ms','Moon', '', 'J2000 9h25m00s 05d12m00s')
    will set the phase center for field 'Moon' to the given direction and recalculate
    the UVW coordinates.

```

flagcmd-task.html

0.1.28 flagcmd

Requires:

Synopsis

Flagging task based on batches of flag-commands

Description

The flagcmd task allows several batch-operations using flag commands. Flag commands follow the mode and parameter names from the flagdata task (also explained below). The available modes are: manual, clip, shadow, quack, elevation, tfcrop, rflag and extend. The summary mode is not supported in this task. Use the flagdata task for that.

The flagcmd task will flag data based on the commands input on inpmode :
table = input from FLAG_CMD table in MS
list = input from text file or list of strings given in ininfile
xml = input from Flag.xml in the MS given by vis

Batch operations include : apply/unapply/list/plot/clear/extract

IMPORTANT: The FLAG_CMD sub-table is meant only for meta-data selections such as online flags. Using it to save other parameters (from modes such as clip, quack, shadow, etc) is possible but carries a risk that in future releases these parameters maybe renamed or changed their default values. Use it at your own risk! There will be no automatic way to rename any parameter that changes in the future.

There is no way to guarantee that a command from the COMMAND column has been applied or not to the MS, even if the APPLIED column is set to True. If you use other ways to flag such as interactive flagging in plotms, the FLAG_CMD will not be updated! Use at your own risk.

NOTE on flagging calibration tables. —————

It is possible to flag cal tables using this task, although we recommend using the flagdata task for this.

When using this task to flag cal tables, only the 'apply' and 'list' actions are supported. Because cal tables do not have a FLAG_CMD sub-table, the default inpmode='table' can only be used if an MS is given in the 'infile' parameter so that flags from the MS are applied to the cal table. Otherwise, the flag commands must be given using inpmode='list', either from a file(s) or from a list of strings. See below for more information about these parameters. Data selection for calibration tables is limited to field, scan, antenna, time, spw and observation. If the calibration table was created before CASA 4.1, this task will create a dummy OBSERVATION column and OBSERVATION sub-table in the input calibration table to adapt it to the new cal table format.

Arguments

Inputs	
vis	Name of MS file or calibration table to flag allowed: string Default:
inpmode	Input mode for flag commands(table/list/xml) allowed: string Default: table
infile	Source of flag commands allowed: any Default: variant
tablerows	Rows of infile to read allowed: intArray Default:
reason	Select by REASON types allowed: any Default: variant any
useapplied	Select commands whose rows have APPLIED column set to True allowed: bool Default: False
tbuff	Time buffer (sec) to pad flags allowed: double Default: 0.0
ants	Allowed flag antenna names to select by allowed: string Default:
action	Action to perform in MS and/or in infile (apply/unapply/list/plot/clear/extract) allowed: string Default: apply
flagbackup	Automatically backup the FLAG column before execution allowed: bool Default: True
clearall	Delete all rows from FLAG_CMD allowed: bool Default: False
rowlist	FLAG_CMD rows to clear allowed: intArray Default:
plotfile	Name of output file to save plot allowed: string Default:
savepars	Save flag commands to the MS or to a file allowed: bool Default: False
outfile	Name of output file to save commands allowed: string Default:
overwrite	Overwrite an existing file to save the flag commands allowed: bool Default: True

Returns

void

Example

Keyword arguments:

```
vis -- Name of input visibility file or calibration table.  
      default: '' (none)  
      example1: vis='uid___A002_X2a5c2f_X54.ms' or  
      example2: vis='cal-X54.B1'
```

-- INPUT of flag commands --

```
inpmode -- Input mode for flag commands.  
          options: 'table','list','xml'  
          default: 'table'
```

inpmode "table" -- input commands from FLAG_CMD table of MS.

```
inpfiler -- path to MS containing FLAG_CMD  
          default: '' (read from FLAG_CMD table in the MS specified via 'vis')
```

Main use is to read flags from internal FLAG_CMD,
but by setting inpfiler to a different MS you can
use this to copy the flags from one MS to another.

One use case is to read the flag commands from an MS given in
inpfiler and apply them to a cal table given in vis. Example:

```
flagcmd(vis='cal-X54.B1', inpmode='table', inpfiler='uid___A002_X2a5c2f_X54.ms')
```

```
tablerows -- list of rows of the FLAG_CMD table to read  
           default: [] (read all rows)  
           example: [0,1,2,10]
```

NOTE: currently only takes integer lists, not
parseable strings with ranges. Use the Python
range function to generate ranges, e.g.
tablerows = range(0,30) + range(50,55)
instead of '0~29,50~54' for now.

```

reason -- select flag commands based on REASON(s)
        default: 'any' (all flags regardless of reason)
               can be a string, or list of strings
example: reason='FOCUS_ERROR'
        reason=['FOCUS_ERROR','SUBREFLECTOR_ERROR']

```

NOTE: what is within the string is literally matched, e.g. reason='' matches only blank reasons, and reason = 'FOCUS_ERROR,SUBREFLECTOR_ERROR' matches this compound reason string only

```

useapplied -- select the flag commands of rows that have column APPLIED=True
options: True,False
default: False

```

If useapplied=True it will read in both applied and unapplied flags.

IMPORTANT: The APPLIED column is set to True after a flag command is applied to the MS. In order to re-apply the same flag command, this parameter should be set to True.

```

inpmode "list" -- input commands from an ASCII file, a list of files or via a list of
                NOTE: You can only apply the flags from a list; you will not be able
                to unapply them. Transfer the flag commands to the FLAG_CMD table
                if you want to unapply the flags. For this, see action='list'

```

```

inpfile -- name of an ASCII file, list of files or a list of Python strings to apply
         MS or cal table.
default: ''
options: [] with flag commands or
         [] with filenames or
         '' with a filename.

```

IMPORTANT: string values must contain quotes around them or the parser will not work. The parser evaluates the commands in the list and considers only existing Python types.

example1: the following commands can be saved to a file or group of files and given to the task (e.g. save it to flags.txt).

```

scan='1~3' mode='manual'
mode='clip' clipminmax=[0,2] correlation='ABS_XX' clipoutside=False
spw='9' mode='tfcrop' correlation='ABS_YY' ntime=51.0

```



```
mode='extend' extendpols=True
```

```
flagcmd(vis, inpmode='list', infile='flags.txt') or  
flagcmd(vis, inpmode='list', infile=['onlineflags.txt', 'flags.txt'])
```

example2: the same commands can be written in a Python list of strings and given to the task.

```
cmd=["scan='1~3' mode='manual'",  
     "mode='clip' clipminmax=[0,2] correlation='ABS_XX' clipoutside=False",  
     "spw='9' mode='tfcrop' correlation='ABS_YY' ntime=51.0",  
     "mode='extend' extendpols=True"]
```

```
flagcmd(vis, inpmode='list', infile=cmd)
```

```
reason -- select flag commands to apply, based on REASON(s)  
default: 'any' (all flags regardless of reason)  
         can be a string, or list of strings  
example: reason='FOCUS_ERROR'  
         reason=['FOCUS_ERROR', 'SUBREFLECTOR_ERROR']
```

If infile is a list of files, the reasons given in this parameter will apply to all the files.

NOTE: what is within the string is literally matched, e.g. reason='' matches only blank reasons, and reason = 'FOCUS_ERROR,SUBREFLECTOR_ERROR' matches this compound reason string only

```
inpmode "xml" -- input online flags from Flag.xml file in the MS. This mode  
               is not available for cal tables. This works only for MSS  
               imported using importevla. It will not work for ALMA MSS.  
NOTE: You can only apply the flags from a XML file; you will not be able  
      to unapply them. Transfer the flag commands to the FLAG_CMD table  
      if you want to unapply the flags. For this, see action='list'
```

```
tbuff -- (float) time padding buffer (seconds, default=1.0)
```

```
ants -- select flags based on antenna,  
       e.g. antenna='ea01'  
default: '' (all flags regardless of antenna)
```

```
reason -- select flag commands based on REASON(s),  
default: 'Any' (all flags regardless of reason)
```

can be a string, or list of strings
example: reason='FOCUS_ERROR'
reason=['FOCUS_ERROR','SUBREFLECTOR_ERROR']

NOTE: what is within the string is literally
matched, e.g. reason='' matches only blank reasons,
and reason = 'FOCUS_ERROR,SUBREFLECTOR_ERROR'
matches this compound reason string only

--ACTIONS--

action -- operation to perform on MS and/or in flag commands from inpfiler.
options: 'apply','clear','list','plot','unapply','extract'
default: 'apply'

action "apply" -- apply flags to MS or cal table.

This operation will apply the commands chosen by inpmode.
If inpmode='table' it will set the APPLIED column to True.

flagbackup -- Automatically backup MS/cal table FLAG column before applying.
options: True,False
default: True

action "unapply" -- unapply flags in MS. (Not available for cal tables).

This operation will unapply the commands chosen by inpmode='table' ONLY.
After unapplying the commands, the task will update the APPLIED column to False.

flagbackup -- Automatically backup MS FLAG column before unapplying?
options: True,False
default: True

action "list" -- list and/or save flag commands.

This operation will list the commands chosen by inpmode on the screen
and save them to the MS or to a file without applying. It will save the commands
to outfile if the parameter savepars is set to True. If outfile is None, it
will save the commands to the MS given in 'vis'.

action "plot" -- plot flags (ant vs. time). (Not available for cal tables)

This operation will plot the flags chosen by inpmode to a

matplotlib gui or to a file. These will be sorted by antenna vs. time. Most useful for showing the online flags.

plotfile -- output plot file
default: '' (plot to matplotlib window)

WARNING: will only reliably plot individual flags
per antenna and timerange (e.g. direct from xml)

action "clear" -- clear flags from FLAG_CMD in MS. (Not available for cal tables)

This operation will delete the selected flag rows from the internal FLAG_CMD table of the MS.

NOTE: choosing this option will disregard anything you set in inpmode and will always work on the FLAG_CMD table in vis.

clearall -- really clear all flags?
default: False (will not clear)

rowlist -- list of FLAG_CMD rows to clear
default: [] (all flags in table)
example: [0,1,2,10]

NOTE: currently only takes integer lists, not parseable strings with ranges. Use the Python range function to generate ranges, e.g.
rowlist = range(0,30) + range(50,55)
instead of '0~29,50~54' for now.

WARNING: this can be dangerous, and you must set clearall=True to use this!!! This will delete the specified rows from the internal FLAG_CMD table for vis regardless of what mode is set to (useful for when you import from xml or file), and decide to redo it). This action will NOT unapply the commands.

action "extract" -- extract internal flag dictionary. (Not available for cal tables)

This option will return the internal flagging dictionary to python.

savepars -- Save the flag commands to the FLAG_CMD table of the MS or to an output text

default: False
options: True/False

outfile -- Name of output file to save the flag commands.
default: ' '; it will save the commands in the FLAG_CMD table of the MS.
example: outfile='flags.txt' will save the parameters in a text file.

overwrite -- Overwrite the existing file given in 'outfile'.
options: True/False
default: True; it will remove the existing file given in 'outfile' and save
commands to a new file with the same name. When set to False,
an error message if the file exist.

=====

-- Internal FLAG_CMD input 'inpmode' useage --

(For inpmode='table')

* It is a good idea to use action='list' first to see what is there
before doing anything else, e.g.

```
inpmode = 'table'
action = 'list'
```

* To apply the flags stored in the FLAG_CMD table in the MS,
simply set inpmode='table' and action='apply'. This is the default setup
of flagcmd. Note that when a flag command is applied, the corresponding APPLIED
column cell will be updated to True.

```
inpmode = 'table'
action = 'apply'
useapplied = False
```

* To re-apply the flags stored in the FLAG_CMD table in the MS,
inpmode='table', action='apply' and useapplied=True.

```
inpmode = 'table'
action = 'apply'
useapplied = True
```

* To merely save to FLAG_CMD but not apply, then

```
inpmode = 'table'
inpfile = 'other.ms'
```

```
action = 'list'
```

* To save commands from a file into the MS without applying.

```
inpmode = 'list'
infile = 'flags.txt'
action = 'list'
```

If you need to select only certain rows from the FLAG_CMD table, use the `tablerows` parameter to control this. Currently this must be a list of individual row numbers (0-based), e.g.

```
tablerows = [0,1,2,3,10,11]
```

or

```
tablerows = range(29)
```

NOTE: the `useapplied=True/False` tag is important if you are going to (re)apply flags marked as APPLIED True in FLAG_CMD. It is common to have a "failed" flagging operation mark the flags as already applied and then they don't show up when you re-run (e.g. in 'list'). Set `useapplied=True` so that it will use these anyway.

* To apply the flag commands from an MS to a calibration table.

```
vis = 'mycaltable'
inpmode = 'table'
infile = 'myMS.ms'
action = 'apply'
```

-- Online flag input inpmode useage --

(For `inpmode='xml'`)

* To list the online flags stored in the Flag.xml file in the MS, simply set:

```
inpmode = 'xml'
action = 'list'
savepars = False
```

* It is then straightforward to save these to FLAG_CMD

```

inpmode = 'xml'
action = 'list'
savepars = True

```

* To directly apply the online flags stored in the Flag.xml file in the MS, set inpmode='xml' and desired buffer, e.g.

```

inpmode = 'xml'
tbuff = 1.0    # pad flag times by 1 sec
action = 'apply'
set savepars to save or not the commands in the MS

```

* You can also specify a set of reasons (a comma separated list) for flags to apply, e.g.

```

reason = 'FOCUS_ERROR,SUBREFLECTOR_ERROR'    # select these flags
reason = 'ANTENNA_NOT_ON_SOURCE'

```

NOTE: The online flag time buffer tbuff is specified in seconds, but in fact should be keyed to the intrinsic online integration time. This is particularly true for EVLA data, where a tbuff value of 0.5x to 1.5x the integration time is needed (currently you should use 1.5x for data taken in early 2011 or before).

Because the Flag.xml is copied to the MS by importevla, you can re-apply the online flags with an increased tbuff simply by running with inpmode='xml' and optype='apply', e.g.

```

inpmode = 'xml'
tbuff = 15.0    # pad flag times by 15 sec for 10sec integrations
optype = 'apply'

```

if you originally used a smaller value (e.g. 1.0) by mistake or you want to try longer values. Note these will be added to the FLAG_CMD table which you would have to clean up manually if you care about this.

-- Flag command useage --

(For inpmode='list')

* For example, a series of commands might be:

```

antenna='ea01' timerange='00:00:00~01:00:00'

```

```

antenna='ea11' timerange='00:00:00~03:00:00' spw='0~4'
mode='clip' clipminmax=[0,5] correlation='ABS_ALL'
mode='quack' quackmode='end' quackinterval=1.0
mode='shadow'

```

Any other mode can also use selection (see the help of flagdata):

```

mode='shadow' antenna='ea01,ea02,ea03'
mode='quack' quackmode='end' quackinterval=1.0 antenna='ea22'

```

These commands can be saved in an ASCII file, e.g. "myflags.txt" and input using inpmode='list', e.g.

```

flagcmd(vis='myvis.ms',inpmode='list',inpfile='myflags.txt')

```

or input from the interface

```

flagcmd(vis='myvis.ms',inpmode='list',
        inpfile=["mode='shadow'",
                  "mode='clip' clipminmax=[0,5] correlation='ABS_ALL'",
                  "mode='quack' quackmode='end' quackinterval=1.0",
                  "antenna='ea01' timerange='00:00:00~01:00:00'",
                  "antenna='ea11' timerange='00:00:00~03:00:00' spw='0~4'"])

```

-- ACTIONs --

The action parameter controls what flagcmd will actually do with the flag commands:

* action = 'apply'

This will apply the selected commands to the data.

If inpmode='table' and inpfile='' then the APPLIED column in FLAG_CMD will be set to True.

* action = 'unapply'

This will unapply any commands on the selected data that come from the FLAG_CMD table.

* action = 'list'

List what is selected and or save in the MS or in a file. It is wise to do this first before doing any other action. It will list the output in the logger

and save them to the FLAG_CMD table of the MS when savepars=True and outfile = '' to a file if outfile is non-blank.

* action = 'plot'

Will pop up a little matplotlib GUI (if outfile='') or plot to a file. Currently only gives an antenna vs time plot, mostly useful for looking at the online flags.

* action = 'clear'

DANGER! This can be used to totally delete rows from the FLAG_CMD table. It ignores what inpmode is pointing to and always works on FLAG_CMD. Use at your own peril but sometimes you need to just blow that table away, e.g.

```
vis = msfile
optype = 'clear'
rowlist = []          # all rows
clearall = True       # disarm the safety
```

Note you have to explicitly set clearall=True to arm the deletion (a minimal precaution).

* action = 'extract'

This option will return the internal flagging dictionary to python. This will allow a power-user to manipulate these commands directly (e.g. for plotting etc.). For example,

```
myflagd = flagcmd(vis=msfile,useapplied=True,action='extract')
```

will extract all the commands (including those already applied) in the FLAG_CMD MS table.

NOTE: There is no extant description of the format of this dictionary, as it is an internal device used by the flagcmd task. This action is provided for the convenience of advanced users.

----- FLAG COMMAND SYNTAX -----

The command syntax is based on the flagdata parameters.

Basic Syntax Rules

Commands are a string (which may contain internal "strings") consisting of KEY=VALUE pairs separated by whitespace (see examples below).

NOTE: There should be no whitespace between KEY=VALUE or within each KEY or VALUE, since the simple parser first breaks command lines on whitespace, then on "=".

Each key should only appear once on a given command line/string

There is an implicit "mode" for each command, with the default being 'manual' if not given.

Comment lines can start with '#' and will be ignored.

1. Data selection parameters (used by all flagging modes)

```
timerange='',  
antenna='',  
spw='',  
correlation='',  
field='',  
scan='',  
feed='',  
array='',  
uvrange='',  
intent='',  
observation='',
```

Note: a command consisting only of selection key-value pairs is a basic "manual" operation, ie. flag the data meeting the selection.

2. Modes specific parameters with default values (for further details and updated default values, refer to the task flagdata).

2.1 Mode manual.

```
autocorr=False
```

2.2 Mode clip.

```
datacolumn='DATA'  
clipminmax=[]  
clipoutside=True  
channelavg=False  
clipzeros=False
```

```

2.3 Mode shadow.
    tolerance=0.0
    addantenna='',

2.4 Mode quack.
    quackinterval=1.0
    quackmode='beg'
    quackincrement=False

2.5 Mode elevation.
    lowerlimit=0.0
    upperlimit=90.0

2.6 Mode tfcrop.
    ntime='scan'
    combinescans=False
    datacolumn='DATA'
    timecutoff=4.0
    freqcutoff=3.0
    timefit='line'
    freqfit='poly'
    maxnpieces=7
    flagdimension='freqtime'
    usewindowstats='none'
    halfwin=1
    extendflags=True

2.7 Mode extend.
    ntime='scan'
    combinescans=False
    extendpols=True
    growtime=50.0
    growfreq=50.0
    growaround=False
    flagneartime=False
    flagnearfreq=False

2.8 Mode rflag.
    ntime='scan'
    combinescans=False
    datacolumn='DATA'
    winsize=3
    timedev='',
    freqdev='',
    timedevscale=5.0

```

```
freqdevscale=5.0
spectralmax=1000000.0
spectralmin=0.0
extendflags=True
```

2.9 Mode unflag.

3. Basic elaboration options for online and interface use

```
id=''          # flag ID tag (not necessary)
reason=''      # reason string for flag
flagtime=''    # a timestamp for when this flag was generated (for
                user history use)
```

NOTE: there is no flagtime column in FLAG_CMD at this time, but we will propose to add this as an optional column

NOTE: These are currently ignored and not used.

4. Extended elaboration options for online and interface use

Note: these are FLAG_CMD columns, but their use is not clear but included here for compatibility and future expansion

```
level=N        # flagging "level" for flags with same reason
severity=N     # Severity code for the flag, on a scale of 0-10 in order
                of increasing severity; user specified
```

flagdata-task.html

0.1.29 flagdata

Requires:

Synopsis

All-purpose flagging task based on data-selections and flagging modes/algorithms.

Description

This task can flag a Measurement Set or a calibration table. It has two main types of operation. One type will read the parameters from the interface and flag using any of the various available modes. The other type will read the commands from a text file, a list of files or a Python list of strings, containing a list of flag commands (each line containing data selection parameters and any parameter specific for the mode being requested). Please see examples at the end of this help.

It is also possible to only save the parameters set in the interface without flagging. The parameters can be saved in the FLAG_CMD sub-table or in a text file. Note that when saving to an external file, the parameters will be appended to the given file.

The available flagging modes are: manual, clip, shadow, quack, elevation, tfcrop, rflag, extend, unflag and summary. For automatic flagging, it is recommended to combine auto-flag modes with extend, via the list mode. The current flags can be automatically backed up before applying new flags if the parameter flagbackup is set. Previous flag versions can be recovered using the flagmanager task.

NOTE on flagging calibration tables. —————

Flagdata can flag many types of calibration tables using mode='manual'. It can only flag using the auto-flagging algorithms (clip, tfcrop or rflag), the cal tables that have the following data columns: CPARAM, FPARAM or SNR. The solution elements of the data columns are given in the correlation parameter using the names 'Sol1', 'Sol2', 'Sol3', or 'Sol4'. See examples at the end of this help on how to flag different cal tables.

When the input is a calibration table, the modes 'elevation' and 'shadow' will be disabled. Data selection for calibration tables is limited to field, scan, time, antenna, spw and observation. It is only possible to save the parameters to an external file. If the calibration table was created before CASA 4.1, this task will create a dummy OBSERVATION column and OBSERVATION sub-table in the input calibration table to adapt it to the new cal table format.

Selecting antennas in some calibration tables have a different meaning compared to selecting the MS. Some calibration tables such as the antenna-based ones, created with some modes of gencal or polcal, have the ANTENNA2 column set to -1. This means that when selecting antenna='ANT', will select the whole ANT and not the cross-correlations between ANT and the other antennas. Similarly, the baseline syntax do not apply to this type of calibration tables. Those values with ampersand do not have any meaning when selecting antenna/baselines in antenna-based cal tables.

The task will flag a subset of data based on the following modes of operation:
list = list of flagging commands to apply to MS/cal table manual = flagging based on specific selection parameters clip = clip data according to values quack = remove/keep specific time range at scan beginning/end shadow = remove antenna-shadowed data elevation = remove data below/above given elevations tfcrop = automatic identification of outliers on the time-freq plane rflag = automatic detection of outliers based on sliding-window RMS filters extend = extend and/or grow flags beyond what the basic algorithms detect summary = report the amount of flagged data unflag = unflag the specified data

Arguments

Inputs	
vis	Name of MS file or calibration table to flag allowed: string Default:
mode	Flagging mode allowed: string Default: manual
autocorr	Flag only the auto-correlations allowed: bool Default: False
infile	Input ASCII file, list of files or Python list of strings with flag commands. allowed: any Default: variant
reason	Select by REASON types allowed: any Default: variant any
tbuff	List of time buffers (sec) to pad timerange in flag commands allowed: any Default: variant 0.0
spw	Spectral-window/frequency/channel: " ==> all, spw='0:17~19' allowed: any Default: variant
field	Field names or field index numbers: " ==> all, field='0~2,3C286' allowed: any Default: variant
antenna	Antenna/baselines: " ==> all, antenna ='3,VA04' allowed: any Default: variant
uvrange	UV range: " ==> all; uvrange ='0~100klambda', default units=meters allowed: any Default: variant
timerange	Time range: " ==> all,timerange='09:14:0~09:54:0' allowed: any Default: variant
correlation	Correlation: " ==> all, correlation='XX,YY' allowed: any Default: 133 variant
scan	Scan numbers: " ==> all allowed: any Default: variant
intent	Scan intent: " ==> all, intent='CAL*POINT*' allowed: any Default: variant

Returns

void

Example

----- Detailed description of keyword arguments -----

```
vis -- Name of input visibility file or calibration table.
      default: '' (none)
      example1: vis='uid___A002_X2a5c2f_X54.ms' or
      example2: vis='cal-X54.B1'
```

Any flagging will only be applied to the specified selections.

```
antenna -- Select data based on baseline
          default: '' (all); example: antenna='DV04&DV06' baseline DV04-DV06
          antenna='DV04&DV06;DV07&DV10' #baselines DV04-DV06 and DV07-DV10
          antenna='DV06' # all cross-correlation baselines between antenna DV06 and
                        all other available antennas
          antenna='DV04,DV06' # all baselines with antennas DV04 and DV06
          antenna='DV06&&DV06' # only the auto-correlation baselines for antenna DV06
          antenna='DV04&&*' # cross and auto-correlation baselines between antenna DV04
                        and all other available antennas
          antenna='0~2&&&' # only the auto-correlation baselines for antennas
                        in range 0~2
```

Note that for some antenna-based calibration tables, selecting baselines with the & syntax do not apply.

```
spw -- Select data based on spectral window and channels
      default: '' (all); example: spw='1'
      spw='<2' #spectral windows less than 2
      spw='>1' #spectral windows greater than 1
      spw='1:0~10' # first 10 channels from spw 1
      spw='1:0~5;120~128' # multiple separated channel chunks.
```

Note : For modes clip, tfcrop and rflag, channel-ranges can be excluded from flagging by leaving them out of the selection range. This is a way to protect known spectral-lines from being flagged by the autoflag algorithms. Example: if spectral-lines fall in channels 6~9, set the selection range to spw='0:0~5;10~63'.

correlation -- Correlation types or expression.
 default: '' (all correlations)
 For modes clip, tfcrop or rflag, the default means ABS_ALL. If the input is cal table that does not contain a complex data column, the default will fall back to REAL_ALL.
 example: correlation='XX,YY' or
 options: Any of 'ABS', 'ARG', 'REAL', 'IMAG', 'NORM' followed by any of 'ALL', 'I', 'XX', 'YY', 'RR', 'LL', 'WVR' 'WVR' refers to the water vapour radiometer of ALMA data.

For calibration tables, the solutions are: 'Sol1', 'Sol2', Sol3, Sol4.

example: correlation='REAL_XX,XY'
 -->correlation selection is not supported for modes other than clip, tfcrop or rflag in cal tables.

Note that the operators ABS,ARG,REAL, etc. are written only once as the first value. if more than one correlation is given, the operator will be applied to all of them.

The expression is used only in modes clip, tfcrop and rflag.

field -- Select data based on field id(s) or name(s)
 default: '' (all); example: field='1'
 field='0~2' # field ids inclusive from 0 to 2
 field='3C*' # all field names starting with 3C

uvrange -- Select data within uvrange (default units meters)
 default: '' (all); example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
 uvrange='0~1000km'; uvrange in kilometers
 -->uvrange selection is not supported for cal tables.

timerange -- Select data based on time range:
 default = '' (all); example,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:0~09:54:0' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time

scan -- Select data based on scan number
 default: '' (all); example: scan='>3'


```
intent -- Select data based on scan intent
    default: '' (all); example: intent='*CAL*,*BAND*'
    -->intent selection is not supported for cal tables.
```

```
array -- Selection based on the antenna array
    default: '' (all);
    -->array selection is not supported for cal tables.
```

```
observation -- Selection based on the observation ID
    default: '' (all); example: observation='1' or observation=1
```

```
feed -- Selection based on the feed - NOT IMPLEMENTED YET
```

```
mode -- Mode of operation.
    options: 'list', 'manual','clip','quack','shadow','elevation', 'tfcrop', 'extend',
            'unflag', 'summary'
    default: 'manual'
```

----- LIST MODE -----

```
list -- Flag according to the data selection and flag commands specified in the input
    The input list may come from a text file, a list of text files or from a Python
    list of strings. Each input line may contain data selection parameters and any
    parameters specific to the mode given in the line. Default values will be used for
    the parameters that are not present in the line. Each line will be taken
    as a command to the task. If data is pre-selected using any of the selection
    parameters, then flagging will apply only to that subset of the MS.
```

For optimization and whenever possible, the task will create a union of the
parameters present in the list and select only that portion of the MS.

NOTE: the flag commands will be applied only when action='apply'. If
action='calculate' the flags will be calculated, but not applied.
This is useful if display is set to something other than 'none'. If
action='' or 'none', the flag commands will not be applied either.
An empty action is useful only to save the parameters of the list
to a file or to the FLAG_CMD sub-table.

```
infile -- Input ASCII file, list of files or a Python list of command strings
    default: ''
    options: [] with flag commands or
            [] with filenames or
            '' with a filename.
```

IMPORTANT: From CASA 4.3 onwards, the parser will be strict and accept only valid flagdata parameters in the list. It will check each parameter name and type and exit with an error if any of them is wrong.

String values must contain quotes around them or the parser will not work. The parser evaluates the commands in the list and considers only existing Python types.

NOTE: There should be no whitespace between KEY=VALUE since the parser first breaks command lines on whitespace, then on "=". Use only one space to separate the parameters (no commas). Scroll down to the bottom for a detailed description of the input list syntax.

Example1: the following commands can be saved to a file or group of files and given to the task (e.g. save it to flags.txt).

```
scan='1~3' mode='manual'
mode='clip' clipminmax=[0,2] correlation='ABS_XX' clipoutside=False
spw='9' mode='tfcrop' correlation='ABS_YY' ntime=51.0
mode='extend' extendpols=True
```

```
flagdata(vis, mode='list', infile='flags.txt') or
flagdata(vis, mode='list', infile=['onlineflags.txt', 'otherflags.txt'])
```

Example2: the same commands can be given in a Python list on the command line to the task.

```
cmd=["scan='1~3' mode='manual'",
     "mode='clip' clipminmax=[0,2] correlation='ABS_XX' clipoutside=False",
     "spw='9' mode='tfcrop' correlation='ABS_YY' ntime=51.0",
     "mode='extend' extendpols=True"]
```

```
flagdata(vis, mode='list', infile=cmd)
```

reason -- select flag commands based on REASON(s) .

default: 'any' (all flags regardless of reason)

can be a string, or list of strings

example: reason='FOCUS_ERROR'

```
reason=['FOCUS_ERROR', 'SUBREFLECTOR_ERROR']
```

If infile is a list of files, the reasons given in this parameter will apply to all the files.

NOTE: what is within the string is literally matched, e.g. reason='' matches only blank reasons, and reason = 'FOCUS_ERROR,SUBREFLECTOR_ERROR' matches this compound reason string only.

See the syntax for writing flag commands at the end of this help.

`tbuff` -- A time buffer or list of time buffers to pad the timerange parameter in flag commands. When a list of 2 time buffers is given, it will subtract the first value from the lower time and the second value will be added to the upper time in the range. The 2 time buffer values can be different, allowing to have an irregular time buffer padding to time ranges. If the list contains only one time buffer, it will use it to subtract from `t0` and add to `t1`. If more than one list of input files is given, `tbuff` will apply to all of the flag commands that have timerange parameter. Each `tbuff` value should be a Float number given in seconds.
default: 0.0 (it will not apply any time padding)

example: `tbuff=[0.5, 0.8]`
 `infile=['online.txt','userflags.txt']`

The timeranges in the `online.txt` file are first converted to seconds. Then, 0.5 is subtracted from `t0` and 0.8 is added to `t1`, where `t0` and `t1` are the two intervals given in timerange. Similarly, `tbuff` will be applied to any timerange in `userflags.txt`.

IMPORTANT: This parameter assumes that `timerange = t0 ~ t1`, therefore it will not work if only `t0` or `t1` is given.

NOTE: The most common use-case for `tbuff` is to apply the online flags that are created by `importasdm` when `savecmds=True`. The value of `tbuff` for a regular time buffer should be `tbuff=0.5*max(integration time)`.

----- MANUAL MODE -----

`manual` -- Flag according to the data selection specified.
 This is the default mode (used when the mode is not specified).

`autocorr` -- Flag only the auto-correlations. Note that this parameter is only active when set to True. If set to False it does NOT mean "do not flag auto-correlations". When set to True, it will only flag data from a processor of type CORRELATOR.
default: False
options: True,False

----- CLIP MODE -----

`clip` -- Clip data according to values of the following subparameters. The polarization

expression is given by the correlation parameter. For calibration tables, the solutions are also given by the correlation parameter.

`datacolumn` -- Column to use for clipping.
default: 'DATA'
options: MS columns: 'DATA', 'CORRECTED', 'MODEL', 'RESIDUAL', 'RESIDUAL_SPECTRUM', 'WEIGHT_SPECTRUM', 'WEIGHT', 'FLOAT_DATA'.
Cal table columns: 'FPARAM', 'CPARAM', 'SNR', 'WEIGHT'.

NOTE1: $\text{RESIDUAL} = \text{CORRECTED} - \text{MODEL}$
 $\text{RESIDUAL_DATA} = \text{DATA} - \text{MODEL}$

NOTE2: when `datacolumn` is `WEIGHT`, the task will internally use `WEIGHT_SPECTRUM`. If `WEIGHT_SPECTRUM` does not exist, it will create one on-the-fly based on the values of `WEIGHT`.

`clipminmax` -- Range of data (Jy) that will NOT be flagged.
default: []; it will flag only NaN and Infs.
example: [0.0,1.5]
It will always flag the NaN/Inf data, even when a range is specified.

`clipoutside` -- Clip OUTSIDE the range?
default: True
example: False; flag data WITHIN the `clipminmax` range.

`channelavg` -- Pre-average data across channels before analyzing visibilities. Partially flagged data is not included in the average unless contributing to a given output channel is flagged. If present, `SIGMA_SPECTRUM` are used to compute a weighted average (`WEIGHT_SPECTRUM` for `CORRECTED_DATA` and `SIGMA_SPECTRUM` for `DATA`).
default: False
options: True/False

NOTE1: Pre-average across channels is not supported in list mode.
NOTE2: Pre-average across channels is not supported for calibration tables.

`chanbin` -- Bin width for channel average in number of input channels. If a list is given, each bin applies to one of the selected SPWs. When `chanbin` is set to 1 all input channels are used considered for average to produce a single output channel, this behaviour aims to be backwards compatible with the previous pre-averaging feature of the task.
default: 1

`timeavg` -- Pre-average data across time before analyzing visibilities for flagged data. Partially flagged data is not included in the average unless contributing to a given output channel is flagged. If present, `WEIGHT_SPECTRUM` are used to compute a weighted average (`WEIGHT_SPECTRUM` for `CORRECTED_DATA` and `SIGMA_SPECTRUM` for `DATA`).

SIGMA_SPECTRUM are used to compute a weighted average (WEIGHT_SPECTRUM * CORRECTED_DATA and SIGMA_SPECTRUM for DATA). Otherwise WEIGHT/SIGMA_SPECTRUM are averaged together data from different integrations.

default: False
options: True/False

NOTE1: Pre-average across time is not supported in list mode.
NOTE2: Pre-average across time is not supported for calibration targets.

timebin -- Bin width for time average in seconds.
default: '0s'

clipzeros -- Clip zero-value data.
default: False

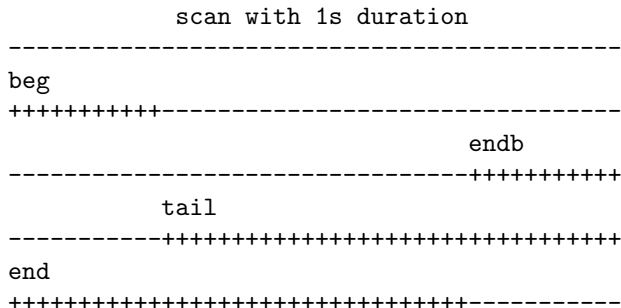
----- QUACK MODE -----

quack -- Option to remove specified part of scan beginning/end.

quackinterval -- Time in seconds from scan beginning or end to flag. Make timebin smaller than the desired time.
default: 0.0

quackmode -- Quack mode
default: 'beg'
options: 'beg' ==> flag an interval at the beginning of scan
'endb' ==> flag an interval at the end of scan
'tail' ==> flag all but an interval at the beginning of scan
'end' ==> flag all but an interval at end of scan

Visual representation of quack mode flagging one scan with 1s duration.
The following diagram shows what is flagged for each quack mode.
quackinterval is set to 0.25s. The flagged part is represented by +.



```

quackincrement -- Quack incrementally in time?
    default: False
        False ==> the quack interval is counted from the
                    beginning of the scan
        True  ==> the quack interval is counted from the
                    first unflagged data in the scan

shadow -- Option to flag data of shadowed antennas. This mode is not available
        for cal tables.

All antennas in the antenna-subtable of the MS (and the corresponding
diameters) will be considered for shadow-flag calculations.
For a given timestep, an antenna is flagged if any of its baselines
(projected onto the uv-plane) is shorter than radius_1 + radius_2 - tolerance.
The value of 'w' is used to determine which antenna is behind the other.
The phase-reference center is used for antenna-pointing direction.

tolerance -- Amount of shadowing allowed (or tolerated), in meters.
    A positive number allows antennas to overlap in projection
    A negative number forces antennas apart in projection
    Zero implies a distance of radius_1+radius_2 between antenna centers
    default: 0.0

addantenna -- It can be either a file name with additional antenna names, positions
             and diameters, or a Python dictionary with the same information.
             You can use the flaghelper functions to create the dictionary from a file.
    default: ''

To create a dictionary inside casapy.
> import flaghelper as fh
> antdic = fh.readAntennaList(antfile)

Where antfile is a text file in disk that contains information such as:
name=VLA01
diameter=25.0
position=[-1601144.96146691, -5041998.01971858, 3554864.76811967]
name=VLA02
diameter=25.0
position=[-1601105.7664601889, -5042022.3917835914, 3554847.245159178]

```

----- ELEVATION MODE -----

```

elevation -- Option to flag based on antenna elevation. This mode is not available
           for cal tables.

```

lowerlimit -- Lower limiting elevation in degrees. Data coming from a baseline where one or both antennas were pointing at a strictly lower elevation (as function of time), will be flagged.
default: 0.0

upperlimit -- Upper limiting elevation in degrees. Data coming from a baseline where one or both antennas were pointing at a strictly higher elevation (as function of time), will be flagged.
default: 90.0

----- TFCROP MODE -----

tfcrop -- Flag using the TFCrop autoflag algorithm.

- For each field, spw, timerange (specified by ntime), and 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 'stddev' of (data - fit).
 - (3) Flag points deviating from the fit by more than N-stddev
 - (4) Repeat (1-3) along the other dimension.

This algorithm is designed to operate on un-calibrated data (step (2)), as well as calibrated data. It is recommended to extend the flags after running this algorithm. See the sub-parameter extendflags below.

ntime -- Timerange (in seconds or minutes) over which to buffer data before running the algorithm.
options: 'scan' or any other float value or string containing the units.
default: 'scan'
example: '1.5min'

: 1.2 (taken in seconds)

The dataset will be iterated through in time-chunks defined here.

WARNING: if ntime='scan' and combinescans=True, all the scans will be loaded at once, thus requesting a lot of memory depending on the available spws.

combinescans -- Accumulate data across scans depending on the value of ntime.
default: False

This parameter should be set to True only when ntime is specified as time-interval (not 'scan'). When set to True, it will remove SCAN frequency sorting columns, therefore it will only accumulate across scans if ntime is not set to 'scan'.

`datacolumn` -- Column to use for flagging. (See also the `datacolumn` explanation)
 default: 'DATA'
 options: MS columns: 'DATA', 'CORRECTED', 'MODEL', 'RESIDUAL', 'RESIDUAL',
 'WEIGHT_SPECTRUM', 'WEIGHT', 'FLOAT_DATA'.
 Cal table columns: 'FPARAM', 'CPARAM', 'SNR', 'WEIGHT'.

`timecutoff` -- Flag threshold in time. Flag all data-points further than N-stds
 from the fit. This threshold catches time-varying RFI spikes (both
 narrow and broad-band), but will not catch RFI that is persistent.
 default: 4.0
 Flagging is done in upto 5 iterations. The stddev calculation is adaptive and
 converges to a value that reflects only the data and no RFI. At each iteration
 the same relative threshold is applied to detect flags. (Step (3))

`freqcutoff` -- Flag threshold in frequency. Flag all data-points further than N-stds
 from the fit.
 default: 3.0
 Same as `timecutoff`, but along the frequency-dimension. This threshold catches
 narrow-band RFI that may or may not be persistent in time.

`timefit` -- Fitting function for the time direction
 default: 'line'
 options: 'line', 'poly'
 A 'line' fit is a robust straight-line fit across the entire timerange selected by
 'ntime'.
 A 'poly' fit is a robust piece-wise polynomial fit across the timerange selected by
 'ntime'.
 Note: A robust fit is computed in upto 5 iterations. At each iteration the fit
 between the data and the fit is computed, values beyond N-stds from the fit
 and the fit and stddev are re-calculated with the remaining points. This
 stddev calculation is adaptive, and converges to a value that reflects only
 the data and no RFI. It also provides a varying set of flags that allows
 deep flagging only when the fit best represents the data.
 Choose 'poly' only if the visibilities are expected to vary significantly
 across the timerange selected by 'ntime', or if there is a lot of strong but
 intermittent RFI.

`freqfit` -- Fitting function for the frequency direction
 default: 'poly'
 options: 'line', 'poly'
 Same as for the 'timefit' parameter.
 Choose 'line' only if you are operating on bandpass-corrected data, and
 expect that the bandshape is linear. The 'poly' option works better for
 uncalibrated bandpasses with narrow-band RFI spikes.

`maxnpieces` -- Maximum number of pieces to allow in the piecewise-polynomial fit
 default: 7
 options: 1 - 9

This parameter is used only if 'timefit' or 'freqfit' are chosen as
 If there is significant broad-band RFI, reduce this number. Using too
 pieces could result in the RFI being fitted in the 'clean' bandpass.
 In later stages of the fit, a third-order polynomial is fit per piece.
 for best results, please ensure that nchan/maxnpieces is at-least 10

flagdimension -- Choose the directions along which to perform flagging
 default: 'freqtime'; first flag along frequency, and then along time
 options: 'time', 'freq', 'timefreq', 'freqtime'
 For most cases, 'freqtime' or 'timefreq' are appropriate, and differences
 between these choices are apparent only if RFI in one dimension is
 significantly stronger than the other. The goal is to flag the dominant
 If there are very few (less than 5) channels of data, then choose 'time'.
 Similarly for 'freq'.

usewindowstats -- Use sliding-window statistics to find additional flags.
 default: 'none'
 options: 'none', 'sum', 'std', 'both'
 Note: This is experimental !
 The 'sum' option chooses to flag a point, if the mean-value in a
 window centered on that point deviates from the fit by more than
 $N \cdot \text{stddev} / 2.0$.
 Note: stddev is calculated between the data and fit as explained in
 This option is an attempt to catch broad-band or
 time-persistent RFI that the above polynomial fits will mistake
 as the clean band. It is an approximation to the sumThreshold
 method found to be effective by Offringa et.al (2010) for LOFAR.
 The 'std' option chooses to flag a point, if the 'local' stddev
 in a window centered on that point is larger than $N \cdot \text{stddev} / 2.0$.
 This option is an attempt to catch noisy RFI that is not excluded by
 polynomial fits, and which increases the global stddev, and results in
 fewer flags (based on the $N \cdot \text{stddev}$ threshold).

halfwin -- Half width of sliding window to use with 'usewindowstats'
 default: 1 (a 3-point window size)
 options: 1,2,3
 Note: This is experimental !

extendflags -- Extend flags along time, frequency and correlation.
 default: True

NOTE: It is usually helpful to extend the flags along time, frequency,
 and correlation using this parameter, which will run the "extend"
 mode after "tfcrop" and extend the flags if more than 50% of the
 timeranges are already flagged, and if more than 80% of the channels
 are already flagged. It will also extend the flags to the other

polarizations. The user may also set extendflags to False and run the "extend" mode in a second step within the same flagging run. See the example below:

```
example :
cmd=["mode='tfcrop' freqcutoff=3.0 usewindowstats='sum' extendflags=False"
    "mode='extend' extendpols=True growtime=50.0 growaround=True"]
```

```
flagdata(vis, mode='list', infile=cmd)
```

channelavg -- Pre-average data across channels before analyzing visibilities. Partially flagged data is not included in the average unless contributing to a given output channel is flagged. If present, SIGMA_SPECTRUM are used to compute a weighted average (WEIGHT_SPECTRUM and SIGMA_SPECTRUM for DATA).

default: False
options: True/False

NOTE1: Pre-average across channels is not supported in list mode.
NOTE2: Pre-average across channels is not supported for calibration

chanbin -- Bin width for channel average in number of input channels. If a list is given, each bin applies to one of the selected SPWs. When chanbin is set to 1 all input channels are used considered for average to produce a single output channel.

default: 1

timeavg -- Pre-average data across time before analyzing visibilities for flagging. Partially flagged data is not included in the average unless contributing to a given output channel is flagged. If present, WEIGHT_SPECTRUM and SIGMA_SPECTRUM are used to compute a weighted average (WEIGHT_SPECTRUM and SIGMA_SPECTRUM for DATA). Otherwise WEIGHT_SPECTRUM and SIGMA_SPECTRUM are averaged together data from different integrations.

default: False
options: True/False

NOTE1: Pre-average across time is not supported in list mode.
NOTE2: Pre-average across time is not supported for calibration

timebin -- Bin width for time average in seconds.

default: '0s'

----- RFLAG MODE -----

rflag -- Detect outliers based on the RFlag algorithm (ref. E.Greisen, AIPS, 2011). The polarization expression is given by the correlation parameter.

Iterate through the data in chunks of time. For each chunk, calculate local statistics, and apply flags based on user supplied (or auto-calculated) thresholds.

Step 1 : Time analysis (for each channel)

```
-- calculate local rms of real and imag visibilities, within a sliding time window
-- calculate the median rms across time windows, deviations of local rms from
  this median, and the median deviation
-- flag if local rms is larger than timedevscale x (medianRMS + medianDeviation)
```

Step 2 : Spectral analysis (for each time)

```
-- calculate avg of real and imag visibilities and their rms across channels
-- calculate the deviation of each channel from this avg, and the median deviation
-- flag if deviation is larger than freqdevscale x medianDeviation
```

It is recommended to extend the flags after running this algorithm. See the sub-parameter `extendflags` below.

Notice that by default the flag implementation in CASA is able to calculate thresholds and apply them on-the-fly (OTF). There is a significant performance gain with this approach, as the visibilities don't have to be read twice, and therefore is highly recommended (see example 1). Otherwise it is possible to reproduce the AIPS usage pattern by doing a first run with 'calculate' mode, and a second run with 'apply' mode. The advantage of this approach is that the flags are calculated using the data from all scans, instead of calculating them from one scan only (see example 3)

Example usage :

- (1) Calculate thresholds automatically per scan, and use them to find flags. Specify scale-factor for time-analysis thresholds, use default for frequency-analysis thresholds.

```
flagdata('my.ms', mode='rflag', spw='9', timedevscale=4.0)
```

- (2) Supply noise-estimates to be used with default scale-factors.

```
flagdata(vis='my.ms', mode='rflag', spw='9', timedev=0.1, freqdev=0.5)
```

- (3) Two-passes. This replicates the usage pattern in AIPS.

```
-- The first pass saves commands in an output text files, with auto-calculated
  thresholds. Thresholds are returned from rflag only when action='calculate'.
  The user can edit this file before doing the second pass,
  but the python-dictionary structure must be preserved.
```

```
-- The second pass applies these commands (action='apply').
```

```
flagdata(vis='my.ms', mode='rflag', spw='9,10', timedev='tdevfile',
         freqdev='fdevfile.txt', action='calculate')
flagdata(vis='my.ms', mode='rflag', spw='9,10', timedev='tdevfile',
         freqdev='fdevfile.txt', action='apply')
```

With action='calculate', display='report' will produce diagnostic plots showing data-statistics and thresholds (the same thresholds as those written out to 'tdevfile.txt' and 'fdevfile.txt').

Note : The RFlag algorithm was originally developed by Eric Greisen AIPS (31DEC11).

AIPS documentation : Section E.5 of the AIPS cookbook
(Appendix E : Special Considerations for EVLA data calibration and <http://www.aips.nrao.edu/cook.html#CEE>)

Note1 : Since this algorithm operates with two passes through each chunk of data (time and freq axes), some data points get flagged twice. This can affect the flag-percentage estimate printed in the logger at runtime. An accurate estimate can be obtained via the summary mode.

Note2: RFlag calculates statistics across all selected correlations. Therefore, if there is a significant amplitude difference between parallel-hand and cross-hand correlations, or between different solutions in a gain table, it is advisable to pre-select subsets of correlations (or sols) on which to run one instance of RFlag. For example, correlation='RR,LL' or correlation='ABS sol1,sol2'.

ntime -- Timerange (in seconds or minutes) over which to buffer data before running the algorithm.

options: 'scan' or any other float value or string containing the units
default: 'scan'
example: '1.5min'
 : 1.2 (taken in seconds)

The dataset will be iterated through in time-chunks defined here.
WARNING: if ntime='scan' and combinescans=True, all the scans will be loaded at once, thus requesting a lot of memory depending on the available spws.

combinescans -- Accumulate data across scans depending on the value of ntime.
default: False

This parameter should be set to True only when ntime is specified as time-interval (not 'scan'). When set to True, it will remove SCAN from sorting columns, therefore it will only accumulate across scans if ntime is not set to 'scan'.

```

datacolumn -- Column to use for flagging. (See also the datacolumn explanation)
             default: 'DATA'
             options: MS columns: 'DATA', 'CORRECTED', 'MODEL', 'RESIDUAL', 'RESIDUAL_
                           'WEIGHT_SPECTRUM', 'WEIGHT', 'FLOAT_DATA'.
                           Cal table columns: 'FPARAM', 'CPARAM', 'SNR', 'WEIGHT'.

winsize -- number of timesteps in the sliding time window ( fparm(1) in AIPS )
          default: 3

timedev -- time-series noise estimate ( noise in AIPS ).
          default: []
          Examples :
              timedev = 0.5 : Use this noise-estimate to calculate flags. Do not
              timedev = [ [1,9,0.2], [1,10,0.5] ] : Use noise-estimate of 0.2 for
                  spw 9, and noise-estimate of 0.5 for field 1, spw 10.
              timedev = [] : Auto-calculate noise estimates.

freqdev -- spectral noise estimate ( scutoff in AIPS ).
          This step depends on having a relatively-flat bandshape.
          Same parameter-options as 'timedev'.
          default: []

timedevscale -- For Step 1 (time analysis), flag a point if local rms around
               is larger than 'timedevscale' x 'timedev' ( fparm(0) in AIPS )
               default: 5.0

freqdevscale -- For Step 2 (spectral analysis), flag a point if local rms around
               is larger than 'freqdevscale' x 'freqdev' ( fparm(10) in AIPS )
               default: 5.0

spectralmax -- Flag whole spectrum if 'freqdev' is greater than spectralmax ( fparm(11) )
              default: 1E6

spectralmin -- Flag whole spectrum if 'freqdev' is less than spectralmin ( fparm(12) )
              default: 0.0

extendflags -- Extend flags along time, frequency and correlation.
              default: True

```

NOTE: It is usually helpful to extend the flags along time, frequency, and correlation using this parameter, which will run the "extend" mode after "rflag" and extend the flags if more than 50% of the timeranges are already flagged, and if more than 80% of the channels are flagged.

are already flagged. It will also extend the flags to the other polarizations. The user may also set extendflags to False and run the "extend" mode in a second step within the same flagging run. See the example below:

example :

```
cmd=["mode='rflag' freqdevscale=3.0 extendflags=False",
     "mode='extend' extendpols=True growtime=50.0 growaround=True"]
```

```
flagdata(vis, mode='list', infile=cmd)
```

channelavg -- Pre-average data across channels before analyzing visibilities. Partially flagged data is not be included in the average unless contributing to a given output channel is flagged. If present, SIGMA_SPECTRUM are used to compute a weighted average (WEIGHT_SPECTRUM and SIGMA_SPECTRUM for DATA).

default: False

options: True/False

NOTE1: Pre-average across channels is not supported in list mode.

NOTE2: Pre-average across channels is not supported for calibration

chanbin -- Bin width for channel average in number of input channels.

If a list is given, each bin applies to one of the selected SPWs.

When chanbin is set to 1 all input channels are used considered for average to produce a single output channel.

default: 1

timeavg -- Pre-average data across time before analyzing visibilities for flags. Partially flagged data is not be included in the average unless contributing to a given output channel is flagged. If present, WEIGHT_SPECTRUM and SIGMA_SPECTRUM are used to compute a weighted average (WEIGHT_SPECTRUM and SIGMA_SPECTRUM for DATA). Otherwise WEIGHT/SIGMA_SPECTRUM are used to compute an average together data from different integrations.

default: False

options: True/False

NOTE1: Pre-average across time is not supported in list mode.

NOTE2: Pre-average across time is not supported for calibration

timebin -- Bin width for time average in seconds.

default: '0s'

----- EXTEND MODE -----

extend -- Extend and/or grow flags beyond what the basic algorithms detect. This mode will extend the accumulated flags available in the MS,

regardless of which algorithm created them.

It is recommended that any autoflag (tfcrop, rflag) algorithm be followed up by a flag extension.

Extensions will apply only within the selected data, according to the settings of extendpols, growtime, growfreq, growaround, flagneartime, flagnearfreq.

Note : Runtime summary counts in the logger can sometimes report larger flag percentages than what is actually flagged. This is because extensions onto already-flagged data-points are counted as new flags. An accurate flag count can be obtained via the summary mode.

ntime -- Timerange (in seconds or minutes) over which to buffer data before running the algorithm.

options: 'scan' or any other float value or string containing the units

default: 'scan'

example: '1.5min'

: 1.2 (taken in seconds)

The dataset will be iterated through in time-chunks defined here.

WARNING: if ntime='scan' and combinescans=True, all the scans will be loaded at once, thus requesting a lot of memory depending on the available spws.

combinescans -- Accumulate data across scans depending on the value of ntime.
default: False

This parameter should be set to True only when ntime is specified as a time-interval (not 'scan'). When set to True, it will remove SCAN frequency sorting columns, therefore it will only accumulate across scans if ntime is not set to 'scan'.

extendpols -- Extend flags to all selected correlations
default: True

options: True/False

For example, to extend flags from RR to only RL and LR, a data-selection of correlation='RR,LR,RL' is required along with extendpols=True.

growtime -- For any channel, flag the entire timerange in the current 2D chunk (set by 'ntime') if more than X% of the timerange is already flagged.
default: 50.0

options: 0.0 - 100.0

This option catches the low-intensity parts of time-persistent RFI.

growfreq -- For any timestep, flag all channels in the current 2D chunk (set by 'growtime') if more than X% of the channels are already flagged.
default: 50.0

options: 0.0 - 100.0
 This option catches broad-band RFI that is partially identified by

growaround -- Flag a point based on the number of flagged points around it.
 default: False
 options: True/False
 For every un-flagged point on the 2D time/freq plane, if more than 3
 surrounding points are already flagged, flag that point.
 This option catches some wings of strong RFI spikes.

flagneartime -- Flag points before and after every flagged one, in the time-domain.
 default: False
 options: True/False
 Note: This can result in excessive flagging.

flagnearfreq -- Flag points before and after every flagged one, in the frequency domain.
 default: False
 options: True/False
 This option allows flagging of wings in the spectral response of strong RFI.
 Note: This can result in excessive flagging

----- UNFLAG MODE -----

unflag -- Unflag according to the data selection specified.

----- SUMMARY MODE -----

summary -- List the number of rows and flagged data points for the MS's meta-data.
 The resulting summary will be returned as a Python dictionary.

minrel -- Minimum number of flags (relative) to include in histogram
 default: 0.0

maxrel -- Maximum number of flags (relative) to include in histogram
 default: 1.0

minabs -- Minimum number of flags (absolute, inclusive) to include in histogram
 default: 0

maxabs -- Maximum number of flags (absolute, inclusive) to include in histogram
 To indicate infinity, use any negative number.
 default: -1

spwchan -- list the number of flags per spw and per channel.
 default: False

spwcorr -- list the number of flags per spw and per correlation.
default: False

basecnt -- list the number of flags per baseline
default: False

fieldcnt -- produce a separated breakdown per field
default: False

name -- Name for this summary, to be used as a key in the returned
Python dictionary. It is possible to call the summary mode
multiple times in list mode. When calling the summary mode as a
command in a list, one can give different names to each one of
them so that they can be easily pulled out of the summary's dictionary.
default: 'Summary'

In summary mode, the task returns a dictionary of flagging statistics.

Example1:

```
s = flagdata(..., mode='summary')
```

Then s will be a dictionary which contains
s['total'] : total number of data
s['flagged'] : amount of flagged data

Example2: two summary commands in list mode, intercalating a manual flagging

```
s = flagdata(..., mode='list', infile=["mode='summary' name='InitFlags' autocorr=True",  
                                         "mode='manual' autocorr=True",  
                                         "mode='summary' name='Autocorr' autocorr=True"])
```

The dictionary returned in 's' will contain two dictionaries, one for each
two summary modes.

```
s['report0']['name'] : 'InitFlags'  
s['report1']['name'] : 'Autocorr'
```

----- ACTIONS -----

action -- Action to perform in MS/cal table or in the input list of parameters.
options: 'none', 'apply', 'calculate'
default: 'apply'

```

'apply' -- Apply the flags to the MS.

display -- Display data and/or end-of-MS reports at run-time. It needs to read
          a datacolumn for the plotting. The default for an MS is DATA, but
          will use FLOAT_DATA for a Sindle-dish MS.
default: 'none'
options: 'none', 'data', 'report', 'both'

'none' --> It will not display anything.

'data' --> display data and flags per-chunk at run-time, within an interactive
          GUI.

          This option opens a GUI to show the 2D time-freq planes of
          the data with old and new flags, for all correlations per baseline.
          -- The GUI allows stepping through all baselines (prev/next) in
          the current chunk (set by 'ntime'), and stepping to the next-chunk.
          -- The 'flagdata' task can be quit from the GUI, in case it becomes
          obvious that the current set of parameters is just wrong.
          -- There is an option to stop the display but continue flagging.

'report' --> displays end-of-MS reports on the screen.

'both' --> displays data per chunk and end-of-MS reports on the screen.

flagbackup -- Automatically backup flags before running the tool.
             Flagversion names are chosen automatically, and are based on the
             mode being used.
default: True
options: True/False

'calculate' -- Only calculate the flags but do not write them to the MS. This is
               useful if used together with the display to analyse the results before
               writing to the MS.

display -- Display data and/or end-of-MS reports at run-time. See extended description
          above.
default: 'none'
options: 'none', 'data', 'report', 'both'

' ' -- When set to empty, the underlying tool will not be executed and no flags
       will be produced. No data selection will be done either. This is useful
       when used together with the parameter savepars to only save the current
       parameters (or list of parameters) to the FLAG_CMD sub-table or to an
       external file.

```

`savepars` -- Save the current parameters to the FLAG_CMD table of the MS or to an output text file. Note that when `display` is set to anything other than 'none', `savepars` will be disabled. This is done because in an interactive mode, the user may skip data which may invalidate the initial input parameters and there is no way to save the interactive commands. When the input is a calibration table it is only possible to save the parameters to a file.
 default: False
 options: True/False

`cmdreason` -- A string containing a reason to save to the FLAG_CMD table or to an output text file given by the `outfile` sub-parameter. If the input contains any reason, they will be replaced with this one. At the moment it is not possible to add more than one reason.
 default: ' '; no reason will be added to output.
 example: `cmdreason='CLIP_ZEROS'`

`outfile` -- Name of output file to save the current parameters.
 default: ' '; it will save the parameters to the FLAG_CMD table of the MS.
 example: `outfile='flags.txt'` will save the parameters in a text file.

`overwrite` -- Overwrite the existing file given in 'outfile'.
 options: True/False
 default: True; it will remove the existing file given in 'outfile' and save the commands to a new file with the same name. When set to False, it will give an error message if the file exist.

---- EXAMPLES ----

NOTE: The vector mode of the `flagdata` task (pre-dating CASA 3.4) can be achieved with the `flagdata` task by using it with `mode='list'` and the commands given in a list in `inpmode=[]`. Example:

```
flagdata('my.ms', inpmode='list', infile=["mode='clip' clipzeros=True","mode='shadow'"])
```

1) Manually flag scans 1~3 and save the parameters to the FLAG_CMD sub-table.

```
flagdata('my.ms', scan='1~3', mode='manual', savepars=True)
```

2) Save the parameters to a file that is open in append mode.

```
flagdata('my.ms', scan='1~3', mode='manual', savepars=True, outfile='flags.txt')
```

3a) Flag all the commands given in the Python list of strings.

```

cmd = ["scan='1~3' mode='manual'",
       "spw='9' mode='tfcrop' correlation='ABS_RR,LL' ntime=51.0",
       "mode='extend' extendpols=True"]

flagdata('my.ms', mode='list', infile=cmd)

```

3b) Flag all the commands given in the file called flags.txt.

```

> cat flags.txt
scan='1~3' mode='manual'
spw='9' mode='tfcrop' correlation='ABS_RR,LL' ntime=51.0
mode='extend' extendpols=True

flagdata('my.ms', mode='list', infile='flags.txt')

```

4) Display the data and flags per-chunk and do not write flags to the MS.

```

flagdata('my.ms', mode='list', infile='flags.txt', action='calculate', display='data')

```

5) Flag all the antennas except antenna=5.

```

flagdata(vis='my.ms', antenna='!5', mode='manual')

```

6) Clip the NaN in the data. An empty clipminmax will flag only NaN.

```

flagdata('my.ms', mode='clip')

```

7) Clip only the water vapour radiometer data.

```

flagdata('my.ms', mode='clip', clipminmax=[0,50], correlation='ABS_WVR')

```

8) Clip only zero-value data.

```

flagdata('my.ms', mode='clip', clipzeros=True)

```

9a) Flag only auto-correlations of non-radiometer data using the autocorr parameter.

```

flagdata('my.ms', autocorr=True)

```

9b) Flag only auto-correlations using the antenna selection.

```

flagdata('my.ms', mode='manual', antenna='*&&&')

```

10a) Flag based on selected reasons from a file.

```

> cat flags.txt

```

```
scan='1~3' mode='manual' reason='MYREASON'
spw='9' mode='clip' clipzeros=True reason='CLIPZEROS'
mode='manual' scan='4' reason='MYREASON'
```

```
flagdata('my.ms', mode='list', infile='flags.txt', reason='MYREASON')
```

10b) The same result of 10a can be achieved using the task `flagcmd`.

```
flagcmd('my.ms', inpmode='file', infile='flags.txt', action='apply', reason='MYREASON')
```

11) Automatic flagging using `rflag`, using auto-thresholds, and specifying a threshold scale-factor to use for flagging.

```
flagdata('my.ms', mode='rflag', spw='9', timedevscale=4.0, action='apply')
```

12) Save the interface parameters to the `FLAG_CMD` sub-table of the MS. Add a reason to the flag command. This `cmdreason` will be added to the `REASON` column of the `FLAG_CMD` sub-table. Apply flags in `flagcmd`.

```
flagdata('my.ms', mode='clip', channelavg=False, clipminmax=[30., 60.], spw='0:0~10',
correlation='ABS_XX,XY', action='', savepars=True, cmdreason='CLIPXX_XY')
```

> Select based on the reason.

```
flagcmd('my.ms', action='apply', reason='CLIPXX_XY')
```

13) Flag antennas that are shadowed by antennas not present in the MS.

> Create a text file with information about the antennas.

> cat ant.txt

```
name=VLA01
```

```
diameter=25.0
```

```
position=[-1601144.96146691, -5041998.01971858, 3554864.76811967]
```

```
name=VLA02
```

```
diameter=25.0
```

```
position=[-1601105.7664601889, -5042022.3917835914, 3554847.245159178]
```

```
name=VLA09
```

```
diameter=25.0
```

```
position=[-1601197.2182404203, -5041974.3604805721, 3554875.1995636248]
```

```
name=VLA10
```

```
diameter=25.0
```

```
position=[-1601227.3367843349, -5041975.7011900628, 3554859.1642644769]
```

```
flagdata('my.vis', mode='shadow', tolerance=10.0, addantenna='ant.txt')
```

The antenna information can also be given as a Python dictionary. To create the

dictionary using the flaghelper functions, do the following inside casapy:

```
> import flaghelper as fh
> antdic = fh.readAntennaList(antfile)
```

```
flagdata('my.vis', mode='shadow', tolerance=10.0, addantenna=antdic)
```

14) Apply the online flags that come from importasdm.

```
> In importasdm, save the online flags to a file.
```

```
importasdm('myasdm', 'asdm.ms', process_flags=True, savecmds=True, outfile='online')
```

```
> You can edit the online_flags.txt to add other flagging commands or apply it directly
```

```
flagdata('asdm.ms', mode='list', infile='online_flags.txt')
```

```
> The same result can be achieved using the task flagcmd.
```

```
flagcmd('asdm.ms', inpmode='file', infile='online_flags.txt', action='apply')
```

15) Clip mode pre-averaging data across channels and across time

```
flagdata(vis='Four_ants_3C286.ms', flagbackup=False, mode='clip', datacolumn='DATA',
timeavg=True, timebin='2s', channelavg=True, chanbin=2)
```

----- EXAMPLES on FLAGGING CALIBRATION TABLES -----

16) Clip zero data from a bandpass calibration table.

```
flagdata('cal-X54.B1', mode='clip', clipzeros=True, datacolumn='CPARAM')
```

17) Clip data from a cal table with SNR <4.0.

```
flagdata('cal-X54.B1', mode='clip', clipminmax=[0.0,4.0], clipoutside=False, datacolumn='DATA')
```

18) Clip the g values of a switched power caltable created using the gencal task. The g values are usually < 1.0.

```
flagdata('cal.12A.syspower', mode='clip', clipminmax=[0.1,0.3], correlation='Sol1,Sol3')
```

19) Now, clip the Tsys values of the same table from above. The Tsys solutions have values between 10 -- 100s.

```
flagdata('cal.12A.syspower', mode='clip', clipminmax=[10.0,95.0], correlation='Sol2,Sol3')
```

---- SYNTAX FOR COMMANDS GIVEN IN A FILE or LIST OF STRINGS ----

Basic Syntax Rules

Commands are strings (which may contain internal "strings") consisting of KEY=VALUE pairs separated by one whitespace only.

NOTE: There should be no whitespace between KEY=VALUE. The parser first breaks command lines on whitespace, then on "=".

Use only ONE white space to separate the parameters (no commas).

Each key should only appear once on a given command line/string.

There is an implicit "mode" for each command, with the default being 'manual' if not given.

Comment lines can start with '#' and will be ignored.

The parser used in flagdata will check each parameter name and type and exit with an error if the parameter is not a valid flagdata parameter or of a wrong type.

Example:

```
scan='1~3' mode='manual'
# this line will be ignored
spw='9' mode='tfcrop' correlation='ABS_XX,YY' ntime=51.0
mode='extend' extendpols=True
scan='1~3,10~12' mode='quack' quackinterval=1.0
```

flagmanager-task.html

0.1.30 flagmanager

Requires:

Synopsis

Enable list, save, restore, delete and rename flag version files.

Description

These flag version files are copies of the flag column for a measurement set. They can be restored to the data set to get back to a previous flag version. On running importvla, a flag version call 'Original' is automatically produced.

Arguments

Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
mode	Operation: list, save, restore, delete, rename allowed: string Default: list
versionname	Flag version name allowed: string Default:
oldname	Flag version to rename allowed: string Default:
comment	Short description of a versionname allowed: string Default:
merge	Merge option: replace will save or over-write the flags allowed: string Default: replace

Returns

void

Example

The flag version files are copies of the FLAG column of a Measurement Set. They can be restored to the data set to obtain a previous flag version. On running importasdm, a flag version called 'Original' is produced by default. It is recommended to save a flagversion at the beginning or after serious editing.

Keyword arguments:

vis -- Name of input visibility file
 default: none. example: vis='ngc5921.ms'

mode -- Flag version operation
 default: 'list'; to list existing flagtables

'save': will save the FLAG column from vis to a specified flag file. If the in versionname already exists, the task will give a warning and rename to a name with a suffix '.old.timestamp'. The respective entry in FLAG will also be updated.

'restore': will place the specified flag file into vis

'delete': will delete specified flag file

'rename': will rename a specified flag file

versionname -- Flag version name
 default: none; example: versionname='original_data'
 No imbedded blanks in the versionname

comment -- Short description of a versionname, when mode is 'save' or 'rename'
 default: ''; example: comment='Clip above 1.85'
 comment = versionname

oldname -- When mode='rename', the flag file to rename

merge -- Merge operation
 Options: 'or', 'and', but not recommended for now.

fluxscale-task.html

0.1.31 fluxscale

Requires:

Synopsis

Bootstrap the flux density scale from standard calibrators

Description

Bootstrap the flux density scale from standard calibrators:

After running gaincal on standard flux density calibrators (with or without an image model), and other calibrators with unknown flux densities (assumed 1 Jy), fluxscale applies the constraint that net system gain was, in fact, independent of field, on average, and that field-dependent gains in the input caltable are solely a result of the unknown flux densities for the calibrators. Using time-averaged gain amplitudes, the ratio between each ordinary calibrator and the flux density calibrator(s) is formed for each antenna and polarization (that they have in common). The average of this ratio over antennas and polarizations yields a correction factor that is applied to the ordinary calibrators' gains. (See also more detailed discussion in Example section below.)

Arguments

Outputs	
fluxd	Dictionary containing the transfer fluxes and their errors. allowed: any Default: variant
Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
caltable	Name of input calibration table allowed: string Default:
fluxtable	Name of output, flux-scaled calibration table allowed: string Default:
reference	Reference field name(s) (transfer flux scale FROM) allowed: stringArray Default:
transfer	Transfer field name(s) (transfer flux scale TO), " -> all allowed: stringArray Default:
listfile	Name of listfile that contains the fit information. Default is " (no file). allowed: string Default:
append	Append solutions? allowed: bool Default: False
refspwmap	Scale across spectral window boundaries. See help fluxscale allowed: intArray Default: -1
gainthreshold	Threshold (% deviation from the median) on gain amplitudes to be used in the flux scale calculation allowed: double Default: -1.0
antenna	antennas to include/exclude allowed: string Default:
timerange	sub selection by timerange allowed: string Default:
scan	sub selection by scan allowed: string Default:
incremental	incremental caltable allowed: bool Default: False
fitorder	order of spectral fitting allowed: int Default: 1
display	display some statistics of flux scaling allowed: bool Default: False

Returns

void

Example

After running gaincal on standard flux density calibrators (with or without an image model), and other calibrators with unknown flux densities (assumed 1 Jy), fluxscale applies the constraint that net system gain was, in fact, independent of field, on average, and that field-dependent gains in the input caltable are solely a result of the unknown flux densities for the calibrators. Using time-averaged gain amplitudes, the ratio between each ordinary calibrator and the flux density calibrator(s) is formed for each antenna and polarization (that they have in common). For incremental=False(default), the median of this ratio over antennas and polarizations yields a correction factor that is applied to the ordinary calibrators' gains. For incremental=True, only the correction factors are written out to the output fluxtable.

The square of the gain correction factor for each calibrator and spw is the presumed flux density of that calibrator, and is reported in the logger. The errors reported with this value reflect the scatter in gain ratio over antennas and polarizations, divided by the square root of the number of antennas and polarizations available. If the flux densities for multiple spws exist, fitted spectral index and (for nspw>2) curvature are also reported. The fit is done for $\log(\text{flux density}) = a_0 + a_1 \cdot (\log(\text{frequency})) + a_2 \cdot (\log(\text{frequency}))^2$ where $\log(\text{frequency})$ is with respect to the mean of $\log(\text{frequency})$. This reference frequency is reported in the logger along with the flux density at that frequency. The fit results are also reported in the returned Python dictionary (the solutions are in 'spidx' in the following order: a_0 [log(S) at the zero point], a_1 [spectral index], and a_2 [curvature]. And their errors are in 'spidxerr'). The MODEL_DATA column is currently _not_ revised to reflect the flux densities derived by fluxscale. Use setjy to set the MODEL_DATA column, if necessary.

The constant gain constraint is usually a reasonable assumption for the electronic systems on typical antennas. It is

important that external time- and/or elevation-dependent effects are separately accounted for when solving for the gain solution supplied to fluxscale, e.g., gain curves, opacity, etc. The fluxscale results can also be degraded by poor pointing during the observation. The parameters, gainthreshold and antenna (and timerange/scan) can be used to control the data to be used in the flux derivation in such cases. The gainthreshold parameter sets the range of the input gain to be used in terms of the percentage deviation from their median values (per field, per spectral window). When the antenna parameter is specified, the sub-parameters timerange and scan are also available to fine tune the data selection for the flux derivation. These parameters uses the general CASA data selection (msselection) syntax. And these are 'AND' operations except when the antenna selection is specified with a negation (e.g. antenna="!6"). In that case, timerange and scan applied to only those antennas appear in the antenna parameter. So, for example, timerange='>02:35:00' with antenna='!6,24', will include the data with time greater than 02:35:00 for antenna ID 6 and 24 but for other antennas the timerange selection is not applied.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'
caltable -- Name of input calibration table
      default: none; example: caltable='ngc5921.gcal'
      This cal table was obtained from task gaincal.
fluxtable -- Name of output, flux-scaled calibration table
      default: none; example: fluxtable='ngc5921.gcal2'
      The gains in this table have been adjusted by the
      derived flux density each calibrator. The MODEL_DATA
      column has NOT been updated for the flux density of the
      calibrator. Use setjy to do this if it is a point source.
reference -- Reference field name(s)
      The names of the fields with a known flux densities or
      visibilities that have been placed in the MODEL column
      by setjy or ft for a model not in the CASA system.
      The syntax is similar to field. Hence field index or
      names can be used.
      default: none; example: reference='1328+307'
transfer -- Transfer field name(s)
      The names of the fields with unknown flux densities.
      These should be point-like calibrator sources
      The syntax is similar to field. Hence source index or
      names can be used.
      default: '' = all sources in caltable that are not specified
      as reference sources. Do not include unknown target sources
```

example: transfer='1445+099, 3C84'; transfer = '0,4'

NOTE: All fields in reference and transfer must have solutions in the caltable.

listfile -- Fit listfile name

The list file contains the flux density, flux density error, S/N, and number of solutions (all antennas and feeds) for each spectral window. NOTE: The nominal spectral window frequencies will be included in the future.

default: '' = no fit listfile will be created.

append -- Append fluxscaled solutions to the fluxtable.

default: False; (will overwrite if already existing)

example: append=True

refspwmap -- Vector of spectral windows enabling scaling across spectral windows

default: [-1]==> none.

Example with 4 spectral windows:

if the reference fields were observed only in spw=1 & 3, and the transfer fields were observed in all 4 spws (0,1,2,3), specify refspwmap=[1,1,3,3].

This will ensure that transfer fields observed in spws 0,1,2,3 will be referenced to reference field solutions only in spw 1 or 3.

gainthreshold -- Threshold in the input gain solutions to be used in % deviation from median values.

default: -1 (no threshold)

example: gainthreshold=0.15 (only used the gain solutions within 15% (inclusive) of the median gain value (per field and per spw)).

antenna --- Antenna selection to be included in the fluxscale determination.

General ms selection syntax is accepted such as antenna id (given as a string) and antenna name.

default: '' (=All antennas will be used)

example: antenna='!23' (exclude antenna id, 23)

* Following sub-parameters are available when the antenna parameter is specified

timerange --- Select time range using the msselection syntax.

If the negation (e.g. '!23') is used in the antenna selection, it will apply the time range selection only to the negated antenna(s). Otherwise, the selection is global (i.e. applied to all antenna and to both reference and transfer fields).

default: '' (all timerange)

example: timerange=">0:58:00"

scan --- Select scan(s) using the msselection syntax. As in the case of the timerange, the selection will be applied to only the negated antenna(s) if the antenna parameter is used with the negation ("!").

default: '' (all scans)
example: '2~5'

incremental -- Create an incremental caltable containing only gain correction factors (flux density= 1/(gain correction factor)**2)
default: False; (older behavior = create flux-scaled gain table)
example: incremental=True (output a caltable containing flux scale factors.)

NOTE: If you use the incremental option, note that BOTH this incremental fluxscale table AND an amplitude vs. time table should be supplied in applycal

fitorder -- Polynomial order of the spectral fitting for valid flux densities with multiple spws. Currently only support 1 (spectral index only) or 2 (spectral index and curvature). It falls back to a lower fitorder if there are not enough solutions to fit with the requested fitorder.
default: 1

display -- Display statistics and/or spectral fitting results. Currently only a histogram of the correction factors to derive the final flux density for each spectral line will be plotted.
default: False
example: display=True

Returned dictionary:

when it is run as fluxres = fluxscale(vis='my.ms',...), the determined flux densities and spectral index information are returned as a Python dictionary in the following format, {fieldIdstr: {spwIdstr: {'fluxd':array([I,Q,U,V]),
'fluxdErr': corresponding errors,
'numSol': corresponding no. of solutions,
'fieldName': field name,
'fitFluxd': fitted flux density at the reference frequency,
'fitFluxdErr': fitted flux density error,
'fitRefFreq': reference frequency,
'spidx': a_0, a_1, a_2
'spidxerr': errors in a_0,a_1, a_2}
'freq': (center) spw frequencies
'spwID': list of spw IDs,
'spwName': list of spw names}}, where fieldIdstr and spwIdstr are field Id and spw Id in string type, respectively.

ft-task.html

0.1.32 ft

Requires:

Synopsis

Insert a source model a visibility set:

Description

A source model (source.model image) or components list is converted into model visibilities that is inserted into the MODEL_DATA column or alternatively is stored in the header of the MS to be served on the fly when requested. This is needed to use more complicated sources than setjy provides; e.g resolved source or off centered sources in gaincal. (Setjy will automatically make this ft step.)

The sources currently available are 3C48, 3C138, 3C147, 3C286 at 1.4, 5.0, 8.4, 15, 22, 43 GHz. Their location is site dependent. In Charlottesville and at the SOC, the models are in /usr/lib/casapy/data/nrao/VLA/CalModels.

Arguments

Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
field	Field selection allowed: string Default:
spw	Spw selection allowed: string Default:
model	Name of input model image(s) allowed: any Default: variant
nterms	Number of terms used to model the sky frequency dependence allowed: int Default: 1
reffreq	Reference frequency (e.g. '1.5e+9' or '1.5GHz') allowed: string Default:
complist	Name of component list allowed: string Default:
incremental	Add to the existing model visibility? allowed: bool Default: False
uscratch	If True predicted visibility is stored in MODEL_DATA column allowed: bool Default: False

Returns

void

Example

A source model (souce.model image) or components list is converted into a model visibility that is inserted into the MODEL_DATA column. This is needed to use resolved source in gaincal and in fluxscale.

Setjy will automatically make this ft step on the sources currently available are 3C48, 3C138, 3C147, 3C286 at 1.4, 5.0, 8.4, 15, 22, 43 GHz. Their location is site dependent. In Charlottesville and at the AOC, the models are in /usr/lib(lib64)/casapy/data/nrao/VLA/CalModels.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'
field -- Field name list
      default: '' ==> all
      NOTE: BUT, only one source can be specified in a multi-source vis.
      field = '1328+307' specifies source '1328+307'
      field = '4' specified field with index 4
spw -- Spw selection
      default: spw = '' (all spw)
model -- Name of input model image
      default: '' ==> None;
      example: model='/usr/lib/casapy/data/nrao/VLA/CalModels/3C286_X.im'
      Note: The model visibilities are scaled from the model frequency
            to the observed frequency of the data.
nterms -- Number of terms used to model the sky frequency dependence
      default: 1 ==> one model image is required
      example : nterms=3 represents a 2nd order Taylor-polynomial in frequency
            and should be used in conjunction with coefficient model images as
model=['xxx.model.tt0','xxx.model.tt1', 'xxx.model.tt2']
reffreq -- Reference-frequency about which this Taylor-expansion is defined.
      default: '' ==> reads the reference frequency from the model image
      example : reffreq = '1.5GHz'
complist -- Name of component list
      default: None; ; example: complist='test.cl'
      component lists are difficult to make.
incremental -- Add model visibility to the existing model visibilities stored in the M
      default: False; example: incremental=True
usescratch -- if True model visibilities will be stored in the scratch column
            MODEL_DATA; when false the model visibilities will be generated
            on the fly (this mode may save some disk space equivalent to
the volume of the observed data).
            default: False; example usescratch=True
```


gaincal-task.html

0.1.33 gaincal

Requires:

Synopsis

Determine temporal gains from calibrator observations

Description

The complex gains for each antenna/spwid are determined from the data column (raw data), divided by the model column, for the specified fields. The gains can be obtained for a specified solution interval for each spectral window, or by a spline fit to all spectral windows simultaneously. Previous calibrations (egs. bandpass) should be applied on the fly.

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
caltable	Name of output gain calibration table allowed: string Default:
field	Select field using field id(s) or field name(s) allowed: string Default:
spw	Select spectral window/channels allowed: string Default:
intent	Select observing intent allowed: string Default:
selectdata	Other data selection parameters allowed: bool Default: True
timerange	Select data based on time range allowed: string Default:
uvrange	Select data within uvrange (default units meters) allowed: any Default: variant
antenna	Select data based on antenna/baseline allowed: string Default:
scan	Scan number range allowed: string Default:
observation	Select by observation ID(s) allowed: any Default: variant
msselect	Optional complex data selection (ignore for now) allowed: string Default:
solint	Solution interval: egs. 'inf', '60s' (see help) allowed: any Default: variant inf
combine	Data axes which to combine for solve (obs, scan, spw, and/or field) allowed: string Default:
preavg	Pre-averaging interval (sec) (rarely needed) allowed: float Default: -1.0
refant	Reference antenna name(s) allowed: string Default:
minblperant	Minimum baselines _per antenna_ required for solve allowed: int Default: 4
minsnr	Reject solutions below this SNR

Example

The complex gains for each antenna/spwid are determined from the data column (raw data) divided by the model column. The gains can be obtained for a specified solution interval, spw combination and field combination. The GSPLINE spline (smooth) option is still under development.

Previous calibrations (egs, bandpass, opacity, parallactic angle) can be applied on the fly. At present with dual-polarized data, both polarizations must be unflagged for any solution to be obtained.

Keyword arguments:

vis -- Name of input visibility file

default: none; example: vis='ngc5921.ms'

caltable -- Name of output gain calibration table

default: none; example: caltable='ngc5921.gcal'

--- Data Selection (see help par.selectdata for more detailed information)

field -- Select field using field id(s) or field name(s).

['go listobs' to obtain the list id's or names]

default: ''=all fields

If field string is a non-negative integer, it is assumed a field index, otherwise, it is assumed a field name

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

field='3C286,3C295'; field named 3C286 and 3C295

field = '3,4C*'; field id 3, all names starting with 4C

DON'T FORGET TO INCLUDE THE FLUX DENSITY CALIBRATOR IF YOU HAVE ONE

spw -- Select spectral window/channels

type 'help par.selection' for more examples.

spw='0~2,4'; spectral windows 0,1,2,4 (all channels)

spw='<2'; spectral windows less than 2 (i.e. 0,1)

spw='0:5~61'; spw 0, channels 5 to 61, INCLUSIVE

spw='*:5~61'; all spw with channels 5 to 61

spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.

spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.

spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60

NOTE ';' to separate channel selections

spw='0:0~10^2,1:20~30^5'; spw 0, channels 0,2,4,6,8,10,

spw 1, channels 20,25,30

intent -- Select observing intent

```

        default: '' (no selection by intent)
        intent='*BANDPASS*' (selects data labelled with
                           BANDPASS intent)
selectdata -- Other data selection parameters
        default: True

Must set selectdata=True to use the following selections:

timerange -- Select data based on time range:
        default = '' (all); examples,
        timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
        Note: if YYYY/MM/DD is missing date defaults to first day in data set
        timerange='09:14:0~09:54:0' picks 40 min on first day
        timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min on NEXT day
        timerange='09:44:00' pick data within one integration of time
        timerange='>10:24:00' data after this time
uvrange -- Select data within uvrange (default units meters)
        default: '' (all); example:
        uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
        uvrange='>4klambda'; uvranges greater than 4 kilo lambda
antenna -- Select data based on antenna/baseline
        default: '' (all)
        If antenna string is a non-negative integer, it is assumed an
        antenna index, otherwise, it is assumed as an antenna name
        antenna='5&6'; baseline between antenna index 5 and index 6.
        antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
        antenna='5&6;7&8'; baselines with indices 5-6 and 7-8
        antenna='5'; all baselines with antenna index 5
        antenna='05'; all baselines with antenna number 05 (VLA old name)
        antenna='5,6,10'; all baselines with antennas 5,6,10 index numbers
scan -- Scan number range.
        Check 'go listobs' to insure the scan numbers are in order.
observation -- Observation ID(s).
        default: '' = all
        example: '0~2,4'
msselect -- Optional complex data selection (ignore for now)

--- Solution parameters
gaintype -- Type of gain solution (G, T, or GSPLINE)
        default: 'G'; example: gaintype='GSPLINE'
        'G' means determine gains for each polarization and sp_wid
        'T' obtains one solution for both polarizations; Hence. their
        phase offset must be first removed using a prior G.
        'GSPLINE' makes a spline fit to the calibrator data. It is
        useful for noisy data and fits a smooth curve through the
        calibrated amplitude and phase. However,

```

at present GSPLINE is somewhat experimental. Use with caution and check solutions.

'K' solves for simple antenna-based delays via FFTs of the spectra on baselines to the reference antenna. (This is not global fringe-fitting.) If combine includes 'spw', multi-band delays are determined; otherwise, per-spw single-band delays will be determined.

'KCROSS' solves for a global cross-hand delay. Use parang=T and apply prior gain and bandpass solutions. Multi-band delay solves (combine='spw') not yet supported for KCROSS.

smodel -- Point source Stokes parameters for source model (experimental)
 default: [] (use MODEL_DATA column)
 example: [1,0,0,0] (I=1, unpolarized)

calmode -- Type of solution
 default: 'ap' (amp and phase); example: calmode='p'
 Options: 'p','a','ap'

solint -- Solution interval (units optional)
 default: 'inf' (~infinite, up to boundaries controlled by combine);
 Options: 'inf' (~infinite),
 'int' (per integration)
 any float or integer value with or without units
 examples: solint='1min'; solint='60s'; solint=60 --> 1 minute
 solint='0s'; solint=0; solint='int' --> per integration
 solint='-1s'; solint='inf' --> ~infinite, up to boundaries
 interacts with combine

combine -- Data axes to combine for solving
 default: '' --> solutions will break at obs, scan, field, and spw boundaries
 Options: '', 'obs', 'scan', 'spw', 'field', or any comma-separated combination in a single string
 For gaintype='K', if combine includes 'spw', multi-band delays will be determined; otherwise, (per-spw) single-band delays will be determined.
 example: combine='scan,spw' --> extend solutions over scan boundaries (up to the solint), and combine spws for solving

refant -- Reference antenna name(s); a prioritized list may be specified
 default: '' => no refant applied
 example: refant='4' (antenna with index 4)
 refant='VA04' (VLA antenna #4)
 refant='EA02,EA23,EA13' (EVLA antenna EA02, use EA23 and EA13 as alternates if/when EA02 drops out)

Use taskname=listobs for antenna listing

minblperant -- Minimum number of baselines required per antenna for each solve


```

default = 4
Antennas with fewer baselines are excluded from solutions.
example: minblperant=10 => Antennas participating on 10 or more
        baselines are included in the solve
minblperant = 1 will solve for all baseline pairs, even if only
        one is present in the data set. Unless closure errors are
        expected, use taskname=gaincal rather than taskname=blcal to
        obtain more options in data analysis.
minsnr -- Reject solutions below this SNR
        default: 3.0
solnorm -- Normalize average solution amps to 1.0 after solution (G, T only)
        default: False (no normalization)
append -- Append solutions to the (existing) table. Appended solutions
        must be derived from the same MS as the existing
        caltable, and solution spws must have the same
        meta-info (according to spw selection and solint)
        or be non-overlapping.
        default: False; overwrite existing table or make new table
splintime -- Spline timescale (sec); used for gaintype='GSPLINE'
        default: 3600 (1 hour); example: splintime=1000
        Typical splintime should cover about 3 to 5 calibrator scans.
npointaver -- Tune phase-unwrapping algorithm for gaintype='GSPLINE'
        default: 3; Keep at this value
phaseswrap -- Wrap the phase for changes larger than this amount (degrees)
        default: 180; Keep at this value

--- Other calibrations to apply on the fly before determining gaincal solution

docallib -- Control means of specifying the caltables:
        default: False ==> Use gaintable, gainfield, interp, spwmap, calwt
        If True, specify a file containing cal library in callib
callib -- If docallib=True, specify a file containing cal
        library directives

gaintable -- Gain calibration table(s) to apply
        default: '' (none);
        examples: gaintable='ngc5921.gcal'
                  gaintable=['ngc5921.ampcal', 'ngc5921.phcal']
gainfield -- Select a subset of calibrators from gaintable(s) to apply
        default: '' ==> all sources in table;
        'nearest' ==> nearest (on sky) available field in table
        otherwise, same syntax as field
        example: gainfield='0~2,5' means use fields 0,1,2,5 from gaintable
                  gainfield=['0~3', '4~6'] means use field 0 through 3
                  from first gain file, field 4 through 6 for second.
interp -- Interpolation type (in time[,freq]) to use for each gaintable.

```

When frequency interpolation is relevant (B, Df, Xf), separate time-dependent and freq-dependent interp types with a comma (freq_after_ the comma). Specifications for frequency are ignored when the calibration table has no channel-dependence. Time-dependent interp options ending in 'PD' enable a "phase delay" correction per spw for non-channel-dependent calibration types.

For multi-obsId datasets, 'perobs' can be appended to the time-dependent interpolation specification to enforce obsId boundaries when interpolating in time.

default: '' --> 'linear,linear' for all gaintable(s)

example: interp='nearest' (in time, freq-dep will be linear, if relevant)

interp='linear,cubic' (linear in time, cubic in freq)

interp='linearperobs,spline' (linear in time per obsId, spline in freq)

interp=',spline' (spline in freq; linear in time by default)

interp=['nearest,spline','linear'] (for multiple gaintables)

Options: Time: 'nearest', 'linear'

Freq: 'nearest', 'linear', 'cubic', 'spline'

spwmap -- Spectral windows combinations to form for gaintable(s)

default: [] (apply solutions from each spw to that spw only)

Example: spwmap=[0,0,1,1] means apply the caltable solutions from spw = 0 to the spw 0,1 and spw 1 to spw 2,3.

spwmap=[[0,0,1,1],[0,1,0,1]]

parang -- If True, apply the parallactic angle correction (required for polarization calibration)

default: False

preavg -- Pre-averaging interval (sec)

default=-1 (none).

Rarely needed. Will average data over periods shorter than the solution interval first.

async -- Run asynchronously

default = False; do not run asynchronously

gencal-task.html

0.1.34 gencal

Requires:

Synopsis

Specify Calibration Values of Various Types

Description

Specify calibration externally.

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
caltable	The new/existing calibration table allowed: string Default:
caltype	The calibration type: 'amp','ph','sbd','mbd','antpos','antposvla','tsys','evlagain','opac','gc' etc. allowed: string Default:
infile	Input ancilliary file allowed: string Default:
spw	Calibration spw(s) selection allowed: string Default:
antenna	Calibration antenna(s) selection allowed: string Default:
pol	Calibration polarizations(s) selection allowed: string Default:
parameter	The calibration values allowed: doubleArray Default:

Returns

void

Example

The gencal task provides a means of specifying antenna-based calibration values manually. The values are put in designated tables and applied to the data using applycal. Several specialized calibrations are also generated with gencal.

Current antenna-based gencal options (caltype) are:

- 'amp' = amplitude correction
- 'ph' = phase correction
- 'sbd' = single-band delay (phase-frequency slope for each spw)
- 'mbd' = multi-band delay (phase-frequency slope over all spw)
- 'antpos' = ITRF antenna position corrections
- 'antposvla' = VLA-centric antenna position corrections
- 'tsys' = Tsys from the SYSCAL table (ALMA)
- 'swpow' = EVLA switched-power gains (experimental)
- 'evlagain' (= 'swpow') (this syntax will deprecate)
- 'rq' = EVLA requantizer gains _only_
- 'swp/rq' = EVLA switched-power gains divided by requantizer gain
- 'opac' = Tropospheric opacity
- 'gc' = Gain curve (zenith-angle-dependent gain) (VLA only)
- 'eff' = Antenna efficiency ($\sqrt{K/Jy}$) (VLA only)
- 'gceff' = Gain curve and efficiency (VLA only)
- 'tecim' = Time-dep TEC image specified in infile (EXPERIMENTAL)

Generic calibration parameters should be specified in the 'parameter' argument as a list. The length of the list must correspond to the net length of the specific polarizations, antennas, and spws specified in the selection arguments. The specified parameters will be duplicated over all members of any unspecified selection axes. E.g., if `pol=antenna=spw=''`, it only makes sense to specify a single parameter value, and this will be duplicated for all pols, antennas, and spws. If multiple parameter values are specified, at least one of the selection arguments must be non-trivial, and the number of specified parameters must be consistent with the explicit selection. E.g., if a non-trivial spw selection is specified, then the parameter list should match the number of spws specified, and

these values will be duplicated for all polarizations and antennas. If more than one selection argument is non-trivially specified, the number of parameters specified should match the product of the number specified selection elements. The parameter values should be sorted by pol (fastest), antenna, and spw (slowest). Un-specified elements on non-trivially specified axes will be filled with nominal values (i.e., it is not necessary to exhaustively specify all elements on any axis or use nominal parameter values explicitly). Please consult the examples provided below for additional guidance. There is currently no support for time-dependent parameter specification. The specified parameters will be assumed constant in time (though their impact on the data may be time-dependent, depending on the caltype). Some caltype options do not require parameter specifications; these are described in detail below.

The same caltable can be specified for multiple runs of gencal, in which case the specified parameters will be incorporated cumulatively. E.g., amplitude parameters (caltype='amp') multiply and phase-like parameters ('ph', 'sbd', 'mbd', 'antpos') add. 'amp' and 'ph' parameters can be incorporated into the same caltable (in separate runs), but each of the other types require their own unique caltable. A mechanism for specifying manual corrections via a text file will be provided in the future.

The caltables are applied to the data by using applycal. Other calibration tables may also be present, if applicable.

For antenna position corrections (caltype='antpos'), the antenna position offsets are specified in the ITRF frame. For EVLA, automated lookup of the antenna position corrections is enabled when antenna is unspecified (antenna='') for this caltype. Note that this requires internet connection to access the EVLA antenna position correction site.

For VLA position corrections in the VLA-centric frame, use caltype='antposvla', and gencal will rotate them to ITRF before storing them in the output caltable.

For Tsys (caltype='tsys', for ALMA) and EVLA switched power corrections (caltype='swpow'), the calibration parameters are derived from information contained in MS subtables. In these cases, specification of spw, antenna, pol, and parameter will be ignored.

EVLA switched power calibration is supported in three modes:

'swpow' (formerly 'evlagain', a syntax which will deprecate) yields the formal EVLA switched power calibration which describes voltage gain as $\sqrt{Pdif/Tcal}$ (used to correct the visibility data) and $Tsys$ as $Psum*Tcal/Pdif/2$ (used to correct the weights). 'swpow' implicitly includes any requantizer gain scale and adjustments.

'rq' yields only the requantizer voltage gains ($Tsys$ is set to 1.0 to avoid weight adjustments).

'swp/rq' yields the ordinary switched power voltage gains divided by the requantizer voltage gain ($Tsys$ is calculate normally). The 'rq' and 'swp/rq' modes are are mainly intended for testing and evaluating the EVLA switched power systems.

For `caltype='opac'`, specify the desired opacity(ies) in the parameter argument. At this time, only constant (in time) opacities are supported via `gencal`.

For `gaincurve` and `efficiency` (`caltype='gc'`, `'gceff'`, or `'eff'`), observatory-provided factors are determined per spw according to the observing frequencies. The parameter argument is ignored. These `caltypes` are currently only supported for VLA processing. (Appropriate factors for ALMA are TBD.)

For `caltype='tecim'`, specify a TEC image in the `infile` parameter. The TEC image can be obtained via the `tec_maps` module; please consult the CASA Cookbook (section 4.3) for more details.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none.  example: vis='ngc5921.ms'
caltable -- Name of input/output caltable.  If it does not
            exist, it will be created.  Specifying an
            existing table will result in the parameters
            being applied cumulatively.  Only a single
            time-stamp for all calibrations are supported,
            currently.  Do not use a caltable
            created by gaincal, bandpass, etc.
            default: none.  example: caltable='test.G'
caltype -- The calibration parameter type being specified.
           Options include:
           'amp' = gain (G) amplitude (1 real parameter per
                  pol, antenna, spw)
           'ph'  = gain (G) phase (deg) (1 real parameter per
                  pol, antenna, spw)
           'sbd' = single-band delays (nsec) (1 real parameter
```

```

        per pol, antenna, spw)
'mbd' = multi-band delay (nsec) (1 real parameter
        per pol, antenna, spw)
'antpos' = antenna position corrections (m) (3 real
        ITRF offset parameters per antenna; spw, pol
        selection will be ignored)
        With antenna='', this triggers an automated lookup
        of antenna positions for EVLA.
'antposvla' = antenna position corrections (m) specified
        in the old VLA-centric coordinate system
'tsys' = Tsys from the SYSCAL table (ALMA)
'evlgain' = EVLA switched-power gains (experimental)
'opac' = Tropospheric opacity (1 real parameter
        per antenna, spw)
'gc' = Antenna zenith-angle dependent gain curve (auto-lookup)
'geff' = Gain curve and efficiency (auto-lookup)
'eff' = Antenna efficiency (auto-lookup)
'tecim' = Time-dep TEC image specified in infile (EXPERIMENTAL)
default: none.
example: caltype='ph'
infile -- For caltype='tecim', the input TEC image from tec_maps
        default: none
spw -- Spectral window selection for specified parameters.
        default: spw='' (specified parameters apply to all spws)
        example: spw = '2,3,4'
antenna -- Antenna selection for specified parameters.
        default: antenna='' (specified parameters apply to all antennas)
        example: antenna='ea02, ea03' (specified parameter(s) to
        apply to ea02 and ea03 only)
pol -- Polarization selection for specified parameters.
        default: pol='' (specified parameters apply to all polarizations)
        example: pol='R' (specified parameters to apply to
        R only)
parameter -- The calibration parameters, specified as a list, to
        store in the caltable for the spw, antenna, and pol
        selection. The required length of the list is
        determined by the caltype and the spw, antenna, pol
        selection. One "set" of parameters (e.g., one value
        for 'amp', 'ph', etc., three values for 'antpos')
        specified the same value for all indicated spw, antenna,
        and pol.
OR,
When specifying a long list of calibration parameter values,
these should be ordered first (fastest) by pol (if pol!=''),
then by antenna (if antenna!=''), and finally (slowest) by
spw (if spw!=''). Unspecified selection axes must not be

```

enumerated in the parameter list

Examples:

```
gencal(vis='test.ms',caltable='test.G',caltype='amp',
       spw='',antenna='',pol='',
       parameter=[3])
```

--> Antenna-based gain amplitude corrections for all spws, antennas, and polarizations will be multiplied by 3. When applied to visibility data, this correction will produce a corrected visibility that is $(1/3 \times 1/3)$ less than the uncorrected visibility.

```
gencal(vis='test.ms',caltable='test.G',caltype='ph',
       spw='',antenna='ea03,ea04',pol='',
       parameter=[45,120])
```

--> Gain phase corrections for antennas ea03 and ea04 will be adjusted (additive) by 45 and 120 degrees (respectively), for all spws and polarizations. When these phases are applied to visibility data, the visibility phases will decrease or increase by the specified amount where the selected antennas occur first or second (respectively) in each baseline. E.g., the phase of baseline ea03-ea04 will change by $(-45+120) = +75$ degrees. Baseline ea01-ea03's phase will change by +45 degrees; baseline ea04-ea05's phase will change by -120 degrees. The same phase sign convention is used for delay and antenna position corrections.

```
gencal(vis='test.ms',caltable='test.G',caltype='ph',
       spw='',antenna='ea05,ea06',pol='R',
       parameter=[63,-34])
```

--> Gain phase corrections for antennas ea05 and ea06 will be adjusted (additive) by 63 and -34 degrees (respectively), in R only, for all spws

```
gencal(vis='test.ms',caltable='test.G',caltype='ph',
       spw='',antenna='ea09,ea10',pol='R,L',
       parameter=[14,-23,-130,145])
```

--> Gain phase corrections in all spws will be adjusted for antenna ea09 by 14 deg in R and -23 deg in L, and for antenna ea10 by -130 deg in R and 145 deg in L.


```

gencal(vis='test.ms',caltable='test.G',caltype='ph',
       spw='2,3',antenna='ea09,ea10',pol='',
       parameter=[14,-23,-130,145])

--> Gain phases corrections in both polarizations will be adjusted for
      antenna ea09 by 14 deg in spw 2 and -23 deg in spw 3, and for
      antenna ea10 by -130 deg in spw 2 and 145 deg in spw 3.

gencal(vis='test.ms',caltable='test.G',caltype='sbd',
       spw='2,3',antenna='ea09,ea10',pol='',
       parameter=[14,-23,-130,145])

--> Delay corrections in both polarizations will be adjusted for
      antenna ea09 by 14 nsec in spw 2 and -23 nsec in spw 3, and for
      antenna ea10 by -130 nsec in spw 2 and 145 nsec in spw
      3. See the above example for caltype='ph' for details
      of the sign convention adopted when applying delay corrections.

gencal(vis='test.ms',caltable='test.G',caltype='antpos',antenna='')

--> *** Currently EVLA observations only ***
      Antenna position corrections will be retrieved automatically
      over internet to generate the caltable with antenna=''.

gencal(vis='test.ms',caltable='test.G',caltype='antpos',
       antenna='ea09,ea10',
       parameter=[0.01,0.02,0.03, -0.03,-0.01,-0.02])

--> Antenna position corrections in meters (in ITRF) for
      antenna ea09 (dBx=0.01, dBy=0.02, dBz=0.03) and for
      antenna ea10 (dBx=-0.03, dBy=-0.01, dBz=-0.02)
      See the above example for caltype='ph' for details
      of the sign convention adopted when applying antpos
      corrections.

gencal(vis='test.ms',caltable='test.G',caltype='antposvla',
       antenna='ea09,ea10',
       parameter=[0.01,0.02,0.03, -0.03,-0.01,-0.02])

--> Antenna position corrections (in the traditional VLA-centric
      frame) will be introduced in meters for
      antenna ea09 (dBx=0.01, dBy=0.02, dBz=0.03) and for
      antenna ea10 (dBx=-0.03, dBy=-0.01, dBz=-0.02)
      These offsets will be rotated to the ITRF frame before
      storing them in the caltable.

```

See the above example for `caltype='ph'` for details of the sign convention adopted when applying antpos corrections.

hanningsmooth-task.html

0.1.35 hanningsmooth

Requires:

Synopsis

Hanning smooth frequency channel data to remove Gibbs ringing

Description

This is the new implementation of hanningsmooth.

* The old implementation is available for a short time as oldhanningsmooth. * Task hanningsmooth2 has been renamed to hanningsmooth. The hanningsmooth2 alias will be removed soon. * Please, update your scripts to call hanningsmooth instead.

The new hanningsmooth task uses the MStTransform framework underneath but keeps roughly the same interface as the old hanningsmooth task. This function Hanning smooths the frequency channels with a weighted running average. The weights are 0.5 for the central channel and 0.25 for each of the two adjacent channels. The first and last channels are flagged. Inclusion of a flagged value in an average causes that data value to be flagged. If the 'CORRECTED' data column is requested for an MS that does not contain this column, it will use 'DATA' to calculate the smoothing and save it to 'DATA' in the output MS. WARNING: by default, all visibility columns will be smoothed.

Arguments

Inputs	
vis	Name of input Measurement set or Multi-MS. allowed: string Default:
outputvis	Name of output Measurement set or Multi-MS. allowed: string Default:
keepmms	If the input is a Multi-MS the output will also be a Multi-MS. allowed: bool Default: True
field	Select field using ID(s) or name(s). allowed: any Default: variant
spw	Select spectral window/channels. allowed: any Default: variant
scan	Select data by scan numbers. allowed: any Default: variant
antenna	Select data based on antenna/baseline. allowed: any Default: variant
correlation	Correlation: " ==> all, correlation='XX,YY'. allowed: any Default: variant
timerange	Select data by time range. allowed: any Default: variant
intent	Select data by scan intent. allowed: any Default: variant
array	Select (sub)array(s) by array ID number. allowed: any Default: variant
uvrange	Select data by baseline length. allowed: any Default: variant
observation	Select by observation ID(s). allowed: any Default: variant
feed	Multi-feed numbers: Not yet implemented. allowed: any Default: variant
datacolumn	Input data column(s) to process.

Example

----- Detailed description of keyword arguments: -----

vis -- Name of input visibility file (MS or MMS)
default: ''; example: vis='ngc5921.ms'

outputvis -- Name of output visibility file (MS or MMS)
default: ''; example: outputvis='out_ngc5921.mms'

keepmms -- Create a Multi-MS as the output if the input is a Multi-MS.
default: True

By default it will create a Multi-MS when the input is a Multi-MS.
The output Multi-MS will have the same partition axis of the input MMS.
See 'help partition' for more information on the MMS format.

--- Data selection parameters ---

field -- Select field using field id(s) or field name(s).
[run listobs to obtain the list of d's or names]
default: ''=all fields If field string is a non-negative
integer, it is assumed to be a field index
otherwise, it is assumed to be a field name
field='0~2'; field ids 0,1,2
field='0,4,5~7'; field ids 0,4,5,6,7
field='3C286,3C295'; fields named 3C286 and 3C295
field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels
default: ''=all spectral windows and channels
spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
spw='<2'; spectral windows less than 2 (i.e. 0,1)
spw='0:5~61'; spw 0, channels 5 to 61
spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.
spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
spw = '*:3~64' channels 3 through 64 for all sp id's
spw = ' :3~64' will NOT work.

NOTE: mstransform does not support multiple channel ranges per
spectral window (';').

```

scan -- Scan number range
      default: ''=all

antenna -- Select data based on antenna/baseline
          default: '' (all)
          Non-negative integers are assumed to be antenna indices, and
          anything else is taken as an antenna name.

examples:
  antenna='5&6': baseline between antenna index 5 and index 6.
  antenna='VA05&VA06': baseline between VLA antenna 5 and 6.
  antenna='5&6;7&8': baselines 5-6 and 7-8
  antenna='5': all baselines with antenna 5
  antenna='5,6,10': all baselines including antennas 5, 6, or 10
  antenna='5,6,10&': all baselines with *only* antennas 5, 6, or
                    10. (cross-correlations only. Use &&
                    to include autocorrelations, and &&&
                    to get only autocorrelations.)
  antenna='!ea03,ea12,ea17': all baselines except those that
                    include EVLA antennas ea03, ea12, or
                    ea17.

correlation -- Correlation types or expression.
              default: '' (all correlations)
              example: correlation='XX,YY'

timerange -- Select data based on time range:
            default: '' (all); examples,
            timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
            Note: if YYYY/MM/DD is missing date, timerange defaults to the
            first day in the dataset
            timerange='09:14:0~09:54:0' picks 40 min on first day
            timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min
            on next day
            timerange='09:44:00' data within one integration of time
            timerange='>10:24:00' data after this time

array -- (Sub)array number range
        default: ''=all

uvrange -- Select data within uvrange (default units meters)
          default: ''=all; example:
          uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
          uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
          uvrange='0~1000km'; uvrange in kilometers

```

```

observation -- Select by observation ID(s)
              default: ''=all

feed -- Selection based on the feed - NOT IMPLEMENTED YET
       default: ''=all

datacolumn -- Which data column to use for processing (case-insensitive).
              default: 'all'; whichever of the visibility data columns that are present.
              options: 'data', 'model', 'corrected', 'all', 'float_data', 'lag_data'.

example1: datacolumn='data'; it will smooth the input DATA column and save the
          smoothed data in DATA of the output MS.
example2: datacolumn='corrected'; it will smooth the input CORRECTED_DATA column
          and save the smoothed data in DATA of the output MS.
example3: datacolumn='all', where the input MS has DATA,CORRECTED_DATA,MODEL_DATA.
          It will smooth all three columns and save the smoothed data in
          DATA, CORRECTED_DATA and MODEL_DATA of the output MS.

```

[imcollapse-task.html](#)

0.1.36 imcollapse

Requires:

Synopsis

Collapse image along one axis, aggregating pixel values along that axis.

Arguments

Inputs	
imagename	Name of the input image allowed: string Default:
function	Aggregate function to apply. This can be set one of flux, max, mean, median, min, rms, stdev, sum, variance. Must be specified. allowed: string Default:
axes	Zero-based axis number(s) or minimal match strings to collapse. allowed: variant Default: [0]
outfile	Name of output CASA image. Must be specified. allowed: string Default:
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: string Default:
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. allowed: string Default:
mask	Mask to use. See help par.mask. Default is none. allowed: string Default:
overwrite	Overwrite existing output file if it exists? allowed: bool Default: False
stretch	Stretch the mask if necessary and possible? See help par.stretch allowed: bool Default: False

Returns

bool

Example

PARAMETER SUMMARY

imagename	Name of the input (CASA, FITS, MIRIAD) image
function	Function used to compute aggregation of pixel values along the collapsed axis. Supported functions are flux, max, mean, median, min, rms, stdev, sum, variance. Minimum match is supported for the function parameter (eg, function="r" will compute the rms of the pixel values).
axis	Zero-based axis number(s) or minimal match strings to compress.
outfile	Name of output CASA image. Must be specified.
overwrite	Controls if an already existing file by the same name can be overwritten. If true, the user is not prompted, the file if it exists is automatically overwritten.
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.
region	Region selection. See "help par.region" for details. Default is to use the full image.
chans	Channels to use. See "help par.chans" for details. Default is to use all channels.
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.
mask	Mask to use. See help par.mask. Default is none.
stretch	Stretch the input mask if necessary and possible. Only used if a mask is specified. See help par.stretch.

This task collapses an image along a specified axis or set of axes of N pixels to a single pixel along the specified axis. Both float valued and complex valued images are supported. It computes the specified aggregate function for pixel values along the specified axes and places those values in the single remaining plane of those axes in the output image. It is an image analysis tool containing the newly created collapsed image if wantreturn=True. Choices for aggregate functions are: flux (see below for constraints), max, mean, median, min, rms, stdev, sum, variance. Minimum match is supported for the function parameter (eg, function="r" will compute the rms of the pixel values, function="m" will compute the median).

If one specifies function='flux', the following constraints must be true:

1. The image must have a direction coordinate,
2. The image must have at least one beam,
3. The specified axes must be exactly the direction coordinate axes,
4. Only one of the non-directional axes may be non-degenerate,

5. The image brightness unit must be conformant with $x \cdot y \text{ Jy/beam}$, where x is an optional unit and y is an optional SI prefix.

Axes can be specified as a single integer or array of integers indicating the zero-based axes which to collapse the image. Axes may also be specified as a single or array of strings which and uniquely match (ignoring case) world axes names in the image (eg "dec" or ["ra", "d"] for collapsing along the declination axis or along the right ascension and declination axes, respectively).

The reference pixel of the collapsed axis is set to 0 and its reference value is set to the value of the first and last values of that axis in the specified region of the input image.

```
# myimage.im is a 512x512x128x4 (ra,dec,freq,stokes) image
imagenname = "myimage.im"
# collapse a subimage of it along its spectral axis avoiding the 8 edge
# channels at each end of the band, computing the mean value of the pixels
# resulting image is 256x256x1x4 in size.
outfile="collapse_spec_mean.im"
function="mean"
axis=2
box="127,127,383,383"
chans="8~119"
imcollapse(imagenname=imagenname, outfile=outfile, function=function, axes=axis, box=box, chans=chans)
```

imcontsub-task.html

0.1.37 imcontsub

Requires:

Synopsis

Estimates and subtracts continuum emission from an image cube

Arguments

Inputs	
imagename	Name of the input spectral line image allowed: string Default:
linefile	Output continuum-subtracted image file name allowed: string Default:
contfile	Output continuum image file name allowed: string Default:
fitorder	Polynomial order for the continuum estimation allowed: int Default: 0
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: string Default:
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: any Default: variant
chans	Channels to use for fitting. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. allowed: string Default:

Returns

void

Example

For each (x, y) column in imagename (or a subset selected by region and/or box), this estimates the continuum by fitting a polynomial to one or more subsets of the channels. The continuum estimate is saved in contfile, and subtracted from imagename (or its subset) to make a spectral line estimate, which is saved in linefile.

Keyword arguments:

imagename -- Input image cube

Default: none; Example: imagename='ngc5921_task.im'

linefile -- Name of output spectral line cube

Default: none; Example: outline='ngc5921_line.im'

contfile -- Name of output continuum cube

Default: none; Example: contfile='ngc5921_cont.im'

fitorder -- Polynomial order for the continuum estimation.

Default: 0; Example fitorder=2

region -- Region selection. See "help par.region" for details. Default is to use the full image.

box -- Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.

chans -- Channels to use for fitting. See "help par.chans" for details. Default is to use all channels.

Example: chans='3~6;>40'

stokes -- Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.

Fit a second order polynomial (fitorder=2) to channels 3-8 and 54-60 to an RA x Dec x Frequency x Stokes cube, selecting the Stokes I plane

```
ch = '3~8, 54~60'
```

```
imcontsub(imagename="myimage.im", linefile="mycontsub.im", fitorder=2, chans=ch, fitorder=2,
```

[imfit-task.html](#)

0.1.38 imfit

Requires:

Synopsis

Fit one or more elliptical Gaussian components on an image region(s)

Arguments

Inputs	
imagenname	Name of the input image allowed: string Default:
box	Rectangular region(s) to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: any Default: variant
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: any Default: variant
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use first Stokes plane. allowed: string Default:
mask	Mask to use. See help par.mask. Default is none. allowed: string Default:
includepix	Range of pixel values to include for fitting. allowed: intArray Default:
excludepix	Range of pixel values to exclude for fitting. allowed: intArray Default:
residual	Name of output residual image. allowed: string Default:
model	Name of output model image. allowed: string Default:
estimates	Name of file containing initial estimates of component parameters. allowed: string Default:
logfile	Name of file to write fit results. allowed: string Default:
append	If logfile exists, append to it if True or overwrite it if False allowed: bool Default: True
newestimates	File to write fit results which can be used as initial estimates for next run. allowed: string Default:
complist	Name of output component list table. allowed: string Default:

Returns

void

Example

PARAMETER SUMMARY

imagename	Name of the input image
box	Rectangular region(s) to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. eg "100, 120, 200, 220, 300, 300, 400, 400" to use two boxes.
region	Region selection. See "help par.region" for details. Default is to use the full image.
chans	Channels to use. See "help par.chans" for details. Default is to use all channels.
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use first Stokes plane.
mask	Mask to use. See help par.mask. Default is none.
includepix	Range of pixel values to include for fitting. Array of two numeric values assumed to have same units as image pixel values. Only one of includepix or excludepix can be specified.
excludepix	Range of pixel values to exclude for fitting. Array of two numeric values assumed to have same units as image pixel values. Only one of includepix or excludepix can be specified.
residual	Name of the residual image to write.
model	Name of the model image to write.
estimates	Name of file containing initial estimates of component parameters (see below for formatting details).
logfile	Name of file to write fit results.
append	If logfile exists, append to it (True) or overwrite it (False).
newestimates	File to write fit results which can be used as initial estimates for next run.
complist	Name of output component list table.
overwrite	Overwrite component list table if it exists?
dooff	Simultaneously fit a zero-level offset?
offset	Initial estimate for the zero-level offset. Only used if dooff is True.
fixoffset	Hold zero-level offset constant during fit? Only used if dooff is True.
stretch	Stretch the input mask if necessary and possible. Only used if a mask is supplied. See help par.stretch.
rms	RMS to use in calculation of various uncertainties, assumed to have units of image. If not positive, the rms of the residual image is used.
noisefwhm	Noise correlation beam FWHM. If numeric value, interpreted as pixel widths. If quantity (dictionary, string), it must have angular units.

OVERVIEW

This application is used to fit one or more two dimensional gaussians to sources in an image as well as an optional zero-level offset. Fitting is limited to a single polarization but can be performed over several contiguous spectral channels. If the image has a clean beam, the report and returned dictionary will contain both the convolved and the deconvolved fit results.

When `dooff` is `False`, the method returns a dictionary with three keys, `'converged'`, `'results'` and `'deconvolved'`. The value of `'converged'` is a boolean array which indicates if the fit converged on a channel by channel basis. The value of `'results'` is a dictionary representing a component list reflecting the fit results. In the case of an image containing beam information the sizes and position angles in the `'results'` dictionary are those of the source(s) convolved with the restoring beam, while the same parameters in the `'deconvolved'` dictionary represent source sizes deconvolved from the beam. In the case where the image does not contain a beam, `'deconvolved'` will be absent. Both the `'results'` and `'deconvolved'` dictionaries can be read into a component list tool (default tool is named `cl`) using the `fromrecord()` method for easier inspection using tool methods, eg

```
cl.fromrecord(res['results'])
```

although this currently only works if the flux density units are conformant with Jy.

There are also values in each component subdictionary not used by `cl.fromrecord()` but meant to supply additional information. There is a `'peak'` subdictionary for each component that provides the peak intensity of the component. It is present for both `'results'` and `'deconvolved'` components. There is also a `'sum'` subdictionary for each component indicating the simple sum of pixel values in the original image enclosed by the fitted ellipse. There is a `'channel'` entry in the `'sum'` subdictionary which provides the zero-based channel number in the input image for which the fit applies. In addition, if the image has a beam(s), then there will be a `'beam'` subdictionary associated with each component in both the `'results'` and `'deconvolved'` dictionaries. This subdictionary will have three keys: `'beamarcsec'` will be a subdictionary giving the beam dimensions in arcsec, `'beampixels'` will have the value of the beam area expressed in pixels, and `'beamster'` will have the value of the beam area expressed in steradians. Also, if the image has a beam(s), in the component dictionaries will be an `'ispoint'` entry with an associated boolean value describing if the component is consistent with a point source.

If `dooff` is `True`, in addition to the specified number of gaussians, a zero-level offset will also be fit. The initial estimate for this offset is specified using the `offset` parameter. Units are assumed to be the same as the image brightness units. The zero level offset can be held constant during the fit by specifying `fixoffset=True`. In the case of `dooff=True`, the returned dictionary contains two additional keys, `'zerooff'` and `'zeroofferr'`, which are both dictionaries containing `'unit'` and `'value'` keys. The values associated with the `'value'` keys are arrays containing the fitted zero level offset value and its error, respectively, for each channel. In cases where the fit did not converge, these values are set to `NaN`.

The value associated with 'unit' is just the image brightness unit.

The region can either be specified by a box(es) or a region.

Ranges of pixel values can be included or excluded from the fit. If specified using the box parameter, multiple boxes can be given using the format

box="blcx1, blcy1, trcx1, trcy1, blcx2, blcy2, trcx2, trcy2, ... , blcxN, blcyN, trcxN, trcyN" where N is the number of boxes. In this case, the union of the specified boxes will be used.

If specified, the residual and/or model images for successful fits will be written.

If an estimates file is not specified, an attempt is made to estimate initial parameters and fit a single Gaussian. If a multiple Gaussian fit is desired, the user must specify initial estimates via a text file (see below for details).

The user has the option of writing the result of the fit to a log file, and has the option of either appending to or overwriting an existing file.

The user has the option of writing the (convolved) parameters of a successful fit to a file which can be fed back to fitcomponents() as the estimates file for a subsequent run.

If specified and positive, the value of rms is used to calculate the parameter uncertainties; otherwise, the rms in the selected region in the relevant channel is used for these calculations.

The noisefwhm parameter represents the noise-correlation beam FWHM. If specified as a quantity, it should have angular units. If specified as a numerical value, it is set equal to that number of pixels. If specified and greater than or equal to the pixel size, it is used to calculate parameter uncertainties using the correlated noise equations (see below). If it is specified less than a pixel width, the uncorrelated noise equations (see below) are used to compute the parameter uncertainties. If it is not specified and the image has a restoring beam, the correlated noise equations are used to compute parameter uncertainties using the geometric mean of the relevant beam major and minor axes as the noise-correlation beam FWHM. If noisefwhm is not specified and the image does not have a restoring beam, then the uncorrelated noise equations are used to compute the parameter uncertainties.

SUPPORTED UNITS

Currently only images with brightness units conformant with Jy/beam, Jy.km/s/beam, and K are supported for fitting. If your image has some other base brightness unit, that unit will be assumed to be equivalent to Jy/pixel and results will be calculated accordingly. In particular, the flux density (reported as Integrated Flux in the logger and associated with the "flux" key in the returned component subdictionary(ies)) for such a case represents the sum of pixel values.

Note also that converting the returned results subdictionary to a component list via cl.from_dictionary() only works properly if the flux density units in the results dictionary are conformant with

If you need to be able to run `cl.fromrecord()` on the resulting dictionary you can first modify the flux density units by hand to be (some prefix)Jy and then run `cl.fromrecord()` on that dictionary bearing in mind your unit conversion.

If the input image has units of K, the flux density of components will be reported in units of [prefix]K*rad*rad, where prefix is an SI prefix used so that the numerical value is between 1 and 1000. To convert to units of K*beam, determine the area of the appropriate beam, which is given by $\pi/(4*\ln(2))*bmaj*bmin$, where bmaj and bmin are the major and minor axes of the beam, and convert to steradians (=rad*rad). This value is included in the beam portion of the component subdictionary (key 'beamster'). Then divide the numerical value of the logged flux density by the beam area in steradians. So, for example

```
\begin{verbatim}
# run on an image with K brightness units
res = imfit(...)
# get the I flux density in K*beam of component 0
comp = res['results']['component0']
flux_density_kbeam = comp['flux']['value'][0]/comp['beam']['beamster']
```

FITTING OVER MULTIPLE CHANNELS

For fitting over multiple channels, the result of the previous successful fit is used as the estimate for the next channel. The number of gaussians fit cannot be varied on a channel by channel basis. Thus the variation of source structure should be reasonably smooth in frequency to produce reliable fit results.

MASK SPECIFICATION

Mask specification can be done using an LEL expression. For example `mask = "myimage">5` will use only pixels with values greater than 5.

INCLUDING AND EXCLUDING PIXELS

Pixels can be included or excluded from the fit based on their values using these parameters. Note that specifying both is not permitted and will cause an error. If specified, both take an array of two numeric values.

ESTIMATES

Initial estimates of fit parameters may be specified via an estimates text file. Each line of this file should contain a set of parameters for a single gaussian. Optionally, some of these parameters can be fixed during the fit. The format of each line is

peak intensity, peak x-pixel value, peak y-pixel value, major axis, minor axis, position angle, fixed

The fixed parameter is optional. The peak intensity is assumed to be in the same units as the image pixel values (eg Jy/beam). The peak coordinates are specified in pixel coordinates. The major and minor axes and the position angle are the convolved parameters if the image has been convolved with a clean beam and are specified as quantities. The fixed parameter is optional and is a string. It may contain any combination of the following characters 'f' (peak intensity), 'x' (peak x position), 'y' (peak y position), 'a' (major axis), 'b' (minor axis), 'p' (position angle).

In addition, lines in the file starting with a # are considered comments.
An example of such a file is:

```
# peak intensity must be in map units
120, 150, 110, 23.5arcsec, 18.9arcsec, 120deg
90, 60, 200, 46arcsec, 23arcsec, 140deg, fxp
```

This is a file which specifies that two gaussians are to be simultaneously fit, and for the second gaussian the specified peak intensity, x position, and position angle are to be held fixed during the fit.

ERROR ESTIMATES

Error estimates are based on the work of Condon 1997, PASP, 109, 166. Key assumptions made are: * The given model (elliptical Gaussian, or elliptical Gaussian plus constant offset) is an adequate representation of the data * An accurate estimate of the pixel noise is provided or can be derived (see above). For the case of correlated noise (e.g., a CLEAN map), the fit region should contain many "beams" or an independent value of rms should be provided. * The signal-to-noise ratio (SNR) or the Gaussian component is large. This is necessary because a Taylor series is used to linearize the problem. Condon (1997) states that the fractional bias in the fitted amplitude due to this assumption is of order $1/(S^2)$, where S is the overall SNR of the Gaussian with respect to the given data set (defined more precisely below). For a 5 sigma "detection" of the Gaussian, this is a 4% effect. * All (or practically all) of the flux in the component being fit falls within the selected region. If a constant offset term is simultaneously fit and not fixed, the region of interest should be even larger. The derivations of the expressions summarized in this note assume an effectively infinite region.

Two sets of equations are used to calculate the parameter uncertainties, based on if the noise is correlated or uncorrelated. The rules governing which set of equations are used have been described above in the description of the noisefwhm parameter.

In the case of uncorrelated noise, the equations used are

$$f(A) = f(I) = f(M) = f(m) = k \cdot s(x)/M = k \cdot s(y)/m = (s(p)/\sqrt{2}) \cdot ((M \cdot M - m \cdot m)/(M \cdot m)) = \sqrt{2}/S$$

where $s(z)$ is the uncertainty associated with parameter z , $f(z) = s(z)/\text{abs}(z)$ is the fractional uncertainty associated with parameter z , A is the peak intensity, I is the flux density, M and m are the FWHM major and minor axes, p is the position angle of the component, and $k = \sqrt{8 \cdot \ln(2)}$. $s(x)$ and $s(y)$ are the direction uncertainties of the component measured along the major and minor axes; the resulting uncertainties measured along the principle axes of the image direction coordinate are calculated by propagation of errors using the 2D rotation matrix which enacts the rotation through the position angle plus 90 degrees. S is the overall signal to noise ratio of the component, which, for the uncorrelated noise case is given by

$$S = (A/(k \cdot h \cdot r)) \cdot \sqrt{\pi \cdot M \cdot m}$$

where h is the pixel width of the direction coordinate and r is the rms noise (see the discussion above for the rules governing how the value of r is determined).

For the correlated noise case, the same equations are used to determine the uncertainties as in the uncorrelated noise case, except for the uncertainty in I (see below). However, S is given by

$$S = (A/(2*r*N)) * \sqrt{M*m} * (1 + ((N*N/(M*M)))^{a/2}) * (1 + ((N*N/(m*m)))^{b/2})$$

where N is the noise-correlation beam FWHM (see discussion of the noisefwhm parameter for rules governing how this value is determined). "a" indicates exponentiation and a and b depend on which uncertainty is being calculated. For $\sigma(A)$, $a = b = 3/2$. For M and x , $a = 5/2$ and $b = 1/2$. For m , y , and p , $a = 1/2$ and $b = 5/2$. $f(I)$ is calculated in the correlated noise case according to

$$f(I) = \sqrt{f(A)*f(A) + (N*N/(M*m))*(f(M)*f(M) + f(m)*f(m))}$$

Note well the following caveats: * Fixing Gaussian component parameters will tend to cause the parameter uncertainties reported for free parameters to be overestimated. * Fitting a zero level offset that is not fixed will tend to cause the reported parameter uncertainties to be slightly underestimated. * The parameter uncertainties will be inaccurate at low SNR (a $\sim 10\%$ for $SNR = 3$). * If the fitted region is not considerably larger than the largest component that is fit, parameter uncertainties may be mis-estimated. * An accurate rms noise measurement, r , for the region in question must be supplied.

Alternatively, a sufficiently large signal-free region must be present in the selected region (at least about 25 noise beams in area) to auto-derive such an estimate. * If the image noise is not statistically independent from pixel to pixel, a reasonably accurate noise correlation scale, N , must be provided. If the noise correlation function is not approximately Gaussian, the correlation length can be estimated using

$$N = \sqrt{(2*\ln(2)/\pi) * \text{double-integral}(dx dy C(x,y))/\sqrt{\text{double-integral}(dx dy C(x,y) * C(x,y))}}$$

where $C(x,y)$ is the associated noise-smoothing function * If fitted model components have significant spatial overlap, the parameter uncertainties are likely to be mis-estimated (i.e., correlations between the parameters of separate components are not accounted for). * If the image being analyzed is an interferometric image with poor uv sampling, the parameter uncertainties may be significantly underestimated.

The deconvolved size and position angle errors are computed by taking the maximum of the absolute values of the differences of the best fit deconvolved value of the given parameter and the deconvolved size of the eight possible combinations of (FWHM major axis +/- major axis error), (FWHM minor axis +/- minor axis error), and (position angle +/- position angle error). If the source cannot be deconvolved from the beam (if the best fit convolved source size cannot be deconvolved from the beam), upper limits on the deconvolved source size are sometimes reported. These limits simply come from the maximum major and minor axes of the deconvolved gaussians taken

from trying all eight of the aforementioned combinations. In the case none of these combinations produces a deconvolved size, no upper limit is reported.

EXAMPLE:

Here is how one might fit two gaussians to multiple channels of a cube using the fit from the previous channel as the initial estimate for the next. It also illustrates how one can specify a region in the associated continuum image as the region to use as the fit for the channel.

```
default imfit
imagename = "co_cube.im"
# specify region using region from continuum
region = "continuum.im:source.rgn"
chans = "2~20"
# only use pixels with positive values in the fit
excludepix = [-1e10,0]
# estimates file contains initial parameters for two Gaussians in channel 2
estimates = "initial_estimates.txt"
logfile = "co_fit.log"
# append results to the log file for all the channels
append = "True"
imfit()
```

imhead-task.html

0.1.39 imhead

Requires:

Synopsis

List, get and put image header parameters

Arguments

Inputs	
imagename	Name of the input image allowed: string Default:
mode	Mode of operation: "add", "del", "get", "history", "list", "put", or "summary". Modes "add", "del", and "put" will not work if the image is read-only (eg a FITS image). allowed: string Default: summary
hdkey	The associated keyword for modes "add", "del", "get", or "put". Only "get" will work if the image is read-only (eg, a FITS image). allowed: string Default:
hdvalue	Value of keyword for modes add or put. allowed: any Default: variant
verbose	Give a full listing of beams or just a short summary? Only used when the image has multiple beams and mode="summary". allowed: bool Default: False

Returns

variant

Example

PARAMETER SUMMARY

imagename Input image name. example: imagename='ngc5921_task.image'
 mode Mode of operation. Supported values: 'list', 'summary', 'history', 'get', 'put', 'add', 'del'.
 NOTE: 'add', 'del', and 'put' should be used with caution, and will not work if the image is read-only (eg FITS images are read-only in CASA).
 hdkey Keyword to use with get, put, add, or del. example: hdkey='telescope'
 hdvalue keyword value used for modes "put" and "add". Also used for mode="del" when hdvalue="masks. example: hdvalue='VLA'

This task allows the user to manipulate metadata associated with a CASA image. Both float and complex valued images are fully supported. The supported mode values

add Add a new metadata value to the image.
 del Delete a key or reset its value to a fiducial value if possible.
 Ignores all but imagename, mode, and hdkey parameters.
 get Return the specified keyword value. Ignores all but imagename, mode, and hdkey parameters.
 history Log image history. Ignores all but imagename and mode parameters.
 list Show supported keywords and their values. Ignores all but imagename and mode parameters.
 put Modify the specified value associated with the keyword.
 summary Log a summary of the image and return a dictionary of various metadata values.
 Ignores all but imagename and mode parameters.

See below for details about how these modes act for specific keywords.

NOTE: Only limited checking is implemented to ensure modifying a specific value will leave the image metadata in a consistent state, so, if one is not careful, one could end up with an image that has an inconsistent set of metadata and is therefore, nonsensical and useless. That is, PROCEED AT YOUR OWN RISK when using modes add, del, or put.

NOTE: For measurement sets, the task vishead should be used.

Supported keywords can be listed using mode = 'list'

beammajor/bmaj Major axis of the clean beam
 beamminor/bmin Minor axis of the clean beam
 beampa/bpa Position angle of the clean beam
 NOTE: If the image contains multiple beams, use mode="summary" to list them with verbose=True.
 bunit Image units (K, Jy/beam, etc)
 cdeltn Pixel size, nth axis. n is one-based.
 crpixn The pixel designated as the reference location, nth axis n is one-based.
 crvaln World coordinate value of the reference pixel for the nth axis. n is one-based.

ctypen	Name of nth axis. n is one-based.
cunitn	Units of nth axis. n is one based.
datamax	Maximum pixel value.
datamin	Minimum pixel value.
date-obs	Date (epoch) of the observation.
equinox	Direction reference frame.
imtype	Image type (eg Intensity)
minpos	World coordinate position of minimum pixel value.
minpixpos	Pixel coordinate position of minimum pixel value.
maxpos	World coordinate position of maximum pixel value.
maxpixpos	Pixel coordinate position of maximum pixel value.
object	Source name
observer	Observer name
projection	Direction coordinate projection (eg 'SIN', 'TAN', or 'ZEA').
reffreqtype	Spectral reference frame.
restfreq	Rest Frequency.
shape	Number of pixels along each axis.
telescope	Telescope name.

NOTES on mode="add"

The behavior of mode="add" depends on the keyword. Below is a summary of the per keyword behavior of this mode. In general, the return value will be True if the operation succeeds, or False if it fails or is not supported. If unsuccessful or not supported, a message is normally logged which describes the failure. In most cases, you probably want to use mode='put' rather than mode='add'. We continue to support mode='add' mainly for backward compatibility.

Keyword	Behavior for mode="del"
beammajor or bmaj	If image has no beam(s), a single, global, circular beam of diameter specified in hdvalue is added. hdvalue must be a valid angular quantity (string or dictionary) or the operation will fail and False will be returned. If the image has a beam(s), the operation fails and False is returned. Examples of acceptable values of hdvalue are "4arcsec", qa.quantity({'unit': 'arcsec', 'value': 4.0}). If you wish an image to have multiple beams, use ia.setrestoringbeams.
beamminor or bmin	Behavior is the same as that for beammajor or bmaj.
beampa or bpa	Operation has no effect and always returns false. If you wish to add beammajor, bmaj, beamminor, or bmin.
bunit	If image has no brightness unit, add the value specified in hdvalue to be a unit supported by CASA. Else do nothing and return False.
cdelt*	No effect. Addition of coordinate system parameters is not supported. False. Use the cs tool to add coordinates.
crpix*	No effect. Addition of coordinate system parameters is not supported.

	False. Use the cs tool to add coordinates.
crval*	No effect. Addition of coordinate system parameters is not supported.
	False. Use the cs tool to add coordinates.
ctype*	No effect. Addition of coordinate system parameters is not supported.
	False. Use the cs tool to add coordinates.
cunit*	No effect. Addition of coordinate system parameters is not supported.
	False. Use the cs tool to add coordinates.
datamax	No effect. Addition of statistical parameters is not supported.
datamin	No effect. Addition of statistical parameters is not supported.
date-obs or epoch	No effect.
equinox	No effect.
imtype	If image type does not exist, add the type specified in hdvalue. hdvalue can be "Undefined", "Intensity", "Beam", "Column Density", "Depolarization Ratio", "Kinetic Temperature", "Magnetic Field", "Optical Depth", "Rotation Measure", "Rotational Temperature", "Spectral Index", "Velocity", or "Velocity Dispersion".
masks	No effect. Addition of masks is not supported. Use <code>ia.calcmask()</code> .
maxpos	No effect. Addition of statistical parameters is not supported.
maxpixpos	No effect. Addition of statistical parameters is not supported.
minpos	No effect. Addition of statistical parameters is not supported.
minpixpos	No effect. Addition of statistical parameters is not supported.
object	If image has no object, add the value specified in hdvalue. Else do nothing.
observer	If image has no observer, add the value specified in hdvalue. Else do nothing.
projection	No effect.
reffreqtype	No effect.
restfreq	If image has a spectral coordinate and no rest frequency, set the rest frequency to the value specified in hdvalue. This value must be a valid CASA quantity in standard units. Else do nothing and return False. Examples of valid values are <code>{'unit': 'GHz', 'value': 1.0}</code> .
shape	No effect.
telescope	If image has no telescope, add the value specified in hdvalue. Else do nothing.
any user defined key	Add the key-value pair if the key does not exist. Else do nothing and return False.

NOTES on mode="del"

The behavior of mode="del" depends on the keyword. Below is a summary of the per keyword behavior of this mode. In general, the return value will be True if the operation succeeds, or False if it fails or is not supported. If unsuccessful or not supported, a warning message is normally logged which describes the failure.

Keyword	Behavior for mode="del"
beammajor or bmaj	Deletes all beams. Returns False if the image has no beams.
beaminor or bmin	Deletes all beams. Returns False if the image has no beams.
beampa or bpa	Deletes all beams. Returns False if the image has no beams.
bunit	Sets the associated value to the empty string.
cdelt*	No effect. Deletion of coordinate system parameters is not supported.
crpix*	No effect. Deletion of coordinate system parameters is not supported.

crval*	No effect. Deletion of coordinate system parameters is not supported.
ctype*	No effect. Deletion of coordinate system parameters is not supported.
cunit*	No effect. Deletion of coordinate system parameters is not supported.
datamax	No effect. Deletion of statistical parameters is not supported.
datamin	No effect. Deletion of statistical parameters is not supported.
date-obs or epoch	No effect.
equinox	No effect.
imtype	No effect.
masks	Deletes the single mask specified in hdvalue, or if hdvalue="", delete
maxpos	No effect. Deletion of statistical parameters is not supported.
maxpixpos	No effect. Deletion of statistical parameters is not supported.
minpos	No effect. Deletion of statistical parameters is not supported.
minpixpos	No effect. Deletion of statistical parameters is not supported.
object	Sets the associated value to the empty string.
observer	Sets the associated value to the empty string.
projection	No effect.
reffreqtype	No effect.
restfreq	No effect.
shape	No effect.
telescope	Sets the associated value to the empty string.
any user defined key	Deletes the key-value pair.

NOTES ON mode='get'

The data type of the value returned by imhead when mode='get' depends on the keyword. Below is a list of keywords on the data type that will be returned when mode='get' for each. A "quantity dictionary" is a dictionary with 'value' and 'unit' keys that can be used as input to various methods of the qa tool.

keyword	data type returned when mode='get'
beammajor	quantity dictionary
beamminor	quantity dictionary
beampa	quantity dictionary
bmaj	quantity dictionary
bmin	quantity dictionary
bpa	quantity dictionary
bunit	string
cdelt*	quantity dictionary
crpix*	float
crval*	quantity dictionary, unless the value for the stokes axis is requested in which case an array of strings is returned
ctype*	string
cunit*	string
datamax	image pixel data type
datamin	image pixel data type

date-obs or epoch	string (YYYY/MM/DD/hh:mm:ss format)
equinox	string
imtype	string
masks	string array
maxpos	string
maxpixpos	integer array
minpos	string
minpixpos	integer array
object	string
observer	string
projection	string
reffreqtype	string
restfreq	quantity dictionary
shape	integer array
telescope	string
any user defined key	string

NOTES on mode='put'

In general, mode='put' will modify the specified key to the specified value, with the following examples. True is returned if the metadatum was successfully modified, False otherwise. Normally, a diagnostic message is logged if there is a failure. Only the parameter specified is modified; eg, no modification of reference direction occurs to implicitly account for precession to a new reference frame. The following are the exceptional cases for

beammajor or bmaj	Will always fail if image has multiple beams. Use <code>ia.setrestoringbeam</code> in this case. If image has no beam(s), a single, global, circular beam is specified in <code>hdvalue</code> is added. <code>hdvalue</code> must be a valid angular quantity (string or dictionary) or the operation will fail and False will be returned. If the image has a single beam, the value of the major axis will be modified unless the specified value is smaller than the minor axis of the existing beam, in which case nothing is modified and False is returned. Examples of acceptable values of <code>hdvalue</code> are "4arcsec", <code>qa.quantity(4.0, unit='arcsec')</code> .
beamminor or bmin	Behavior is the same for <code>bmaj</code> , although of course if the image already has a single beam, the specified value must be less than the existing major axis value, or nothing is modified and False is returned.
beampa or bpa	If the image does not already have a single beam, nothing is modified and False is returned. Angular units are required.
bunit	Fails if <code>hdvalue</code> is not a supported CASA unit.
cdelt*	One-based axis must be less than or equal to the number of axes in the image. <code>hdvalue</code> type must be a number (in which case the unit of the corresponding axis is assumed) or a quantity (string or dictionary). If a quantity, the units must conform to the existing axis unit.
crpix*	One-based axis must be less than or equal to the number of axes in the image. <code>hdvalue</code> type must be a number. Will fail if the polarization axis is specified.
crval*	One-based axis must be less than or equal to the number of axes in the image.

	If not the polarization/stokes axis, hdvalue type must be a number (in which case the unit of the corresponding axis is assumed), a quantity (string or dictionary), or a valid measure format (such as a sexagesimal direction specification for an axis with angular units). If a quantity, the unit must conform to the existing axis unit. If the stokes/polarization axis, c must provide an array of stokes/polarization strings (["I", "Q", "XX"]) the same length as the stokes axis. If the stokes axis is degenerate, one must alternatively provide a string indicating the stokes value (eg "U").
ctype*	One-based axis must be less than or equal to the number of axes in the image. hdvalue type must be a string.
cunit*	One-based axis must be less than or equal to the number of axes in the image. unit must conform to the existing axis unit. Will fail if stokes/polarization axis.
datamax	This cannot be modified. False is always returned.
datamin	This cannot be modified. False is always returned.
date-obs or epoch	A valid time specification must be given.
equinox	A valid direction reference frame specification string must be given.
imtype	A CASA-supported image type string must be given or the image type will default to be set to 'Intensity'
masks	Masks may not be modified. False is always returned.
maxpos	This cannot be modified.
maxpixpos	This cannot be modified.
minpos	This cannot be modified.
minpixpos	This cannot be modified.
object	hdvalue must be a string.
observer	hdvalue must be a string.
projection	hdvalue must be a string representing a supported CASA projection specification.
reffreqtype	hdvalue must be a string representing a supported CASA velocity reference specification.
restfreq	hdvalue can be a number (in which case frequency axis units are assumed) or a quantity string or quantity dictionary in which case the unit must conform. Only the active rest frequency may be modified. For more functionality use <code>cs.setrestfrequency()</code> .
shape	This cannot be modified.
telescope	hdvalue must be a string.
any user defined key	hdvalue can be practically any supported input parameter type.

EXAMPLES

```
# mode='get'. Image has direction and spectral coordinates
epoch = imhead(imagename=imagename, mode="get", hdkey="date-obs")
observer = imhead(imagename=imagename, mode="get", hdkey="observer")
projection = imhead(imagename=imagename, mode="get", hdkey="projection")
restfreq = imhead(imagename=imagename, mode="get", hdkey="restfreq")

# mode='add'
if imhead(imagename=imagename, mode="add", hdkey="mykey", hdvalue="myvalue"):
```

```

        print "mykey added".
    else:
        print "addition of mykey failed".

# mode="del"
if imhead(imagename=imagename, mode="del", hdkey="mykey"):
    print "mykey deleted".
else:
    print "deletion of mykey failed".

# mode="put"
# change the reference RA value
key = 'crval1'
imhead(imagename=imagename, mode="put", hdkey=key, hdvalue="3:00:00")
# or equivalently
imhead(imagename=imagename, mode="put", hdkey=key, hdvalue="45deg")

# change the direction reference frame (NOTE, no precession of the existing
# reference values is done!)
imhead(imagename=imagename, mode="put", hdkey="equinox", hdvalue="GALACTIC")

# change the object
imhead(imagename=imagename, mode="put", hdkey="object", hdvalue="Milliways, also known as TH")

```

imhistory-task.html

0.1.40 imhistory

Requires:

Synopsis

Retrieve and modify image history

Arguments

Inputs	
imagename	Name of the input image allowed: string Default:
mode	Mode to run in, "list" to retrieve history, "append" to append a record to history. allowed: string Default: list
verbose	Write history to logger if mode="list"? allowed: bool Default: True
origin	Origin of appended message. Only used for mode="append". allowed: string Default: imhistory
message	eMessage to append. Only used of mode="append". allowed: string Default:

Returns

variant

Example

PARAMETER SUMMARY

imagename	Name of the input (CASA, FITS, MIRIAD) image
mode	Operating mode. Either "list" or "append" is accepted.

verbose	Log history if mode="list"?
origin	Origin of appended message if mode="append".
message	Message to append if mode="append".

This task provides access to the logtable of an image, where generally history information is stored. Two operation modes are supported. When mode="list", the history messages are returned as an array of strings. If verbose=True, this information is also written to the logger. When mode="append", a specified message (along with its specified origin) are appended to the log and True is returned if successful.

```
# get history and list to logger
imhistory("my.im", mode="list", verbose=True)

# append a message
imhistory("my.im", origin="myorigin", "ETI signal found in this image!")
```


immath-task.html

0.1.41 immath

Requires:

Synopsis

Perform math operations on images

Description

math on images

Arguments

Inputs	
imagename	<p>a list of input images</p> <p>allowed: any</p> <p>Default: variant</p>
mode	<p>mode for math operation (evalexpr, spix, pola, poli)</p> <p>allowed: string</p> <p>Default: evalexpr</p>
outfile	<p>File where the output is saved</p> <p>allowed: string</p> <p>Default: immath_results.im</p>
expr	<p>Mathematical expression using images</p> <p>allowed: string</p> <p>Default: IM0</p>
varnames	<p>a list of variable names to use with the image files</p> <p>allowed: any</p> <p>Default: variant</p>
sigma	<p>standard deviation of noise for debiasing</p> <p>allowed: string</p> <p>Default: 0.0mJy/beam</p>
polithresh	<p>Threshold in linear polarization intensity image below which to mask pixels.</p> <p>allowed: string</p> <p>Default:</p>
mask	<p>Mask to use. See help par.mask. Default is none.</p> <p>allowed: string</p> <p>Default:</p>
region	<p>Region selection. See "help par.region" for details. Default is to use the full image.</p> <p>allowed: string</p> <p>Default:</p>
box	<p>Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.</p> <p>allowed: string</p> <p>Default:</p>
chans	<p>Channels to use. See "help par.chans" for details. Default is to use all channels.</p> <p>allowed: string</p> <p>Default:</p>
stokes	<p>Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.</p> <p>allowed: string</p> <p>Default:</p>
stretch	<p>Stretch the mask if necessary and possible? See help stretch.par</p> <p>allowed: bool bool</p> <p>Default: False</p>
imagemd	<p>An image name from which metadata should be copied. The input can be either an image listed under imagename or any other image on disk. Leaving this parameter unset may copy header metadata from any of the input images, which one is not guaranteed.</p> <p>allowed: string</p> <p>Default:</p>

Returns

bool

Example

This task evaluates mathematical expressions involving existing image files. The results of the calculations are stored in the designated output file. Options are available to specify mathematical expression directly or pre-defined expression for calculation of spectral index image, and polarization intensity and position angle images are available. The image file names imbedded in the expression or specified in the `imagename` parameter for the pre-defined calculations may be CASA images or FITS images.

NOTE: Index values start at 0 Use the `imhead` task to see the range of index values for each axes.

Keyword arguments:

`imagename` input image name(s)

Default: none;

Examples: `mode='evalexpr'; imagename=['image1.im', 'image2.im']`

The text 'IM0' is replaced by 'image1.im' in the expression and 'IM1' is replaced with 'image2.im'

`mode='spix'; imagename=['image1.im','image2.im']` will calculate an image of $\log(S1/S2)/\log(f1/f2)$, where S1 and S2 are fluxes and f1 and f2 are frequencies

`mode='pola'; imagename='multistokes.im'` (where that image contains both Q and stokes planes) or `imagename=['imageQ.im','imageU.im']` will calculate

an image of polarization angle distribution, where imageQ.im and

imageU.im are Stokes Q and U images, respectively. Calculate $0.5 \cdot \arctan(U/Q)$

`mode='poli'; imagename=['imageQ.im','imageU.im','imageV.im']` will calculate total polarization intensity image, where imageQ.im, imageU.im, imageV.im

are Stokes Q, U, and V images, respectively. Alternatively, with

`imagename = ['imageQ.im','imageU.im']` the linear polarization intensity

image will be calculated. In the case where imagename is a single multi-stokes image, the total polarization image will be calculated if all of the Q, U, and

V stokes planes are present, and the linear polarization intensity image will be calculated if the Q and U (but not V) planes are present.

mode mode for mathematical operation
Default: evalexpr
Options: 'evalexpr' : evaluate a mathematical expression defined in 'expr'
 'spix' : spectralindex image
 'pola' : polarization position angle image
 'poli' : polarization intensity image

>>> mode expandable parameters

sigma (for mode='poli') standard deviation of noise of Stokes images with
 Jy/beam to correct for bias
 Default: '0.0Jy/beam' (= no debiasing)

polithresh (for mode='pola') Quantity (eg '30uJy/beam') describing the linear
 the Stokes V contribution is not included) polarization threshold.
 is written to the output image and is False for all corresponding
 values below this threshold. This parameter overrides the mask in
 (below). Default (') means use the value given in mask, or no mask
 value is empty as well.

expr (for mode='evalexpr') A LEL expression with images.
Image file names are specified in the imagenames parameter, and
the variables IM0, IM1, ... (or optionally via the varnames parameter)
are used to represent these files
in the expression. Explicit notations of file names in the
expression are also supported, in which cases the file names must
be enclosed in double quotes (") and imagename is ignored.

Examples:

Make an image that is image1.im - image2.im
expr=' (IM0 - IM1)'
or with an explicit notation,
expr='("image1.im" - "image2.im")'
Clip an image below a value (0.5 in this case)
expr = ' iif(IM0 >=0.5, IM0, 0.0) '

Note: iif (a, b, c) a is the boolean expression
 b is the value if true
 c is the value if false

Take the rms value of two images
expr = ' sqrt(IM0 * IM0 + IM1 * IM1) '

Note: No exponentiation available?

Build an image pixel by pixel from the minimum of (image2.im, 2*image1.im)
expr='min(IM1,2*max(IM0))'

varnames For mode="evalexpr". Instead of the default variable names IM0, IM1, ...
the names in this array to represent the input images.

outfile The output image. Overwriting an existing outfile is not permitted.
Default: immath_results.im; Example: outfile='results.im'

mask Mask to use. See help par.mask. Default is none. Also see polithresh.

stretch Stretch the input mask if necessary and possible. See below.

region Region selection. See "help par.region" for details. Default is to use the full
box Rectangular region to select in direction plane. See "help par.box" for details.

is to use the entire direction plane.
 Example: `box='10,10,50,50'`

`chans` Channels to use. See "help par.chans" for details. Default is to use all chans

`stokes` Stokes planes to use. See "help par.stokes" for details. Default is to use all
 Not used in for cases of `mode='poli'` or `mode='pola'`

`imagemd` The image from which metadata should be copied to the output. Default means r
 which image is used. The image must exist and should conform to the output i

Available functions in the `expr` and `mask` parameters:

`pi()`, `e()`, `sin()`, `sinh()`, `asinh()`, `cos()`, `cosh()`, `tan()`, `tanh()`,
`atan()`, `exp()`, `log()`, `log10()`, `pow()`, `sqrt()`, `complex()`, `conj()`
`real()`, `imag()`, `abs()`, `arg()`, `phase()`, `amplitude()`, `min()`, `max()`
`round()`, `isgn()`, `floor()`, `ceil()`, `rebin()`, `spectralindex()`, `pa()`,
`iif()`, `indexin()`, `replace()`, ...

If the mask has fewer dimensions than the image and if the shape
 of the dimensions the mask and image have in common are the same,
 the mask will automatically have the missing dimensions added so
 it conforms to the image.

For a full description of the allowed syntax see the
 Lattice Expression Language (LEL) documentation on the at:
<http://aips2.nrao.edu/docs/notes/223/223.html>

NOTE: where indexing and axis numbering are used in the above
 functions they are 1-based, ie. numbering starts at 1.

If `stretch` is true and if the number of mask dimensions is less than
 or equal to the number of image dimensions and some axes in the
 mask are degenerate while the corresponding axes in the image are not,
 the mask will be stretched in the degenerate axis dimensions. For example,
 if the input image has shape `[100, 200, 10]` and the input
 mask has shape `[100, 200, 1]` and `stretch` is true, the mask will be
 stretched along the third dimension to shape `[100, 200, 10]`. However if
 the mask is shape `[100, 200, 2]`, stretching is not possible and an
 error will result.

Note that when multiple image are used in the expression, there is
 no garauntee about which of those images will be used to create the metadata
 of the output image, unless `imagemd` is specified. If `imagemd` is specified, the following
 rules of metadata copying will be followed:

1. The pixel data type of the image specified by `imagemd` and the output image must
 be the same.
2. The metadata copied include the coordinate system (and so of course the dimensionality
 the output image must correspond to the coordinate system to be copied), the `image_info`

(which contains things like the beam(s)), the misc_info record (should one exist in the specified by imagemd), and the units.

3. If the output image is a spectral image, the brightness units are set to the empty string.

4. If the output image is a polarization angle image, the brightness unit is set to "deg" and the stokes coordinate is set to have a single plane of type of Pangle.

Examples:

Double all values in an image.

```
immath( imagesname='myimage.im', expr='IM0*2', outfile='double.im' )
```

or with an explicit notation,

```
immath( expr='"myimage.im"*2', outfile='double.im' )
```

Taking the sin of an image and adding it to another

Note that the images need to be the same size

```
immath(images=['image1.im', 'image2.im'], expr='sin(IM1)+IM0;', outfile='newImage.im')
```

Adding only the plane associated with the 'V' stokes value and

the 1st channel together in two images

```
immath(imagenname=[image1', 'image2'], expr='IM0+IM1', chans='1', stokes='V')
```

Selecting a single plane (5th channel), of the 3-D cube and

adding it to the original image. In this example the 2-D plane

gets expanded out and the values are applied to each plane in the

3-D cube.

```
default('immath')
```

```
imagenname='ngc7538.image'
```

```
outfile='chanFive.im'
```

```
expr='IM0'
```

```
chans='5'
```

```
go
```

```
default('immath')
```

```
imagenname=['ngc7538.image', 'chanFive.im']
```

```
outfile='ngc7538_chanFive.im'
```

```
expr='IM0+IM1'
```

```
go
```

Selecting and saving the inner 3/4 of an image for channels 40,42,44

as well as channels less than 10

```
default('immath')
```

```
imagenname='my_image.im'
```

```
expr='IM0'
```

```
box='25,25,123,123'
```

```
chans='<10;40,42,44'
```

```
outfile='my_image_inner.im' )
```

```
go
```

```

# Dividing an image by another, making sure we aren't dividing by zero
default('immath')
imagename=['orion.image', 'my.image']
expr='IM0/iif(IM1==0,1.0,IM1)'
outfile='my_orion.image'
go

# Applying a mask to all of the images in the expression
default('immath')
imagename=['ngc7538.image','ngc7538_clean.image']
expr='(IM0*10)+IM1'
mask='"ngc7538.mask"'
outfile='really_noisy_ngc7538.image'
go

# Applying a pixel mask contained in the image information
default('immath')
imagename='ngc5921.image'
expr='IM0*10'
mask='mask("ngc5921.mask")'
outfile='ngc5921.masked.image'
go

# Creating a total polarization intensity image from an multi-stokes image
# containing IQUV.
default('immath')
outfile='pol_intensity'
stokes=''
# in imagename, you can also specify a list containing single stokes images
# of Q and U (for linear polarization intensity) and V (for total
# polarization intensity)
imagename='3C138_pcal'
mode='poli'
go

# Creating a polarization position angle image
default('immath')
outfile='pol_angle.im'
mode='pola'
# you can also do imagename=['Q.im','U.im'] for single stokes images, order of
# the two Stokes images does not matter
imagename='3C138_pcal' # multi-stokes image containing at least Q and U stokes
go

```

```
# same as before but write a mask with values of False for pixels for which the
# corresponding linear polarization ( sqrt(Q*Q+U*U)) is less than 30 microJy/beam
polithresh='30uJy/beam'
go
```

```
# Creating a spectral index image from the images at two different observing frequencies
default('immath')
outfile='mySource_sp.im'
mode='spix'
imagename=['mySource_5GHz.im', 'mySource_8GHz.im']
go
```

TEMPORARY IMAGES

At this time, it is usually necessary for this task to create intermediate, temporary di. The names of these images start with '_immath' and are created in the directory in which is run. The task makes reasonable attempts to remove these images before it exits, but there are conceivably instances where the temporary images may not be automatically deleted. It is safe to delete them by hand, assuming no immath instance is currently in progress.

The hope and plan is that the necessity of these images will decrease in the future (ie. will require only RAM and not temporary persistent storage of intermediate results).

immoments-task.html

0.1.42 immoments

Requires:

Synopsis

Compute moments from an image

Description

Arguments

Inputs	
imagenname	Name of the input image allowed: string Default:
moments	List of moments you would like to compute allowed: intArray Default: 0
axis	The momement axis: ra, dec, lat, long, spectral, or stokes allowed: any Default: variant spectral
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: any Default: variant
box	Rectangular region(s) to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. allowed: string Default:
mask	Mask to use. See help par.mask. Default is none. allowed: string Default: variant
includepix	Range of pixel values to include allowed: any Default: variant -1
excludepix	Range of pixel values to exclude allowed: any Default: variant -1
outfile	Output image file name (or root for multiple moments) allowed: string Default:
stretch	Stretch the mask if necessary and possible? allowed: bool Default: False

Returns

bool

Example

The spectral moment distributions at each pixel are determined. See the cookbook and User Reference Manual for mathematical details.

The main control of the calculation is given by parameter `moments`:

```
moments=-1 - mean value of the spectrum
moments=0  - integrated value of the spectrum
moments=1  - intensity weighted coordinate;traditionally used to get
            'velocity fields'
moments=2  - intensity weighted dispersion of the coordinate; traditionally
            used to get "velocity dispersion"
moments=3  - median of I
moments=4  - median coordinate
moments=5  - standard deviation about the mean of the spectrum
moments=6  - root mean square of the spectrum
moments=7  - absolute mean deviation of the spectrum
moments=8  - maximum value of the spectrum
moments=9  - coordinate of the maximum value of the spectrum
moments=10 - minimum value of the spectrum
moments=11 - coordinate of the minimum value of the spectrum
```

Keyword arguments:

<code>imagename</code>	Name of input image default: none; example: <code>imagename="ngc5921_task.image"</code>
<code>moments</code>	List of moments you would like to compute default: 0 (integrated spectrum);example: <code>moments=[0,1]</code> see list above
<code>axis</code>	The moment axis default: (spectral axis); example: <code>axis=spec</code> options: ra, dec, latitude, longitude, spectral, stokes
<code>mask</code>	Mask to use. See help <code>par.mask</code> . Default is none.
<code>stretch</code>	Stretch the input mask if necessary and possible. See below.
<code>region</code>	Region selection. See "help <code>par.region</code> " for details. Default is to use the full image.
<code>box</code>	Rectangular region to select in direction plane. See

```

    "help par.box" for details. Default is to use the entire direction plane.
    Example: box="10,10,50,50"
    box = "10,10,30,30,35,35,50,50" (two boxes)
chans    Channels to use. See "help par.chans" for details. Default is to use
         all channels.
         See "help par.chans" for examples.
stokes   Stokes planes to use. See "help par.stokes" for details. Default is to
         use all Stokes planes.
         Example: stokes="IQUV";
         Example: stokes="I,Q"
includepix Range of pixel values to include
         default: [-1] (all pixels); example=[0.02,100.0]
excludepix Range of pixel values to exclude
         default: [-1] (don't exclude pixels); example=[100.,200.]
outfile  Output image file name (or root for multiple moments)
         default: "" (input+auto-determined suffix); example: outfile="source_moment"

```

If stretch is true and if the number of mask dimensions is less than or equal to the number of image dimensions and some axes in the mask are degenerate while the corresponding axes in the image are not, the mask will be stretched in the degenerate axis dimensions. For example, if the input image has shape [100, 200, 10] and the input mask has shape [100, 200, 1] and stretch is true, the mask will be stretched along the third dimension to shape [100, 200, 10]. However if the mask is shape [100, 200, 2], stretching is not possible and an error will result.

```

Example for finding the 1-momment, intensity-weighted
coordinate, often used for finding velocity fields.
immoments( axis="spec", imagename="myimage", moment=1, outfile="velocityfields" )

```

```

Example finding the spectral mean, -1 moment, on a specified region
of the image as defined by the box and stokes parameters
taskname="immoments"
default()
imagename = "myimage"
moment    = -1

axis      = "spec"
stokes    = "I"
box       = [55,12,97,32]
go

```

Example using a mask created with a second file to select the data used to calculate the 0-moments, integrated values. In this case the mask is from the calibrated.im file and all values

that have a value greater than 0.5 will be positive in the mask..

```
immoments( "clean.image", axis="spec", mask="calibrated.im>0.5", outfile="mom_withma
```

If an image has multiple (per-channel beams) and the moment axis is equal to the spectral axis, each channel will be convolved with a beam that is equal to the beam having the largest area in the beamset prior to moment determination.

`impbcor-task.html`

0.1.43 impbcor

Requires:

Synopsis

Construct a primary beam corrected image from an image and a primary beam pattern.

Arguments

Inputs	
imagename	Name of the input image allowed: string Default:
pbimage	Name of the primary beam image which must exist or array of values for the pb response. Default "" allowed: any Default: variant ""
outfile	Output image name. If empty, no image is written. Default "" allowed: string Default:
overwrite	Overwrite the output if it exists? Default False allowed: bool Default: False
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: any Default: variant
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. allowed: string Default:
mask	Mask to use. See help par.mask. Default is none. allowed: string Default:
mode	Divide or multiply the image by the primary beam image. Minimal match supported. Default "divide" allowed: string Default: divide
cutoff	PB cutoff. If mode is "d", all values less than this will be masked. If "m", all values greater will be masked. Less than 0, no cutoff. Default no cutoff allowed: double Default: -1.0
stretch	Stretch the mask if necessary and possible? See help par.stretch allowed: bool Default: False

Returns

bool

Example

PARAMETER SUMMARY

imagename	Name of the input (CASA, FITS, MIRIAD) image
pbimage	Name of the image (CASA, FITS, MIRIAD) of the primary beam pattern or an array
outfile	Name of output CASA image. Must be specified.
overwrite	If output file is specified, controls if an already existing file by the same name can be overwritten. If true, the user is not prompted, the file if it exists is automatically overwritten.
box	Rectangular region to select in direction plane. See "help par.box" for details. If not specified, is to use the entire direction plane.
region	Region selection. See "help par.region" for details. Default is to use the entire region.
chans	Channels to use. See "help par.chans" for details. Default is to use all channels.
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all stokes planes.
mask	Mask to use. See help par.mask. Default is none.
stretch	Stretch the input mask if necessary and possible. See help par.mask.
mode	Divide or multiply the image by the primary beam image. Minimal match supported.
cutoff	PB cutoff. If mode is "d", all values less than this will be masked. If "m", all values greater than this will be masked.

DESCRIPTION

Correct an image for primary beam attenuation using an image of the primary beam pattern. The primary beam pattern can be provided as an image, in which case 1. it must have the same shape as the input image and its coordinate system must be the same, or 2. it must be a 2-D image in which case its coordinate system must consist of a (2-D) direction coordinate which is the same as the direction coordinate in the input image and its direction plane must be the same shape as that of the input image. Alternatively, pbimage can be an array of pixel values in which case the same dimensionality and shape constraints apply.

One can choose between dividing the image by the primary beam pattern (mode="divide") or multiplying the image by the primary beam pattern (mode="multiply"). One can also choose to specify a cutoff limit for the primary beam pattern. For mode="divide", for all pixels below this cutoff in the primary beam pattern, the output image will be masked. In the case of mode="multiply", all pixels in the output will be masked corresponding to pixels with values greater than the cutoff in the primary beam pattern. A negative value for cutoff means that no cutoff will be applied, which is the default.

EXAMPLE

```
impbcor(imagename="attenuated.im", pbimage="mypb.im", outname="pbcorrected.im", mode="divide",
```


[importatca-task.html](#)

0.1.44 importatca

Requires:

Synopsis

Import ATCA RPFITS file(s) to a measurement set

Description

Imports an arbitrary number of ATCA RPFITS format data sets into a casa measurement set. If more than one band is present, they will be put in the same measurement set but in a separate spectral window. The task will handle both old ATCA and new CABB (after April 2009) archive data.

Arguments

Inputs		
files		Name of input ATCA RPFits file(s) allowed: stringArray Default:
vis		Name of output visibility file (MeasurementSet) allowed: string Default:
options		Processing options: birdie, reweight, noxycorr, fastmo- saic, hires, noac (comma separated list) allowed: string Default:
spw		Specify the spectral windows to use, default=all allowed: intArray Default: -1
nscans		Number of scans to skip followed by number of scans to read allowed: intArray Default: 0,0
lowfreq	GHz	Lowest reference frequency to select allowed: doubleGHz Default: 0.1
highfreq	GHz	Highest reference frequency to select allowed: doubleGHz Default: 999
fields		List of field names to select allowed: stringArray Default:
edge		Percentage of edge channels to flag. For combined zooms, this specifies the percentage for a single zoom window allowed: double Default: 8

Example

```
importatca(['2001-01-02_1*.C999', '2001-01-03_0*.C999'], 'c999.ms',
options='birdie,reweight', edge=10)
Imports a number of ATCA RPFITS-format data sets into
a casa measurement set. We are applying birdie flagging and lag spectrum reweighting
(as appropriate for old 33 channel continuum data) as well as flagging the 10% edge channels
```

importasap-task.html

0.1.45 importasap

Requires:

Synopsis

Convert ASAP Scantable data into a CASA visibility file (MS)

Arguments

Inputs	
infile	Name of input ASAP Scantable data allowed: string Default:
outputvis	Root name of the ms to be created. Note the .ms is NOT added allowed: string Default:
flagbackup	Back up flag column before applying flags. allowed: bool Default: True
overwrite	Over write an existing MS(s) allowed: bool Default: False

Returns

bool

Example

Keyword arguments:

```
asdm -- Name of input ASAP Scantable data  
      default: none; example: asdm='mydata.asap'
```

```
outputvis -- Root ms name. Note that a prefix (.ms) is NOT appended to this name.  
           default: none; example outputvis='myms.ms'
```

```
flagbackup  -- Backup original flags in >ms<.flagversions
              default: True

overwrite   -- overwrite an existing MS
              default: False (do not overwrite)
```

`importasdm-task.html`

0.1.46 importasdm

Requires:

Synopsis

Convert an ALMA Science Data Model observation into a CASA visibility file (MS)

Arguments

Inputs	
asdm	Name of input asdm directory (on disk) allowed: string Default:
vis	Root name of the ms to be created. Note the .ms is NOT added allowed: string Default:
createmms	Create a multi-MS output allowed: bool Default: False
separationaxis	Axis to do parallelization across(scan, spw, baseline, auto) allowed: string Default: auto
numsubms	The number of SubMSs to create (auto or any number) allowed: any Default: variant auto
corr_mode	specifies the correlation mode to be considered on input. A quoted string containing a sequence of ao, co, ac, or all separated by whitespaces is expected allowed: string Default: all
srt	specifies the spectral resolution type to be considered on input. A quoted string containing a sequence of fr, ca, bw, or all separated by whitespaces is expected allowed: string Default: all
time_sampling	specifies the time sampling (INTEGRATION and/or SUBINTEGRATION) to be considered on input. A quoted string containing a sequence of i, si, or all separated by whitespaces is expected allowed: string Default: all
ocorr_mode	output data for correlation mode AUTO_ONLY (ao) or CROSS_ONLY (co) or CROSS_AND_AUTO (ca) allowed: string Default: ca
compression	Flag for turning on data compression allowed: bool Default: False
lazy	Make the MS DATA column read the ASDM Binary data directly (faster import, smaller MS) allowed: bool Default: False
asis	Creates verbatim copies of the ASDM tables in the output measurement set. Value given must be a string of table names separated by spaces; A * wildcard is allowed. allowed: string Default:
wvr_corrected_data	Specifies which values are considered in the SDM binary data to fill the DATA column in the MAIN table of the MS. Expected values for this option are: no, for uncorrected data (default), yes, for the corrected data, and both, for for corrected and uncorrected data. Note if both is selected two measurement sets are created, one

Returns

void

Example

Keyword arguments:

asdm -- Name of input ASDM file (directory)

default: none; example: asdm='ExecBlock3'

vis -- Root ms name. Note that a prefix (.ms) is NOT appended to this name.

default: none

createmms -- Create a multi-MS partitioned according to the given separation axis.

For more detailed documentation on partition, Multi-MS and the MPI use in CASA, please see the help partition and help mstransform.

default: True

separationaxis -- Axis to do parallelization across.

default: 'auto'

Options: 'scan', 'spw', 'baseline', 'auto'

* The 'auto' option will partition per scan/spw to obtain optimal load balancing following criteria:

1 - Maximize the scan/spw/field distribution across sub-MSs

2 - Generate sub-MSs with similar size

* The 'scan' or 'spw' axes will partition the MS into scan or spw. The individual axes will not be balanced with respect to the number of rows.

* The 'baseline' axis is mostly useful for Single-Dish data. This axis will partition based on the available baselines. If the user wants only auto-correlations, use `ocorr_mode='ao'`. Note that if `numsubms='auto'`, partition will try to create as many sub-MSs as the number of available servers in the cluster. If the user wants to have one sub-MS per server, set the `numsubms` parameter to a number higher than the number of baselines to

numsubms -- The number of sub-MSs to create in the Multi-Ms.

default: 'auto'

Options: any integer number (example: numsubms=4)

The default 'auto' is to partition using the number of available servers given in the configuration. If the task is unable to determine the number of running servers, or the user

using mpicasa, numsubms will use 8 as the default.

Example: Launch CASA with 5 engines, where 4 of them will be used to create engine is used as the MPIClient.

```
mpicasa -n 5 casa --nogui --log2term
CASA> importasdm('uid__A1', createmms=True)
```

corr_mode -- correlation mode to be considered on input. Could be one or more of the following, ao, co, ac, or all
default: 'all'

srt -- spectral resolution type. Could be one or more of the following, fr, ca, bw, or all
default: 'all'

time_sampling -- specifies the time sampling, INTEGRATION and/or SUBINTEGRAION. could be one or more of the following i, si, or all.
default: 'all'

ocorr_mode -- output data for correlation mode AUTO_ONLY (ao) or CROSS_ONLY (co) or CROSS_AND_AUTO (ca)
default: 'ca'

compression -- produces compressed columns in the resulting measurement set.
default: False

lazy -- Make the MS DATA column read the ASDM Binary data directly (faster import, smaller MS). Instead of writing a copy of the visibilities into a standard DATA column, lazy=True will make importasdm only write a lookup-table such that later access to the DATA column will read the ASDM binary visibility data directly. This requires that the ASDM not be removed from its location as long the the DATA column is needed. Use method ms.asdmref() to query and manipulate the reference to the ASDM. lazy=True will save ca. 50% disk space and accelerate the DATA column access by ca. 10%. lazy=True will only work when there is visibility data in the ASDM, not with pure radiometer data.
default: False

asis -- creates verbatim copies of the ASDM tables in the output measurement set. The value given to this option must be a list of table names separated by space characters; the wildcard character '*' is allowed in table names.

```

        default: none

wvr_corrected_data -- specifies which values are considered in the
                    ASDM binary data to fill the DATA column in
                    the MAIN table of the MS. Expected values for
                    this option are 'no' for the uncorrected data
                    (this is the default), 'yes' for the corrected
                    data and 'both' for corrected and uncorrected
                    data. In the latter case, two measurement sets
                    are created, one containing the uncorrected
                    data and the other one, whose name is suffixed
                    by '-wvr-corrected', containing the corrected
                    data.
        default: 'no'

scans -- processes only the scans specified in the option's value. This value is a semicolon
        separated list of scan specifications. A scan specification consists in a
        scan index followed by the character ':' followed by a comma separated list of scan
        index ranges. A scan index is relative to the exec block it belongs to. Scan
        indices are 0-based while exec blocks' are 0-based. '0:1' or '2:2~6' or '0:1,1:2~6,8'
        are valid values for the option. '3:' alone will be interpreted as 'all the
        exec block#3'. An scan index or a scan index range not preceded by an exec
        block# will be interpreted as 'all the scans with such indexes in all the exec blocks'.
        If no scan index is specified, all the scans are considered.
        default: none (all scans)

ignore_time -- All the rows of the tables Feed, History, Pointing, Source, SysCal, CalDevice
              and Weather are processed independently of the time range of the selected exec
              block.
        default: False

process_syspower -- The SysPower table is processed if and only if this parameter is set to
                  True
        default: True

process_caldevice -- The CalDevice table is processed if and only if this parameter is set to
                   True
        default: True

process_pointing -- The Pointing table is processed if and only if this parameter is set to
                  True
        default: True

process_flags -- Create online flags based on the Flag.xml, Antenna.xml and SpectralWindow.xml
               and copy them to the FLAG_CMD sub-table of the MS. The flags will NOT be
               saved if the parameter applyflags is set to False. Optionally, the flags can also
               be saved to an external ASCII file if savecmds is set to True.
        default: True

>>> process_flags expandable parameter

```

```

tbuff    -- Time padding buffer (in seconds)
           default: 0.0

NOTE: this time is in seconds. You should currently
set the value of tbuff to be 1.5x the correlator
integration time if greater than 1 second. For
example, if the SDM has integrations of 3 seconds,
set tbuff=4.5. Likewise, set tbuff=15.0 for 10-sec
integrations.

applyflags -- Apply the online flags to the MS.
            default: False

savecmds  -- Save the flag commands to an ASCII file given by the parameter c
            default: False

outfile   -- Filename or list of filenames where to save the online flag comm
            default: ' ' --> by default it will save on a filename composed from the M
            Example: vis='uid_A02.ms', the outfile will be 'uid_A02_cmd.txt'.
                   vis='uid_A02-wvr-corrected.ms', the outfile will be 'uid_A02-w

flagbackup -- Backup original flags in >ms<.flagversions
            default: True

verbose   -- produce log output as asdm2MS is being run
            default: False

overwrite -- overwrite an existing MS or MS(s), if the option wvr_corrected_data='both'
            default: False (do not overwrite)

NOTE: the overwrite parameter affects all the output of the task. If any of
      exist, it will not overwrite them. MS(s), .flagversions, online flag f
      True, it will overwrite the MS, .flagversions and online flag file.

showversion -- report the version of the asdm2MS being used.
              default: False

useversion -- Selects the version of asdm2MS to be used ('\v3\' (default, should work for al
              default: v3

bdfflags  -- Set the MS FLAG column according to the ASDM _binary_ flags
            default: false

with_pointing_correction -- add (ASDM::Pointing::encoder - ASDM::Pointing::pointingDirection

```

```

                                to be written in MS::Pointing::direction

remove_ref_undef -- if set to True then apply fixspwbackport on the resulting MSes.
                    default: False

convert_ephem2geo -- ALMA uses ephemerides with observer location equal to the ALMA site.
                    For later processing of the radial velocity information in, e.g. cvel,
                    a geocentric ephemeris is needed.
                    Setting this option to True will perform the conversion of positions and
                    velocities on all attached ephemerides in the imported MS.
                    This will neither change the time-spacing nor the duration of the ephemeris.
                    No interpolation in time is done.

```

`importevla-task.html`

0.1.47 importevla

Requires:

Synopsis

Convert an Science Data Model observation into a CASA Measurement Set

Arguments

Inputs	
asdm	Name of input asdm directory (on disk) allowed: string Default:
vis	Root name of the ms to be created. Note the .ms is NOT added allowed: string Default:
ocorr_mode	Fill correlation mode AUTO_ONLY (ao), CROSS_ONLY (co) or CROSS_AND_AUTO (ca) allowed: string Default: co
compression	Flag for turning on data compression allowed: bool Default: False
asis	Create verbatim copies of these SDM tables in the MS. allowed: string Default:
scans	List of scans to fill (default is all scans). allowed: string Default:
verbose	Output lots of information while the filler is working allowed: bool Default: False
overwrite	Over write an existing MS allowed: bool Default: False
online	Create online flags allowed: bool Default: True
tbuff	Time padding buffer (in seconds) allowed: double Default: 0.0
flagzero	Create flag commands for zero points allowed: bool Default: True
flagpol	Create flag commands for cross-hand correlations allowed: bool Default: True
shadow	Create flag commands for shadowed data allowed: bool Default: True
tolerance	Amount of shadow allowed (in meters) allowed: double Default: 0.0
addantenna	File name or dictionary with additional antenna names, positions and diameters allowed: 245 variant Default:
applyflags	Apply flag commands to MS allowed: bool Default: False
savecmds	Save flag commands to an ASCII file allowed: bool Default: False

Returns

void

Example

Convert a Science Data Model (SDM) dataset into a CASA Measurement Set (MS). Will place online flags and specified clip/shadow flags into FLAG_CMD table and optionally apply to MS.

Warning: This version is under development and is geared to handling EVLA specific flag and system files, and is otherwise identical to `importasdm`.

HISTORY: Task created v1.0 S.T. Myers 2010-03-11 (3.0.1)
Last updated v9.0 S.M. Castro 2012-03-13 (3.4) code+doc

Keyword arguments:

`asdm` -- Name of input SDM file (directory)
 default: none;
 Example: `asdm='ExecBlock3'`

`vis` -- Root ms or scantable name, note a .ms is NOT appended to name
 default: none

`ocorr_mode` -- output data for correlation mode AUTO_ONLY
 (ao) or CROSS_ONLY (co) or CROSS_AND_AUTO (ca)
default: co (for EVLA)

`compression` -- produces compressed columns in the resulting measurement set.
 default: False

`asis` -- creates verbatim copies of the ASDM tables in
 the output measurement set. The value given to
this option must be a list of table names separated
by space characters; the wildcard character '*' is
 allowed in table names.

`scans` -- processes the scans requested in this parameter (default is
 all scans). For simplest use provide a comma-separated list of
scan ranges, e.g. `scans='1~3,5,10~20'`.
 default: '' = all scans

NOTE: A scan specification technically consists of an ExecBlock

index followed by the character ':' followed by a comma separated list of scan indexes or scan index ranges. The EVLA does not currently include more than one ExecBlock in a SDM so this specification prefix is not needed.

By default all the scans are considered.

overwrite -- Over write an existing MS

verbose -- produce log output as asdm2MS is being run

EVLA-specific parameters:

online -- create flagging commands for online flags. The commands will be saved to the sub-table of the MS. Optionally, it can also be saved to an ASCII file when to True.
default: True

>>> online expandable parameters
tbuff -- (float) time padding buffer (in seconds)
default: 0.0

NOTE: this time is in seconds. You should currently set the value of tbuff to be 1.5x the correlator integration time if greater than 1 second. For example, if the SDM has integrations of 3 seconds, set tbuff=4.5. Likewise, set tbuff=15.0 for 10-sec integrations.

flagzero -- create flags to clip out visibilities with zero values. The command will be saved to sub-table of the MS. Optionally, it can also be saved to an ASCII file when to True.
default: True

>>> flagzero expandable parameter(s)
flagpol -- (boolean) also zero-clip on cross-hands (default=False)

shadow -- create flags for antennas that are shadowed. The command will be saved to the sub-table of the MS. Optionally, it can also be saved to an ASCII file when to True.
default: True

>>> shadow expandable parameter
tolerance -- Amount of shadowing allowed in meters.

default: 0.0

addantenna -- It can be either a file name with additional antenna names, p
and diameters, or a Python dictionary with the same informati
You can use the flaghelper functions to create the dictionary
default: ''

To create a dictionary inside casapy.
> import flaghelper as fh
> antdic = fh.readAntennaList(antfile)

Where antfile is a text file in disk that contains information such as:
name=VLA01
diameter=25.0
position=[-1601144.96146691, -5041998.01971858, 3554864.76811967]
name=VLA02
diameter=25.0
position=[-1601105.7664601889, -5042022.3917835914, 3554847.245159178]

applyflags -- apply the online and specified flags to the MS
default: False

savecmds -- Save the flag commands to an ASCII file given by the parameter outfile. It wil
flag commands from online, flagzero and/or shadow if they are set to True.
default: False

>>> savecmds expandable parameter
outfile -- Filename where to save the flag commands.
default: ' ' --> by default it will save on a filename composed from the MS
Example: vis='evla.ms', the outfile will be 'evla_cmd.txt'.

NOTE: The file is open to save in append mode.

flagbackup -- Backup original flags in >ms<.flagversions
default: True

Examples:

- 1) Produces MS CLowTest_000.ms with autocorrelations.
You will find the online, zero, and shadow flags in the FLAG_CMD table
for later application. Does not apply any flags.

```
importevla(asdm='CLowTest_000',ocorr_mode='ca')
```

- 2) Produces MS CLowTest_000.ms without autocorrelations.

```
importevla(asdm='CLowTest_000')
```

- 3) Will apply online flags and uses a more conservative 2sec buffer before the start and after the end timeranges.

```
importevla(asdm='CLowTest_000',online=True,tbuff=2.0,applyflags=True)
```

- 4) This will create the FLAG_CMD sub-table using online flags only, but will not apply them to the MS.

```
importevla(asdm='CLowTest_000',online=True,flagzero=False,shadow=False)
```

- 5) This will write the online flags to the FLAG_CMD table. It will also save commands to clip zeros and to flag shadowed antennas to the table. The commands will be applied to the data and the APPLIED column of the FLAG_CMD will be updated to True.

```
importevla(asdm='CLowTest_000',online=True,flagzero=True,shadow=True,applyflags=True)
```

- 6) Import only scans 1, 2, 3, 5, 7, 9, save the online, shadow and clip commands to CLowTest_000_cmd.txt. Do not apply the flags. The commands will be saved to CLowTest_000_cmd.txt.

```
importevla(asdm='CLowTest_000',scans='1~3,5,7,9',online=True,flagzero=True,shadow=True,applyflags=False,savecmds=True)
```

You can use either flagdata or flagcmd to apply the flags later with the following commands:

Apply all the flags in the file using flagdata

```
flagdata('CLowTest_000.ms', mode='list', ininfile='CLowTest_000_cmd.txt')
```

Select by reason on the file

```
flagdata('CLowTest_000.ms',mode='list', ininfile='CLowTest_000_cmd.txt',  
         reason=['ANTENNA_NOT_POINTING','CORRELATOR_DATA_INVALID'])
```

Apply all the flags in the file using flagcmd

```
flagcmd('CLowTest_000.ms',inpmode='list',ininfile='CLTest_000_cmd.txt',action='apply')
```

HISTORY: Task last updated v9.0 S.M. Castro 2012-03-8 (3.4.0)
Docs last updated v9.0 S.M. Castro 2012-03-13 (3.4.0)

importfits-task.html

0.1.48 importfits

Requires:

Synopsis

Convert an image FITS file into a CASA image

Description

Convert an image FITS file into a CASA image

Keyword arguments: fitsimage – Name of input image FITS file default: none; example='3C273XC1.fits' imagename – Name of output CASA image default: none; example: imagename='3C273XC1.image' whichrep – If fits image has multiple coordinate reps, choose one. default: 0 means first; example: whichrep=1 whichhdu – If fits file contains multiple images, choose this one (0 == first) default=-1 use the first valid one; example: whichhdu=1 zeroblanks – Set blanked pixels to zero (not NaN) default=True; example: zeroblanks=True overwrite – Overwrite pre-existing imagename default=False; example: overwrite=True defaultaxes – Add the default 4D coordinate axes where they are missing default=False, example: defaultaxes=True defaultaxesvalues – List of values to assign to added degenerate axes when defaultaxes==True (ra,dec,freq,stokes) default = [], example: defaultaxesvalues=['13.5h', '-2.5deg', '88.5GHz', 'Q'] beam – List of values to be used to define the synthesized beam [BMAJ,BMIN,BPA] (as in the FITS keywords) default = [] (i.e.take from FITS file), example: beam=['0.35arcsec', '0.24arcsec', '25deg']

Arguments

Inputs	
fitsimage	Name of input image FITS file allowed: string Default:
imagename	Name of output CASA image allowed: string Default:
whichrep	If fits image has multiple coordinate reps, choose one. allowed: int Default: 0
whichhdu	If its file contains multiple images, choose one (0 = first HDU, -1 = first valid image). allowed: int Default: -1
zeroblanks	Set blanked pixels to zero (not NaN) allowed: bool Default: True
overwrite	Overwrite pre-existing imagename allowed: bool Default: False
defaultaxes	Add the default 4D coordinate axes where they are missing allowed: bool Default: False
defaultaxesvalues	List of values to assign to added degenerate axes when defaultaxes==True (ra,dec,freq,stokes) allowed: variant Default: []
beam	List of values to be used to define the synthesized beam [BMAJ,BMIN,BPA] (as in the FITS keywords) allowed: variant Default: []

Example

```
importfits(fitsimage='ngc3256.fits', imagename='ngc3256.im', overwrite=True)
```

importfitsidi-task.html

0.1.49 importfitsidi

Requires:

Synopsis

Convert a FITS-IDI file to a CASA visibility data set

Description

Convert a FITS-IDI file to a CASA visibility data set.

Arguments

Inputs	
fitsidifile	Name(s) of input FITS-IDI file(s) allowed: stringArray Default:
vis	Name of output visibility file (MS) allowed: string Default:
constobsid	If True, give constant obs ID==0 to the data from all input fitsidi files (False = separate obs id for each file) allowed: bool Default: False
scanreindexgap_s	min time gap (seconds) between integrations to start a new scan allowed: double Default: 0.

Example

If several files are given, they will be concatenated into one MS.

Keyword arguments:

fitsidifile -- Name(s) of input FITS-IDI file(s)
 default: none; must be supplied

```

example='3C273XC1.IDI'
example=['3C273XC1.IDI1','3C273XC1.IDI2']
    vis -- Name of output visibility file (MS)
           default: none; example: vis='3C273XC1.ms'
constobsid -- If True a constant obs id == 0 is given to all input files
           default = False (new obs id for each input file)
scanreindexgap_s -- if > 0., a new scan is started whenever the gap between two
           integrations is > the given value (seconds) or when a new field starts
           or when the ARRAY_ID changes.
           default = 0. (no reindexing)
    async -- Run asynchronously
           default = false; do not run asynchronously

```

importgmrt-task.html

0.1.50 importgmrt

Requires:

Synopsis

Convert a UVFITS file to a CASA visibility data set

Description

Convert a GRMT FITS file to a CASA visibility data set. Also read GMRT flag file(s) and flag data based on the contents of the files.

Arguments

Inputs	
fitsfile	Name of input UV FITS file allowed: string Default:
flagfile	Name of output visibility file (MS) allowed: any Default: variant
vis	Name of output visibility file (MS) allowed: string Default:

Example

Convert a GMRT FITS file to a CASA visibility data set:

Keyword arguments:

```
fitsfile -- Name of input UV FITS file
           default: none; example='3C273XC1.fits'
flagfile -- List of files containing flagging information.
           default: none; example='3c273XC1.flag'
           example=['3c273Cc1_1.flag', '3c273Cc2_1.flag', '']
```



```
vis      -- Name of output visibility file (MS)
          default: none; example: vis='3C273XC1.ms'
async    -- Run asynchronously
          default = false; do not run asychronously
```

Note: Don't forget to flag autocorrections using
taskname flagdata, autocorr = true

importmiriad-task.html

0.1.51 importmiriad

Requires:

Synopsis

Convert a Miriad visibility file into a CASA MeasurementSet

Description

Convert a Miriad visibility file into a CASA MeasurementSet with optional selection of spectral windows and weighting scheme

Arguments

Inputs	
mirfile	Name of input Miriad visibility file allowed: string Default:
vis	Name of output MeasurementSet allowed: string Default:
tsys	Use the Tsys to set the visibility weights allowed: bool Default: False
spw	Select spectral windows allowed: string Default: all
vel	Select velocity reference (TOPO,LSRK,LSRD) allowed: string Default:
linecal	(CARMA) Apply line calibration allowed: bool Default: False
wide	(CARMA) Select wide window averages allowed: string Default: all
debug	Display increasingly verbose debug messages allowed: int Default: 0

Returns

void

Example

```
importmiriad(mirfile='ngc5921.uv', vis='ngc5921.ms',tsys=True)
```

importuvfits-task.html

0.1.52 importuvfits

Requires:

Synopsis

Convert a UVFITS file to a CASA visibility data set

Description

Convert a UVFITS file to a CASA visibility data set. Don't forget to flag autocorrelations using taskname flagdata, autocorr = true

Arguments

Inputs	
fitsfile	Name of input UV FITS file allowed: string Default:
vis	Name of output visibility file (MS) allowed: string Default:
antnamescheme	VLA/EVLA/CARMA only; 'new' or 'old'; 'VA04' or '04' for VLA ant 4 allowed: string Default: old

Example

Convert a UVFITS file to a CASA visibility data set:

Keyword arguments:

```
fitsfile -- Name of input UV FITS file
           default = none; example='3C273XC1.fits'
vis -- Name of output visibility file (MS)
      default = none; example: vis='3C273XC1.ms'
antnamescheme -- Naming scheme for VLA/JVLA/CARMA antennas
```

```

default = old;
old: Antenna name is a number, '04'
    This option exists for backwards compatibility
    but can lead to ambiguous results when antenna
    indices are used for data selection.
new: Antenna name is not a number, e.g., 'VA04' or 'EA04'
    With this scheme, data selection via
    antenna names and indices is non-ambiguous.
async -- Run asynchronously
default = false; do not run asynchronously

```

Note: Don't forget to flag autocorrections using
taskname flagdata, autocorr = true

`importvla-task.html`

0.1.53 importvla

Requires:

Synopsis

Import VLA archive file(s) to a measurement set

Description

Imports an arbitrary number of VLA archive-format data sets into a casa measurement set. If more than one band is present, they will be put in the same measurement set but in a separate spectral window. The task will handle old style and new style VLA (after July 2007) archive data and apply the `tsys` to the data and to the weights.

Arguments

Inputs		
archivefiles		Name of input VLA archive file(s) allowed: stringArray Default:
vis		Name of output visibility file allowed: string Default:
bandname		VLA frequency band name:"=>obtain all bands in the archive file allowed: string Default:
frequencytol	Hz	Frequency shift to define a unique spectra window (Hz) allowed: doubleHz Default: 150000.0
project		Project name: " => all projects in files allowed: string Default:
starttime		start time to search for data allowed: string Default:
stoptime		end time to search for data allowed: string Default:
applytsys		apply nominal sensitivitiy scaling to data and weights allowed: bool Default: True
autocorr		import autocorrelations to ms, if set to True allowed: bool Default: False
antnamescheme		'old' or 'new'; 'VA04' or '04' for VLA ant 4 allowed: string Default: new
keepblanks		Fill scans with blank (empty) source names (e.g. tipping scans) allowed: bool Default: False
evlabands		Use updated eVLA frequencies and bandwidths for bands and wavelengths allowed: bool Default: False

Example

Imports an arbitrary number of VLA archive-format data sets into a casa measurement set. If more than one band is present, they will be put in the same measurement set but in a separate spectral window. The task will handle old style and new style VLA (after July 2007) archive data and apply the tsys to the data and to the weights.

Keyword arguments:

```
archivefiles -- Name of input VLA archive file(s)
               default: none. Must be supplied
               example: archivefiles = 'AP314_A959519.xp1'
               example: archivefiles=['AP314_A950519.xp1','AP314_A950519.xp2']
vis -- Name of output visibility file
      default: none. Must be supplied.
      example: vis='NGC7538.ms'
      Will not over-write existing ms of same name.
      A backup flag-file version 'Original' will be made in
      vis.flagversions. See help flagmanager
bandname -- VLA Frequency band
           default: => '' = all bands
           example: bandname='K'
           Options: '4'=48-96 MHz, 'P'=298-345 MHz, 'L'=1.15-1.75 GHz,
           'C'=4.2-5.1 GHz, 'X'=6.8-9.6 GHz, 'U'=13.5-16.3 GHz,
           'K'=20.8-25.8 GHz, 'Q'=38-51 GHz
frequencytol -- Tolerance in frequency shift in making spectral windows
               default: => 150000 (Hz). For Doppler shifted data, <10000 Hz may
               may produce too many unnecessary spectral windows.
               example: frequencytol = 1500000.0 (units = Hz)
project -- Project name to import from archive files:
          default: '' => all projects in file
          example: project='AL519'
          project = 'al519' or AL519 will work. Do not include
          leading zeros; project = 'AL0519' will not work.
starttime -- Time after which data will be considered for importing
            default: '' => all: Date must be included.
            syntax: starttime = '2003/1/31/05:05:23'
stoptime -- Time before which data will be considered for importing
           default: '' => all: Date must be included.
           syntax: stoptime = '2003/1/31/08:05:23'
applytsys -- Apply data scaling and weight scaling by nominal
            sensitivity (~Tsys)
            default: True. Strongly recommended
autocorr -- import autocorrelations to ms
           default: => False (no autocorrelations)
antnamescheme -- 'old' or 'new' antenna names.
```



```

    default => 'new' gives antnenna names
    'VA04' or 'EA13 for VLA telescope 04 and 13 (EVLA)
    'old' gives names '04' or '13'
keepblanks -- Should sources with blank names be filled into the data base
    default => false. Do not fill
    These scans are tipping scans (as of June 1, 2009) and should not
    be filled in the visibility data set.
evlabands -- Use the EVLA's center frequency and bandwidths for frequencies
    specified via wavelength or band.
default => True.
async -- Run asynchronously
    default = False; do not run asychronously

```

[imrebin-task.html](#)

0.1.54 imrebin

Requires:

Synopsis

Rebin an image by the specified integer factors

Arguments

Inputs	
imasename	Name of the input image allowed: string Default:
outfile	Output image name. allowed: string Default:
factor	Binning factors for each axis. Use imhead or ia.summary to determine axis ordering. allowed: intArray Default:
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: any Default: variant
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. Stokes planes cannot be rebinned. allowed: string Default:
mask	Mask to use. See help par.mask. Default is none. allowed: string Default:
dropdeg	Drop degenerate axes? allowed: bool Default: False
overwrite	Overwrite the output if it exists? Default False allowed: bool Default: False
stretch	Stretch the mask if necessary and possible? See help par.stretch allowed: bool Default: False
crop	Remove pixels from the end of an axis to be rebinned if there are not enough to form an integral bin? allowed: bool Default: True

Returns

bool

Example

PARAMETER SUMMARY

imagename	Name of the input (CASA, FITS, MIRIAD) image
outfile	Name of output CASA image. Must be specified.
factor	Array of binning factors for each axis, eg [2,3]. Use imhead or ia.summary to determine order of axes in your image.
region	Region selection. See "help par.region" for details. Default is to use the full image.
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.
chans	Channels to use. See "help par.chans" for details. Default is to use all channels.
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. Stokes planes cannot be rebinned.
mask	Mask to use. See help par.mask. Default is none.
dropdeg	Drop degenerate axes?
overwrite	Should the image of the same name as specified in outfile be overwritten? If true, the file if it exists is automatically overwritten.
stretch	Stretch the input mask if necessary and possible. See help par.mask.
crop	Only considered if the length of the input axis is not an integral multiple of the associated binning factor. If True, pixels at the end of the axis that do not form a complete bin are not included in the binning. If False, the remaining pixels are averaged to form the final bin along the axis.

DESCRIPTION

This application rebins the specified image by the specified integer binning factors for each axis. It supports both float valued and complex valued images. The corresponding output pixel value is the average of the input pixel values. The output pixel will be masked False if there were no good input pixels. A polarization axis cannot be rebinned.

The binning factors array must contain at least one element and no more elements than the number of input image axes. If the number of elements specified is less than the number of image axes, then the remaining axes not specified are not rebinned. All specified values must be positive. A value of one indicates that no rebinning of the associated axis will occur. Should this array contain any float values, they will be rounded to the next lowest integer. Note that in many images with both frequency and polarization

axes, the polarization axis preceeds the frequency axis. If you wish to rebin the frequency axis, it is recommended that you inspect your image with `imhead` or `ia.summary()` to determine the axis ordering.

Binning starts from the origin pixel of the bounding box of the selected region or the origin pixel of the input image if no region is specified. The value of `crop` is used to determine how to handle cases where there are pixels at the end of the axis that do not form a complete bin. If `crop=True`, extra pixels at the end of the axis are discarded. If `crop=False`, the remaining pixels are averaged into the final bin along that axis. Should the length of the axis to be rebinned be an integral multiple of the associated binning factor, the value of `crop` is irrelevant.

A value of `dropdeg=True` will result in the output image not containing axes that are degenerate in the specified region or in the input image if no region is specified. Note that, however, the binning factors array must still account for degenerate axes, and the binning factor associated with a degenerate axis must always be 1.

EXAMPLE

```
# rebin the first two axes (normally the direction axes)
imrebin(imagename="my.im", outfile="rebinned.im", factor=[2,3])

# rebin the frequency axis, which is the fourth axis in this image
imrebin(imagename="my2.im", outfile="rebinned2.im", factor=[1,1,1,4])
```

imreframe-task.html

0.1.55 imreframe

Requires:

Synopsis

Change the frame in which the image reports its spectral values

Arguments

Inputs	
imagename	Name of the input image allowed: string Default:
output	Name of the output image; " => modify input image allowed: string Default:
outframe	Spectral frame in which the frequency or velocity values will be reported by default allowed: string Default: lsrk
epoch	Epoch to be associated with this image e.g '2000/12/25/18:30:00.10' allowed: string Default:
restfreq	restfrequency to use for velocity values (e.g "1.420GHz" for the HI line) allowed: string Default:

Returns

void

Example

```
imagename -- name of casa image file to process on
```

```

output      -- name of output image '' means modify the input image itself
             default: '';
outframe    -- new spectral frame in which the frequency or
             velocity will be reported for.
             Options: 'lsrk','lsrd','bary','geo','topo','galacto',
             'lgroup','cmb'
default: 'lsrk'
>>>
epoch      -- when outframe is 'topo' or 'geo' a time in UTC is needed
             to decide when to do the frequency conversion. '' is to use
             the epoch of the input image
             default= ''

restfreq -- Specify rest frequency to use for output image
             default=''; '' means use the restfrequency already in input image
             For example for
             NH_3 (1,1) put restfreq='23.694496GHz'

```

[imregrid-task.html](#)

0.1.56 imregrid

Requires:

Synopsis

regrid an image onto a template image

Description

Imregrid will regrid an input image onto a new coordinate system from a template image or to a new directional reference frame. If a template image is used, then the input and template images must have the same coordinate structure.

Arguments

Inputs	
imagename	Name of the source image allowed: string Default:
template	A dictionary, refcode, or name of an image that provides the output shape and coordinate system allowed: any Default: variant get
output	Name for the regridded image allowed: string Default:
asvelocity	Regrid spectral axis in velocity space rather than frequency space? allowed: bool Default: True
axes	The pixel axes to regrid. -1 => all. allowed: intArray Default: -1
shape	Shape of the output image. Only used if template is an image. If not specified (-1), the output image shape will be the same as the template image shape along the axes that are regridded and the same as input image shape along the axes which are not regridded. allowed: intArray Default: -1
interpolation	The interpolation method. One of "nearest", "linear", "cubic". allowed: string Default: linear
decimate	Decimation factor for coordinate grid computation allowed: int Default: 10
replicate	Replicate image rather than regrid? allowed: bool Default: False
overwrite	Overwrite (unprompted) pre-existing output file? allowed: bool Default: False

Example

The imregrid task currently finds the nearest input pixel center and interpolates to the output pixel center. No averaging is done in any direction!

Imregrid will regrid an input image onto a new coordinate system from a template image or to a new directional reference frame. If a template image is used, then the input and template images must have the same coordinate structure.

Keyword arguments:

imagename	Name of the source image that needs to be regridded. Must be specified. example: imagename='orion.image'
template	Dictionary, directional reference code, or imagename defining the new shape and coordinate system, or 'get' to return the template dictionary for imagename. Recognized directional reference codes are: 'J2000', 'B1950', 'B1950_VLA', 'GALACTIC', 'HADEC', 'AZEL', 'AZELSW', 'AZELNE', 'ECLIPTIC', 'MECLIPTIC', 'TECLIPTIC', and 'SUPERGAL'. default: 'get'; example: template='orion_j2000.im' (for a template image), template='J2000' (to regrid the input image to J2000 coordinates).
shape	Shape of the output image. Only used if template is an image. If not specified (-1), the output image will be the same as the template image shape along the axes which are regridded and the same as the input image shape along the axes which are not regridded. If specified and the axis ordering of the input image and the template are not the same, the values in the array of shape will be reordered to match the axis ordering of the input image; the output image will have the same shape as the input image. Ignored if template is set equal to a reference code. If template is a dictionary, the output shape is retrieved from the dictionary so the shape input parameter is ignored.
output	Name for the regridded image. Must be specified. example: imagename='orion_shifted.im'
asvelocity	If True, regrid spectral axis with respect to velocity, not frequency. If False, regrid with respect to frequency. default: True
axes	The pixel axes to regrid. Default value [-1] => all except Stokes. Ignored if template is set equal to a reference code (in which case only the directional axes are regridded). If specified, this should be provided as an array. example axes=[0,1] (only regrid the first two axes, which are normally the directional axes).
interpolation	The interpolation method. One of 'nearest', 'linear', 'cubic'.
decimate	Decimation factor for coordinate grid computation
replicate	Replicate image rather than regrid?
overwrite">	Overwrite (unprompted) pre-existing output file?
async	Run task in a separate process (return CASA prompt) default: False; example: async=True

The new coordinate system is defined by the template parameter, which can be:

- * a recognized directional reference frame string. This will rotate the image and the coordinate system so that the new reference frame's axes are aligned to the cardinal directions (left-right, up-down).

Rotation occurs about the center direction pixel. If this pixel is not the reference pixel, a temporary copy of the original image is created and the coordinate system is adjusted so that the center direction pixel is the reference pixel. The coordinate system of the input image is not modified and the output image's reference direction pixel is the center pixel. Note that the conversion between one frame and another in general becomes less accurate as distance from the output image's reference pixel increases. Before the rotation occurs, the image is padded with masked pixels to ensure that all good pixels are used in the rotation (so that the corners of the image are not cropped after the rotation). After the image is rotated, the remaining pixels along the edges of the image in the directional coordinate are cropped, so that only the masked slices in the directional coordinate along the edges of the final image remain.

- * a {'csys': [valid coordinate system dictionary], 'shap': [int array describing the output image shape]}. This is normally obtained by first running `regrid` with `template='get'`. In this case it is a necessary dictionary.
- * 'get', which does not regrid but returns the template dictionary for `imagename`, suitable for modification and reuse (see the point immediately above),
- * the name of an image from which to get the coordinate system and shape. The input and template images must have the same coordinate structure.

Regridding of complex-valued images is supported. The real and imaginary parts are regridded independently and the resulting regridded pixel values are combined to form the regridded, complex-valued image.

The argument `{\stfaf replicate}` can be used to simply replicate pixels rather than regridding them. Normally (`{\stfaf replicate=F}`), for every output pixel, its world coordinate is computed and the corresponding input pixel found (then a little interpolation grid is generated). If you set `{\stfaf replicate=T}`, then what happens is that for every output axis, a vector of regularly sampled input pixels is generated (based on the ratio of the output and input axis shapes). So this just means the pixels get replicated (by whatever interpolation scheme you use) rather than regridded in world coordinate space. This process is much faster, but it's not a true world coordinate based regrid.

As described above, when `{\stfaf replicate}` is False, a coordinate is computed for each output pixel; this is an expensive operation. The argument `{\stfaf decimate}` allows you to decimate the computation of that coordinate grid to a sparse grid, which is then filled in via fast interpolation. The default for `{\stfaf decimate}` is 10. The number of pixels per axis in the sparse grid is the number of output pixels for that axis divided by the decimation factor. A factor of 10 does pretty well. You may find that for very non-linear coordinate systems (e.g. very close to the pole) that you have to reduce the decimation factor. You may also have to reduce the decimation factor if the number of pixels in the output image along an axis to be regridded is less than about 50, or the output image may be completely masked.

If one of the axes to be regridded is a spectral axis and `asvelocity=T`, the axis will be regridded to match the velocity, not the frequency, coordinate of the template coordinate system. Thus the output pixel values will correspond only to the velocity, not the frequency, of the output axis.

A variety of interpolation schemes are provided (only the first three characters to be specified). The cubic interpolation is substantially slower than linear, and often the improvement is modest. By default linear interpolation is used.

If an image has per-plane beams and one attempts to regrid the spectral axis, an exception is thrown.

RULES USED FOR GENERATING OUTPUT IMAGES IN SPECIFIC CASES

There are numerous rules governing the shape and coordinate system of the output image depending on the input image, template image, and whether default values of the axes and shape parameters are used. They are enumerated below.

NOTE: If you want to be certain of what type of output you will get, it is highly recommended you specify both axes and shape to avoid any ambiguity.

1. Rules governing Stokes axes

- 1.1. If the input image has no stokes axis, then the output image will have no stokes axis.
- 1.2. If the input image has a stokes axis, but the template image/coordinate system does not, and if the default value of the shape parameter is used or if shape is specified as `length`, then the specified value for the length stokes axis in equal to the length of the input image's stokes axis, then all stokes in the input image will be present in the output image.
- 1.3. If the input image has a stokes axis, but the template image/coordinate system does not, and if the value of the shape parameter is specified but the length of the resulting stokes axis is not equal to the length of the input image's stokes axis, a failure will occur.
- 1.4. If the input image has a stokes axis, if the template parameter is an image name, and the template image has a degenerate stokes axis, if the axes parameter is not specified, but does not contain the input stokes axis number, and if the shape parameter is not specified, then all stokes planes in the input image will be present in the output image.
- 1.5. If the input image has a stokes axis, if the template parameter is an image name, and the template image has a degenerate stokes axis, if the axes parameter is not specified, but does not contain the input stokes axis number, if the shape parameter is specified, then the specified length of the stokes axis is not equal to the length of the input stokes axis, a failure will occur.
- 1.6. If the input image has a stokes axis, if the template parameter is an image name, and the template image has a degenerate stokes axis, if the axes parameter is specified, but does not contain the input stokes axis number, then use the applicable rule of rules 1.7. and 1.8. for the input stokes axis number.

- image having a nondegenerate stokes axis.
- 1.7. If the input image has a stokes axis, if the template parameter is an image name, if the template image has a nondegenerate stokes axis, and if axes is not specified or if it is and does not contain the input stokes axis number, then only the stokes parameters common to both the input image and the template image will be present in the output image. If the input image and the template image have no common stokes parameters, failure will occur. If shape is specified and the specified stokes axis is not equal to the number of common stokes parameters in the template image, then failure will result.
 - 1.8. If the input image has a stokes axis, if the template parameter is an image name, if the template image has a nondegenerate stokes axis, and if axes is specified but does not contain the input image stokes axis number, then all stokes present in the input image will be present in the output image. If shape is also specified but the length of the specified stokes axis does not equal the input stokes axis, then failure will result.
2. Rules governing spectral axes
- In all cases, if the shape parameter is specified, the spectral axis length must be consistent with what one would normally expect in the special cases, or a failure will result.
- 2.1. If the input image does not have a spectral axis, then the output image will not have a spectral axis.
 - 2.2. If the input image has a degenerate spectral axis, if the template parameter is an image name, if the template image has a spectral axis, if axes is not specified or if it is and does not contain the input image spectral axis number, then the spectral coordinate of the input image will be copied to the output image and the output image will have a degenerate spectral axis.
 - 2.3. If the input image has a degenerate spectral axis, if the template parameter is an image name, if the template image has a spectral axis, if axes is specified and it contains the input image spectral axis number, then the spectral coordinate of the input image will be copied to the output image. If shape is not specified, the output image will have the same number of channels as the input image. If shape is specified, the output image will have the number of channels specified in shape for the spectral axis. In these cases, the pixel and mask values for all channels will be identical; the regridded single spectral plane is simply replicated n times where n is the number of channels in the output image.
 - 2.4. If the input image has a spectral axis, if the template parameter is an image name, if the template image does not have a spectral axis, if axes is not specified or if it is and does not contain the input image spectral axis number, then the spectral coordinate of the input image will be copied to the output image and the output image will have the same number of channels as the input image.
 - 2.5. If the input image has a spectral axis, if the template parameter is an image name, if the template image does not have a spectral axis, if axes is specified it contains the input image spectral axis number, then failure will result.
 - 2.6. If the input image has a spectral axis, if the template parameter is an image name, if the template image has a degenerate spectral axis, and if axes is unspecified or if it is and does not contain the spectral axis number of the input image, the spectral coordinate of the input image will be copied to the output image and the output image will have the same number of channels as the input image.
 - 2.7. If the input image has a spectral axis, if the template parameter is an image name, if the template image has a nondegenerate spectral axis, and if axes is unspecified or if it is and does not contain the spectral axis number of the input image, regrid the spectral axis of the input image to match the spectral axis of the template.

IMPORTANT NOTE ABOUT FLUX CONSERVATION

in general regridding is inaccurate for images that the angular resolution is poorly sampled. A check is done for such cases and a warning message is emitted if a beam present. However, no such check is done if there is no beam present. To add a restoring beam to an image, use `ia.setrestoringbeam()`.

Basic Examples

Regrid an image to the "B1950" or "GALACTIC" coordinate systems

```
imregrid(imagename="input.image", output="output.image", template="B1950")
imregrid(imagename="input.image", output="output.image", template="GALACTIC")
```

Note that when regridding to another coordinate system in the manner above, if the input image's direction coordinate is already in the frame specified by template, a straight copy of the image is made. No regridding is actually done.

Obtain a template dictionary from an image and then use it to regrid another image

```
temp_dict = imregrid(imagename="target.image", template="get")
imregrid(imagename="input.image", output="output.image", template=temp_dict)
```

In this example, the `template="get"` option is used in the first command in order to characterize the desired shape and coordinate system used, and a new dictionary, `temp_dict`, is generated accordingly. This is then used when performing the actual regridding of `input.image` in the second command.

More Advanced Examples

It is also possible to directly use a template image for regridding with `imregrid`. For this to work reliably and predictably, the dimensionality (i.e. which dimensions are present in an image) and the axis ordering of the input image must be the same. The type and ordering of the axes of both the input and template images can (and should) first be examined using the CASA `imhead` task. Any necessary reordering of axes can be performed using the CASA `imtrans` task.

Unless the user explicitly specifies which dimensions to regrid using the `axes` parameter (see the following example), `imregrid` will also attempt to regrid degenerate axes (i.e. image axes of length one pixel). Stokes axes are never regridded.

In the case where template is an image name and the default value of shape is specified, the output image's shape will be the same as the template image's shape along the axes which are regridded and the same as the input image's shape along the axes which are not regridded.

So for example, if the input image has a shape of [20, 30, 40] and the template image has a shape of [10, 40, 70] and only axes=[0, 1], the output image will have a shape of [10, 40, 40]. If only axes=[0], the output image will have a shape of [20, 30, 70].

Regrid input.image by directly using target.image as a template

```
imregrid(imagename="input.image", output="output.image", template="target.image", shape=[10, 40, 70])
```

In this example, it is assumed that the axis order of the input image is of the form (direction_x, direction_y, spectral, Stokes), where 'direction_x' and 'direction_y' are the directional coordinates on the sky (in some reference frame), 'spectral' is a velocity/frequency axis, and 'Stokes' contains polarization information. In this example, input.image might typically be a data cube of shape [100, 100, 40, 1]. Note that the default value of asvelocity (True) will be used so that the spectral axis will be regridded to the same velocity system as that of the template image.

Regrid only the first two axes of an image

Firstly, the user should inspect the type and ordering of the axes with imhead, and then correct with imtrans if necessary.

```
imregrid(imagename="input.image", output="output.image", template="target.image", axes=[0, 1])
```

The above command will regrid only the first two axes (normally the directional axes) of input.image and leave all other axes unchanged. The output image will have the shape of the template image along axes [0, 1] and the shape of the input image along the other axes since the shape parameter was explicitly specified.

Regrid the third axis, considering velocity rather than frequency units

```
imregrid(imagename="input.image", output="output.image", template="target.image", axes=[2])
```

This example regrids the spectral axis (zero-based axis number 2) with respect to velocity. The parameter asvelocity has been set to True. This is useful when eg, regridding a cube containing one spectral line to the shape of another cube containing a different spectral line.

Regrid the third axis, considering velocity rather than frequency units but first set the rest frequency

```
imhead("input.image", mode="put", hdkey="restfreq", hdvalue="110GHz")
imregrid(imagename="input.image", output="output.image", template="target.image", axes=[2])
```

The first command in this example uses the imhead task to set the value of the image rest frequency to a value of 110GHz in input.image. The following

`imregrid` command then performs a frequency units regridding only of the third axis listed (zero-based axis) (2), taking account of the `input.image` rest frequency in the `i`

imsmooth-task.html

0.1.57 imsmooth

Requires:

Synopsis

Smooth an image or portion of an image

Description

Arguments

Inputs	
imagename	Name of the input image. Must be specified. allowed: string Default:
kernel	Type of kernel to use. Acceptable values are "b", "box", or "boxcar" for a boxcar kernel, "g", "gauss", or "gaussian" for a gaussian kernel, "c", "common", or "commonbeam" to use the common beam of an image with multiple beams as the gaussian to which to convolve all the planes, "i" or "image" to use an image as the kernel. allowed: string Default: gauss
major	Major axis for the kernels. Standard quantity representation. Must be specified for kernel="boxcar". Example: "4arcsec". allowed: any Default: variant
minor	Minor axis. Standard quantity representation. Must be specified for kernel="boxcar". Example: "2arcsec". allowed: any Default: variant
pa	Position angle used only for gaussian kernel. Standard quantity representation. Example: "40deg". allowed: any Default: variant
targetres	If gaussian kernel, specified parameters are to be resolution of output image (True) or parameters of gaussian to convolve with input image (False). allowed: bool Default: False
kimage	Kernel image name. Only used if kernel="i" or "image". allowed: string Default:
scale	Scale factor. -1.0 means auto-scale. Only used if kernel="i" or "image". allowed: double Default: -1.0
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: any Default: variant
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. 281 allowed: string Default:
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.

Returns

any

Example

This task performs a Fourier-based convolution to 'smooth' the direction plane of an image. Smoothing is typically performed in order to reduce the noise in an image.

Keyword arguments:

imagename	Input image name. Must be specified.
outfile	Output smoothed image file name. Must be specified.
kernel	Type of kernel to use when smoothing ("g", "gauss", or "gaussian" for a gaussian kernel or "b", "box", or "boxcar" for a boxcar kernel), or if the image has multiple channels and kernel="commonbeam" (or "c", or "common"), convolve all channels to the smallest beam that encloses all beams in the input image, 'commonbeam' to use an image as the kernel. For boxcar smoothing, the major axis is parallel to the y-axis of the image and the minor axis is parallel to the x-axis. For a Gaussian, the orientation is specified by a position angle. A value of 0 degrees means the major axis is parallel to the y-axis and an increasing value of the position angle results in a counter-clockwise rotation of the ellipse. default: 'gauss'
major	Major axis of kernel which must be specified for boxcar smoothing. For Gaussian smoothing, the kernel parameters can alternatively be specified in the beam parameter. Standard quantity representations are supported. Example "4arcsec".
minor	Minor axis of kernel which must be specified for boxcar smoothing. For Gaussian smoothing, the kernel parameters can alternatively be specified in the beam parameter. Standard quantity representations are supported. Example "3arcsec".
pa	Position angle to use for gaussian kernel, unused for boxcar. The Gaussian kernel parameters can alternatively be specified in the beam parameter. Standard quantity representations are supported. Example "40deg".
beam	Record specifying Gaussian beam parameters. Do not specify any of major, minor, or pa if you choose to specify this parameter. Example: {"major": "5arcsec", "minor": "2arcsec", "pa": "20deg"}
targetres	Boolean used only for kernel='gauss'. If True, kernel parameters (major/minor/position angle or beam) are the resolution of the output image. If false, a gaussian with these parameters is convolved with the input image to produce

the output image.

`kimage` The image to be used as the convolution kernel. Only used if `kernel="image"` or `kernel="i"`.

`scale` Scale factor to use if `kernel="i"` or `kernel="image"`. -1.0 means auto-scale, which is the default.

`mask` Mask to use. See `help par.mask`. Default is none.

`region` Region selection. See `"help par.region"` for details. Default is to use the full image.

`box` Rectangular region to select in direction plane. See `"help par.box"` for details. Default is to use the entire direction plane. Example: "5, 10, 100, 200".

`chans` Channels to use. See `"help par.chans"` for details. Default is to use all channels.

`stokes` Stokes planes to use. See `"help par.stokes"` for details. Default is to use all Stokes planes. Example: 'I'

GAUSSIAN KERNEL

The direction pixels must be square. If they are not, use `imregrid` to regrid your image onto a grid of square pixels.

Under the hood, `ia.convolve2d()` is called with `scale=-1` (auto scaling). This means that, when using a restoring beam, pixel values in the output image are scaled in such a way as to conserve flux.

Major and minor are the full width at half maximum (FWHM) of the Gaussian. `pa` is the position angle of the Gaussian. The beam parameter offers an alternate way of describing the convolving Gaussian. If used, neither major, minor, nor `pa` can be specified. The beam parameter must have exactly two fields: "major", "minor", and "pa" (or "positionangle"). This is the record format for the output of `ia.restoringbeam()`. For example

```
beam = {"major": "5arcsec", "minor": "2arcsec", "pa": "20deg"}
```

If both beam and any of major, minor, and/or `pa` is specified for a Gaussian kernel, an exception will be thrown.

Alternatively, if the input image has multiple beams, setting `kernel='commonbeam'` will result in the smallest beam that encloses all beams in the image to be used as the target resolution to which the image is convolved.

In addition, the `targetres` parameter indicates if the specified Gaussian is to be the resolution of the final image (True) or if it is to be used to convolve the input image. If True, the input image must have a restoring beam. Use `imhead()` or `ia.restoringbeam()` to check for its existence. If the image has multiple beams and `targetres=True`, all planes in the image will be convolved so that the resulting resolution is that specified by the kernel parameters. If the image has multiple beams and `targetres=False`, each plane will be convolved with a Gaussian specified by beam (and hence, in general, the output image will also have multiple beams that vary with spectral channel and/or polarization).

If the units on the original image include Jy/beam, the units on the output image will be rescaled by the ratio of the input and output beams as well as rescaling by the area of convolution kernel.

If the units on the original image include K, then only the image convolution kernel rescaling is done.

BOXCAR KERNEL

major is length of the box along the y-axis and minor is length of the box along the x-axis. pa is not used and beam should not be specified. The value of targetres is not used.

IN GENERAL

The major, minor, and pa parameters can be specified in one of three ways

Quantity -- for example major=qa.quantity(1, 'arcsec')

Note that you can use pixel units, such as

major=qa.quantity(1, 'pix')

String -- for example minor='1pix' or major='0.5arcsec'

(i.e. a string that the Quanta quantity function accepts).

Numeric -- for example major=10.

In this case, the units of major and minor are assumed to be in arcsec and units of pa are assumed to be degrees.

Note: Using pixel units allows you to convolve axes with different units.

IMAGE KERNEL

If kernel="i" or "image", the image specified by kimage is used to convolve the input image. The coordinate system of the convolution image is ignored; only the pixel values are considered.

Fourier-based convolution is performed.

The provided kernel can have fewer dimensions than the image being convolved. In this case, it will be padded with degenerate axes. An error will result if the kernel has more dimensions than the image.

The scaling of the output image is determined by the argument {\stfaf scale}. If this is left unset, then the kernel is normalized to unit sum. If {\stfaf scale} is not left unset, then the convolution kernel will be scaled (multiplied) by this value.

Masked pixels will be assigned the value 0.0 before convolution.

The output mask is the combination (logical OR) of the default input \pixelmask\ (if any) and the OTF mask. Any other input \pixelmasks\

will not be copied. The function
maskhandler
should be used if there is a need to copy other masks too.

EXAMPLES

```
# smoothing with a gaussian kernel 20arseconds by 10 arseconds
imsmooth( imagename='my.image', kernel='gauss', major='20arcsec', minor='10arcsec', pa="0deg")

# the same as before, just a different way of specifying the kernel parameters
mybeam = {'major': '20arcsec', 'minor': '10arcsec', 'pa': '0deg'}
imsmooth( imagename='my.image', kernel='gauss', beam=mybeam)

# Smoothing using pixel coordinates and a boxcar kernel.
imsmooth( imagename='new.image', major='20pix', minor='10pix', kernel='boxcar')
```

[imstat-task.html](#)

0.1.58 imstat

Requires:

Synopsis

Displays statistical information from an image or image region

Arguments

Inputs	
imagename	Name of the input image allowed: string Default:
axes	List of axes to evaluate statistics over. Default is all axes. allowed: any Default: variant -1
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: string Default:
box	Rectangular region(s) to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. allowed: string Default:
listit	Print stats and bounding box to logger? allowed: bool Default: True
verbose	Print additional messages to logger? allowed: bool Default: True
mask	Mask to use. See help par.mask. Default is none. allowed: string Default:
stretch	Stretch the mask if necessary and possible? See help par.stretch allowed: bool Default: False
logfile	Name of file to write fit results. allowed: string Default:
append	If logfile exists, append to it if True or overwrite it if False allowed: bool Default: True
algorithm	Algorithm to use. Supported values are "chauvenet", "classic", "fit-half", and "hinges-fences". Minimum match is supported. allowed: string Default: classic
fence	Fence value for hinges-fences. A negative value means use the entire data set (ie default to the "classic" algorithm). Ignored if algorithm is not "hinges-fences". allowed: double Default: -1
center	Center to use for fit-half. Valid choices are "mean",

Returns

void

Example

Many parameters are determined from the specified region of an image. For this version, the region can be specified by a set of rectangular pixel coordinates, the channel ranges and the Stokes.

For directed output, run as
myoutput = imstat()

Keyword arguments:

imagename	Name of input image Default: none; Example: imagename='ngc5921_task.im'
axes	axes to compute statistics over. -1 => all axes.
region	Region selection. See "help par.region" for details. Default is to use the full image.
box	Rectangular region(s) to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. Example: box='10,10,50,50' box = '10,10,30,30,35,35,50,50' (two boxes)
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. Example: chans='3~20'
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. Example: stokes='I,Q'
listit	Print stats and bounding box to logger?
verbose	Print additional messages to logger?
mask	Mask to use. See help par.mask. Default is none.
stretch	Stretch the mask if necessary and possible? See help par.stretch
logfile	Name of file to write fit results.
append	If logfile exists, append to it (True) or overwrite it (False).
alogortihm	Algorithm to use to compute statistics. Supported values are "classic" and "hinges-fences" (minimum match supported.)
fence	Fence factor when algorithm = "hinges-fences". Negative values are not applicable and in these cases, the classic algorithm is used.
center	Center to use for "fit-half". Valid choices are "mean" (mean value of the selected pixels), "median" (median value of the selected pixels), and "zero" (0.0 is used as the center value). Ignored if algorithm is not "fit-half".

lside	For fit-half, use values \leq center for the real data? If false, use values \geq center as the real data. Ignored if algorithm is not "fit-half"
zscore	For chauvenet, this is the target maximum number of standard deviations data may have to be included. If negative, use Chauvenet's criterion. Ignored if algorithm is not "chauvenet".
maxiter	For chauvenet, this is the maximum number of iterations to attempt. Iterating will stop when either this limit is reached, or the zscore criterion is met. If negative, iterate until the zscore criterion is met. Ignored if algorithm is not "chauvenet".
clmethod	Method to use for calculating classical statistics. Supported methods are "auto", "tiled", and "framework". Ignored if algorithm is not "classic".

General procedure:

1. Specify inputs, then
2. myoutput = imstat()
or specify inputs directly in calling sequence to task
myoutput = imstat(imageName='image.im', etc)
3. myoutput['KEYS'] will contain the result associated with any of the keys given below

KEYS CURRENTLY AVAILABLE

blc	- absolute PIXEL coordinate of the bottom left corner of the bounding box surrounding the selected region
blcf	- Same as blc, but uses WORLD coordinates instead of pixels
trc	- the absolute PIXEL coordinate of the top right corner of the bounding box surrounding the selected region
trcf	- Same as trc, but uses WORLD coordinates instead of pixels
flux	- the flux or flux density. See below for details.
npts	- the number of unmasked points used
max	- the maximum pixel value
min	- minimum pixel value
maxpos	- absolute PIXEL coordinate of maximum pixel value
maxposf	- Same as maxpos, but uses WORLD coordinates instead of pixels
minpos	- absolute pixel coordinate of minimum pixel value
minposf	- Same as minpos, but uses WORLD coordinates instead of pixels
sum	- the sum of the pixel values: $\sum I_i$
sumsq	- the sum of the squares of the pixel values: $\sum I_i^2$
mean	- the mean of pixel values: $\bar{I} = \sum I_i / n$
sigma	- the standard deviation about the mean: $\sigma^2 = (\sum I_i - \bar{I})^2 / (n-1)$
rms	- the root mean square: $\sqrt{\sum I_i^2 / n}$

median - the median pixel value
 medabsdevmed - the median of the absolute deviations from the
 median
 quartile - the inner-quartile range. Find the points
 which are 25% largest and 75% largest (the median is
 50% largest).
 q1 - the first quartile.
 q3 - the third quartile

ALGORITHMS

Several types of statistical algorithms are supported:

- * classic: This is the familiar algorithm, in which all unmasked pixels are used. One may choose one of two methods, which vary only by performance, for computing classic statistics, via the `clmethod` parameter. The "tiled" method is the old method and is fastest in cases where there is a large number of individual sets of statistics to be computed and a small number of data points per set. This can occur when one sets the `axes` parameter, which causes several individual statistics to be computed. The "framework" method uses the new statistics framework to compute statistics. This method is fastest in the regime where one has a small number of individual sets of statistics to calculate, and each set has a large number of points. For example, this method is fastest when computing statistics over an entire image in one go (no axes specified). A `choice` option, "auto", chooses which method to use by predicting which will be faster based on the number of pixels in the image and the choice of the `axes` parameter.
- * fit-half: This algorithm calculates statistics on a dataset created from real and virtual data. The real values are determined by the input parameters `center` and `lside`. The parameter `center` tells the algorithm where the center value of the combined real+virtual dataset should be. It can be the mean or the median of the input image's pixel values, or at zero. The `lside` parameter tells the algorithm on which side of this center the real pixel values are located. True indicates that the real pixel values to be used are \leq center. False indicates the real pixel values to be used are \geq center. The virtual part of the dataset is then created by reflecting all the real data through the center value, to create a perfectly symmetric dataset composed of a real and a virtual component. Statistics are then calculated on this resultant dataset. These two parameters are ignored if algorithm is not "fit-half". Because the maximum value is virtual if `lside` is True, the minimum value is virtual if `lside` is False, the value of the maximum position (if `lside`=True) and the minimum position (if `lside`=False) is not reported in the returned record.
- * hinges-fences: This algorithm calculates statistics by including data in a range between $Q1 - f \cdot D$ and $Q3 + f \cdot D$, inclusive, where $Q1$ is the first quartile of the distribution of unmasked data, subject to any specified pixel ranges, $Q3$ is the third quartile, $D = Q3 - Q1$ (the inner quartile range), and f is the user-specified fence factor. Negative values of f indicate that the full distribution is to be used (ie, the classic algorithm is used). Small positive values of f will also be equivalent to using the classic algorithm. For $f = 0$, only data in the inner quartile range is used for computing statistics. The value of fence is silent.

ignored if algorithm is not "hinges-fences".

* **chauvenet**: The idea behind this algorithm is to eliminate outliers based on a maximum z-score. A z-score is the number of standard deviations a point is from the mean of a distribution. This method thus is meant to be used for (nearly) normal distributions. In general, this is an iterative process, with successive iterations discarding additional outliers as the remaining points get closer to forming a normal distribution. Iterating stops when no additional points lie beyond a specified zscore value, or, if zscore is negative, when Chauvenet's criterion is met (see [Chauvenet's criterion](#)). The parameter **maxiter** can be set to a non-negative value to prematurely abort this iterative process. When **verbose=T**, the "N iter" column in the table that is logged represents the number of iterations that were executed.

Chauvenet's criterion allows the target z-score to decrease as the number of points in the distribution decreases on subsequent iterations. Essentially, the criterion is that the probability of having one point in a normal distribution at a maximum z-score of **z_max** must be at least 0.5. **z_max** is therefore a function of (only) the number of points in the distribution and is given by

$$npts = 0.5/erfc(z_max/\sqrt{2})$$

where **erfc()** is the complementary error function. As iterating proceeds, the number of remaining points decreases as outliers are discarded, and so **z_max** likewise decreases. Convergence occurs when all remaining points fall within a z-score of **z_max**. Below is an illustrative table of **z_max** and their corresponding **npts** values. For example, it is likely that there will be a 5-sigma "bump" in a perfectly noisy image with one million independent elements.

z_max	npts
1.0	1
1.5	3
2.0	10
2.5	40
3.0	185
3.5	1,074
4.0	7,893
4.5	73,579
5.0	872,138
5.5	13,165,126
6.0	253,398,672
6.5	6,225,098,696
7.0	195,341,107,722

NOTES ON FLUX DENSITIES AND FLUXES

Fluxes and flux densities are not computed if any of the following conditions is met:

1. The image does not have a direction coordinate
2. The image does not have a intensity-like brightness unit. Examples of such units

- are Jy/beam (in which case the image must also have a beam) and K.
3. There are no direction axes in the cursor axes that are used.
 4. If the (specified region of the) image has a non-degenerate spectral axis, and the image has a tabular spectral axis (axis with varying increments)
 5. Any axis that is not a direction nor a spectral axis that is included in the cursor axes is not degenerate within in specified region

Note that condition 4 may be removed in the future.

In cases where none of the above conditions is met, the flux density(ies) (intensities integrated over direction planes) will be computed if any of the following conditions are met:

1. The image has no spectral coordinate
2. The cursor axes do not include the spectral axis
3. The spectral axis in the chosen region is degenerate

In the case where there is a nondegenerate spectral axis that is included in the cursor axes, the flux (flux density integrated over spectral planes) will be computed. In this case, the spectral portion of the flux unit will be the velocity unit of the spectral coordinate if it has one (eg, if the brightness unit is Jy/beam and the velocity unit is km/s, the flux will have units of Jy.km/s). If not, the spectral portion of the flux unit will be the frequency unit of the spectral axis (eg, if the brightness unit is K and the frequency unit is Hz, the resulting flux unit will be K.arcsec².Hz).

In both cases of flux density or flux being computed, the resulting numerical value is assigned to the "flux" key in the output dictionary.

ADDITIONAL EXAMPLES

```
# Selected two box region
# box 1, bottom-left coord is 2,3 and top-right coord is 14,15
# box 2, bottom-left coord is 30,31 and top-right coord is 42,43
imstat( 'myImage', box='2,3,14,15;30,31,42,43' )

# Select the same two box regions but only channels 4 and 5
imstat( 'myImage', box='2,3,14,15;30,31,42,43', chan='4~5' )

# Select all channels greater the 20 as well as channel 0.
# Then the mean and standard deviation are printed
results = imstat( 'myImage', chans='>20;0' )
print "Mean is: ", results['mean'], " s.d. ", results['sigma']

# Find statistical information for the Q stokes value only
# then the I stokes values only, and printing out the statistical
# values that we are interested in.
```

```

s1 = imstat( 'myimage', stokes='Q' )
s2 = imstat( 'myimage', stokes='I' )
    print "          | MIN | MAX | MEAN"
    print "  Q      | ",s1['min'][0]," | ",s1['max'][0]," | ",s1['mean'][0]
    print "  I      | ",s2['min'][0]," | ",s2['max'][0]," | ",s2['mean'][0]

# evaluate statistics for each spectral plane in an ra x dec x frequency image
myim = "noisy.im"
ia.fromshape(myim, [20,30,40])
# give pixels non-zero values
ia.addnoise()
ia.done()
# These are the display axes, the calculation of statistics occurs
# for each (hyper)plane along axes not listed in the axes parameter,
# in this case axis 2 (the frequency axis)
# display the rms for each frequency plane (your mileage will vary with
# the values).
stats = imstat(imagename=myim, axes=[0,1])
stats["rms"]
Out[10]:
array([ 0.99576014,  1.03813124,  0.97749186,  0.97587883,  1.04189885,
        1.03784776,  1.03371549,  1.03153074,  1.00841606,  0.947155  ,
        0.97335404,  0.94389403,  1.0010221 ,  0.97151822,  1.03942156,
        1.01158476,  0.96957082,  1.04212773,  1.00589049,  0.98696715,
        1.00451481,  1.02307892,  1.03102005,  0.97334671,  0.95209879,
        1.02088714,  0.96999902,  0.98661619,  1.01039267,  0.96842754,
        0.99464947,  1.01536798,  1.02466023,  0.96956468,  0.98090756,
        0.9835844 ,  0.95698935,  1.05487967,  0.99846411,  0.99634868])

```

imsubimage-task.html

0.1.59 imsubimage

Requires:

Synopsis

Create a (sub)image from a region of the image

Arguments

Inputs	
imagename	Input image name. Default is unset. allowed: string Default:
outfile	Output image name. Default is unset. allowed: string Default:
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. allowed: string Default:
region	Region selection. See "help par.region" for details. Default is to use the full image. allowed: string Default:
chans	Channels to use. See "help par.chans" for details. Default is to use all channels. allowed: string Default:
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes. allowed: string Default:
mask	Mask to use. See help par.mask. Default is none. allowed: any Default: variant
dropdeg	Drop degenerate axes allowed: bool Default: False
overwrite	Overwrite (unprompted) pre-existing output file? allowed: bool Default: False
verbose	Post additional informative messages to the logger allowed: bool Default: True
stretch	Stretch the mask if necessary and possible? allowed: bool Default: False

Returns

image

Example

PARAMETER SUMMARY

imagename	Name of the input image
outfile	Name of output file. Must be specified.
box	Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.
region	Region selection. See "help par.region" for details. Default is to use the full image.
chans	Channels to use. See "help par.chans" for details. Default is to use all channels.
stokes	Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.
mask	Mask to use. See help par.mask. Default ("") is none.
dropdeg	If True, all degenerate axes in the input image will be excluded in the output.
overwrite	If True, a pre-existing file of the same name as outfile will be overwritten.
verbose	Post additional informative messages to the logger.
stretch	Stretch the input mask if necessary and possible. Only used if a mask is specified. See help par.stretch.

OVERVIEW

This task copies all or part of the image to a new image specified by outfile. Both float and complex valued images are supported.

Sometimes it is useful to drop axes of length one (degenerate axes). Set `{\stfaf dropdeg}` equal to True if you want to do this.

The output mask is the combination (logical OR) of the default input `\pixelmask\` (if any) and the OTF mask. Any other input `\pixelmasks\` will not be copied. Use function `maskhandler` if you need to copy other masks too.

If the mask has fewer dimensions than the image and if the shape of the dimensions the mask and image have in common are the same, the mask will automatically have the missing dimensions added so it conforms to the image.

If `stretch` is true and if the number of mask dimensions is less than or equal to the number of image dimensions and some axes in the mask are degenerate while the corresponding axes in the image are not, the mask will be stretched in the degenerate dimensions. For example, if the input image has shape `[100, 200, 10]` and the input mask has shape `[100, 200, 1]` and `stretch` is true, the mask will be

stretched along the third dimension to shape [100, 200, 10]. However if the mask is shape [100, 200, 2], stretching is not possible and an error will result.

EXAMPLES

```
# make a subimage containing only channels 4 to 6 of the original image,
imsubimage(imagename="my.im", outfile="first.im", chans="4~6")
```

```
# Same as above command, just specifying chans in an alternate, more verbose
# way
imsubimage(imagename="my.im", outfile="second.im", chans="range=[4pix,6pix]")
```

```
# Same as the above command, but even more verbose way of specifying the spectral
# selection. Assumes the direction axes are axes numbers 0 and 1.
ia.open("my.im")
shape = ia.shape()
axes = ia.coordsys().names()
ia.done()
xmax = shape[axes.index("Right Ascension")] - 1
ymax = shape[axes.index("Declination")] - 1
reg = "box[[0pix,0pix],[ " + str(xmax) + "pix, " + str(ymax) + "pix]] range=[4pix,6pix]"
imsubimage(imagename="my.im", outfile="third.im", region=reg)
```

imtrans-task.html

0.1.60 imtrans

Requires:

Synopsis

Reorder image axes

Arguments

Inputs	
imagename	Name of the input image which must be specified. allowed: string Default:
outfile	Name of output CASA image. allowed: string Default:
order	New zero-based axes order. allowed: any Default: variant

Returns

bool

Example

PARAMETER SUMMARY

imagename	Name of the input image
outfile	Name of output CASA image. Must be specified.
order	Output axes mapping

This task reorders (transposes) the axes in the input image to the specified order. The associated pixel values and coordinate system are transposed.

The order parameter describes the mapping of the input axes to the output axes. It can be one of three types: a non-negative integer, a string, or a list of

strings. If a string or non-negative integer, it should contain zero-based digits describing the new order of the input axes. It must contain the same number of (unique) digits as the number of input axes. For example, specifying `order="1032"` or `order=1032` for a four axes image maps input axes 1, 0, 3, 2 to output axes 0, 1, 2, 3. In the case of `order` being a nonnegative integer and the zeroth axis in the input being mapped to zeroth axis in the output, the zeroth digit is implicitly understood to be 0 so that to transpose an image where one would use a string `order="0321"`, one could equivalently specify an int `order=321`. IMPORTANT: When specifying a non-negative integer and mapping the zeroth axis of the input to the zeroth axis of the output, do *not* explicitly specify the leading 0; eg, specify `order=321` rather than `order=0321`. Python interprets an integer with a leading 0 as an octal number.

Because of ambiguity for axes numbers greater than nine, using string or integer order specifications cannot handle images containing more than 10 axes. The `order` parameter can also be specified as a list of strings which uniquely match, ignoring case, the first characters of the image axis names (`ia.coordsys().names()`). So to reorder an image with right ascension, declination, and frequency axes, one could specify `order=["d", "f", "r"]` or equivalently `["decl", "frequ", "right a"]`. Note that specifying "ra" for the right ascension axis will result in an error because "ra" does not match the first two characters of "right ascension". Axes can be simultaneously inverted in cases where `order` is a string or an array of strings by specifying negative signs in front of the axis/axes to be inverted. So, in a 4-D image, `order="-10-3-2"` maps input axes 1, 0, 3, 2 to output axes 0, 1, 2, 3 and reverses the direction and values of input axes 1, 3, and 2.

EXAMPLE:

```
# Swap the stokes and spectral axes in an RA-Dec-Stokes-Frequency image
imagenname = "myim.im"
outfile = "outim.im"
order = "0132"
imtrans()
```

or

```
outfile = "myim_2.im"
order = 132
imtrans()
```

or

```
outfile = "myim_3.im"
order = ["r", "d", "f", "s"]
imtrans()
```

or

```
utfile = "myim_4.im"  
order = ["rig", "declin", "frequ", "stok"]  
imtrans()
```

imval-task.html

0.1.61 imval

Requires:

Synopsis

Get the data value(s) and/or mask value in an image.

Arguments

Outputs	
blc	<p>Bottom-left corner of the bounding box that encloses the region being examined..</p> <p>allowed: any</p> <p>Default: variant</p>
trc	<p>top-right corner of the bounding box that encloses the region being examined.</p> <p>allowed: any</p> <p>Default: variant</p>
axes	<p>A listing of the axis index numbers and the data stored along that axis.</p> <p>allowed: any</p> <p>Default: variant</p>
unit	<p>The units the data values are stored and displayed in.</p> <p>allowed: any</p> <p>Default: variant</p>
data	<p>The mask values found at the give point(s).</p> <p>allowed: any</p> <p>Default: variant</p>
mask	<p>The mask values found at the give point(s).</p> <p>allowed: any</p> <p>Default: variant</p>
Inputs	
imagename	<p>Name of the input image</p> <p>allowed: string</p> <p>Default:</p>
region	<p>Region selection. See "help par.region" for details. Default is to use the full image.</p> <p>allowed: any</p> <p>Default: variant</p>
box	<p>Rectangular region(s) to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.</p> <p>allowed: string</p> <p>Default:</p>
chans	<p>Channels to use. See "help par.chans" for details. Default is to use all channels.</p> <p>allowed: string</p> <p>Default:</p>
stokes	<p>Stokes planes to use. See "help par.stokes" for details. Planes specified must be contiguous. Default is to use all Stokes planes.</p> <p>allowed: string</p> <p>Default:</p>

Returns

void

Example

The data point(s) to be retrieved are those found in the specified region, which may be:

1. A region file or text string (see help par.region), with the following caveat:
 - * If the specified region is complex (eg, a union or intersection of multiple regions), only the first simple region in this set is used.
 - * If the region is not rectangular, then the rectangular region that circumscribes the specified region (ie the bounding box) is used to retrieve values, since the result arrays must be rectangular. The resulting mask values in this case are the result of ANDing the image mask values with the specified region mask values, eg if a pixel falls outside the specified region but within the bounding box, its mask value will be false even if its image mask value is true.
2. A region specified by a set of rectangular pixel coordinates, the channel ranges and/or the Stokes.

For directed output, run as

```
myoutput = imval()
```

Keyword arguments:

imagename -- Name of input image

Default: none; Example: imagename='ngc5921_task.im'

region -- Region selection. See "help par.region" for details. Default is to use the

Example: region='myimage.im.rgn'

region='region1'

box -- Rectangular region to select in direction plane. See "help par.box" for details. Default

Default: '' (referencepixel values for the Directional coord);

Example: box='10,10,50,50'

box = '10,10,30,30,35,35,50,50' (two boxes)

chans -- Channels to use. See "help par.chans" for details. Default is to use all channels.

stokes -- Stokes planes to use. See "help par.stokes" for details. Planes specified must be

Example: stokes='IQUV';

stokes='I,Q'

General procedure:

1. Specify inputs, then

2. `myoutput = imval()`
or specify inputs directly in calling sequence to task
`myoutput = im sval(imagename='image.im', etc)`
3. `myoutput['KEYS']` will contain the result associated with any
of the keys given below

KEYS CURRENTLY AVAILABLE

<code>blc</code>	- absolute PIXEL coordinate of the bottom left corner of the bounding box surrounding the selected region
<code>trc</code>	- the absolute PIXEL coordinate of the top right corner of the bounding box surrounding the selected region
<code>axes</code>	- List the data stored in each axis of the data block.
<code>unit</code>	- unit of the returned data values.
<code>data</code>	- data value(s) found in the given region
<code>mask</code>	- mask value(s) found in the given region. See important note above regarding returned mask values for non-rectangular regions.

NOTE: The data returned is in the same order as it is internally
stored, typically RA, DEC, spectral, stokes. Also both the data
and mask values are returned as Python Numpy arrays, for more
information on how to manipulate them see
http://numpy.scipy.org/#array_interface

Additional Examples

```
# The value and mask value at a single point (5,17,2,Q)
imval( 'myImage', box='5,5,17,17', chans=2, stokes='Q' )

# Select and report on two box regions
# box 1, bottom-left coord is 2,3 and top-right coord is 14,15
# box 2, bottom-left coord is 30,31 and top-right coord is 42,43
# Note that only the boxes for the
imval( 'myImage', box='2,3,14,15;30,31,42,43' )

# Select the same two box regions but only channels 4 and 5
imval( 'myImage', box='2,3,14,15;30,31,42,43', chan='4~5' )

# Select all channels greater the 20 as well as channel 0.
# Then the mean and standard deviation are printed
# Note that the data returned is a Python numpy array which
# has built in operations such as min, max, and means as
# demonstrated here.
results = imval( 'myImage', chans='>20;0' )
imval_data=results['data']
```

```
mask=results['mask']
# holds the absolute coordinates of the associated pixels in imval_data
coords = results['coords']
print "Data max: ", imval_data.max(), " mean is ", imval_data.mean()
swapped_data=imval_data.swapaxes(0,2)
swapped_mask=mask.swapaxes(0,2)
print "Data values for 21st channel: \n", swapped_data[0]
print "Mask values for 21st channel: \n", swapped_mask[0]
```

imview-task.html

0.1.62 imview

Requires:

Synopsis

View an image

Description

The imview task will display images in raster, contour, vector or marker form. Images can be blinked, and movies are available for spectral-line image cubes. Executing the imview task will bring up a display panel window, which can be resized. If no data file was specified, a Load Data window will also appear. Click on the desired data file and choose the display type; the rendered data should appear on the display panel.

A Data Display Options window will also appear. It has drop-down subsections for related options, most of which are self-explanatory.

The state of the imview task – loaded data and related display options – can be saved in a 'restore' file for later use. You can provide the restore filename on the command line or select it from the Load Data window.

Arguments

Inputs	
raster	<p>(Optional) Raster filename (string) or complete raster config dictionary. The allowed dictionary keys are file (string), scaling (numeric), range (2 element numeric vector), colormap (string), and colorwedge (bool).</p> <p>allowed: any</p> <p>Default: variant</p>
contour	<p>(Optional) Contour filename (string) or complete contour config dictionary. The allowed dictionary keys are file (string), levels (numeric vector), unit (float), and base (float).</p> <p>allowed: any</p> <p>Default: variant</p>
zoom	<p>(Optional) zoom can specify intermental zoom (integer), zoom region read from a file (string) or dictionary specifying the zoom region. The dictionary can have two forms. It can be either a simple region specified with blc (2 element vector) and trc (2 element vector) [along with an optional coord key ("pixel" or "world"; pixel is the default) or a complete region rectangle e.g. loaded with "rg.fromfileto record()". The dictionary can also contain a channel (integer) field which indicates which channel should be displayed.</p> <p>allowed: any</p> <p>Default: variant 1</p>
axes	<p>(Optional) this can either be a three element vector (string) where each element describes what should be found on each of the x, y, and z axes or a dictionary containing fields "x", "y" and "z" (string).</p> <p>allowed: any</p> <p>Default: variant</p>
out	<p>(Optional) Output filename or complete output config dictionary. If a string is passed, the file extension is used to determine the output type (jpg, pdf, eps, ps, png, xbm, xpm, or ppm). If a dictionary is passed, it can contain the fields, file (string), scale (float), dpi (int), or orient (landscape or portrait). The scale field is used for the bitmap formats (i.e. not ps or pdf) and the dpi parameter is used for scalable formats (pdf or ps).</p> <p>allowed: any</p> <p>Default: variant</p>

Returns

void

Example

The `imview` task provides access to a subset of all of the configuration options for loading and configuring the display of images in the `casaviewer`. This interface will evolve and eventually provide access to nearly all of the image options available in the `casaviewer`.

To simply create a `casaviewer` to set up interactively, you can use:

```
imview
```

To open a particular image:

```
imview "ngc5921.clean.image"
```

to open an image and overlay a contour:

```
imview "ngc5921.clean.image", "ngc5921.clean.image"
```

or equivalently:

```
imview( raster="ngc5921.clean.image", contour="ngc5921.clean.image" )
```

to output an image:

```
imview( raster="ngc5921.clean.image", out="ngc5921-01.png" )
```

There are five optional parameters for `imview` -- `raster`, `contour`, `zoom`, `axes`, and `out`. Each of these parameters can take a few different forms and are treated as python dictionaries:

```
raster -- (string) image file to open
         (dict)  file (string)    => image file to open
                  scaling (float) => scaling power cycles
                  range (float*2) => data range
                  colormap (string) => name of colormap
                  colorwedge (bool) => show color wedge?
contour -- (string) file to load as a contour
```

```

        (dict)   file (string)      => file to load
                levels (float*N)    => relative levels
                base (numeric)      => zero in relative levels
                unit (numeric)      => one in the relative levels
zoom  -- (int)   integral zoom level
        (string) region file to load as the zoom region
        (dict)   blc (numeric*2)   => bottom left corner
                trc (numeric*2)   => top right corner
                coord (string)    => pixel or world
                channel (int)      => channel to display
        (dict)   <region record>   => record loaded
                                e.g. rg.fromfileto record( )
axes  -- (string*3) demension to display on the x, y, and z axes
        (dict)   x                => dimension for x-axes
                y                => dimension for y-axes
                z                => dimension for z-axes
out   -- (string) file with a supported extension
        [jpg, pdf, eps, ps, png, xbm, xpm, ppm]
        (dict)   file (string)    => filename
                format (string)   => valid ext (filename ext overrides)
                scale (numeric)   => scale for non-eps, non-ps output
                dpi (numeric)     => dpi for eps or ps output
                orient (string)   => portrait or landscape

```

Examples:

- 1) A subset (zoom) of a raster image. Note the notation of curly brackets:

```

imview(raster="ngc5921.clean.image", out="ngc5921-02.png",
       zoom={'channel': 10, 'blc': [113,109], 'trc': [141,136]} )

```

- 2) An overlay of a raster image, ngc5921.clean.image, with a contour map of the same image ngc5921.clean.image. Data ranges are selected, as well as the colormap and the scaling cycles of the raster image. Contours are autogenerated and The x-axis will be Declination. The image is written out to a file named myout.png in the png format.

```

imview(raster={'file': 'ngc5921.clean.image',
               'range': [-0.01,0.03],
               'colormap': 'Hot Metal 2',
               'scaling': -1},
       contour={'file': 'ngc5921.clean.image'},
       axes={'x': 'Declination'} ,
       zoom={'channel': 7, 'blc': [75,75], 'trc': [175,175],

```

```

        'coord': 'pixel'},
    out='myout.png')

```

3) As example (2) but with an integral zoom level and no output to a file

```

imview(raster={'file': 'ngc5921.clean.image',
                'range': [-0.01,0.03],
                'colormap': 'Hot Metal 2'},
       contour={'file': 'ngc5921.clean.image'},
       axes={'x':'Declination'} ,
       zoom=2)

```

4) Now, the contour levels are explicitly given, a region file is used to define the zoom area

```

imview(raster={'file': 'ngc5921.clean.image',
                'range': [-0.01,0.03],
                'colormap': 'Hot Metal 2'},
       contour={'file': 'ngc5921.clean.image',
                'levels': [-0.2, 0.2, 0.25, 0.3, 0.35, 0.4, 0.6, 0.8] },
       zoom='myregion.rgn')

```

specifying "zoom={'file': 'myregion.rgn', 'channel': 10}" would result in the same level of zoom and would display channel number 10 from the cube.

initweights-task.html

0.1.63 initweights

Requires:

Synopsis

Initializes weight information in the MS

Arguments

Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
wtmode	Initialization mode allowed: string Default: nyq
tsystable	Tsys calibration table to apply on the fly allowed: string Default:
gainfield	Select a subset of calibrators from Tsys table allowed: string Default:
interp	Interpolation type in time[,freq]. default==linear,linear allowed: string Default:
spwmap	Spectral windows combinations to form for gaintables(s) allowed: intArray Default:
downtsp	Initialize the WEIGHT_SPECTRUM column. allowed: bool Default: False

Returns

void

Example

This task provides for initialization of the weight information in the MS. For ALMA interferometry and EVLA data, it should not generally be necessary to use this task, as the weight information should have been initialized properly at fill time (v4.2.2 and later).

Several initialization modes are supported via the `wtmode` parameter.

If `wtmode='nyq'` (the default), `SIGMA` and `WEIGHT` will be initialized according to bandwidth and integration time. This is the theoretically correct mode for raw normalized visibilities. (e.g., ALMA). For the EVLA, this is correct if switched-power and bandpass calibration will later be applied.

If `wtmode='sigma'`, `WEIGHT` will be initialized according to the existing `SIGMA` column.

If `mode='weight'`, `WEIGHT_SPECTRUM` will be initialized according to the existing `WEIGHT` column; `dowtspec=T` must be specified in this case.

If `wtmode='ones'`, `SIGMA` and `WEIGHT` will be initialized with 1.0, globally. This is a traditional means of initializing weight information, and is adequate when the integration time and bandwidth are uniform. It is not recommended for modern instruments (ALMA, EVLA), where variety in observational setups is common, and properly initialized and calibrated weights will be used for imaging sensitivity estimates.

There are two EXPERIMENTAL modes, `wtmode='tsys'` and `'tinttsys'`. In the modes, `SIGMA` and `WEIGHT` will be initialized according to `Tsys`, bandwidth, and integration time (used only in `'tinttsys'`), i.e.,

```
tsys      : weight=bw/Tsys^2
tinttsys: weight=bw*t_int/Tsys^2
```

These modes use `Tsys` values to calculate weight as is done in `Tsys` calibration. `Tsys` values are taken from a `tsys` calibration table given as `tsystable`. Selection of gain field (`gainfield`), interpolation method (`interp`), and spectral window mapping (`spwmap`) are supported, too.

Available types of interpolation are,

```
Time: 'nearest', 'linear', the variation of those with 'perobs',
      e.g., 'linearperobs' (enforce obsId boundaries in interpolation)
Freq: 'nearest', 'linear', 'cubic', 'spline', and the variation
      of those with 'flag', e.g., 'linearflag' (with
```

channelized flag).

See the help of `applycal` for details of interpolations.

Note if the weight in an MS is initialized with these modes and Tsys calibration table is applied with `calwt=True` after that, the weight would be contaminated by being divided by square of Tsys twice.

!!! USERS ARE ADVISED TO USE THESE EXPERIMENTAL MODES WITH CARE !!!

For the above `wtmodes`, if `dowtsp=T` (or if the `WEIGHT_SPECTRUM` column already exists), the `WEIGHT_SPECTRUM` column will be initialized (uniformly in channel in `wtmode='nyq'`, `'sigma'`, `'weight'`, and `'ones'`), in a manner consistent with the disposition of the `WEIGHT` column. If the `WEIGHT_SPECTRUM` column does not exist, `dowtsp=T` will force its creation.

Use of the `WEIGHT_SPECTRUM` column is only meaningful for ALMA data which will be calibrated with channelized Tsys information, or if the weights will become channelized after calibration, e.g., via averaging over time- and channel-dependent flagging. (A task for channel-dependent weight estimation from the data itself is also currently under development).

In non-channelized modes (`wtmode='nyq'`, `'sigma'`, `'weight'`, and `'ones'`) or when `dowtsp=F`, `SIGMA_SPECTRUM` column will be removed from MS. On the other hand, `SIGMA_SPECTRUM` column is added and initialized in channelized modes (`wtmode='tsys'` and `'tinttsys'`) if `dowtsp=T` or `WEIGHT_SPECTRUM` already column exists.

Two additional modes are available for managing the spectral weight info columns; these should be used with extreme care: If `wtmode='delwtsp'`, the `WEIGHT_SPECTRUM` column will be deleted (if it exists). If `wtmode='delsigsp'`, the `SIGMA_SPECTRUM` column will be deleted (if it exists). Note that creation of `SIGMA_SPECTRUM` is not supported via this method.

Note that this task does not support any prior selection. Initialization of the weight information must currently be done globally or not at all. This is to maintain consistency.

listcal-task.html

0.1.64 listcal

Requires:

Synopsis

List antenna gain solutions

Description

This task lists antenna gain solutions in tabular form. The table is organized as follows. Solutions are output by 1) Spectral window, 2) Antenna, 3) Time, 4) Channel, 5) and Polarization; where the inner-most loop is over polarization.

The listcal output table contains two table headers. The top-level header is printed each time the spectral window changes. This header lists the spectral window ID (SpwID), the date of observation (Date), the calibration table name (CalTable), and the measurement set name (MS name).

A lower-level header is printed when the the top-level header is printed, when the antenna names change, and every ‘pagerows’ of output. The lower-level header columns are described here:

Column Name	Description	Ant
Antenna name	(contains sub-columns: Amp, Phs, F)	
Time	Visibility timestamp corresponding to gain solution	
Field name	Chn	Channel number
Amp	Complex solution amplitude	
Phs	Complex solution phase	
F	Flag	

Elements of the “F” column contain an ‘F’ when the datum is flagged, and ‘ ’ (whitespace) when the datum is not flagged.

Presently, the polarization mode names (for example: R, L) are not given, but the ordering of the polrization modes (left-to-right) is equivalent to the order output by task listobs (see “Feeds” in listobs output).

Input Parameters:

vis Name of input visibility file default: none; example: vis=’ngc5921.ms’

caltable Name of input calibration table default: none; example:

caltable=’ngc5921.gcal’

field Select data based on field ID(s) or name(s) default: ”=,all; example:

field=’1’ field=’0~2’ field ids inclusive from 0 to 2 field=’3C*’ all field names starting with 3C

antenna Select calibration data based on antenna default: ”=,all; example:

antenna=’5’ antenna=’5,6’ antenna index 5 and 6 solutions

antenna=’VA05’,’VA06’ VLA antenna 5 and 6

spw Select spectral window, channel to list default: " -i All spws and channels; spw='2:34' spectral window 2, channel 34 will only list one spw, one channel at a time

listfile write output to disk; will not overwrite default: " -i write to screen

pagerows rows per page of listing default: 50; 0 -i do not paginate

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
caltable	Input calibration table to list allowed: string Default:
field	Field name or index; "==">all allowed: string Default:
antenna	Antenna name or index; "==">all; antenna='3' allowed: string Default:
spw	Spectral window and channel: "==">all; spw='5:0~10' allowed: string Default:
listfile	Disk file to write output: "==">to terminal allowed: string Default:
pagerows	Rows per page allowed: int Default: 50

Returns

void

Example

This task lists antenna gain solutions in tabular form. The table

is organized as follows. Solutions are output by

- 1) Spectral window,
- 2) Antenna,
- 3) Time,
- 4) Channel,
- 5) and Polarization;

where the inner-most loop is over polarization.

The listcal output table contains two table headers. The top-level header is printed each time the spectral window changes. This header lists the spectral window ID (SpwID), the date of observation (Date), the calibration table name (CalTable), and the measurement set name (MS name).

A lower-level header is printed when the the top-level header is printed, when the antenna names change, and every 'pagerows' of output. The lower-level header columns are described here:

Column Name	Description
-----	-----
Ant	Antenna name (contains sub-columns: Amp, Phs, F)
Time	Visibility timestamp corresponding to gain solution
Field	Field name
Chn	Channel number
Amp	Complex solution amplitude
Phs	Complex solution phase
F	Flag

Elements of the 'F' column contain an 'F' when the datum is flagged, and ' ' (whitespace) when the datum is not flagged.

Presently, the polarization mode names (for example: R, L) are not given, but the ordering of the polrization modes (left-to-right) is equivalent to the order output by task listobs (see 'Feeds' in listobs output).

Input Parameters:

vis	Name of input visibility file default: none; example: vis='ngc5921.ms'
caltable	Name of input calibration table default: none; example: caltable='ngc5921.gcal'
field	Select data based on field ID(s) or name(s) default: ''==>all; example: field='1' field='0~2' field ids inclusive from 0 to 2

field='3C*' all field names starting with 3C

antenna Select calibration data based on antenna name
 default: '' --> all; example: antenna='5';
 antenna='5,6' antenna index 5 and 6 solutions
 antenna='VA05','VA06' VLA antenna 5 and 6

spw Select spectral window(s), channel(s) to list
 default: '' --> All spws and channels;
 spw='2:34' spectral window 2, channel 34;
 spw='1:5,3~5:7~9' spectral window 1, channel
 5 and spectral windows 3 thru 5, channels
 7 thru 9.

listfile write output to disk; will not overwrite
 default: '' --> write to screen

pagerows rows per page of listing
 default: 50; 0 --> do not paginate

Example:

```
# Get path to CASA home dir
pathname=os.environ.get('CASAPATH').split()[0]
# Select uv-data (FITS) file
fitsdata=pathname+'/data/demo/NGC5921.fits'
# MS name; write to current directory
msdata='NGC5921.ms'
# import FITS data to MS
importuvfits(fitsfile=fitsdata, vis=msdata)
# Create model data for flux calibrator
setjy(vis=msdata)
# Calibration table name
caldata=msdata+'.bcal'
# Bandpass calibration
bandpass(vis=msdata, caltable=caldata)
# List a subset of calibration factors
listcal(vis=msdata, caltable=caldata, field='N5921_2, 0, 1',
        antenna='1,2,5;10~14', spw='0:1,0:22~25', pagerows=0)
```

Example Output:

```
SpwID = 0, Date = 1995/04/13, CalTable = NGC5921.ms.bcal (B Jones), MS name = /users/jcross
-----
Time      Field                      | Ant = 1                      | Ant = 2                      |
Chn|    Amp    Phs F    Amp    Phs F|    Amp    Phs F    Amp    Phs F|
```

Time	Field	Chn	Amp	Phs F	Amp	Phs F	Amp	Phs F	Amp	Phs F
09:21:46.0	1331+30500002_0	1	0.165	7.9	0.117	21.3	0.168	98.8	0.161	-116.8
10:05:27.9	1445+09900002_0	1	0.260	10.3	0.185	20.0	0.266	102.3	0.250	-116.1
10:09:05.3	N5921_2	1	0.047	54.2	0.030	50.7	0.057	-64.6	0.041	36.5
			Ant = 11				Ant = 12			

SpwID = 0, Date = 1995/04/13, CalTable = NGC5921.ms.bcal (B Jones), MS name = /users/jcross

Time	Field	Chn	Ant = 1				Ant = 2			
			Amp	Phs F	Amp	Phs F	Amp	Phs F	Amp	Phs F
09:21:46.0	1331+30500002_0	22	0.319	4.6	0.323	-6.8	0.311	109.6	0.315	-109.0
09:21:46.0	1331+30500002_0	23	0.318	4.4	0.323	-6.8	0.309	109.7	0.315	-108.8
09:21:46.0	1331+30500002_0	24	0.318	4.2	0.323	-6.6	0.309	109.8	0.316	-108.6
09:21:46.0	1331+30500002_0	25	0.319	4.3	0.323	-6.6	0.308	109.5	0.315	-108.4
10:05:27.9	1445+09900002_0	22	0.502	7.0	0.508	-7.9	0.483	112.2	0.499	-108.5
10:05:27.9	1445+09900002_0	23	0.498	7.2	0.509	-8.2	0.489	112.6	0.502	-108.8
10:05:27.9	1445+09900002_0	24	0.496	6.3	0.506	-7.1	0.487	111.9	0.502	-108.3
10:05:27.9	1445+09900002_0	25	0.489	6.3	0.512	-8.2	0.483	113.0	0.498	-108.7
10:09:05.3	N5921_2	22	0.089	53.9	0.084	38.8	0.135	-84.0	0.148	54.9
10:09:05.3	N5921_2	23	0.068	50.4	0.073	31.5	0.117	-80.7	0.150	50.5
10:09:05.3	N5921_2	24	0.068	51.4	0.080	45.1	0.125	-89.0	0.146	47.3
10:09:05.3	N5921_2	25	0.060	45.8	0.060	42.5	0.124	-85.4	0.146	47.8
			Ant = 11				Ant = 12			

Time	Field	Chn	Amp	Phs F	Amp	Phs F	Amp	Phs F	Amp	Phs F
09:21:46.0	1331+30500002_0	22	0.302	-99.8	0.301	-10.5	0.341	169.8	0.350	-137.6
09:21:46.0	1331+30500002_0	23	0.301	-99.9	0.302	-10.6	0.341	169.7	0.349	-138.0
09:21:46.0	1331+30500002_0	24	0.300	-100.0	0.301	-10.9	0.342	169.6	0.348	-138.4
09:21:46.0	1331+30500002_0	25	0.301	-100.1	0.300	-11.0	0.339	169.9	0.347	-138.5
10:05:27.9	1445+09900002_0	22	0.478	-97.3	0.482	-9.7	0.535	171.3	0.544	-138.1
10:05:27.9	1445+09900002_0	23	0.481	-97.4	0.479	-10.4	0.531	171.4	0.549	-138.9
10:05:27.9	1445+09900002_0	24	0.482	-97.6	0.484	-10.1	0.532	172.7	0.544	-139.3
10:05:27.9	1445+09900002_0	25	0.479	-98.4	0.484	-10.1	0.534	172.4	0.553	-139.0
10:09:05.3	N5921_2	22	0.127	44.8	0.142	128.9	0.090	-94.4	0.090	-48.5
10:09:05.3	N5921_2	23	0.135	43.1	0.132	126.0	0.087	-89.3	0.103	-38.2
10:09:05.3	N5921_2	24	0.135	49.4	0.137	136.1	0.092	-95.9	0.084	-42.7
10:09:05.3	N5921_2	25	0.144	49.8	0.119	130.0	0.086	-96.5	0.074	-42.8

Listed 120 antenna solutions.

listhistory-task.html

0.1.65 listhistory

Requires:

Synopsis

List the processing history of a dataset:

Description

List the processing history of a dataset: The list of all task processing steps will be given in the logger.

Arguments

Inputs	
vis	Name of input visibility file (MS)
	allowed: string
	Default:

Returns

void

Example

The list of all task processing steps in a visibility data set are listed in the logger.

Keyword arguments:

```
vis -- Name of input visibility file
default: none; example: vis='ngc5921.ms'
    async -- Run asynchronously
    default = False; do not run asynchronously
```


listfits-task.html

0.1.66 listfits

Requires:

Synopsis

List the HDU and typical data rows of a fits file:

Description

List the HDU and typical data rows of a fits file: The list will be given in the logger.

Arguments

Inputs	
fitsfile	Name of input fits file
	allowed: string
	Default:

Returns

void

Example

The HDU and typical data rows in a fits file are listed in the logger.

Keyword arguments:

fitsfile -- Name of input fits file

default: none; example: fitsfile='ngc5921.uvfits'

async -- Run asynchronously

default = False; do not run asynchronously

listobs-task.html

0.1.67 listobs

Requires:

Synopsis

List the summary of a data set in the logger or in a file

Description

List the summary information of a data set in the logger or in a file, based on a data selection. Only rows can be selected and printed. No in-row selection is possible (channel or correlation).

Lists the following properties of a measurement set: scan list, field list, spectral window list with correlators, antenna locations, ms table information.

Arguments

Inputs	
vis	Name of input visibility file (MS) allowed: string Default:
selectdata	Data selection parameters allowed: bool Default: True
spw	spectral-window/frequency/channel allowed: any Default: variant
field	Field names or field index numbers: "==">all, field='0~2,3C286' allowed: any Default: variant
antenna	antenna/baselines: "==">all, antenna ='3,VA04' allowed: any Default: variant
uvrange	uv range: "==">all; uvrange ='0~100klambda', default units=meters allowed: any Default: variant
timerange	time range: "==">all,timerange='09:14:0~09:54:0' allowed: any Default: variant
correlation	Select data based on correlation allowed: any Default: variant
scan	scan numbers: "==">all allowed: any Default: variant
intent	Select data based on observation intent: "==">all allowed: any Default: variant
feed	multi-feed numbers: Not yet implemented allowed: any Default: variant
array	(sub)array numbers: "==">all allowed: any Default: variant
observation	Select data based on observation ID: "==">all allowed: any Default: variant
verbose	 allowed: bool Default: True

Returns

void

Example

List the summary information of a data set in the logger or in a file, based on a data selection. Only rows can be selected and printed. No in-row selection is possible (channel or correlation). Refer to the task `listvis` to list visibilities.

Lists the following properties of a measurement set:
scan list, field list, spectral window list with
correlators, antenna locations, ms table information.

Keyword arguments:

`vis` -- Name of input visibility file

default: none. example: `vis='ngc5921.ms'`

`selectdata` -- Select a subset of data for flagging

default: False

options: True,False

The summary listing will only apply to the specified selection.

`antenna` -- Select data based on baseline

default: '' (all); example: `antenna='5&6'` baseline 5-6

`antenna='5&6;7&8'` #baseline 5-6 and 7-8

`antenna='5'` # all cross-correlation baselines between antenna 5 and all
antennas

`antenna='5,6'` # all baselines with antennas 5 and 6

`antenna='1&&1'` # only the auto-correlation baselines for antenna 1

`antenna='1&&*''` # cross and auto-correlation baselines between antenna 1
and all other available antennas

`antenna='1~7&&&'` # only the auto-correlation baselines for antennas in 1

`spw` -- Select data based on spectral window and channels

default: '' (all); example: `spw='1'`

`spw='<2'` #spectral windows less than 2

`spw='>1'` #spectral windows greater than 1

`correlation` -- Correlation types

default: '' (all);

example: `correlation='RR LL'`

`field` -- Select data based on field id(s) or name(s)

default: '' (all); example: `field='1'`

```

        field='0~2' # field ids inclusive from 0 to 2
        field='3C*' # all field names starting with 3C
    uvrange -- Select data within uvrange (default units meters)
        default: '' (all); example:
        uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
        uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
        uvrange='0~1000km'; uvrange in kilometers
    timerange -- Select data based on time range:
        default = '' (all); example,
        timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
        Note: YYYY/MM/DD can be dropped as needed:
        timerange='09:14:0~09:54:0' # this time range
        timerange='09:44:00' # data within one integration of time
        timerange='>10:24:00' # data after this time
        timerange='09:44:00+00:13:00' #data 13 minutes after time
    scan -- Select data based on scan number
        default: '' (all); example: scan='>3'
    intent -- Select data based on observation intent
        default: '' (all); example: intent='*CAL*,*BAND*'
    feed -- Selection based on the feed - NOT IMPLEMENTED YET
    array -- Selection based on the antenna array
    observation -- Selection based on the observation ID
        default: '' (all); example: observation='1' or observation=1

verbose -- level of detail
    verbose=True: (default); scan and antenna lists
    verbose=False: less information

listfile -- name of disk file to write output.
    default: None. Example: listfile='list.txt'

listunfl -- List unflagged row counts? If true, it can have significant negative per

cachesize -- maximum size of the memory cache in megabytes in which data structures are
    stored. For very large datasets this can be increased for possibly better
    performance. THIS IS ONLY EXPERIMENTAL FOR NOW, AND INCREASING THE VALUE OF THIS PARAMETER
    SPEED. DEPENDING ON ITS (LACK OF) USEFULNESS, IT MAY BE REMOVED IN THE FUTURE.

```

The 'Int (s)' column is the average of the MS's INTERVAL column for each scan, so in a time-averaged MS 'Int' = 9.83s more likely means 5 10s integrations and 1 9s integration (timebin) than 6 9.83s integrations.

DESCRIPTION OF ALGORITHM TO CALCULATE THE NUMBER OF UNFLAGGED ROWS

The number of unflagged rows are only computed if `listunfl=True`. Computing these quantities can have a negative performance impact, especially for large datasets. The number of unflagged rows (the `nUnflRows` columns in the scans and fields portions of the output) is calculated by summing the fractional unflagged bandwidth for each row (and hence why the number of unflagged rows, in general, is not an integer). Thus a row which has half of its total bandwidth flagged contributes 0.5 rows to the unflagged row count. A row with 20% of its total bandwidth flagged contributes $20/32 = 0.625$ rows to the unflagged row count. A row with all of its bandwidth flagged in the `FLAG_ROW` column is not counted in the number of unflagged rows.

listpartition-task.html

0.1.68 listpartition

Requires:

Synopsis

List the summary of a multi-MS data set in the logger or in a file

Description

Lists the following properties of a multi-measurement set: sub-MS name, scan list, spw list, list of number of channels per spw, number of rows for all scans.

Arguments

Inputs	
vis	Name of multi-MS or normal MS. allowed: string Default:
createdict	Create and return a dictionary with sub-MS information allowed: bool Default: False
listfile	Name of ASCII file to save output: "==">to terminal allowed: string Default:

Returns

void

Example

A multi-measurement set (MMS) is an MS that has been split into sub-MSs. An MMS contains a reference MS in the top directory and the sub-MSs are located in a directory called SUBMSS inside the MMS directory.

Example of a MS that was partitioned in the 'scan' axis using the task partition:

```
> ls ngc5921.mms
ANTENNA          FLAG_CMD      POLARIZATION  SPECTRAL_WINDOW  table.dat
DATA_DESCRIPTION HISTORY        PROCESSOR     STATE             table.info
FEED             OBSERVATION   SORTED_TABLE  SUBMSS            WEATHER
FIELD           POINTING      SOURCE        SYSCAL
```

```
> ls ngc5921.mms/SUBMSS/
ngc5921.0000.ms/  ngc5921.0002.ms/  ngc5921.0004.ms/  ngc5921.0006.ms/
ngc5921.0001.ms/  ngc5921.0003.ms/  ngc5921.0005.ms/
```

The task lists the following properties of a multi-MS or MS:
sub-MS name, scan, spw list, list of number of channels per spw,
number of rows for each scan and the size in disk. Example of logger output:

Sub-MS	Scan	Spw	Nchan	Nrows	Size
ngc5921.0000.ms	1	[0]	[63]	4509	11M
ngc5921.0001.ms	2	[0]	[63]	1890	6.4M
ngc5921.0002.ms	3	[0]	[63]	6048	13M
ngc5921.0003.ms	4	[0]	[63]	756	4.9M
ngc5921.0004.ms	5	[0]	[63]	1134	6.4M
ngc5921.0005.ms	6	[0]	[63]	6804	15M
ngc5921.0006.ms	7	[0]	[63]	1512	6.4M

----- Detailed description of keyword arguments -----

vis -- Name of multi-MS or normal MS.
default: ''.
example: vis='pScan.mms'

createdict -- Create and return a dictionary containing scan summaries of each
sub-MS.
default: False

If set to True, the returned dictionary will contain information from
ms.getscansummary() and ms.getspectralwindowinfo(), with the addition of an
index as the top key and the sub-MS name.
Example:

```
{0: {'MS': 'ngc5921.0000.ms',
      'scanId': {1: {'nchans': array([63], dtype=int32),
                    'nrows': 4509,
                    'spwIds': array([0], dtype=int32)}},
      'size': '11M'},
 1: {'MS': 'ngc5921.0001.ms',
```



```
'scanId': {2: {'nchans': array([63], dtype=int32),  
              'nrows': 1890,  
              'spwIds': array([0], dtype=int32)}}},  
'size': '6.4M'}}
```

```
listfile -- Name of ASCII file to save output to. If empty, it will  
           list on the logger/terminal.  
default: ''  
example: listfile='pscan.txt'
```

listsdm-task.html

0.1.69 listsdm

Requires:

Synopsis

Lists observation information present in an SDM directory.

Description

Given an SDM directory, this task will print observation information to the logger and return a dictionary keyed by scan.

Arguments

Inputs	
sdm	Name of input SDM directory
	allowed: string
	Default:

Returns

void

Example

The listsdm task reads SDM XML tables, processes the observation information contained therein, and prints this information to the CASA log. It will also return a dictionary keyed on scan number. The dictionary contains the following information:

```
'baseband'  list of baseband name(s)
'chanwidth' list of channel widths (Hz)
'end'       observation end time (UTC)
```

'field'	field ID
'intent'	scan intent(s)
'nchan'	list of number of channels
'nsubs'	number of subscans
'reffreq'	list of reference frequencies (Hz)
'source'	source name
'spws'	list of spectral windows
'start'	observation start time (UTC)
'timerange'	start time - end time range (UTC)

Example:

```
myscans = listsdm(sdm='AS1039_sb1382796_2_000.55368.51883247685')
```

Prints information about the requested SDM to the CASA logger and returns a dictionary with scan information in 'myscans'.

Keyword argument:

```
sdm -- Name of input SDM directory.
      example: sdm='AG836_sb1377811_1.55345.300883159725'
```

listvis-task.html

0.1.70 listvis

Requires:

Synopsis

List measurement set visibilities.

Description

This task lists measurement set visibility data under a number of input selection conditions. The measurement set data columns that can be listed are: the raw data, float_data, corrected data, model data, and residual (corrected - model) data.

The output table format is dynamic. Field, Spectral Window, and Channel columns are not displayed if the column contents are uniform. For example, if "spw = '1'" is specified, the spw column will not be displayed. When a column is not displayed, a message is sent to the logger and terminal indicating that the column values are uniform and listing the uniform value.

Table column descriptions:

COLUMN NAME	DESCRIPTION
Date/Time	Date/Time stamp of data sample (YYMMDD/HH:MM:SS UT)
Intrf	Interferometer baseline (antenna names)
UVDist	uv-distance (units of wavelength)
Fld	Field ID (if more than 1)
SpW	Spectral Window ID (if more than 1)
Chn	Channel number (if more than 1)
(Correlated polarization)	(Correlated polarizations (eg: RR, LL, XY)
Sub-columns are:	Amp, Phs, Wt, F
Amp	Visibility amplitude
Phs	Visibility phase (deg)
Wt	Weight of visibility measurement
F	Flag: 'F' = flagged datum; ' ' = unflagged
UVW	UVW coordinates (meters)

Input Parameters: vis Name of input visibility file default: none; example: vis='ngc5921.ms'

options List options: default = 'ap' Not yet implemented for suboptions

datacolumn Visibility file data column: default = 'data': options are data, float_data, corrected, model, residual (corrected-model)

field Select data based on field id(s) or name(s) default: "=="all; example:

field='1' field='0~2' field ids inclusive from 0 to 2 field='3C*' all field names starting with 3C

spw Select spectral window, channel to list default: '0:0' -i spw=0, chan=0

spw='2:34' spectral window 2, channel 34

selectdata Toggle the following 7 selection parameters. default: False; example:

selectdata=True If false, the following parameters are reset to default values.

antenna Select calibration data based on antenna default: "-i all; examples:
 antenna = '5,6'; antenna index 5 and 6 solutions antenna = '05,06'; antenna
 names '05' and '06' solutions
 timerange Select time range to list default: "-i all; examples:
 timerange='10:37:50.1'; list data for this sampling interval
 timerange='j10:37:25'; list data before 10:37:25
 correlation Select polarization correlations to list default: "-i all; examples:
 correlation='RR LL'; list RR and LL correlations correlation='XX XY'; list
 XX and XY correlations
 scan Select scans to list default: "-i all; examples: scan='2'; list scan 2
 scan='j2'; list scan numbers greater than 2
 feed (not yet implemented)
 array (not yet implemented)
 observation Select by observation ID.
 uvrange Select baseline lengths to list. default: "-i all; examples:
 uvrange='j5klambda'; less than 5 kilo-wavelengths Caution: Input units
 default to meters. Listed units are always wavelengths.
 average (not yet implemented)
 showflags (not yet implemented)
 pagerows rows per page of listing default: 50; 0 -i do not paginate
 listfile write output to disk; will not overwrite default: " -i write to screen
 listfile = 'solutions.txt'
 async Run asynchronously default = False; do not run asynchronously

Arguments

Inputs	
vis	Name of input visibility file allowed: string Default:
options	List options: ap only allowed: string Default: ap
datacolumn	Column to list: data, float_data, corrected, model, residual allowed: string Default: data
field	Field names or index to be listed: "==">all allowed: string Default:
spw	Spectral window:channels: "==">all, spw='1:5~57' allowed: string Default: *
selectdata	Other data selection parameters allowed: bool Default: False
antenna	Antenna/baselines: "==">all, antenna = '3' allowed: string Default:
timerange	Time range: "==">all allowed: string Default:
correlation	Correlations: "==">all, correlation = 'RR RL' allowed: string Default:
scan	Scan numbers allowed: string Default:
feed	Multi-feed numbers (Not yet implemented) allowed: string Default:
array	Array numbers (Not yet implemented) allowed: string Default:
observation	Select by observation ID(s) allowed: any Default: variant
uvrange	uv range: "==">all; not yet implemented allowed: string Default:
average	Averaging mode: "==">none (Not yet implemented) allowed: string Default: 333
showflags	Show flagged data (Not yet implemented) allowed: bool Default: False
pagerows	Rows per page allowed: int Default: 50
listfile	Output file allowed: string

Returns

void

Example

This task lists measurement set visibility data under a number of input selection conditions. The measurement set data columns that can be listed are: the raw data, float_data, corrected data, model data, and residual (corrected - model) data.

The output table format is dynamic. Field, Spectral Window, and Channel columns are not displayed if the column contents are uniform. For example, if "spw = '1'" is specified, the spw column will not be displayed. When a column is not displayed, a message is sent to the logger and terminal indicating that the column values are uniform and listing the uniform value.

Table column descriptions:

COLUMN NAME	DESCRIPTION
-----	-----
Date/Time	Time stamp of data sample (YYMMDD/HH:MM:SS UT)
Intrf	Interferometer baseline (antenna names)
UVDist	uv-distance (units of wavelength)
Fld	Field ID (if more than 1)
SpW	Spectral Window ID (if more than 1)
Chn	Channel number (if more than 1)
(Correlated	Correlated polarizations (eg: RR, LL, XY)
polarization)	Sub-columns are: Amp, Phs, Wt, F
Amp	Visibility amplitude
Phs	Visibility phase (deg)
Wt	Weight of visibility measurement
F	Flag: 'F' = flagged datum; ' ' = unflagged
UVW	UVW coordinates (meters)

Input Parameters:

vis Name of input visibility file
 default: none; example: vis='ngc5921.ms'

options List options: default = 'ap'

Not yet implemented for suboptions

datacolumn Visibility file data column:
default = 'data': options are
data, float_data, corrected, model, residual (corrected-model)

field Select data based on field id(s) or name(s)
default: ''==>all; example: field='1'
field='0~2' field ids inclusive from 0 to 2
field='3C*' all field names starting with 3C

spw Select spectral window, channel to list
default: '0:0' --> spw=0, chan=0
spw='2:34' spectral window 2, channel 34

selectdata Toggle the following 7 selection parameters.
default: False; example: selectdata=True
If false, the following parameters are reset
to default values.

antenna Select calibration data based on antenna
default: ''-->all; examples:
antenna = '5,6'; antenna index 5 and 6 solutions
antenna = '05,06'; antenna names '05' and '06' solutions

timerange Select time range to list
default: ''-->all; examples:
timerange='10:37:50.1'; list data for this sampling interval
timerange='<10:37:25'; list data before 10:37:25

correlation Select polarization correlations to list
default: ''-->all; examples:
correlation='RR LL'; list RR and LL correlations
correlation='XX XY'; list XX and XY correlations

scan Select scans to list
default: ''-->all; examples:
scan='2'; list scan 2
scan='>2'; list scan numbers greater than 2

feed (not yet implemented)

array (not yet implemented)

observation Select by observation ID(s).
default: ''-->all;

example: observation='0' (select obsID 0)

uvrange Select baseline lengths to list.
default: ''--> all; examples:
uvrange='<5klambda'; less than 5 kilo-wavelengths
Caution: Input units default to meters.
Listed units are always wavelengths.

average (not yet implemented)

showflags (not yet implemented)

pagerows rows per page of listing
default: 50; 0 --> do not paginate

listfile write output to disk; will not overwrite
default: '' --> write to screen
listfile = 'solutions.txt'

async Run asynchronously
default = False; do not run asynchronously

[makemask-task.html](#)

0.1.71 makemask

Requires:

Synopsis

Makes and manipulates image masks

Description

Construct masks based on various criteria, convert between mask-types, and generate a mask for clean

Arguments

Inputs	
mode	Mask method (list, copy, expand, delete, setdefaultmask) allowed: string Default: list
inpimage	Name of input image. allowed: any Default: variant
inpmask	mask(s) to be processed: image masks, T/F internal masks (Need to include parent image names), regions (for copy mode) allowed: any Default: variant
output	Name of output mask (imagenam or image-name:internal_maskname) allowed: string Default:
overwrite	overwrite output if exists? allowed: bool Default: False
inpfreqs	List of chans/freqs (in inpmask) to read masks from allowed: any Default: variant
outfreqs	List of chans/freqs (in output) on which to expand the mask allowed: any Default: variant

Returns

void

Example

Modes :

list : list internal masks in inpimage to the log

copy : Copy/merge masks and regrid if necessary to a new or existing mask
 expand : Expand a mask from one range of freqs to another range
 delete : delete an internal mask from an image (if the deleted mask was a default mask,
 the task chooses the first one in the remaining internal mask list (as appears
 in the log when do listing with mode='list')
 setdefaultmask : set a specified internal mask as a default internal mask

In all cases (for output mask is expected), if the output image has a different coordinate system
 result of input and processing, the mask will be re-gridded to the output
 coordinate system.

Parameter Descriptions and rules:

 inpimage : Name of input image to use as a reference for the output coordinates (if output coordinates
 Also used as a reference image when regions are specified in inpmask for copy mode.
 If output is a new image specified with an internal T/F mask, the pixel values in
 are copied to the output image and the regions specified in inpmask are merged (if
 specified) and treated as a valid region therefore will be UNMASKED in output.
 default: none (must specify for list, copy, expand modes)

Expandable parameters for mode='copy', 'expand', 'delete' and 'setdefaultmask':

inpmask : Name(s) of input mask(s)
 default: none
 To specify an image (zero/non-zero) mask, just give a image name (e.g. myimage1.im).
 To specify an internal (T/F) mask, you must give a parent image name and the internal mask
 separated by a colon. (e.g. myimage1.im:mask0). The internal mask names can be found in
 the makemask task in mode='list'.

(expand mode)

'myimage:mask0' : use(true/false) internal mask

'myimage' : use the inpimage values to make a mask (zero/non-zero).

Non-zero values are normalized to one in the process.

(copy mode)

Specify the image mask(s), T/F mask(s), and region(s) to be merged in a list of strings.
 The regions can be specified directly in the CASA region format or in the text file format.
 the regions.

(delete and setdefaultmask mode)

Specify the internal mask with the format, image:mask

output : Name of output image.

```

default: none
*The resultant mask is written as an image (zero/one) mask if the output is a plain
*The resultant mask is written as an internal (T/F) mask if the output name is the
The created mask is set as a default internal mask.
*To re-grid a mask to a different coordinate system,
give an image with the target coordinate system in inpimage. Or make a copy an ima
with the target coordinate system and specified the name of the copy in output.

- If output is specified as a plain image, if it exists, it will regrid the mask to
the new coordinate system and modify output (if overwrite=True).
- If output is specified as an image with an internal mask, if the internal mask exists,
it will regrid the mask to the new coordinate system and modify the internal mask onl
- If output does not exist, it will only copy inpimage.
- If output == inpimage, do not regrid. Only modify in-place.

*** Please note that the term 'mask' is used in the image analysis and clean tasks in op
sense. In the image analysis, the masked region in general a region to be excluded v
clean's input mask defines the region to be used as a clean box/region.
In the makemask task, since the most common use case of output image mask is to use
an input mask in clean, when it converts an internal mask to the image mask,
the 'masked' region (where the pixels are masked and have the Boolean values 'False'
of the internal mask is translated to the pixels with value of 0 in output image mas

overwrite : overwrite the mask specified in output? (see also the output rules above)
default: False
* Note that for a cube mask, overwrite=True generally overwrites in the specific
so any pre-existed masks in other channels will be remain untouched.

Additional expandable parameters for mode='expand':
inpfreqs : input channel/frequency/velocity range
Specify channels in a list of integers. for frequency/velocity,
a range is specified in a string with '~', e.g. '1.5MHz~1.6MHz', '-8km/s~-14km/s'
(for the cube with ascending frequencies)
default: [] - all channels
* Note that the range in frequency or velocity needs to be specified as the same
as in the template cube specified in inpimage. E.g., if a template cube has desc
frequencies, then the range will be, for example, '1.6MHz~1.5MHz' or '-14km/s~-8

outfreqs : output channel/frequency/velocity range
Specify same way as inpfreqs
default: [] - all channels

```

Usage examples :

- (1) (list mode):
`makemask(mode='list', inpimage='mymask.im')`
it prints out a list of the internal mask(s) exist in mymask.im to the log
- (2) (copy mode):
Regrid a Boolean mask from one coordinate system to another and save as Boolean mask in the output image.

`makemask(mode='copy', inpimage='oldmask.im', inpmask='oldmask.im:mask0', output='newmask.im')`
- (3) (copy mode):
Same as (1), but save as integer mask in the output image.

`makemask(mode='copy', inpimage='oldmask.im', inpmask='oldmask.im:mask0', output='newmask.im')`

* mask0 is translated so that pixels in oldmask.im that appears as 'masked' in the view has the pixel mask value = 'False' when extracted in imval, are to have pixel value 0 in the output image, newmask.im.
- (4) (copy mode):
Convert a Boolean(true/false) mask to an integer(one/zero) mask in the same image

`makemask(mode='copy', inpimage='oldmask.im', inpmask='oldmask.im:mask0', output='', over=0)`
- (5) (copy mode):
Convert an integer(one/zero) mask to a Boolean(true/false) mask in the same image

`makemask(mode='copy', inpimage='oldmask.im', inpmask='oldmask.im', output='oldmask.im:mask0')`
- (6) (copy mode):
Copy a CRTF mask defined in mybox.txt to a Boolean(true/false) mask in a new image

`makemask(mode='copy', inpimage='image1.im', inpmask='mybox.txt', output='image2.im:mask0')`

The pixel values of image1.im will be copied to image2.im and the region outside mybox.txt will be masked.
- (7) (copy mode):
Apply a region defined in a CRTF file to mask part of an image

`makemask(mode='copy', inpimage='image1.im', inpmask='myregion.crtf', output='image1.im:mask0')`

The region is copied as a T/F mask (mask0) inside the image, image1.im. The region outside the mask will be masked.

(8) (copy mode):

Merge a (one/zero) mask and T/F masks, using the input coordinate-sys of inpimage and saving in a new output file. Remember, if the image specified in output already exist and has a different coordinate system from inpimage, the mask will be regridded to it. All masks to be merged are specified in a list in inpmask. The name of internal masks must be given in the format, 'parent_image_name:internal_mask_name' as shown the example below.

In the example below, image1.im (the 1/0 mask), the internal masks, mask0 from image1.im and mask1 from image2.im, and a region (on image1.im as defined in inpimage) are combined. The output, newmask.im is a new mask name which has not yet exist so image specified in inpimage, image1.im's coordinates are used as a target image coordinates. If image1.im and image2.im has different coordinates, image2.im:mask1 will be regridded before it is combined to the other two masks.

```
makemask(mode='copy',
         inpimage='image1.im',
         inpmask=['image1.im', image1.im:mask0', 'image2.mask:mask1', 'circle[[15pix , 15pix]',
         output='newmask.im);
```

(9) (expand mode):

Expand a (one/zero) mask from continuum imaging to use as an input mask image for spectral line imaging. Use an existing spectral line clean image as a template by specified in inpimage. The inpfreqs is left out as it uses a default (=[], means all channels).

```
makemask(mode='expand', inpimage='spec.clean.image', inpmask='cont.clean.mask',
         outfreqs=[4,5,6,7], output='spec.clean.mask')
```

(10) (expand mode):

Expand a Boolean mask from one range of channels to another range in the same image.

```
makemask(mode='expand', inpimage='oldmask.im', inpmask='oldmask.im:mask0', inpfreqs=[5,6,7],
         output='oldmask.im:mask0', overwrite=True)
```

(11) (expand mode):

Expand a Boolean mask from a range of channels in the input image to another range of channels in a different image with a different spectral-coordinate system. Save the mask as ones/zeros so that it can be used as an input mask in the clean task. As the inpimage is used as a template for the CoordinateSystem of the output cube, it is a prerequisite to have the cube image (a dirty image, etc). In this particular example, it is assumed that bigmask.im is a working copy made from the cube image of a previous

execution. It is used as an input template and the resultant mask is overwritten to the

Specify the infreqs and outfreqs in frequency (assuming here bigmask.im has frequencies
makemask(mode='expand', inpimage='bigmask.im', inpmask='smallmask.im:mask0',
 inpfreqs='1.5MHz~1.6MHz', outfreqs='1.2MHz~1.8MHz', output='bigmask.im', overv

or to specify the ranges in velocities,
makemask(mode='expand', inpimage='bigmask.im', inpmask='smallmask.im:mask0',
 inpfreqs='4.0km/s~0.5km/s', outfreqs='6.5km/s~-2.4km/s', output='bigmask.im', o

(12) (delete mode)

Delete an internal mask from an image.

```
makemask(mode='delete', inpmask='newmask.im:mask0')
```

(13) (setdefaultmask mode)

Set an internal mask as a default internal mask.

```
makemask(mode='setdefaultmask', inpmask='newmask.im:mask1')
```


mosaic-task.html

0.1.72 mosaic

Requires:

Synopsis

Create a multi-field deconvolved image with selected algorithm

Description

Form images from visibilities. Handles continuum and spectral line cubes.

Arguments

Inputs		
vis		name of input visibility file allowed: string Default:
imagename		Pre-name of output images allowed: string Default:
mode		Type of selection (mfs, channel, velocity, frequency) allowed: string Default: mfs
alg		Algorithm to use (clark, hogbom, multiscale) allowed: string Default: clark
imsize		Image size in pixels (nx,ny), symmetric for single value allowed: intArray Default: 256256
cell	arcsec	The image cell size in arcseconds [x,y]. allowed: doubleArrayarcsec Default: 1.01.0
phasecenter		Field Identifier or direction of the image phase center allowed: any Default: variant
stokes		Stokes params to image (I,IV,QU,IQUV,RR,LL,XX,YY,RRL,LL,XXYY) allowed: string Default: I
niter		Maximum number of iterations allowed: int Default: 500
gain		Loop gain for cleaning allowed: double Default: 0.1
threshold		Flux level to stop cleaning (unit mJy assumed) allowed: double Default: 0.0
mask		Set of mask images used in cleaning allowed: stringArray Default:
cleanbox		clean box regions or file name or 'interactive' allowed: any Default: variant
nchan		Number of channels in output image allowed: int Default: 1
start		Start channel allowed: any Default: variant 0
width		Channel width (value > 1 indicates channel averaging) allowed: any Default: variant 1
field		Field Name allowed: any Default: variant

Returns

void

Example

Two types of point-source deconvolution, as well as multi-scale deconvolution, are available. A continuum image (mfs) is produced by gridding together all spectral data. Individual channels or groups of channels can also be images and then placed in an output image cube.

The cleaning regions can be specified by an input mask image, from a file containing rectangular regions, or interactively as the deconvolution progresses.

The mosaic task only uses the "corrected" datacolumn which is made from the "data" data column using applycal with the appropriate calibration tables. Many Stokes combinations are available.

Keyword arguments:

```
vis -- Name of input visibility file
      default: none; example: vis='ngc5921.ms'
imagenam -- Pre-name of output images:
      default: none; example: imagenam='m2'
      output images are:
          m2.image; cleaned and restored image
          m2.flux; relative sky sensitivity over field
          m2.model; image of clean components
          m2.residual; image of residuals
          m2.interactive.mask; image containing clean regions
mode -- Frequency Specification:
NOTE: See examples below:
      default: 'mfs'
          mode = 'mfs' means produce one image from all specified data.
          mode = 'channel'; Use with nchan, start, width to specify
              output image cube. See examples below
          mode = 'velocity', means channels are specified in velocity.
mode = 'frequency', means channels are specified in frequency.
>>> mode expandable parameters (for modes other than 'mfs')
```

Start, width are given in units of channels, frequency or velocity as indicated by mode, but only channel is complete.

nchan -- Number of channels (planes) in output image
 default: 1; example: nchan=3

start -- Start input channel (relative-0)
 default=0; example: start=5

width -- Output channel width (>1 indicates channel averaging)
 default=1; example: width=4

examples:

```

spw = '0,1'; mode = 'mfs'
  will produce one image made from all channels in spw 0 and 1
spw='0:5~28^2'; mode = 'mfs'
  will produce one image made with channels (5,7,9,...,25,27)
spw = '0'; mode = 'channel': nchan=3; start=5; width=4
  will produce an image with 3 output planes
  plane 1 contains data from channels (5+6+7+8)
  plane 2 contains data from channels (9+10+11+12)
  plane 3 contains data from channels (13+14+15+16)
spw = '0:0~63^3'; mode=chann; nchan=21; start = 0; width = 1
  will produce an image with 20 output planes
  Plane 1 contains data from channel 0
  Plane 2 contains date from channel 2
  Plane 21 contains data from channel 61
spw = '0:0~40^2'; mode = 'channel'; nchan = 3; start = 5; width = 4
  will produce an image with three output planes
  plane 1 contains channels (5,7)
  plane 2 contains channels (13,15)
  plane 3 contains channels (21,23)

```

alg -- Algorithm to use (expandable):
 default: 'clark': Options: 'clark','hogbom','multiscale','entropy'
 'hogbom' Cleans from the images only. Only inner quarter
 of image is cleaned
 'clark' Cleans from gridded us data. Only inner quarter of
 image is cleaned
 'multiscale' cleans with several resolutions using hobgom clean
 Currently much slower than single resolution. For extended
 sources, try single resolution with interactive and
 'entropy' Maximum entropy algorithm is still experimental
 and not recommended for general use

```

>>> multiscale expandable parameter
scales -- in pixel numbers; the size of component to deconvolve
default = [0,3,10]
  recommended sizes are 0 (point), 3 (points per clean beam), and
  10 (about a factor of three lower resolution)
negcomponent' -- Stop component search when the largest
scale has found this number of negative components; -1 means

```

```

        continue component search even if the largest component is
        negative.
            default: 2; example: negcomponent=-1
    >>> entropy (MEM) expandable parameters (experimental)
sigma -- Target image sigma
            default: '0.001Jy'; example: sigma='0.1Jy'
targetflux -- Target flux for final image
            default: '1.0Jy'; example: targetflux='200Jy'
constrainflux -- Constrain image to match target flux;
                otherwise, targetflux is used to initialize model only.
                    default: False; example: constrainflux=True
prior -- Name of MEM prior images
            default: ['']; example: prior='source_mem.image'
imsize -- Image pixel size (x,y)
            default = [256,256]; example: imsize=[350,350]
            imsize = 500 is equivalent to [500,500]
cell -- Cell size (x,y)
            default= none;
            example: cell=['0.5arcsec','0.5arcsec'] or
            cell=['1arcmin', '1arcmin']
            cell = '1arcsec' is equivalent to ['1arcsec','1arcsec']
NOTE:cell = '2' makes default cell size of 2 radians!
phasecenter -- direction measure or fieldid for the mosaic center
            default: 0 (imply field=0 as center); example: phasecenter=6
            or phasecenter='J2000 19h30m00 -40d00m00'
stokes -- Stokes parameters to image
            default='I'; example: stokes='IQUV';
            Options: 'I','IV','QU','IQUV','RR','LL','XX','YY','RRLl','XXYY'
niter -- Maximum number iterations, set to zero for no CLEANing
            default: 500; example: niter=500
gain -- Loop gain for CLEANing
            default: 0.1; example: gain=0.5
threshold -- Flux level at which to stop CLEANing (units=mJy)
            default: 0.0; example: threshold=0.0
mask -- Name of mask image used for CLEANing
            default '' means no mask;
            example: mask='orion.mask'.
It is useful to use a mask from a previous interactive mosaic
session for a new execution. The mask image shape
must be the same as the new mosaic.
cleanbox -- Cleaning region:
            default: [] defaults to inner quarter of image
            Three specification types:
            (a) 'interactive' allows the user to build the cleaning
                mask interactively using the viewer. The viewer will
                appear every npercycle iteration, but modify as needed

```

The final interactive mask is saved in the file
 imagename_interactive.mask.

(b) Explicit pixel ranges

example: cleanbox=[110,110,150,145]

clean region with blc=110,100; trc=150,145 (pixel values)

Only one clean region can be given this way.

(c) Filename with pixel values with ascii format:

<fieldindex blc-x blc-y trc-x trc-y> on each line

1 45 66 123 124

2 23 100 300 340

>>> 'interactive' expandable parameter

npercycle -- this is the number of iterations between each clean
 to update mask interactively. Set to about niter/5, can also
 be changed interactively.

field -- Select fields in mosaic. Use field id(s) or field name(s).

['go listobs' to obtain the list id's or names]

default: ''=all fields

If field string is a non-negative integer, it is assumed to

be a field index otherwise, it is assumed to be a field name

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

field='3C286,3C295'; field named 3C286 and 3C295

field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels

NOTE: This selects the data passed as the INPUT to mode

default: ''=all spectral windows and channels

spw='0~2,4'; spectral windows 0,1,2,4 (all channels)

spw='<2'; spectral windows less than 2 (i.e. 0,1)

spw='0:5~61'; spw 0, channels 5 to 61

spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.

spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.

spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60

spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,

spw 1, channels 20-30, and spw 2, channels, 1,2 and 3

timerange -- Time range:

default = '' (all); examples,

selecttime = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'

Note: if YYYY/MM/DD is missing date defaults to first day

in data set

timerange='09:14:0~09:54:0' picks 40 min on first day

timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min on next day

timerange='09:44:00' data within one integration of time

timerange='>10:24:00' data after this time

restfreq -- Specify rest frequency to use for image

default=''

Occasionally it is necessary to set this (for example some VLA

```

spectral line data). For example for
    NH_3 (1,1) put restfreq='23.694496GHz'
sdimage -- Input Single Dish image to use for model
    default='' (no image); example: sdimage='n4826_12mchan.im'
modelimage -- Name of output(/input) model image
    default='' (none=imagenamemodel); modelimage='orion.model'
    Note: This specifies the output model if a single dish
    image is input or the output model name from the imaging
weighting -- Weighting to apply to visibilities:
    default='natural'; example: weighting='uniform';
    Options: 'natural','uniform','briggs','radial','superuniform'
>>> Weighting expandable parameters
    For weighting='briggs'
        rmode -- Robustness mode (see help mosaic)
            default='norm'; example='abs';
            Options: 'norm','abs','none'
        robust -- Brigg's robustness parameter
            default=0.0; example: robust=0.5;
            Options: -2.0 to 2.0; -2 (uniform)/+2 (natural)
        noise -- noise parameter to use for rmode='abs' in
        briggs weighting
            example noise='1.0mJy'
    For superuniform/briggs weighting
        npixels -- number of pixels to determine uv-cell size
        for weight calculation
            example npixels=7
mosweight -- Individually weight the fields of the mosaic
    default: False; example: mosweight=True
    This can be useful if some of your fields are more
    sensitive than others (i.e. due to time spent on-source);
    this parameter will give more weight to higher sensitivity
    fields in the overlap regions.
ftmachine -- Gridding method for the image;
    Options: ft (standard interferometric gridding), sd
    (standard single dish) both (ft and sd as appropriate),
mosaic (gridding use PB as convolution function)
    default: 'mosaic'; example: ftmachine='ft'
cyclefactor -- Change the threshold at which the deconvolution cycle will
    stop, degrid and subtract from the visibilities. For poor PSFs,
    reconcile often (cyclefactor=4 or 5); For good PSFs, use
    cyclefactor 1.5 to 2.0.
    default: 1.5; example: cyclefactor=4
    cycle threshold = cyclefactor * max sidelobe * max residual
cyclespeedup -- Cycle threshold doubles in this number of iterations
    default: -1; example: cyclespeedup=500
scaletype -- Controls scaling of pixels in the image plane.

```

```

default='SAULT'; example: scaletype='PBCOR'
Options: 'PBCOR','SAULT'
'SAULT' scale makes an output image where the noise is constant
across the whole mosaic. However, the image is NOT
corrected for the PB pattern, and therefore is not "flux
correct". Division of the SAULT image_name.image image
by the image_name.flux image will produce a "flux correct image".
The 'PBCOR' option uses the SAULT scaling scheme for
deconvolution, but when interactively cleaning shows the
primary beam corrected image; the final PBCOR image is "flux
correct"
minpb -- Minimum PB level to use
        default=0.1; example: minpb=0.01
async -- Run asynchronously
        default = False; do not run asynchronously

```


msview-task.html

0.1.73 msview

Requires:

Synopsis

View a visibility data set

Description

The msview task will display measurements in raster form. Many display and editing options are available.

Executing the msview task will bring up a display panel window, which can be resized. If no data file was specified, a Load Data window will also appear.

Click on the desired measurement set, and the rendered data should appear on the display panel.

A Data Display Options window will also appear. It has drop-down subsections for related options, most of which are self-explanatory.

The state of the msview task – loaded data and related display options – can be saved in a 'restore' file for later use. You can provide the restore filename on the command line or select it from the Load Data window.

See the cookbook for more details on using the msview task.

Arguments

Inputs	
infile	(Optional) Name of file to visualize. allowed: string Default:
displaytype	(Optional) Type of visual rendering (raster, contour, vector or marker). lel if an lel expression is given for infile (advanced). allowed: string Default: raster
channel	(Optional) access a specific channel in the image cube allowed: int Default: 0
zoom	(Optional) zoom in/out by increments allowed: int Default: 1
outfile	(Optional) name of the output file to generate allowed: string Default:
outscale	(Optional) amount to scale output bitmap formats (non-PS, non-PDF) allowed: double Default: 1.0
outdpi	(Optional) output DPI for PS/PDF allowed: int Default: 300
outformat	(Optional) format of the output e.g. jpg or pdf (this is overridden by the output files extension allowed: string Default: jpg
outlandscape	(Optional) should the output mode be landscape (PS or PDF) allowed: bool Default: False
gui	(Optional) Display the panel in a GUI. allowed: bool Default: True

Returns

void

Example

examples of usage:

```
msview  
msview "mymeasurementset.ms"  
msview "myrestorefile.rstr"
```

Keyword arguments:

infile -- Name of file to visualize

default: ''

example: infile='my.ms'

If no infile is specified the Load Data window will appear for selecting data.

displaytype -- (optional): method of rendering data visually (raster, contour, vector or marker).

You can also set this parameter to 'lcl' and provide an lcl expression for infile (advanced).

default: 'raster'

Note: there is no longer a filetype parameter; typing of data files is now done automatically.

example: msview infile='my.ms'

obsolete: msview infile='my.ms', filetype='ms'

mstransform-task.html

0.1.74 mstransform

Requires:

Synopsis

Split the MS, combine/separate/regrid spws and do channel and time averaging

Description

The task mstransform can do the same functionalities available in cvel, partition, hanningsmooth and split without the need to read and write the output to disk multiple times. The main features of this task are:

- * take an input MS or Multi-MS (MMS)
- * ability to create an output MS or MMS
- * spw combination and separation
- * channel averaging taking flags and weights into account
- * time averaging taking flags and weights into account
- * reference frame transformation
- * Hanning smoothing

All these transformations will be applied on the fly without any writing to disk to optimize I/O. The user can ask to create a Multi-MS in parallel using CASA's cluster infrastructure using the parameter createmms. See MPIInterface for more information on the cluster infrastructure.

This task is implemented in a modular way to preserve the functionalities available in the replaced tasks. One can choose which functionality to apply or apply all of them by setting the corresponding parameters to True. Note that there is an order in which the transformations are applied to the data that makes logical sense on the point of view of the data analysis.

This task can create a multi-MS as the output. General selection parameters are included, and one or all of the various data columns (DATA, LAG_DATA and/or FLOAT_DATA, and possibly MODEL_DATA and/or CORRECTED_DATA) can be selected. It can also be used to create a normal MS, split-based on the given data selection parameters.

The mstransform task creates a Multi-MS in parallel, using the CASA MPI framework. The user should start CASA as follows in order to run it in parallel.

- 1) Start CASA on a single node with 8 engines. The first engine will be used as the MPIClient, where the user will see the CASA prompt. All other engines will be used as MPIServers and will process the data in parallel. mpicasa -n 8 casa -nogui -log2term mstransform(.....)
- 2) Running on a group of nodes in a cluster. mpicasa -hostfile user_hostfile casa mstransform(.....)

where `user_hostfile` contains the names of the nodes and the number of engines to use in each one of them. Example: `pc001234a, slots=5 pc001234b, slots=4`. If CASA is started without `mpicasa`, it is still possible to create an MMS, but the processing will be done in sequential.

The resulting `WEIGHT_SPECTRUM` produced by `mstransform` is in the statistical sense correct for the simple cases of channel average and time average, but not for the general re-gridding case, in which the error propagation formulas applicable for `WEIGHT_SPECTRUM` are yet to be defined. Currently, as in `cvel` and in the imager, `WEIGHT_SPECTRUM` is transformed in the same way as the other data columns. Notice that this is not formally correct from the statistical point of view, but is a good approximation at this stage.

NOTE: the input/output in `mstransform` have a one-to-one relation. input MS – output MS input MMS – output MMS

unless the user sets the parameter `createmms` to `True` to create the following: input MS – output MMS

Arguments

Inputs	
vis	Name of input Measurement set or Multi-MS. allowed: string Default:
outputvis	Name of output Measurement Set or Multi-MS. allowed: string Default:
createmms	Create a multi-MS output from an input MS. allowed: bool Default: False
separationaxis	Axis to do parallelization across(scan,spw,auto,baseline). allowed: string Default: auto
numsubms	The number of Sub-MSs to create (auto or any number) allowed: any Default: variant auto
tileshape	List with 1 or 3 elements giving the tile shape of the disk data columns. allowed: intArray Default: 0
field	Select field using ID(s) or name(s). allowed: any Default: variant
spw	Select spectral window/channels. allowed: any Default: variant
scan	Select data by scan numbers. allowed: any Default: variant
antenna	Select data based on antenna/baseline. allowed: any Default: variant
correlation	Correlation: " ==> all, correlation='XX,YY'. allowed: any Default: variant
timerange	Select data by time range. allowed: any Default: variant
intent	Select data by scan intent. allowed: any Default: variant
array	Select (sub)array(s) by array ID number. allowed: any Default: variant
uvrange	Select data by baseline length. allowed: any Default: variant

Example

Detailed description of keyword arguments:

--- Input/Output parameters ---

vis -- Name of input visibility file
default: ''; example: vis='ngc5921.ms'

outputvis -- Name of output visibility file or Multi-MS
default: ''; example: outputvis='ngc5921.mms'

createmms -- Create an output Multi-MS from an input MS.
default: False

This parameter only has effect if set to True, when it will try to create an output Multi-MS from an input MS. The one-to-one relation of input/output in mstransform is:

input MS -- output MS
input MMS -- output MMS

by setting createmms=True, the following is possible:
input MS -- output MMS

NOTE: See information on processing input Multi-MS at the end of this help section.

separationaxis -- Axis to do parallelization across.
default: 'auto'
options: 'scan', 'spw', 'auto', 'baseline'

* The 'auto' option will partition per scan/spw to obtain optimal load balancing following criteria:

- 1 - Maximize the scan/spw/field distribution across sub-MSs
- 2 - Generate sub-MSs with similar size

* The 'scan' or 'spw' axes will partition the MS into scan or spw. The individuals will not be balanced with respect to the number of rows.

* The 'baseline' axis is mostly useful for Single-Dish data. This axis will partition based on the available baselines. If the user wants only auto-correlations, use antenna selection such as antenna='*&&&' together with this separation axis. If numsubms='auto', partition will try to create as many subMSs as the number of servers in the cluster. If the user wants to have one subMS for each baseline,

parameter to a number higher than the number of baselines to achieve this.

numsubms -- The number of sub-MSs to create.
default: 'auto'
Options: any integer number (example: numsubms=4)

The default 'auto' is to partition using the number of available servers in the system.
If the task is unable to determine the number of running servers, or the user specifies a value less than 8, the task will use 8 as the default.
using mpicasa, numsubms will use 8 as the default.

tileshape -- List with 1 or 3 elements describing the tile shape that will be used
to save the columns to disk. (list)
default: [0]
options: [0] or [1] or [int,int,int]. When list has only one element, it should
be either 0 or 1. When the list has three elements, they should be the
number of correlations, channels, rows.

--- Data selection parameters ---

field -- Select field using field id(s) or field name(s).
[run listobs to obtain the list of d's or names]
default: ''=all fields If field string is a non-negative
integer, it is assumed to be a field index
otherwise, it is assumed to be a field name
field='0~2'; field ids 0,1,2
field='0,4,5~7'; field ids 0,4,5,6,7
field='3C286,3C295'; fields named 3C286 and 3C295
field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels
default: ''=all spectral windows and channels
spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
spw='<2'; spectral windows less than 2 (i.e. 0,1)
spw='0:5~61'; spw 0, channels 5 to 61
spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.
spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
spw = '*:3~64' channels 3 through 64 for all sp id's
spw = ' :3~64' will NOT work.

NOTE: mstransform does not support multiple channel ranges per
spectral window (';').

scan -- Scan number range
default: ''=all

antenna -- Select data based on antenna/baseline


```

default: '' (all)
    Non-negative integers are assumed to be antenna indices, and
    anything else is taken as an antenna name.

examples:
    antenna='5&6': baseline between antenna index 5 and index 6.
    antenna='VA05&VA06': baseline between VLA antenna 5 and 6.
    antenna='5&6;7&8': baselines 5-6 and 7-8
    antenna='5': all baselines with antenna 5
    antenna='5,6,10': all baselines including antennas 5, 6, or 10
    antenna='5,6,10&': all baselines with *only* antennas 5, 6, or
                        10. (cross-correlations only. Use &&
                        to include autocorrelations, and &&&
                        to get only autocorrelations.)
    antenna='!ea03,ea12,ea17': all baselines except those that
                        include EVLA antennas ea03, ea12, or
                        ea17.

correlation -- Correlation types or expression.
    default: '' (all correlations)
    example: correlation='XX,YY'

timerange -- Select data based on time range:
    default: '' (all); examples,
        timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
        Note: if YYYY/MM/DD is missing date, timerange defaults to the
        first day in the dataset
        timerange='09:14:0~09:54:0' picks 40 min on first day
        timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min
        on next day
        timerange='09:44:00' data within one integration of time
        timerange='>10:24:00' data after this time

array -- (Sub)array number range
    default: ''=all

uvrange -- Select data within uvrange (default units meters)
    default: ''=all; example:
        uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
        uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
        uvrange='0~1000km'; uvrange in kilometers

observation -- Select by observation ID(s)
    default: ''=all

feed -- Selection based on the feed - NOT IMPLEMENTED YET

```

default: ''=all

datacolumn -- Which data column to use for processing (case-insensitive).
default: 'corrected'; example: datacolumn='data'
options: 'data', 'model', 'corrected', 'all', 'float_data', 'lag_data',
 'float_data,data', 'lag_data,data'.

NOTE: 'all' = whichever of the above that are present. If the requested
column does not exist, the task will exit with an error.

When datacolumn is set to either one of the values 'model','all',
'data,model,corrected', a sub-parameter realmodelcol will be enabled.
See description below.

realmodelcol -- Make real a virtual MODEL column. If set to True, a real MODEL_DATA
column will be added to the output MS based on the existing SOURCE_M
column.
default: False

keepflags -- Keep completely flagged rows in the output or drop them. This has no
effect on partially flagged rows. All of the channels and correlations
of a row must be flagged for it to be droppable, and a row must be
well defined to be keepable.

IMPORTANT: Regardless of this parameter, flagged data is never included in
channel averaging. On the other hand, partially flagged rows will
always be included in time averaging. The average value of the
flagged data for averages containing ONLY flagged data in the relevant
output channel will be written to the output with the corresponding
flag set to True, while only unflagged data is used on averages where
there is some unflagged data with the flag set to False.

default: True (keep completely flagged rows in the output)

usewtspectrum -- Create a WEIGHT_SPECTRUM column in the output MS. When set to True,
a WEIGHT_SPECTRUM column will be created using the input WEIGHT column,
such that each channel in the WEIGHT_SPECTRUM will get WEIGHT/nChannels.
default: False

--- SPW combination parameters ---

combinespws -- Combine the input spws into a new output spw.
default: False

NOTE: Whenever the data to be combined has different EXPOSURE values in the spectral windows, mstransform will use the WEIGHT_SPECTRUM for the combination. If WEIGHT_SPECTRUM is not available, it will use the values from the WEIGHT column. Each output channel is calculated using the following equation:

$$\text{outputChannel}_j = \frac{\text{SUM}(\text{inputChannel}_i * \text{contributionFraction}_i * \text{inputWeightSpectrum}_i)}{\text{SUM}(\text{contributionFraction}_i * \text{inputWeightSpectrum}_i)}$$

--- Channel averaging parameters ---

chanaverage -- Average data across channels. Partially flagged data is not be included unless all data contributing to a given output channel is flagged. In this case, mstransform calculates the average of all flagged data, and writes it to the output channel with the corresponding flag set to true. If present, WEIGHT_SPECTRUM/SIGMA_SPECTRUM are used together with the channelized flags (FLAG), to compute a weighted average (using WEIGHT_SPECTRUM for CORRECTED_DATA and SIGMA_SPECTRUM for DATA).
default: False

chanbin -- Bin width for channel average in number of input channels.
If a list is given, each bin applies to one of the selected SPWs.

default: 1 => no channel averaging.
options: (int) or [int]
example: chanbin=[2,3] => average 2 channels of 1st selected spectral window and 3 in the second one.

NOTE: WEIGHT_SPECTRUM/SIGMA_SPECTRUM will be used (if present) in addition to the flags to compute a weighted average. The calculations is done as follows:

- 1) When WEIGHT_SPECTRUM/SIGMA_SPECTRUM are not present:
$$\text{Avg} = \text{SUM}(\text{Chan}_i * \text{Flag}_i) / \text{SUM}(\text{Flag}_i)$$
- 2) When WEIGHT_SPECTRUM/SIGMA_SPECTRUM are present:
$$\text{Avg} = \text{SUM}(\text{Chan}_i * \text{Flag}_i * \text{WeightSpectrum}_i) / \text{SUM}(\text{Flag}_i * \text{WeightSpectrum}_i)$$

--- Hanning smoothing parameters ---

hanning -- Hanning smooth frequency channel data to remove Gibbs ringing.
default: False

--- Regrid parameters ---

regridms -- Transform channel labels and visibilities to a different spectral reference

Notice that u,v,w data is not transformed.
default: False

mode -- Regridding mode.
default: 'channel'; produces equidistant grid based on first selected channel.
options: 'velocity', 'frequency', 'channel_b'.

When set to velocity or frequency, it means that the channels must be specified in the respective units. When set to channel_b it means an alternative 'channel' mode that does not force an equidistant grid. It is faster.

nchan -- Number of channels in the output spw (int).
default: -1

start -- First channel to use in the output spw (depends on the mode)
default: 0 --> when mode='channel'

When mode='channel', 'start' means the first channel in the input spw to use when creating the output spw. When mode='frequency', 'start' means the lowest frequency of the output spw. If this information is not available, leave it blank and mstransform will calculate it.

width -- Width of input channels that are used to create an output channel.
default: 1

Note that mstransform will only shift spws with channel widths of the same sign in a single operation. If you are regridding spws with mixed positive and negative channel widths, you should run this task separated for each group of spws. You can verify the channel widths for your MS using listobs for example, and looking at the SPW table, column ChanWid.

nspw -- Number of output spws to create in the output MS/MMS (int).
default: 1 --> it means, do not separate the spws.

One can regrid the MS or not and further separate the output into a given number of spws. Internally, the framework will combine the selected spws before separating them so that channel gaps and overlaps are taken into account. This parameter will create a regular grid of spws in the output MS. If nchan is set, it will refer to the number of output channels in each of the separated spws.

interpolation -- Spectral interpolation method.
default: 'linear'
options: 'nearest', 'cubic', 'spline', 'fftshift'

```

phasecenter -- Direction measure or fieldid. To be used in mosaics to indicate
               the center direction to be used in the spectral coordinate transformation.
               default: '' (first selected field)
               options: FIELD_ID (int) or center coordinate measure (str).
               example: phasecenter=6 or phasecenter='J2000 19h30m00 -40d00m00'

restfreq -- Specify rest frequency to use for output.
            default: ''; occasionally it is necessary to set this.
            example1 for some VLA spectral line data.
            example2 for NH_3 (1,1) put restfreq='23.694496GHz'.

outframe -- Output reference frame (case-insensitive).
            default: ''; it will keep the input reference frame.
            options: 'LSRK', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB', 'GEO', 'TOPO'.

veltype -- Definition of velocity (as used in mode).
            default: 'radio'

--- Time averaging parameters ---
timeaverage -- Average data across time. Partially flagged data is not be included in the
               unless all data contributing to a given output channel is flagged. In this
               mstransform calculates the average of all flagged data, and writes it to
               with the corresponding flag set to true. If keepflags=False, the fully flagged
               is not be written to the output MS. If present, WEIGHT_SPECTRUM/SIGMA_SPECTRUM
               are used together with the channelized flags (FLAG), to compute a weighted
               (using WEIGHT_SPECTRUM for CORRECTED_DATA and SIGMA_SPECTRUM for DATA). (If
               WEIGHT/SIGMA are used instead to average together data from different integrations)

               default: False

timebin -- Bin width for time average in seconds.
            default: '0s'

timespan -- Let the timebin span across scan, state or both.
            State is equivalent to sub-scans. One scan may have several
            state ids. For ALMA MSs, the sub-scans are limited to about
            30s duration each. In these cases, the task will automatically
            add state to the timespan parameter. To see the number of states
            in an MS, use the msmd tool. See help msmd.

            default: '' (separate time bins by both of the above)
            options: 'scan', 'state', 'state,scan'

examples:
timespan = 'scan'; can be useful when the scan number

```

```

        goes up with each integration as in many WSRT MSs.
timespan = ['scan', 'state']: disregard scan and state
        numbers when time averaging.
timespan = 'state,scan'; same as above.

maxuvwdistance -- Provide a maximum separation of start-to-end baselines
        that can be included in an average. (int)
default: 0.0 (given in meters)

--- On-the-fly calibration parameters ---
docallib -- Enable on-the-fly (OTF) calibration as in task applycal
        default: False

callib -- Path to calibration library file, which is a ascii file containing
        the parameters to correct the data as task in task applycal, namely
        gaintable/gainfield/interp/spwmap/calwt. In a Cal Library file, each
        row expresses the calibration apply instructions for a particular
        caltable and (optionally) a specific selection of data in the MS to
        which it is to be applied.

default: '' (there is no default callib file)

examples:

caltable='cal.G' tinterp='linear' calwt=True

-> Arrange a caltable called cal.G to be applied (with no detailed selection)
to all MS data with linear interpolation in time, and with the weights also
calibrated.

caltable='cal.G' tinterp='linear' fldmap='nearest' spwmap=[0,1,1,3] calwt=True
caltable='cal.B' finterp='linear' fldmap='3' spwmap=[0,0,0,0] calwt=False

-> In this case, solutions from cal.G will be selected based on directional
proximity ('nearest') for each MS field via the fldmap parameter, and spw 2
will be calibrated by spw 1 solutions. For cal.B, solutions from field id 3
will be used exclusively, with spw 0 calibrating all MS spws.

----- Multi-MS Processing and Heuristics -----

** Input Multi-MS (MMS) **

```

Task mstransform will process an input MMS in parallel whenever possible. Each sub-MS of the MMS will be processed in a separate engine and the results will be post-processed at the end to create an output MMS. The output MMS will have the same separationaxis of the input MMS, which will be written to the table.info file inside the MMS directory.

Naturally, some transformations available in mstransform require more care when the user first partitions the MS. If one wants to do a combination of spws by setting the parameter combinespws = True in mstransform, the input MMS needs to contain all the selected spws in each of the sub-MSs or the processing will fail. For this, one may set separationaxis to scan or use the default auto with a proper numsubms set so that each sub-MS of the MMS is self-contained with all the necessary spws for the combination.

The task will check if the sub-MSs contain all the selected spws when combinespws=True and if not, it will issue a warning and process the input MMS as a monolithic MS. In this case, the separation axis of the output MMS will be set to scan, regardless of what the original axis was.

A similar case happens when the separation axis of the input MMS is per scan and the user asks to do time averaging with time spanning across scans. If the individual sub-MSs are self-contained of the necessary scans and the duration of the scans is shorter than the timebin, the spanning will not be possible. In this case, the task will process the input as a monolithic MS and will set the axis of the output MMS to spw.

It is important that the user sets the separation axis correctly when first partitioning the MMS. See the table below for when it is possible to process the input MMS in parallel or not. mstransform.

input MMS axis	combinespws=True	nspw > 1	timeaverage=True, timespan='scan'
scan	YES	YES	NO
spw	NO	NO	YES
auto	MAYBE	MAYBE	MAYBE

----- EXAMPLES -----

More documentation on mstransform can be found here:

<http://www.eso.org/~scastro/ALMA/casa/MST/MSTransformDocs/MSTransformDocs.html>

- 1) Split out a single channel.

```
mstransform(vis='ctb80-vsm.ms', outputvis='mychn.ms', datacolumn='data', spw='0:25')
```
- 2) Only combine the selected spws into a single output spw.

```
mstransform(vis='Four_ants.ms', outputvis='myspw.ms', combinespws=True, spw='0~3')
```
- 3) Combine two spws and regrid one field, using two input channels to make one output.

- ```
mstransform(vis='jupiter6cm.demo.ms',outputvis='test1.ms',datacolumn='DATA',field='11',
 spw='0,1', combinespws=True, regridms=True, nchan=1, width=2)
```
- 4) Combine 24 spws and regrid in frequency mode to create 21 output channels. Change the phase center.
 

```
mstransform(vis='g19_d2usb_targets_line.ms', outputvis='test2.ms', datacolumn='DATA',
 combinespws=True, regridms=True, mode='frequency', nchan=21, start='229587.0',
 width='1600kHz', phasecenter="J2000 18h25m56.09 -12d04m28.20")
```
  - 5) Only apply Hanning smoothing to MS.
 

```
mstransform(vis='g19_d2usb_targets_line.ms', outputvis='test3.ms', datacolumn='DATA',
 hanning=True)
```
  - 6) Change the reference frame and apply Hanning smoothing after combining all spws.
 

```
mstransform(vis='g19_d2usb_targets_line.ms', outputvis='test4.ms', datacolumn='DATA',
 combinespws=True, regridms=True, mode="channel", outframe="BARY",
 phasecenter="J2000 18h25m56.09 -12d04m28.20", hanning = True)
```
  - 7) Apply time averaging using a bin of 30 seconds on the default CORRECTED column.
 

```
mstransform(vis='g19_d2usb_targets_line.ms', outputvis='test5.ms', timeaverage=True,
 timebin='30s')
```
  - 8) Apply OTF calibration to ngc5921 using a calibration library
 

```
mstransform(vis='ngc5921.ms', outputvis='ngc5921_calibrated.ms',docallib=True,
 callib='unittest/mstransform/ngc5921_regression/ngc5921_callib.txt')
```



msuvbin-task.html

### **0.1.75 msuvbin**

Requires:

#### **Synopsis**

grid the visibility data onto a defined uniform grid (in the form of an ms);  
multiple MS's can be done onto the same grid

#### **Description**

msuvbin is a uv gridding task. The use is for large volumes of data (from multiple epochs) that needs to be imaged into one image. One way of proceeding is to image the epochs and average them afterwards. Rather than doing this averaging the visibilities on a common uv grid has several convenience advantages like easily doing the proper weighted averaging and imaging. If an output grid already exists and a second ms is gridded on the grid then the output grid parameters is ignored but the existant grid is used.

#### **Arguments**

|             |                                                                                 |
|-------------|---------------------------------------------------------------------------------|
| Inputs      |                                                                                 |
| vis         | Name of input visibility file (MS)<br>allowed: string<br>Default:               |
| field       | Field selection of input ms<br>allowed: string<br>Default:                      |
| spw         | Spw selection<br>allowed: string<br>Default:                                    |
| taql        | TaQl string for data selection<br>allowed: string<br>Default:                   |
| outvis      | name of output uvgrid<br>allowed: string<br>Default:                            |
| phasecenter | phase center of uv grid<br>allowed: string<br>Default:                          |
| nx          | Number of pixels of grid along the x-axis<br>allowed: int<br>Default: 1000      |
| ny          | Number of pixels of grid along the y-axis<br>allowed: int<br>Default: 1000      |
| cell        | pixel cell size defined in sky dimension<br>allowed: string<br>Default: 1arcsec |
| ncorr       | number of correlations to store in grid<br>allowed: int<br>Default: 1           |
| nchan       | Number of spectral channels in grid<br>allowed: int<br>Default: 1               |
| fstart      | Frequency of first spectral channel<br>allowed: string<br>Default: 1GHz         |
| fstep       | spectral channel width<br>allowed: string<br>Default: 1kHz                      |
| wproject    | Do wprojection correction while gridding<br>allowed: bool<br>Default: False     |
| memfrac     | Limit how much of memory to use<br>allowed: double<br>Default: 0.5              |

## Returns

void

## Example

```
Keyword arguments:
vis -- Name of input visibility file
 default: none; example: vis='ngc5921.ms'
field -- Field name list
 default: '' ==> all
 field = '1328+307' specifies source '1328+307'
 field = '4' specified field with index 4
spw -- Spw selection
 default: spw = '' (all spw)
spw='2'
taql --Taql expression for data selection (see http://www.astron.nl/casacore/trunk/)
 default taql=''
Example select all data where U > 1 m in the ms
taql='UVW[0] > 1'
outvis -- name of output grid
 default: '' The user has to give something here
phasecenter -- phasecenter of the grid
 default= ''
 phasecenter='J2000 18h03m04 -20d00m45.1'
nx -- number of pixels along the x axis of the grid
 default: 1000
nx=1200
ny -- number of pixels along the y axis of the grid
 default: 1000
ny=1200
cell -- cellsize of the grid (given in sky units)
 default: '1arcsec'
 cell='0.1arcsec'
ncorr -- number of correlation/polarization plane in uv grid (allowed 1, 2, 4)
 default: 1
 ncorr=4
nchan -- number of spectral channel
 default: 1
```

```
 nchan=2000
fstart -- frequency of the first channel
 default: '1GHz'; User has to give something useful here
fstep -- spectral channel width
 default: '1kHz'
wproject -- do wprojection correction while gridding
 default: False
 wproject=True
memfrac -- control how much of computer's memory is available for gridding
 default=0.5
 memfrac=0.9
```

oldhanningsmooth-task.html

## 0.1.76 oldhanningsmooth

Requires:

### Synopsis

Hanning smooth frequency channel data to remove Gibbs ringing

### Description

**THIS TASK IS DEPRECATED IT WILL BE REMOVED SOON**

This function Hanning smoothes the frequency channels with a weighted running average. The weights are 0.5 for the central channel and 0.25 for each of the two adjacent channels. The first and last channels are flagged. Inclusion of a flagged value in an average causes that data value to be flagged. If an 'outputvis' filename is given, the task will copy the input file to the output file name first, including all columns that are present in the input MS. After that step it will smooth the column(s) as requested in the 'datacolumn' parameter. Alternatively, if no 'outputvis' is specified, oldhanningsmooth will work directly on the input visibility file. If the 'CORRECTED' data column is requested for an MS that does not contain this column, it will be filled from the 'DATA' column and then smoothed.

WARNING: by default, all visibility columns will be smoothed. This will modify the DATA column of the output MS in order to make sure that later gaincal will work on the smoothed data, e.g. as part of self-cal.

### Arguments

| Inputs     |                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------|
| vis        | Name of input visibility file (MS)<br>allowed: string<br>Default:                                  |
| datacolumn | the name of the MS column into which to write the smoothed data<br>allowed: string<br>Default: all |
| outputvis  | name of the output visibility file (MS)<br>allowed: string<br>Default:                             |

## Returns

void

## Example

Keyword arguments:

vis -- Name of input visibility file (MS)

default: none; example: vis='ngc5921.ms'

datacolumn -- the name of the MS column to be Hanning smoothed

default='all'; example: datacolumn='corrected'

options: 'corrected', 'data', 'all'

outputvis -- name of the output visibility file (MS)

default=none (write to the input MS); example: outputvis='ngc5921\_src.ms'

oldhanningsmooth(vis='ngc4852.ms', datacolumn='data', outputvis='ngc4852-hs.ms')

oldsplit-task.html

### **0.1.77 oldsplit**

Requires:

#### **Synopsis**

Create a visibility subset from an existing visibility set

#### **Description**

**THIS TASK IS DEPRECATED IT WILL BE REMOVED SOON**

Oldsplit is the general purpose program to make a new data set that is a subset or averaged form of an existing data set. General selection parameters are included, and one or all of the various data columns (DATA, LAG\_DATA and/or FLOAT\_DATA, and possibly MODEL\_DATA and/or CORRECTED\_DATA) can be selected.

Oldsplit is often used after the initial calibration of the data to make a smaller measurement set with only the data that will be used in further flagging, imaging and/or self-calibration. Oldsplit can average over frequency (channels) and time (integrations).

#### **Arguments**

|             |                                                                                                |
|-------------|------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                |
| vis         | Name of input measurement set<br>allowed: string<br>Default:                                   |
| outputvis   | Name of output measurement set<br>allowed: string<br>Default:                                  |
| datacolumn  | Which data column(s) to Oldsplit out<br>allowed: string<br>Default: corrected                  |
| field       | Select field using ID(s) or name(s)<br>allowed: any<br>Default: variant                        |
| spw         | Select spectral window/channels<br>allowed: any<br>Default: variant                            |
| width       | Number of channels to average to form one output channel<br>allowed: any<br>Default: variant 1 |
| antenna     | Select data based on antenna/baseline<br>allowed: any<br>Default: variant                      |
| timebin     | Bin width for time averaging<br>allowed: string<br>Default: 0s                                 |
| timerange   | Select data by time range<br>allowed: string<br>Default:                                       |
| scan        | Select data by scan numbers<br>allowed: string<br>Default:                                     |
| intent      | Select data by scan intents<br>allowed: string<br>Default:                                     |
| array       | Select (sub)array(s) by array ID number<br>allowed: string<br>Default:                         |
| uvrange     | Select data by baseline length<br>allowed: string<br>Default:                                  |
| correlation | Select correlations<br>allowed: any<br>Default: variant                                        |
| observation | 375<br>Select by observation ID(s)<br>allowed: any<br>Default: variant                         |
| combine     | Let time bins span changes in scan and/or state<br>allowed: any<br>Default: variant            |



## Example

Keyword arguments:

```
vis -- Name of input visibility file
 default: none; example: vis='ngc5921.ms'
outputvis -- Name of output visibility file
 default: none; example: outputvis='ngc5921_src.ms'

datacolumn -- Which data column to split out
 default='corrected'; example: datacolumn='data'
 Options: 'data', 'model', 'corrected', 'all',
 'float_data', 'lag_data', 'float_data,data', and
 'lag_data,data'.
 N.B.: 'all' = whichever of the above that are present.
 Otherwise the selected column will go to DATA (or
 FLOAT_DATA) in the output.
 Splitting with the default datacolumn='corrected'
 before clean is normally required for self-calibration!

--- Data Selection (see help par.selectdata for more detailed
 information)

field -- Select field using field id(s) or field name(s).
 [run listobs to obtain the list id's or names]
 default: ''=all fields If field string is a non-negative
 integer, it is assumed to be a field index
 otherwise, it is assumed to be a field name
 field='0~2'; field ids 0,1,2
 field='0,4,5~7'; field ids 0,4,5,6,7
 field='3C286,3C295'; fields named 3C286 and 3C295
 field = '3,4C*'; field id 3, all names starting with 4C
spw -- Select spectral window/channels
 default: ''=all spectral windows and channels
 spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
 spw='<2'; spectral windows less than 2 (i.e. 0,1)
 spw='0:5~61'; spw 0, channels 5 to 61
 spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.
 spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
 spw = '*:3~64' channels 3 through 64 for all sp id's
 spw = ' :3~64' will NOT work.
 Oldsplit does not support multiple channel ranges per spectral
 window (';') because it is not clear whether to keep the ranges
 in the original spectral window or make a new spectral window
```

```

 for each additional range.
width -- Defines the number of channel to average to form the one
 output channel.
default: '1' => no channel averaging
example: width=[2,3] => average 2 channels of 1st
 spectral window selected and 3 in the second one.
antenna -- Select data based on antenna/baseline
default: '' (all)
 Non-negative integers are assumed to be antenna indices, and
 anything else is taken as an antenna name.

Examples:
antenna='5&6': baseline between antenna index 5 and index 6.
antenna='VA05&VA06': baseline between VLA antenna 5 and 6.
antenna='5&6;7&8': baselines 5-6 and 7-8
antenna='5': all baselines with antenna 5
antenna='5,6,10': all baselines including antennas 5, 6, or 10
antenna='5,6,10&': all baselines with *only* antennas 5, 6, or
 10. (cross-correlations only. Use &&
 to include autocorrelations, and &&&
 to get only autocorrelations.)
antenna='!ea03,ea12,ea17': all baselines except those that
 include EVLA antennas ea03, ea12, or
 ea17.

timebin -- Interval width for time averaging.
 default: '0s' or '-1s' (no averaging)
 example: timebin='30s'
 '10' means '10s'

timerange -- Select data based on time range:
 default = '' (all); examples,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: if YYYY/MM/DD is missing date, timerange defaults to the
 first day in the dataset
 timerange='09:14:0~09:54:0' picks 40 min on first day
 timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min
 on next day
 timerange='09:44:00' data within one integration of time
 timerange='>10:24:00' data after this time

array -- (Sub)array number range
 default: ''=all

uvrange -- Select data within uvrange (default units meters)
 default: ''=all; example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
 uvrange='0~1000km'; uvrange in kilometers

scan -- Scan number range

```

```

 default: ''=all
intent -- Select by scan intent (state). Case sensitive.
 default: '' = all
Examples:
 intent = 'CALIBRATE_ATMOSPHERE_REFERENCE'
 intent = 'calibrate_atmosphere_reference'.upper() # same as above
 # Select states that include one or both of CALIBRATE_WVR.REFERENCE
 # or OBSERVE_TARGET_ON_SOURCE.
 intent = 'CALIBRATE_WVR.REFERENCE, OBSERVE_TARGET_ON_SOURCE'
correlation -- Select correlations, e.g. 'rr, ll' or ['XY', 'YX'].
 default '' (all).
observation -- Select by observation ID(s).
 default: '' = all
combine -- Let time bins span changes in scan and/or state.
 default = '' (separate time bins by both of the above)
 combine = 'scan': Can be useful when the scan number
 goes up with each integration,
 as in many WSRT MSes.
 combine = ['scan', 'state']: disregard scan and state
 numbers when time averaging.
 combine = 'state,scan': Same as above.
keepflags -- If practical, keep completely flagged rows instead of
 dropping them. This has absolutely no effect on averaging
 calculations, or partially flagged rows. All of the
 channels and correlations of a row must be flagged for it
 to be droppable, and a row must be well defined to be
 keepable. The latter condition means that this option has
 no effect on time averaging - in that case fully flagged
 rows are automatically omitted. Regardless of this
 parameter, flagged data is never included in averaging
 calculations.

The only time keepflags matters is if
1. the input MS has some completely flagged rows
and
2. time averaging is not being done.

Then, if keepflags is False, the completely flagged rows
will be omitted from the output MS. Otherwise, they will
be included (subject to the selection parameters).

keepmms -- If true and the input is a multi-MS, make the output one, too.
 Otherwise, the output will be a normal MS without partitioning.
(experimental)
 Default: False

```



plotants-task.html

## 0.1.78 plotants

Requires:

### Synopsis

Plot the antenna distribution in the local reference frame:

### Description

The location of the antennas in the MS will be plotted with X-toward local east; Y-toward local north.

### Arguments

| Inputs  |                                                                     |
|---------|---------------------------------------------------------------------|
| vis     | Name of input visibility file (MS)<br>allowed: string<br>Default:   |
| figfile | Save the plotted figure to this file<br>allowed: string<br>Default: |

### Returns

void

### Example

```
Plot the antenna distribution in the local reference frame:
```

```
The location of the antennas in the MS will be plotted with
X-toward local east; Y-toward local north.
```

```
Keyword arguments:
```

```
vis -- Name of input visibility file.
```

default: none. example: vis='ngc5921.ms'

figfile -- Save the plotted figure in this file.  
default: ''. example: figfile='myFigure.png'

The name of each antenna (egs. vla=antenna number) is  
shown next to its respective location.

DO NOT use the buttons on the Mark Region line. These are  
not implemented yet and might abort CASA.

You can zoom in by pressing the magnifier button (bottom,  
third from left) and making a rectangular region with  
the mouse. Press the home button (left most button) to  
remove zoom.

A hard-copy of this plot can be obtained by pressing the  
button on the right at the bottom of the display. This  
produces a png format file.

plotbandpass-task.html

## **0.1.79 plotbandpass**

Requires:

### **Synopsis**

Makes detailed plots of Tsys and bandpass solutions.

### **Description**

Developed at the NAASC, this is a generic task to display CASA Tsys and bandpass solution tables with options to overlay them in various combinations, and/or with an atmospheric transmission or sky temperature model. It works with both the 'new' (casa 3.4) and 'old' calibration table formats, and allows for mixed mode spws (e.g. TDM and FDM for ALMA). It uses the new msmd tool to access the information about an ms. This task is still being developed as new ALMA observing modes are commissioned. So if you encounter problems, please report them.

### **Arguments**

|             |                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                                                                                         |
| caltable    | Input table name, either a bandpass solution or a Tsys solution<br>allowed: string<br>Default:                                                                          |
| antenna     | A comma-delimited string list of antennas (either names or integer indices) for which to display solutions. Default = all antennas.<br>allowed: any<br>Default: variant |
| field       | A comma-delimited string list of fields (either names or integer indices) for which to display solutions. Default = all fields.<br>allowed: any<br>Default: variant     |
| spw         | A comma-delimited string list of spws for which to display solutions. Default = all spws.<br>allowed: any<br>Default: variant                                           |
| yaxis       | The quantity to plot on the y-axis ("amp", "phase", "both", "tsys", append "db" for dB).<br>allowed: string<br>Default: amp                                             |
| xaxis       | The quantity to plot on the x-axis ("chan" or "freq").<br>allowed: string<br>Default: chan                                                                              |
| figfile     | The name of the plot file to produce.<br>allowed: string<br>Default:                                                                                                    |
| plotrange   | The axes limits to use [x0,x1,y0,y1].<br>allowed: doubleArray<br>Default: 0,0,0,0                                                                                       |
| caltable2   | A second cal table, of type BPOLY or B, to overlay on a B table<br>allowed: string<br>Default:                                                                          |
| overlay     | Show multiple solutions in same frame in different colors (time, antenna, spw, baseband, or time,antenna)<br>allowed: string<br>Default:                                |
| showflagged | Show the values of the solution, even if flagged<br>allowed: bool<br>Default: False                                                                                     |
| timeranges  | Show only these timeranges, the first timerange being 0<br>allowed: string<br>Default:                                                                                  |
| buildpdf    | If True, assemble all the pngs into a pdf<br>allowed: bool<br>Default: False                                                                                            |
| caltable3   | A third cal table, of type BPOLY, to overlay on the first two tables<br>allowed: string<br>Default:                                                                     |



## Returns

variant

## Example

```
plotbandpass('X3c1.tsys',overlay='antenna',yaxis='amp',field='0~1,4',xaxis='chan',figfile='t')
plotbandpass('bandpass.bcal',caltable2='bandpass.bcal_smooth',xaxis='freq')
plotbandpass('bandpass.bcal',caltable2='bandpass.bcal_smooth',xaxis='freq',poln='X',showatm=1)
plotbandpass('bandpass.bcal',channeldiff='5')
```

This task returns void unless the `channeldiff` option is selected, in which case it returns a dictionary containing the statistics of the solutions, keyed by the antenna name, followed by the spw, timerange, polarization, and finally 'amp' and/or 'phase' depending on the yaxis selection.

### Keyword arguments:

`antenna`: must be either an ID (int or string or list), or a single antenna name or list  
`basebands`: show only spws from the specified baseband or list of basebands (default: ''=[])  
`buildpdf`: True/False, if True and `figfile` is set, assemble pngs into a pdf  
`caltable`: a bandpass table, of type B or BPOLY  
`caltable2`: a second cal table, of type BPOLY or B, to overlay on a B table  
`caltable3`: a third cal table, of type BPOLY, to overlay on the first two  
`channeldiff`: set to value > 0 to plot derivatives of amplitude, the value is also used as `chanrange`  
`chanrange`: set xrange (e.g. "5~100") over which to autoscale y-axis for `xaxis='freq'`  
`chanrangeSetXrange`: if True, then `chanrange` also sets the xrange to display  
`convert`: full path for convert command (in case it's not found)  
`density`: dpi to use in creating PNGs and PDFs (default=108)  
`edge`: the number of edge channels to ignore in finding outliers (for `channeldiff>0`)  
`field`: must be an ID, source name, or list thereof; can use trailing \*: 'J\*'  
`figfile`: the base\_name of the png files to save: base\_name.antX.spwY.png  
`figfileSequential`: naming scheme, False: name by spw/antenna (default)  
                  True: figfile.1.png, figfile.2.png, etc.  
`gs`: full path for ghostscript command (in case it's not found)  
`interactive`: if False, then `figfile` will run to completion automatically  
`lo1`: specify the LO1 setting (in GHz) for the observation  
`overlay`: 'antenna', 'time', 'spw', or 'baseband', make 1 plot with different items in colors  
`markersize`: size of points (default=3)

```

ms: name of the ms for this table, in case it does not match the string in the caltable
parentms: name of the parent ms, in case the ms has been previously split
pdftk: full path for pdftk command (in case it's not found)
phase: the y-axis limits to use for phase plots when yaxi='both'
platformingSigma: declare platforming if the amplitude derivative exceeds this many times t
platformingThreshold: if platformingSigma=0, then declare platforming if the amplitude
 derivative exceeds this percentage of the median
plotrange: define axis limits: [x0,x1,y0,y1] where 0,0 means auto
poln: polarizations to plot (e.g. 'XX','YY','RR','LL' or '' for both)
pwv: define the pwv to use for the showatm option: 'auto' or value in mm
resample: channel expansion factor to use when computing MAD of derivative (for channeldiff)
scans: show only solutions for the specified scans (int, list, or string)
showatm: compute and overlay the atmospheric transmission curve (on B or Tsys solutions)
showatmfield: use first observation of this fieldID or name
showatmPoints: draw atmospheric curve with points instead of a line
showBasebandNumber: put the BBC_NO in the title of each plot
showfdm: when showing TDM spws with xaxis='freq', draw locations of FDM spws
showflagged: show the values of data, even if flagged
showimage: also show the atmospheric curve for the image sideband (in black)
showtsky: compute and overlay the sky temperature curve instead of transmission
showlines: draw lines connecting the data (default=T for amp, F for phase)
showpoints: draw points for the data (default=F for amp, T for phase)
solutionTimeThresholdSeconds: consider 2 solutions simultaneous if within this interval (d
spw: must be single ID or list or range (e.g. 0~4, not the original ID)
subplot: 11..81,22,32 or 42 for RowsxColumns (default=22), any 3rd digit is ignored
timeranges: show only these timeranges, the first timerange being 0
xaxis: 'chan' or 'freq'
yaxis: 'amp', 'tsys', 'phase', or 'both' amp+phase == 'ap'. Append 'db' for dB
zoom: 'intersect' will zoom to overlap region of caltable with caltable2

```

plotcal-task.html

### **0.1.80 plotcal**

Requires:

#### **Synopsis**

An all-purpose plotter for calibration results

#### **Description**

An all-purpose plotter for calibration results. The values for all calibration solutions (G, T, GSPLINE, B, BPOLY, D) can be displayed for a variety of polarization combinations and calibrations. The solutions may be iterated through antennas/spw/fields during one execution.

#### **Arguments**

|            |                                                                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                        |
| caltable   | Name of input calibration table<br>allowed: string<br>Default:                                                                                         |
| xaxis      | Value to plot along x axis (time,chan,freq, antenna,antenna1,antenna2,scan, amp,phase,real,imag,snr, tsys,delay,spgain)<br>allowed: string<br>Default: |
| yaxis      | Value to plot along y axis (amp,phase,real,imag,snr, antenna,antenna1,antenna2,scan, tsys,delay,spgain,tec)<br>allowed: string<br>Default:             |
| poln       | Antenna polarization to plot (RL,R,L,XY,X,Y,/)<br>allowed: string<br>Default:                                                                          |
| field      | field names or index of calibrators: "==">all<br>allowed: string<br>Default:                                                                           |
| antenna    | antenna/baselines: "==">all, antenna = '3,VA04'<br>allowed: string<br>Default:                                                                         |
| spw        | spectral window:channels: "==">all, spw='1:5~57'<br>allowed: string<br>Default:                                                                        |
| timerange  | time range: "==">all<br>allowed: string<br>Default:                                                                                                    |
| subplot    | Panel number on display screen (yxn)<br>allowed: int<br>Default: 111                                                                                   |
| overplot   | Overplot solutions on existing display<br>allowed: bool<br>Default: False                                                                              |
| clearpanel | Specify if old plots are cleared or not (ignore)<br>allowed: string<br>Default: Auto                                                                   |
| iteration  | Iterate plots on antenna,time,spw,field<br>allowed: string<br>Default:                                                                                 |
| plotrange  | plot axes ranges: [xmin,xmax,ymin,ymax]<br>allowed: doubleArray<br>Default:                                                                            |
| showflags  | If true, show flagged solutions<br>allowed: bool<br>Default: False                                                                                     |
| plotsymbol | pylab plot symbol<br>allowed: string<br>Default: o                                                                                                     |
| plotcolor  | initial plotting color<br>allowed: string<br>Default: blue                                                                                             |
| markersize | Size of plotted marks<br>allowed: double<br>Default: 5.0                                                                                               |
| fontsize   | Font size for labels                                                                                                                                   |

## Returns

void

## Example

The values for all calibration solutions (G, T, GSPLINE, B, BPOLY, D, M) can be displayed for a variety of polarization combinations and calibrations. The plot solutions may be iterated through antennas/spw/fields during one execution, and many frames can be obtained in each plot.

The plotter permits zooming, listing and flagging of solutions, although the results of flagged solutions are not yet available.

The plotter permits zooming, listing and flagging of solutions, although the implications of flagged solutions are not yet made. See some hints at the end of this description.

Keyword arguments:

caltable -- Name of input calibration table

default: none; example: caltable='ngc5921.gcal'

The type of calibration table is determined automatically.

xaxis -- Value to plot on the x axis

Options: 'time','scan','chan','freq','antenna','amp','phase','real','imag','snr'

Default: cal type dependent, usually 'time'

yaxis -- Value to plot on the y-axis

Options: 'amp','phase','real','imag','snr','antenna','tsys','delay','spgain'

Default: cal type dependent, usually 'amp'

poln -- Polarization (or combination) to plot

default: '' (RL); all polarizations

Options: '' = ('RL'),'R','L','XY','X','Y',

'/' --> form complex poln ratio

(amp ratio and phase difference)

field -- Select field using field id(s) or field name(s).

['go listobs' to obtain the fieldt id's or names]

default: ''=all fields

If field string is a non-negative integer, it is assumed a field index, otherwise it is assumed a field name

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

```

 field='3C286,3C295'; field named 3C286 and 3C295
 field = '3,4C*'; field id 3, all names starting with 4C
antenna -- Antenna selection (baseline syntax ignored)
 default: '' (all);
 example: antenna='1,3~5' means antenna
 indices 1,3,4,5.
spw -- Select spectral window (channel syntax ignored, except for D)
 default: ''=all spectral windows
 spw='0~2,4'; spectral windows 0,1,2,4
 spw='<2'; spectral windows less than 2
timerange -- Time selection
 default: '' (all)
 example: timerange='1995/04/13/09:15:00~1995/04/13/09:25:00'

--- Plot Options ---
subplot -- Panel number on the display screen
 default: 111 (full screen display);
 examples:
 if iteration = 'antenna'; subplot=321 then
 a plot frame will contain the first 6 antennas, in three
 rows and two columns. Follow instructions on screen to
 cycle through the frames
 if iteration = ''; then one frame can be filled with many
 plots in a piecemeal fashion; for example
 antenna='0'; subplot=221; plotcal()
 antenna='1'; subplot=222; plotcal()
 antenna='2'; subplot=223; plotcal()
 antenna='3'; subplot=224; plotcal()
overplot -- Overplot these values on current plot (if possible)
 default: False;
 True (overplotting) can be done ONLY IF iteration=''
clearpanel -- Ignore this parameter.
 Clear nothing on the plot window, automatically
 clear plotting area, clear the current plot area, or
 clear the whole plot panel.
 options: None, Auto, Current, All (None and Auto not supported)
 default: Auto
 example: clearpanel='Current'
iteration -- Create a sequence of plots, iterating over antenna, time,
 field, and/or spw
 default: '' --> create in all in one plot
 example: iteration='antenna' --> create a sequence of
 separate plots separated by antenna. Flagging cannot
 be done in iteration mode.
plotrange -- Control the x and y ranges of the plot, as a list of
 values, e.g., [xmin,xmax,ymin,ymax]

```

```

default=[] --> plot will self-scale
Note: time plotting ranges are cumbersome to use.
 Use the zoom option
showflags -- If true, only flagged solutions will be plotted
 default: false --> only show unflagged solutions
plotsymbol -- pylab plot symbol. See cookbook for details
 default: '.' : large points
 ',' = small points (see markersize)
 '-' = connect points by line
 colors are cycled automatically for multi-function plots
plotcolor -- Initial color to use on each plot
 default: 'blue'
markersize -- Control the size of plot symbols
 default: 5.0 --> a nice size for symbols
fontsize -- Control the font size of title (axes labels will be
 80% of this size)
 default: 10.0
showgui -- Whether or not to display the plotting GUI
 default: True; example showgui=False
figfile -- File name to save the plotted figure to.
 default: ''; example figfile=myPlot.png

Hints on using plotxy (see section 3.4 in cookbook)

Useful Buttons at bottom left:
 5th--magnifying glass. Click on this,
 left mouse button rectangle drag will zoom
 right mouse button rectangle drag will unzoom a certain amount
 1st--restore original magnification

Useful regions just above:
 Quit will terminate plotter
 Next will go to next plot as specified by iteration
 To locate, you must click 'Mark Region' first
 then make appropriate region(s)
 then click locate to list points on logger
 DO NOT USE Flag, Unflag at the present time.

```

plotms-task.html

### **0.1.81 plotms**

Requires:

#### **Synopsis**

A plotter/interactive flagger for visibility data.

#### **Description**

Task for plotting and interacting with visibility data. Limited support for caltable plotting is also included as of CASA v4.1.

A variety of axes choices (including data column) along with MS selection and averaging options are provided for data selection. Flag extension parameters are also available for flagging operations in the plotter.

All of the provided parameters can also be set using the GUI once the application has been launched. Additional and more specific operations are available through the GUI and/or through the plotms tool (pm).

Most basic functions (plotting, iteration, locate, flagging) will work for most CalTables. Parameterized CalTables (delays, antpos, gaincurve, opacity), will, at best, currently just plot the simple parameters contained in the table, not the effective amplitudes or phases sampled at observing times, frequencies etc. BPOLY and GSPLINE tables are not yet supported. Features currently unsupported for CalTables include Averaging, Transformations (velocity conversions, etc.), and some details of selection (channel and polarization selection are not yet enabled) and axes choices (geometry options are not yet enabled). In the plotms gui, many options irrelevant for CalTables are not yet hidden when interacting with a CalTable, and such settings will be ignored (when benign) or cause an error message.

#### **Arguments**



|               |                                                                                                            |
|---------------|------------------------------------------------------------------------------------------------------------|
| Inputs        |                                                                                                            |
| vis           | input MS (or CalTable) (blank for none)<br>allowed: string<br>Default:                                     |
| gridrows      | Number of subplot rows (default 1).<br>allowed: int<br>Default: 1                                          |
| gridcols      | Number of subplot columns (default 1).<br>allowed: int<br>Default: 1                                       |
| rowindex      | Row location of the plot (0-based, default 0)<br>allowed: int<br>Default: 0                                |
| colindex      | Column location of the plot (0-based, default 0)<br>allowed: int<br>Default: 0                             |
| plotindex     | Index to address a subplot (0-based, default 0)<br>allowed: int<br>Default: 0                              |
| xaxis         | plot x-axis (blank for default/current)<br>allowed: string<br>Default:                                     |
| xdatacolumn   | data column to use for x-axis (blank for default/current)<br>allowed: string<br>Default:                   |
| yaxis         | plot y-axis (blank for default/current)<br>allowed: any<br>Default: variant                                |
| ydatacolumn   | data column to use for y-axis (blank for default/current)<br>allowed: any<br>Default: variant              |
| yaxislocation | whether to use a left or right y-axis for the data (blank for default)<br>allowed: any<br>Default: variant |
| selectdata    | data selection parameters<br>allowed: bool<br>Default: True                                                |
| field         | field names or field index numbers (blank for all)<br>allowed: string<br>Default:                          |
| spw           | spectral windows:channels (blank for all)<br>allowed: string<br>Default:                                   |
| timerange     | time range (blank for all)<br>allowed: string<br>Default:                                                  |
| uvrange       | uv range (blank for all)<br>allowed: string<br>Default:                                                    |
| antenna       | antenna/baselines (blank for all)<br>allowed: string<br>Default:                                           |

## Returns

void

## Example

Task for plotting and interacting with visibility data. Limited support for caltable plotting is also included as of CASA v4.1.

A variety of axes choices (including data column) along with MS selection and averaging options are provided for data selection. Flag extension parameters are also available for flagging operations in the plotter.

All of the provided parameters can also be set using the GUI once the application has been launched. Additional and more specific operations are available through the GUI and/or through the plotms tool (pm).

Most basic functions (plotting, iteration, locate, flagging) will work for most CalTables. Parameterized CalTables (delays, antpos, gaincurve, opacity), will, at best, currently just plot the simple parameters contained in the table, not the effective amplitudes or phases sampled at observing times, frequencies etc. BPOLY and GSPLINE tables are not yet supported. Features currently unsupported for CalTables include Averaging, Transformations (velocity conversions, etc.), and some details of selection (channel and polarization selection are not yet enabled) and axes choices (geometry options are not yet enabled). In the plotms gui, many options irrelevant for CalTables are not yet hidden when interacting with a CalTable, and such settings will be ignored (when benign) or cause an error message.

Keyword arguments:

```
vis -- input MS or CalTable
 default: '' (will merely launch the gui)
gridrows -- Number of subplot rows
 default: 1
gridcols -- Number of subplot columns
 default: 1
```

```

rowindex -- Row location of the subplot (0-based).
 default: 0
colindex -- Column location of the subplot (0-based).
 default: 0
plotindex -- Index to address a subplot (0-based).
 default: 0
axis, yaxis -- what to plot on the two axes
 default: '' (defaults are xaxis='time',
 yaxis='amp' on first execution;
 thereafter the most recent
 settings are used)
valid options (=indicates valid synonyms):
MS Ids and other meta info:
'scan' (number)
'field' (index)
'time',
'interval'='timeint'='timeinterval'='time_interval'
'spw' (index)
'chan'='channel' (index)
'freq'='frequency' (GHz)
'vel'='velocity' (km/s)
'corr'='correlation' (index)
'ant1'='antenna1' (index)
'ant2'='antenna2' (index)
'baseline' (a baseline index)
'row' (absoute row Id from the MS)
'observation' (index)
'intent' (index)
'feed1' (index)
'feed2' (index)
Visibility values, flags:
'amp'='amplitude'
'phase' (deg)
'real'
'imag'='imaginary'
'wt'='weight' (unchannelized)
'wtsp'='weightspectrum'
'flag'
'flagrow'
Observational geometry:
'uvdist' (meters)
'uvwave'='uvdistl'='uvdist_l' (wavelengths, per channel)
'u' (meters)
'v' (meters)
'w' (meters)
'uwave' ('u' in wavelengths, per channel)

```

```

'vwave' ('v' in wavelengths, per channel)
'wwave' ('w' in wavelengths, per channel)
'azimuth' (at array reference; degrees)
'elevation' (at array reference; degrees)
'hourang'='hourangle' (at array reference; hours)
'parang'='parangle'='parallacticangle' (at array reference; degrees)
Antenna-based (only works vs. data Ids):
'ant'='antenna'
'ant-azimuth'
'ant-elevation'
'ant-parang'='ant-parangle'
Calibration:
'gainamp'='gamp'
'gainphase'='gphase'
'gainreal'='greal'
'gainimag'='gimag'
'delay'='del'
'opacity'='opac'
'swpower'='swp'='switchedpower'

>>> xaxis, yaxis expandable parameters
 xdatacolumn,
 ydatacolumn -- which data column to use for Visibility values:
 default: '' ('data' on first execution;
 thereafter the most recent
 setting is used)
 valid options: 'data' (observed)
 'corrected'='corr'
 'model'
 'residual' (aliases 'corrected-model')
 'corrected-model'
 'data-model'
 'float'

selectdata -- data selection parameters flag
 default: True (reveals data selection parameters
 described below)
 Consult listobs output for data selection values,
 and see help par.selectdata for more detailed
 information on syntax; also, visit
 http://casa.nrao.edu/other_doc.shtml and click
 on "Measurement Set selection syntax" for more
 tips on using data selection parameters in CASA)

>>> selectdata expandable parameters:

```

```

field -- Select field using field id(s) or field name(s).
 default: ''=all fields
 If field string is a non-negative integer, it is assumed a
 field index, otherwise, it is assumed a field name
 field='0~2'; field ids 0,1,2
 field='0,4,5~7'; field ids 0,4,5,6,7
 field='3C286,3C295'; field named 3C286 and 3C295
 field = '3,4C*'; field id 3, all names starting with 4C
spw -- Select spectral window/channels
 type 'help par.selection' for more examples.
 spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
 spw='<2'; spectral windows less than 2 (i.e. 0,1)
 spw='0:5~61'; spw 0, channels 5 to 61, INCLUSIVE
 spw='*:5~61'; all spw with channels 5 to 61
 spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.
 spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
 spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60
 NOTE ';' to separate channel selections

timerange -- Select data based on time range:
 default = '' (all); examples,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: if YYYY/MM/DD is missing date defaults to first day in data set
 timerange='09:14:0~09:54:0' picks 40 min on first day
 timerange= '25:00:00~27:30:00' picks 1 hr to 3 hr 30min on NEXT day
 timerange='09:44:00' pick data within one integration of time
 timerange='>10:24:00' data after this time
uvrange -- Select data within uvrange (default units meters)
 default: '' (all); example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo lambda

antenna -- Select data based on antenna/baseline
 default: '' (all, including auto-correlations, if present)
 If antenna string is a non-negative integer, it is assumed an
 antenna index, otherwise, it is assumed as an antenna name
 antenna='5&6'; baseline between antenna index 5 and index 6.
 antenna='!ea02'; exclude EVLA antenna 2.
 antenna='ea13;!ea22'; EVLA antenna 13, excluding antenna 22.
 antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
 antenna='5&6;7&8'; baselines with indices 5-6 and 7-8
 antenna='5'; all baselines with antenna index 5
 antenna='05'; all baselines with antenna number 05 (VLA old name)
 antenna='5,6,10'; all baselines with antennas 5,6,10 index numbers
 NB: For explicit selections, use a single ampersand (&) to

```

```

select only cross-correlations among the specified antennas,
double ampersands (&&) to select cross- and
auto-correlations among the specified antennas, and
triple ampersands (&&&) to select only
auto-correlations. E.g.:
antenna='*&'; selects all cross-correlation baseline
(excludes all auto-correlations)
antenna='*&&&'; selects all auto-correlation baselines
(excludes all cross-correlations)
antenna='1&&1,2,3'; selects baselines 1-1 (auto), 1-2,1-3 (cross)
antenna='VA05&&&'; selects the VA05 autocorrelation
See the link noted above for more information.
scan -- Scan numbers or ranges.
 default: '' (all scans)
 scan='1,2,6,43'; scans 1, 2, 6, and 43
 scan='3~14'; scans 3 through 14, inclusive
correlation -- Select by correlation
 default: '' (all correlations)
 options: 'RR','RL','LR','LL','XX','XY','YX','YY',
 or any comma-separated combination; use
 basis (R/L or X/Y) appropriate to the MS)
array -- Select the array id
 default: '' (all array ids)
observation -- Select by observation ID(s).
 default: ''-->all;
 observation='0' (select obsID 0)
intent -- Select observing intent
 default: '' (no selection by intent)
 intent='*BANDPASS*' (selects data labelled with
 BANDPASS intent)
feed -- Select by feed IDs
 default: '' (all feeds)
 feed='1~2'
msselect -- Optional TaQL data selection

averagedata -- data averaging parameters flag
 default: True (reveals expandable parameters
 described below)
>>> averagedata expandable parameters
 avgchannel -- average over channel? either blank for none, or a value
 in channels.
 default: '' (no channel averaging).
 avgtime -- average over time? either blank for none, or a value in
 seconds.
 default: '' (no time averaging).
 avgscan -- average over scans? only valid if time averaging is turned

```

```

 on.
 default: False.
avgfield -- average over fields? only valid if time averaging is
 turned on.
 default: False.
avgbaseline -- average over selected baselines; mutually
 exclusive with avgantenna.
 default: False. (no averaging over baseline)
avgantenna -- form per-antenna averages; mutually exclusive with
 avgbaseline.
 default: False. (no per-antenna averaging)
avgspw -- average over selected spectral windows?
 default: False. (no average of spectral windows)
scalar -- scalar averaging?
 default: False (i.e., do vector averaging)

transform -- apply various transformations on data for plotting
 default: False.
>>> transform expandable parameters
 freqframe -- the coordinate frame in which to render frequency and velocity axes
 default: '' (unspecified: will use frame in which data were taken)
 options: TOPO, GEO, BARY, LSRK, LSRD
 restfreq -- the rest frequency to use in velocity conversions (MHz)
 default: '' (use spw central frequency and show relative velocity)
 example: '22235.08MHz'
 veldef -- the velocity definition to use
 default: 'RADIO'
 options: 'RADIO','OPT','TRUE'
 shift -- adjust phase according to a phase center shift [dx,dy] (arcsec)
 NB: the phase shift in plots is an approximate transformation
 default: [0,0] (no shift)

extendflag -- have flagging extend to other data points?
 default: False.
>>> extendflag expandable parameters
 extcorr -- extend flags based on correlation?
 default: False.
 extchannel -- extend flags based on channel?

iteraxis -- axis upon which iterate plots (one plot per page, for now)
 default: '' (no iteration)
 options: 'scan','field','spw','baseline','antenna','time',''
>>> iteraxis expandable parameters
 xselfscale -- If true, iterated plots should share a common x-axis label per column.
 yselfscale -- If true, iterated plots should share a common y-axis label per row.
 default: false, which will scale all plots globally

```

```

xsharedaxis -- If true, iterated plots should share a common x-axis (must also set x
 default: false, each plot will have its own x-axis.
ysharedaxis -- If true, iterated plots should share a common y-axis (must also set y
 default: false, each plot will have its own y-axis.

customsymbol -- If true, use a custom symbol for drawing unflagged points
 default: False
>>> customsymbol expandable parameters
symbolshape -- If true, use a custom shape to draw unflagged symbols
 default: 'autoscaling' (ignores symbolsize)
 options: 'autoscaling', 'circle', 'square', 'diamond', 'pixel', 'nosymbol'
symbolsize -- size of the unflagged symbols in pixels
 default: 2
symbolcolor -- color to use for unflagged symbols; can be a RGB hex code or a color name
 default: '0000ff'
 example: 'purple'
symbolfill -- type of fill to use for unflagged symbols
 default: 'fill'
 options: 'fill', 'mesh1', 'mesh2', 'mesh3', 'nofill'
symboloutline -- If true, outline unflagged symbols in black

coloraxis -- axis upon which to colorize the plotted points
 options (= indicates synonyms):
 'scan', 'field', 'spw', 'antenna1'='ant1', 'antenna2'='ant2',
 'baseline', 'channel'='chan', 'corr'='correlation', 'time',
 'observation', 'intent'
 default: '' (use a single color for all points)

customflaggedsymbol -- If true, use a custom symbol for drawing flagged points
 default: False
>>> customflaggedsymbol expandable parameters
symbolshape -- If true, use a custom shape to draw flagged symbols
 default: 'nosymbol'
 options: 'autoscaling', 'circle', 'square', 'diamond', 'pixel', 'nosymbol'
symbolsize -- size of the flagged symbols in pixels
 default: 2
symbolcolor -- color to use for flagged symbols; can be a RGB hex code or a color name
 default: '0000ff'
 example: 'purple'
symbolfill -- type of fill to use for flagged symbols
 default: 'fill'
 options: 'fill', 'mesh1', 'mesh2', 'mesh3', 'nofill'
symboloutline -- If true, outline flagged symbols in black

plotrange -- manual plot axis ranges: [xmin,xmax,ymin,ymax]
 Does not affect data selection.

```



```

 default: []; both axes will be autoscaled according
 to the ranges found in the selected data
 If xmin=xmax (or ymin=ymax) then that axis will
 be autoscaled, e.g.:
 [0,0,-2.0,14.0]; autoscale the xaxis, and use
 ymin=-2.0, ymax=14.0

title -- title along top of plot (called "canvas" in some places)

xlabel -- text to label horizontal axis, with formatting using '%%'
ylabel -- text to label horizontal axis, with formatting using '%%'

showmajorgrid -- show major grid lines (horiz and vert.)
 default: False
>>> showmajorgrid expandable parameters
 majorwidth -- line width in pixels of major grid lines
 majorstyle -- major grid line style: solid dash dot none
 majorcolor -- color in hex code of major grid lines

showminorgrid -- show minor grid lines (horiz and vert.)
 default: False
>>> showminorgrid expandable parameters
 minorwidth -- line width in pixels of minor grid lines
 minorstyle -- minor grid line style: solid dash dot none
 minorcolor -- color in hex code of minor grid lines

plotfile -- name of plot file to save automatically
 default: '' (i.e., draw an interactive plot in the gui)
>>> plotfile expandable parameters
 expformat -- export format type; if 'txt' is used an ASCII dump of the plotted points
 default: '' (plotfile extension will be used)
 options: 'jpg', 'png', 'ps', 'pdf', 'txt'
 exprange -- pages to export for iteration plots
 default: ''
 options: 'current', 'all'
 highres -- use high resolution in exported plot
 default: False (use screen resolution)
 dpi -- DPI of exported plot
 default: -1 (not set)
 width -- width of exported plot
 default: -1 (not set)
 height -- height of exported plot
 default: -1 (not set)
 overwrite -- overwrite plot file if it already exists
 default: False

```

`callib -- calibration library string, list of strings, or filename`  
`default: ''`

`showgui - Whether or not to display the plotting GUI`  
`default: True`

plotuv-task.html

## **0.1.82 plotuv**

Requires:

### **Synopsis**

Plot the baseline distribution

### **Description**

Plots the selected baselines of vis one field at a time, in kilowavelengths.

### **Arguments**

|             |                                                                                |
|-------------|--------------------------------------------------------------------------------|
| Inputs      |                                                                                |
| vis         | Name of input visibility file (MS)<br>allowed: string<br>Default:              |
| field       | Select field using ID(s) or name(s)<br>allowed: any<br>Default: variant        |
| antenna     | Select data based on antenna/baseline<br>allowed: any<br>Default: variant      |
| spw         | Select spectral window/channels<br>allowed: any<br>Default: variant            |
| observation | Select by observation ID(s)<br>allowed: any<br>Default: variant                |
| array       | Select (sub)array(s) by array ID number<br>allowed: any<br>Default: variant    |
| maxnpts     | Maximum number of points per plot.<br>allowed: int<br>Default: 100000          |
| colors      | a list of matplotlib color codes<br>allowed: stringArray<br>Default: r y g b   |
| symb        | A matplotlib plot symbol code<br>allowed: string<br>Default: ,                 |
| ncycles     | How many times to cycle through colors per plot.<br>allowed: int<br>Default: 1 |
| figfile     | Save the plotted figure(s) using this name<br>allowed: string<br>Default:      |

## Returns

bool

## Example

Plots the uv coverage of vis in klambda. ncycles of colors will be allocated to representative wavelengths.

Keyword arguments:

vis -- Name of input visibility file  
default: none; example: vis='ngc5921.ms'

--- Data Selection (see help par.selectdata for more detailed information)

field -- Select field using field id(s) or field name(s).  
[run listobs to obtain the list IDs or names]  
default: ''=all fields. If field is a non-negative integer, it is assumed to be a field index. Otherwise, it is assumed to be a field name  
field='0~2'; field ids 0,1,2  
field='0,4,5~7'; field ids 0,4,5,6,7  
field='3C286,3C295'; fields named 3C286 and 3C295  
field = '3,4C\*'; field id 3, all names starting with 4C

antenna -- Select data based on antenna/baseline  
default: '' (all)  
Non-negative integers are assumed to be antenna indices, and anything else is taken as an antenna name.

Examples:

antenna='5&6': baseline between antenna index 5 and index 6.  
antenna='VA05&VA06': baseline between VLA antenna 5 and 6.  
antenna='5&6;7&8': baselines 5-6 and 7-8  
antenna='5': all baselines with antenna 5  
antenna='5,6,10': all baselines including antennas 5, 6, or 10  
antenna='5,6,10&': all baselines with *only* antennas 5, 6, or 10. (cross-correlations only. Use && to include autocorrelations, and &&& to get only autocorrelations.)  
antenna='!ea03,ea12,ea17': all baselines except those that include EVLA antennas ea03, ea12, or ea17.

spw -- Select spectral windows. Channel selection is ignored for now.  
default: ''=all spectral windows  
spw='0~2,4'; spectral windows 0,1,2,4  
spw='<2'; spectral windows less than 2 (i.e. 0,1)  
spw='0'; spw 0

```

 spw='0,10,3'; spws 0, 10, and 3
observation -- Select by observation ID(s). default: '' = all
array -- (Sub)array number range. default: ''=all

maxnpts -- Save memory and/or screen space by plotting a maximum of maxnpts
 (or all of them if maxnpts < 1). There is a very sharp
 slowdown if the plotter starts swapping.
 default: 100000
colors -- a list of matplotlib color codes, used in order of decreasing
 visibility wavelength.
 default: ['r', 'y', 'g', 'b'] (red, yellow, green, blue)
symb -- One of matplotlib's codes for plot symbols: .:,o^v<>s+xDd234hH|_
 default: ',': The smallest points I could find.
ncycles -- The number of times colors will be cycled through per plot.
 default: 1
figfile -- If not '', save the plots using names based on figfile.
 Example: if figfile is 'test.png', and field is '1,2,4', the plots
 will be saved to test_fld1.png, test_fld2.png,
and test_fld4.png.
 default: '' (Do not save)

```

plotweather-task.html

### 0.1.83 plotweather

Requires:

#### Synopsis

Plot elements of the weather table; estimate opacity.

#### Arguments

| Inputs          |                                                                     |
|-----------------|---------------------------------------------------------------------|
| vis             | MS name<br>allowed: string<br>Default:                              |
| seasonal_weight | weight of the seasonal model<br>allowed: double<br>Default: 0.5     |
| doPlot          | set this to True to create a plot<br>allowed: bool<br>Default: True |
| plotName        | (Optional) the name of the plot file<br>allowed: string<br>Default: |

#### Example

Generates opacity estimates from both the weather data and a seasonal model; intended for VL  
By default the returned opacity is the mean of these predictions, but this can be adjusted w

These methods and models are described in detail in EVLA Memo 143, VLA Test Memo 232, VLA S

Saves the plot to the following default file: MS name + .plotweather.png  
Custom plot filenames must end in one of: .png, .pdf, .ps, .eps or .svg

If run as a function, will return the mean zenith opacity per spectral window.

The wind direction is defined as the direction where the wind is coming from.  
The wind direction is thus in the opposite side of the arrow, with north at  
the top and counterclockwise through west, south, and east.

Written by Josh Marvil, revised 02/06/12

example:

```
myTau = plotweather(vis='myMS.ms',seasonal_weight=0.5, doPlot=True)
```



partition-task.html

### 0.1.84 partition

Requires:

#### Synopsis

Task to produce Multi-MSs using parallelism

#### Description

Partition is a task to create a Multi-MS out of an MS. General selection parameters are included, and one or all of the various data columns (DATA, LAG\_DATA and/or FLOAT\_DATA, and possibly MODEL\_DATA and/or CORRECTED\_DATA) can be selected.

The partition task creates a Multi-MS in parallel, using the CASA MPI framework. The user should start CASA as follows in order to run it in parallel.

1) Start CASA on a single node with 8 engines. The first engine will be used as the MPIClient, where the user will see the CASA prompt. All other engines will be used as MPIServers and will process the data in parallel. `mpicasa -n 8 casa -nogui -log2term partition(.....)`

2) Running on a group of nodes in a cluster. `mpicasa -hostfile user_hostfile casa .... partition(.....)`

where `user_hostfile` contains the names of the nodes and the number of engines to use in each one of them. Example: `pc001234a, slots=5 pc001234b, slots=4`  
If CASA is started without `mpicasa`, it is still possible to create an MMS, but the processing will be done in sequential.

A multi-MS is structured to have a reference MS on the top directory and a sub-directory called SUBMSS, which contain each partitioned sub-MS. The reference MS contains links to the sub-tables of the first sub-MS. The other sub-MSs contain a copy of the sub-tables each. A multi-MS looks like this in disk.

```
ls ngc5921.mms ANTENNA FLAG_CMD POLARIZATION
SPECTRAL_WINDOW table.dat DATA_DESCRIPTION HISTORY
PROCESSOR STATE table.info FEED OBSERVATION SORTED_TABLE
SUBMSS WEATHER FIELD POINTING SOURCE SYSCAL
ls ngc5921.mms/SUBMSS/ ngc5921.0000.ms/ ngc5921.0002.ms/
ngc5921.0004.ms/ ngc5921.0006.ms/ ngc5921.0001.ms/ ngc5921.0003.ms/
ngc5921.0005.ms/
```

Inside `casapy`, one can use the task `listpartition` to list the information from a multi-MS.

When partition processes an MMS in parallel, each sub-MS is processed independently in an engine. The log messages of the engines are identified by the string MPIServer-#, where # gives the number of the engine running that process. When the task runs sequentially, it shows the MPIClient text in the origin of the log messages or does not show anything.

## **Arguments**

|                |                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                  |
| vis            | Name of input measurement set<br>allowed: string<br>Default:                                     |
| outputvis      | Name of output measurement set<br>allowed: string<br>Default:                                    |
| createmms      | Should this create a multi-MS output<br>allowed: bool<br>Default: True                           |
| separationaxis | Axis to do parallelization across(scan, spw, baseline, auto)<br>allowed: string<br>Default: auto |
| numsubms       | The number of SubMSs to create (auto or any number)<br>allowed: any<br>Default: variant auto     |
| flagbackup     | Create a backup of the FLAG column in the MMS.<br>allowed: bool<br>Default: True                 |
| datacolumn     | Which data column(s) to process.<br>allowed: string<br>Default: all                              |
| field          | Select field using ID(s) or name(s).<br>allowed: any<br>Default: variant                         |
| spw            | Select spectral window/channels.<br>allowed: any<br>Default: variant                             |
| scan           | Select data by scan numbers.<br>allowed: any<br>Default: variant                                 |
| antenna        | Select data based on antenna/baseline.<br>allowed: any<br>Default: variant                       |
| correlation    | Correlation: " ==> all, correlation='XX,YY'.<br>allowed: any<br>Default: variant                 |
| timerange      | Select data by time range.<br>allowed: any<br>Default: variant                                   |
| intent         | Select data by scan intent.<br>allowed: any<br>Default: variant                                  |
| array          | Select (sub)array(s) by array ID number.<br>allowed: any<br>Default: variant                     |
| uvrange        | Select data by baseline length.                                                                  |

## Example

----- Detailed description of keyword arguments -----

vis -- Name of input visibility file  
default: none; example: vis='ngc5921.ms'

outputvis -- Name of output visibility file  
default: none; example: outputvis='ngc5921.mms'

createmms -- Create a multi-MS as the output.  
default: True  
If False, it will work like the split task and create a normal MS, split according to the given data selection parameters. Note that, when this parameter is set to False, a cluster will not be used.

separationaxis -- Axis to do parallelization across.  
default: 'auto'  
Options: 'scan', 'spw', 'baseline', 'auto'

\* The 'auto' option will partition per scan/spw to obtain optimal load balancing following criteria:

- 1 - Maximize the scan/spw/field distribution across sub-MSs
- 2 - Generate sub-MSs with similar size

\* The 'scan' or 'spw' axes will partition the MS into scan or spw. The individual axes will not be balanced with respect to the number of rows.

\* The 'baseline' axis is mostly useful for Single-Dish data. This axis will partition based on the available baselines. If the user wants only auto-correlations, use antenna selection such as antenna='\*&&&' together with this separation axis. If numsubms='auto', partition will try to create as many subMSs as the number of servers in the cluster. If the user wants to have one subMS for each baseline, set numsubms to a number higher than the number of baselines to achieve this.

numsubms -- The number of sub-MSs to create.  
default: 'auto'  
Options: any integer number (example: numsubms=4)

The default 'auto' is to partition using the number of available servers in the cluster.

If the task is unable to determine the number of running servers, or the user is using mpicasa, numsubms will use 8 as the default.

Example: Launch CASA with 5 engines, where 4 of them will be used to create the MMS and one engine is used as the MPIClient.

```
mpicasa -n 5 casa --nogui --log2term
CASA> partition('uid__A1', outputvis='test.mms')
```

flagbackup -- Make a backup of the FLAG column of the output MMS. When the MMS is created, the .flagversions of the input MS are not transferred, therefore it is necessary to re-create it for the new MMS. Note that multiple backups from the input MS will not be preserved. This will create a single backup of all the flags present in the input MS at the time the MMS is created.

default: True

datacolumn -- Which data column to use when partitioning the MS.

default='all'; example: datacolumn='data'

Options: 'data', 'model', 'corrected', 'all', 'float\_data', 'lag\_data', 'float\_data,data', and 'lag\_data,data'.

N.B.: 'all' = whichever of the above that are present.

---- Data selection parameters (see help par.selectdata for more detailed information)

field -- Select field using field id(s) or field name(s).

[run listobs to obtain the list of field ids or names]

default: ''=all fields If field string is a non-negative integer, it is assumed to be a field index otherwise, it is assumed to be a field name

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

field='3C286,3C295'; fields named 3C286 and 3C295

field = '3,4C\*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels

default: ''=all spectral windows and channels

spw='0~2,4'; spectral windows 0,1,2,4 (all channels)

spw='<2'; spectral windows less than 2 (i.e. 0,1)

spw='0:5~61'; spw 0, channels 5 to 61

spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.

spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.

spw = '\*:3~64' channels 3 through 64 for all sp id's

spw = ':3~64' will NOT work.

```

spw = '*:0;60~63' channel 0 and channels 60 to 63 for all IFs
 ',' needed to separate different channel ranges in one spw
spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60
spw='0:0~10,1:20~30,2:1;2;4'; spw 0, channels 0-10,
 spw 1, channels 20-30, and spw 2, channels, 1, 2 and 4

antenna -- Select data based on antenna/baseline
default: '' (all)
 Non-negative integers are assumed to be antenna indices, and
 anything else is taken as an antenna name.

Examples:
antenna='5&6': baseline between antenna index 5 and index 6.
antenna='VA05&VA06': baseline between VLA antenna 5 and 6.
antenna='5&6;7&8': baselines 5-6 and 7-8
antenna='5': all baselines with antenna 5
antenna='5,6,10': all baselines including antennas 5, 6, or 10
antenna='5,6,10&': all baselines with *only* antennas 5, 6, or
 10. (cross-correlations only. Use &&
 to include autocorrelations, and &&&
 to get only autocorrelations.)
antenna='!ea03,ea12,ea17': all baselines except those that
 include EVLA antennas ea03, ea12, or
 ea17.

timerange -- Select data based on time range:
default = '' (all); examples,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: if YYYY/MM/DD is missing date, timerange defaults to the
 first day in the dataset
 timerange='09:14:0~09:54:0' picks 40 min on first day
 timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min
 on next day
 timerange='09:44:00' data within one integration of time
 timerange='>10:24:00' data after this time

array -- (Sub)array number range
default: ''=all

uvrange -- Select data within uvrange (default units meters)
default: ''=all; example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
 uvrange='0~1000km'; uvrange in kilometers

scan -- Scan number range

```

```

 default: ''=all

observation -- Select by observation ID(s)
 default: ''=all

----- EXAMPLES -----

1) Create a Multi-MS of some spws, partitioned per spw. The MS contains 16 spws.
 partition('uid001.ms', outputvis='source.mms', spw='1,3~10', separationaxis='spw')

2) Create a Multi-MS but select only the first channels of all spws. Do not back up the FLAG
column.
 partition('uid0001.ms', outputvis='fechans.mms', spw='*:1~10', flagbackup=False)

3) Create a Multi-MS using both separation axes.
 partition('uid0001.ms', outputvis='myuid.mms', createmms=True, separationaxis='auto')

4) Create a single-dish Multi-MS using the baseline axis only for the auto-correlations.
 partition('uid0001.ms', outputvis='myuid.mms', createmms=True, separationaxis='baseline')

```

polcal-task.html

## **0.1.85 polcal**

Requires:

### **Synopsis**

Determine instrumental polarization calibrations

### **Description**

The complex instrumental polarization factors (D-terms) for each antenna/spwid are determined from the data for the specified calibrator sources. Previous calibrations can be applied on the fly.

### **Arguments**



|             |                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                             |
| vis         | Name of input visibility file<br>allowed: string<br>Default:                                                |
| caltable    | Name of output gain calibration table<br>allowed: string<br>Default:                                        |
| field       | Select field using field id(s) or field name(s)<br>allowed: string<br>Default:                              |
| spw         | Select spectral window/channels<br>allowed: string<br>Default:                                              |
| intent      | Select observing intent<br>allowed: string<br>Default:                                                      |
| selectdata  | Other data selection parameters<br>allowed: bool<br>Default: True                                           |
| timerange   | Select data based on time range<br>allowed: string<br>Default:                                              |
| uvrange     | Select data within uvrange (default units meters)<br>allowed: any<br>Default: variant                       |
| antenna     | Select data based on antenna/baseline<br>allowed: string<br>Default:                                        |
| scan        | Scan number range<br>allowed: string<br>Default:                                                            |
| observation | Select by observation ID(s)<br>allowed: any<br>Default: variant                                             |
| msselect    | Optional complex data selection (ignore for now)<br>allowed: string<br>Default:                             |
| solint      | Solution interval<br>allowed: any<br>Default: variant inf                                                   |
| combine     | Data axes which to combine for solve (obs, scan, spw, and/or field)<br>allowed: string<br>Default: obs,scan |
| preavg      | Pre-averaging interval (sec)<br>allowed: double<br>Default: 300.0                                           |
| refant      | Reference antenna name(s)<br>allowed: string<br>Default:                                                    |
| minblperant | Minimum baselines _per antenna_ required for solve<br>allowed: int<br>Default: 4                            |
| minsnr      | Reject solutions below this SNR                                                                             |

## Example

The instrumental polarization factors (D-terms), the calibrator polarization, and the R-L polarization angle can be determined using polcal. The solutions can be obtained for each antenna/spwid and even individual channels, if desired. Previous calibrations of the total intensity data should be applied on the fly when running polcal, since polcal uses the 'data' column, not the 'corrected' column.

After calibrating the gain, bandpass, and (if relevant, for channelized data) cross-hand delay, the simplest way to calibrate the polarization data is:

- a) Run polcal with poltype = 'D+QU' on the main 'calibrator' source. The D terms and polarization (QU) of the calibrator will be determined. Relatively good parallactic angle coverage is needed.
- b) If there is little parallactic angle coverage, place the known polarization of the main calibrator (or 0) using setjy with the appropriate fluxdensity. Then run polcal with poltype = 'D'. Run plotcal with xaxis = 'real'; yaxis = 'imag' to view solutions. It is best to use an unpolarized calibrator in this instance; large systematic offsets from zero indicate significant source polarization that will bias the polarization calibration. A mechanism to constrain this bias will be made available in the near future.
- c) To determine R-L polarization angle, use setjy to put the fluxdensity of the polarization calibrator [I,Q,U,0.0] in the model column. For resolved sources put in values associated with an appropriate u-v range. Polarized models are not yet available for the major polarization standard sources, so very resolved polarized sources should not be used.
- d) Run polcal with poltype = 'X' and include polarization standard. Make sure to include all previous calibrations, especially the D results. Run plotxy with correlation = 'RL LR' and make sure polarization angles are as expected.
- e) Run applycal with all calibration table, include the D and X tables. Make sure that parang = T

NOTE: For very high dynamic range, use poltype='Df' or 'Df+QU' to determine D terms for each channel. Similarly, poltype='Xf' can be used to determine a channel-dependent R-L phase "bandpass".

NOTE: Rather than use setjy in b and c above, the new smodel parameter may be used in polcal to specify a simple

point source Stokes model.

Keyword arguments:

vis -- Name of input visibility file  
      default: none; example: vis='ngc5921.ms'  
caltable -- Name of output gain calibration table  
          default: none; example: caltable='ngc5921.dcal'

--- Data Selection (see help par.selectdata for more detailed information)

field -- Select field using field id(s) or field name(s).  
      [run listobs to obtain the list id's or names]  
      default: ''=all fields.  
      Most likely, the main calibrator source should be picked.  
      If field string is a non-negative integer, it is assumed a field index  
      otherwise, it is assumed a field name  
      field='0~2'; field ids 0,1,2  
      field='0,4,5~7'; field ids 0,4,5,6,7  
      field='3C286,3C295'; field named 3C286 and 3C295  
      field = '3,4C\*'; field id 3, all names starting with 4C  
spw -- Select spectral window/channels  
      type 'help par.selection' for more examples.  
      spw='0~2,4'; spectral windows 0,1,2,4 (all channels)  
      spw='<2'; spectral windows less than 2 (i.e. 0,1)  
      spw='0:5~61'; spw 0, channels 5 to 61, INCLUSIVE  
      spw='\*:5~61'; all spw with channels 5 to 62  
      spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.  
      spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.  
      spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60  
      NOTE ';' to separate channel selections  
      spw='0:0~10~2,1:20~30~5'; spw 0, channels 0,2,4,6,8,10,  
      spw 1, channels 20,25,30  
intent -- Select observing intent  
      default: '' (no selection by intent)  
      intent='\*BANDPASS\*' (selects data labelled with  
                          BANDPASS intent)  
selectdata -- Other data selection parameters  
      default: True  
timerange -- Select data based on time range:  
      default = '' (all); examples,  
      timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'  
      Note: if YYYY/MM/DD is missing defaults to first day in data set  
      timerange='09:14:0~09:54:0' picks 40 min on first day  
      timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min on next day  
      timerange='09:44:00' data within one integration of time  
      timerange='>10:24:00' data after this time

```

uvrange -- Select data within uvrange (default units meters)
default: '' (all); example:
uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
antenna -- Select data based on antenna/baseline
default: '' (all)
If antenna string is a non-negative integer, it is assumed an antenna index
otherwise, it is assumed as an antenna name
antenna='5&6'; baseline between antenna index 5 and index 6.
antenna='VA05&VA06'; baseline between VLA antenna 5 and 6.
antenna='5&6;7&8'; baseline 5-6 and 7-8
antenna='5'; all baselines with antenna index 5
antenna='05'; all baselines with antenna name 05, i.e. VLA ant 5
antenna='5,6,10'; all baselines with antennas 5, 6 and 10
scan -- Scan number range
observation -- Observation ID(s).
default: '' = all
example: '0~2,4'
msselect -- Optional complex data selection (ignore for now)

--- Solution parameters
poltype -- Type of instrumental polarization solution
'D+QU' (or 'Df+QU') solve also for apparent source polarization (channelized)
Need relatively good parallactic angle coverage for this
'D' (or 'Df') solve only for instrumental polarization (channelized). The
I, Q, U flux density of the source can be placed in the model column using
setjy. Use for poor parallactic angle coverage.
'X' (or 'Xf') = solve only for position angle correction (channelized).
The source must have its I, Q, U flux density in the model column
or specified in smodel. If the source is resolved, use a limited
uvrange that is appropriate.
'D+X' (or 'Df+X') = solve also for position angle offset (channelized D) as
well as the D-term. Not normally done.
default: 'D+QU'
The solution used the traditional linear approximation. Non-linearized option
will be available soon.
smodel -- Point source Stokes parameters for source model (experimental)
default: [] (use MODEL_DATA column)
examples: [1,0,0,0] (I=1, unpolarized)
[5.2,0.2,0.3,0.0] (I=5.2, Q=0.2, U=0.3, V=0.0)
solint -- Solution interval (units optional)
default: 'inf' (~infinite, up to boundaries controlled by combine);
Options: 'inf' (~infinite), 'int' (per integration), any float
or integer value with or without units
examples: solint='1min'; solint='60s', solint=60 --> 1 minute
solint='0s'; solint=0; solint='int' --> per integration

```

```

 solint='-1s'; solint='inf' --> ~infinite, up to boundaries
 enforced by combine
combine -- Data axes to combine for solving
 default: 'obs,scan' --> solutions will break at field and spw
 boundaries but may extend over multiple obs and scans
 (per field and spw) up to solint.
 Options: '', 'obs', 'scan', 'spw', 'field', or any comma-separated
 combination in a single string
 example: combine='scan,spw' --> extend solutions over scan boundaries
 (up to the solint), and combine spws for solving
preavg -- Pre-averaging interval (sec)
 default=300
 Interval to apply parallactic angle.
refant -- Reference antenna name
 default: '' => refant = '0'
 example: refant='13' (antenna with index 13)
 refant='VA04' (VLA antenna #4)
 refant='EA02,EA23,EA13' (EVLA antenna EA02, use
 EA23 and EA13 as alternates if/when EA02
 drops out)
 Use 'go listobs' for antenna listing.
 USE SAME REFERENCE ANTENNA AS USED FOR I CALIBRATION.
minblperant -- Minimum number of baselines required per antenna for each solve
 Antennas with fewer baselines are excluded from solutions. Amplitude
 solutions with fewer than 4 baselines, and phase solutions with fewer
 than 3 baselines are only trivially constrained, and are no better
 than baseline-based solutions.
 default: 4
 example: minblperant=10 => Antennas participating on 10 or more
 baselines are included in the solve
minsnr -- Reject solutions below this SNR
 default: 3.0
append -- Append solutions to the (existing) table. Appended solutions
 must be derived from the same MS as the existing
 caltable, and solution spws must have the same
 meta-info (according to spw selection and solint)
 or be non-overlapping.
 default: False; overwrite existing table or make new table

--- Other calibrations to apply on the fly before determining polcal solution

docallib -- Control means of specifying the caltables:
 default: False ==> Use gaintable, gainfield, interp, spwmap, calwt
 If True, specify a file containing cal library in callib
callib -- If docallib=True, specify a file containing cal
 library directives

```

```

gaintable -- Gain calibration table(s) to apply
default: '' (none); BUT I CALIBRATION TABLES SHOULD GENERALLY BE INCLUDED
examples: gaintable='ngc5921.gcal'
 gaintable=['ngc5921.ampcal','ngc5921.phcal']
gainfield -- Select a subset of calibrators from gaintable(s)
default: '' ==> all sources in table;
'nearest' ==> nearest (on sky) available field in table
otherwise, same syntax as field
example: gainfield='0~3'
 gainfield=['0~3','4~6'] means use field 0 through 3
 from first gain file, field 4 through 6 for second.
interp -- Interpolation type (in time[,freq]) to use for each gaintable.
When frequency interpolation is relevant (B, Df, Xf),
separate time-dependent and freq-dependent interp
types with a comma (freq_after_ the comma).
Specifications for frequency are ignored when the
calibration table has no channel-dependence.
Time-dependent interp options ending in 'PD' enable a
"phase delay" correction per spw for non-channel-dependent
calibration types.
For multi-obsId datasets, 'perobs' can be appended to
the time-dependent interpolation specification to
enforce obsId boundaries when interpolating in time.
default: '' --> 'linear,linear' for all gaintable(s)
example: interp='nearest' (in time, freq-dep will be
 linear, if relevant)
 interp='linear,cubic' (linear in time, cubic
 in freq)
 interp='linearperobs,spline' (linear in time
 per obsId,
 spline in freq)
 interp=',spline' (spline in freq; linear in
 time by default)
 interp=['nearest,spline','linear'] (for multiple gaintables)
Options: Time: 'nearest', 'linear'
 Freq: 'nearest', 'linear', 'cubic', 'spline'
spwmap -- Spectral windows combinations to form for gaintable(s)
default: [] (apply solutions from each spw to that spw only)
Example: spwmap=[0,0,1,1] means apply the caltable solutions
 from spw = 0 to the spw 0,1 and spw 1 to spw 2,3.
 spwmap=[[0,0,1,1],[0,1,0,1]]
async -- Run asynchronously
default = False; do not run asynchronously

```

[predictcomp-task.html](#)

## **0.1.86 predictcomp**

Requires:

### **Synopsis**

Make a component list for a known calibrator

### **Description**

Writes a component list named `clist` to disk and returns a dict of `'clist': clist`, `'objname': objname`, `'standard': standard`, `'epoch': epoch`, `'freqs': pl.array` of frequencies, in GHz, `'antennalist': a simdata type configuration file`, `'amps': pl.array` of predicted visibility amplitudes, in Jy, `'savedfig': False` or, if made, the filename of a plot. or False on error.

### **Arguments**

|             |                                                                               |
|-------------|-------------------------------------------------------------------------------|
| Inputs      |                                                                               |
| objname     | Object name<br>allowed: string<br>Default:                                    |
| standard    | Flux density standard<br>allowed: string<br>Default: Butler-JPL-Horizons 2010 |
| epoch       | Epoch<br>allowed: string<br>Default:                                          |
| minfreq     | Minimum frequency<br>allowed: string<br>Default:                              |
| maxfreq     | Maximum frequency<br>allowed: string<br>Default:                              |
| nfreqs      | Number of frequencies<br>allowed: int<br>Default: 2                           |
| prefix      | Prefix for the component list directory name.<br>allowed: string<br>Default:  |
| antennalist | Plot for this configuration<br>allowed: string<br>Default:                    |
| showplot    | Plot S vs —u— to the screen?<br>allowed: bool<br>Default: False               |
| savefig     | Save a plot of S vs —u— to this filename<br>allowed: string<br>Default:       |
| symb        | A matplotlib plot symbol code<br>allowed: string<br>Default: .                |
| include0amp | Force the amplitude axis to start at 0?<br>allowed: bool<br>Default: False    |
| include0bl  | Force the baseline axis to start at 0?<br>allowed: bool<br>Default: False     |
| blunit      | unit of the baseline axis<br>allowed: string<br>Default:                      |
| showbl0flux | Print the zero baseline flux ?<br>allowed: bool<br>Default: False             |



## Returns

record

## Example

Writes a component list to disk and returns a dict of  
{'clist': filename of the component list,  
  'objname': objname,  
  'angdiam': angular diameter in radians (if used in clist),  
  'standard': standard,  
  'epoch': epoch,  
  'freqs': pl.array of frequencies, in GHz,  
  'antennalist': pl.array of baseline lengths, in m,  
  'amps': pl.array of predicted visibility amplitudes, in Jy,  
  'savedfig': False or, if made, the filename of a plot.}  
or False on error.

objname: An object supported by standard.

standard: A standard for calculating flux densities, as in setjy.

Default: 'Butler-JPL-Horizons 2010'

epoch: The epoch to use for the calculations. Irrelevant for  
extrasolar standards. (Uses UTC)

Examples: '2011-12-31/5:34:12', '2011-12-31-5:34:12'

minfreq: The minimum frequency to use.

Example: '342.0GHz'

maxfreq: The maximum frequency to use.

Default: minfreq

Example: '346.0GHz'

Example: '', anything <= 0, or None: use minfreq.

nfreqs: The number of frequencies to use.

Default: 1 if minfreq == maxfreq,  
2 otherwise.

prefix: The component list will be saved to

prefix + 'spw0\_<objname>\_<minfreq><epoch>.cl'

Default: ''

Example: "Bands3to7\_"

(which could produce 'Bands3to7\_spw0\_Uranus\_100GHz55877d.cl',

depending on the other parameters)

antennalist: 'Observe' and plot the visibility amplitudes for this  
antenna configuration. The file should be in a format usable  
by simdata. The search path is:

```

 .:casa['dirs']['data'] + '/alma/simmos/'
 Default: '' (None, just make clist.)
 Example: 'alma.cycle0.extended.cfg'

Subparameters of antennalist:
showplot: Whether or not to show a plot of S vs. |u| on screen.
 Subparameter of antennalist.
 Default: Necessarily False if antennalist is not specified.
 True otherwise.
savefig: Filename for saving a plot of S vs. |u|.
 Subparameter of antennalist.
 Default: False (necessarily if antennalist is not specified)
 Examples: '' (do not save the plot)
 'myplot.png' (save to myplot.png)
symb: One of matplotlib's codes for plot symbols: .:,o^v<>s+xDd234hH|_
 Default: '.'
include0amp: Force the amplitude axis to start at 0?
 Default: False
include0bl: Force the baseline axis to start at 0?
 Default: False
blunit: unit of the baseline axis ('' or 'klambda')
 Default: ''=use a unit in the data
showbl0flux: Print the zero baseline flux?
 Default: False

```

`impv-task.html`

### **0.1.87   `impv`**

Requires:

#### **Synopsis**

Construct a position-velocity image by choosing two points in the direction plane.

#### **Arguments**

|           |                                                                                                                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                                                                                                                                       |
| imagename | Name of the input image<br>allowed: string<br>Default:                                                                                                                                                                                |
| outfile   | Output image name. If empty, no image is written.<br>allowed: string<br>Default:                                                                                                                                                      |
| mode      | If "coords", use start and end values. If "length", use center, length, and pa values.<br>allowed: string<br>Default: coords                                                                                                          |
| start     | The starting pixel in the direction plane (array of two values).<br>allowed: any<br>Default: variant                                                                                                                                  |
| end       | The ending pixel in the direction plane (array of two values).<br>allowed: any<br>Default: variant                                                                                                                                    |
| center    | The center point in the direction plane (array of two values). If specified, length and pa must also be specified and neither of start nor end may be specified.<br>allowed: any<br>Default: variant                                  |
| length    | The length of the segment in the direction plane. If specified, center and pa must also be specified and neither of start nor end may be specified.<br>allowed: any<br>Default: variant                                               |
| pa        | The position angle of the segment in the direction plane, measured from north through east. If specified, center and length must also be specified and neither of start nor end may be specified.<br>allowed: any<br>Default: variant |
| width     | Width of slice for averaging pixels perpendicular to the slice. Must be an odd positive integer or valid quantity. See help for details.<br>allowed: any<br>Default: variant 1                                                        |
| unit      | Unit for the offset axis in the resulting image. Must be a unit of angular measure.<br>allowed: string<br>Default: arcsec                                                                                                             |
| overwrite | Overwrite the output if it exists?<br>allowed: bool<br>Default: False                                                                                                                                                                 |
| region    | Region selection. Default is entire image. No selection is permitted in the direction plane. See help par.region.<br>allowed: any<br>Default: variant ""                                                                              |

## Returns

image

## Example

### PARAMETER SUMMARY

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| imagename | Name of the input (CASA, FITS, MIRIAD) image                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| outfile   | Name of output CASA image. Must be specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| mode      | Indicates which sets of parameters to use for defining the slice. mode="coords" means use center, start, and end parameters. mode="length" means use center, length, and position angle to define the slice.                                                                                                                                                                                                                                                                                                                                                                                                       |
| start     | The starting pixel in the direction plane (array of two values), such as [200, 100]. Used iff mode="coords".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| end       | The ending pixel in the direction plane (array of two values), such as [200, 100]. Used iff mode="coords".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| center    | The center of the slice in the direction plane (array of two values), such as [200, 100]. Used iff mode="length".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| length    | The length of the slice in the direction plane. May be specified as a single value, in which case it is interpreted as the number of pixels, or as a valid quantity which can be converted to the direction axes units (eg "40arcsec", {"value": 40, "unit": "arcsec"}). Used iff mode="length".                                                                                                                                                                                                                                                                                                                   |
| pa        | Position angle of the slice, measured from the direction of positive latitude (eg north through east in an equatorial coordinate system). Must be expressed as a valid quantity (eg "40deg", {"value": 40, "unit": "deg"}). Used iff mode="length".                                                                                                                                                                                                                                                                                                                                                                |
| width     | Width of slice for averaging pixels perpendicular to the slice which must be a valid quantity. The averaging using this value occurs after the image has been rotated. An integer value is interpreted as the number of pixels to average. A value of 1 means no averaging. A value of 3 means average one pixel on each side of the slice and the pixel on the slice. A value of 5 means average 2 pixels on each side of the slice and the pixel on the slice, etc. If a quantity (eg "3pixels") is specified, the equivalent number of pixels is calculated, and if necessary, rounded to the next odd integer. |
| unit      | Allows the user to set the unit for the angular offset axis. Must be a unit of angular measure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| overwrite | If output file is specified, this parameter controls if an already existing file with the same name can be overwritten. If true, the user is not prompted, the file is created if it exists is automatically overwritten.                                                                                                                                                                                                                                                                                                                                                                                          |
| region    | Region specification. See help par.region. Default is to not use a region. If specified, the entire direction plane must be specified. If specified do not specify center or length.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| chans     | Optional contiguous frequency channel number specification. Default is all channels. If specified, do not specify region. See "help par.chans" for examples.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

|         |                                                                                                      |
|---------|------------------------------------------------------------------------------------------------------|
| stokes  | Contiguous stokes planes specification. If specified, do not specify region                          |
| mask    | Mask to use. See help par.mask. Default is none.                                                     |
| stretch | Stretch the input mask if necessary and possible. Only used if a mask is sp<br>See help par.stretch. |

Create a position-velocity image. The way the slice is specified is controlled by the mode parameter. If mode="coords", start and end are used to specify the points between which a slice is taken in the coordinate system. If mode="length" center, pa (position angle), and length are used to specify the extent of the resulting image will be that provided by the region specification or the entire input image if no region is specified. One may not specify a region in direction space; specifying the slice as described previously. The parameters start and end may be specified as arrays of numerical values, in which case these values will be interpreted as pixel coordinates. Alternatively, they may be expressed as arrays of two strings each representing the coordinates in either represent quantities (eg ["40.5deg", "0.5rad"]) or be sexagesimal format (eg ["14:20:20.5s", "-30.45.25.4"]). In addition, they may be expressed as a single string containing latitude-like values and optionally a reference frame value, eg "J2000 14:20:20.5s -30.45.25.4". The length parameter may be specified as a single numerical value, which will be interpreted as the length in pixels, or a valid quantity, in which case it must have units of length or direction axes units. The pa (position angle) parameter must be specified as a valid quantity. The position angle is interpreted in the usual astronomical sense; eg measured from north through east in the coordinate system. The slice in this case starts at the specified position angle and ends at the specified center. Thus pa="45deg" means start at a point at a pa of 45 degrees relative to the center. The slice ends at a point at a pa of 215 degrees relative to the center. Either start/end or center/pa must be specified. If a parameter from one of these sets is specified, a parameter from the other set may not be specified. In this case, the end points of the segment must fall within the input image, and they both must be within the edge of the input image to facilitate rotation (see below).

One may specify a width, which represents the number of pixels centered along and perpendicular to the direction slice that are used for averaging along the slice. The width may be specified as a single numerical value, in which case it must be positive and odd. Alternatively, it may be specified as a valid quantity string, in which case it must be a quantity record (eg qa.quantity("4arcsec")). In this case, units must be conformant to the direction axes units and the specified quantity will be rounded up, if necessary, to the next highest integer value of pixels. The default value of 1 represents no averaging.

A value of 3 means average one pixel on each side of the slice and the pixel on the slice. Note that this width is applied to pixels in the image after it has been rotated (see below for details of the algorithm used).

One may specify the unit for the angular offset axis.

Internally, the image is first rotated, padding if necessary to include relevant pixels that would otherwise be excluded by the rotation operation, so that the slice is horizontal, with the starting pixel at the left and the ending pixel at the right. Then, the pixels within the specified width of the slice are averaged and the result is written and/or returned. The output image has a linear coordinate in place of the direction coordinate, and the corresponding axis represents angular offset with the center pixel having an offset of zero.

The equivalent coordinate system, with a (usually) rotated direction coordinate (eg, RA and Dec).

to the output image as a table record. It can be retrieved using the table tool as shown in

```
create a pv image with the position axis running from ra, dec pixel positions of [45, 50]
in the input image
impv(imagename="my_spectral_cube.im", outfile="mypv.im", start=[45,50], end=[100,120])
analyze the pv image, such as get statistics
pvstats = imstat("mypv.im")
#
get the alternate coordinate system information
tb.open("mypv.im")
alternate_csys_record = tb.getkeyword("misc")["secondary_coordinates"]
tb.done()
```

`rmfit-task.html`

## **0.1.88   rmfit**

Requires:

### **Synopsis**

Calculate rotation measure.

### **Arguments**



|           |                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                                            |
| imagename | Name(s) of the input image(s). Must be specified.<br>allowed: any<br>Default: variant                                                      |
| rm        | Output rotation measure image name. If not specified, no image is written.<br>allowed: string<br>Default:                                  |
| rmerr     | Output rotation measure error image name. If not specified, no image is written.<br>allowed: string<br>Default:                            |
| pa0       | Output position angle (degrees) at zero wavelength image name. If not specified, no image is written.<br>allowed: string<br>Default:       |
| pa0err    | Output position angle (degrees) at zero wavelength error image name. If not specified, no image is written.<br>allowed: string<br>Default: |
| nturns    | Output number of turns image name. If not specified, no image is written.<br>allowed: string<br>Default:                                   |
| chisq     | Output reduced chi squared image name. If not specified, no image is written.<br>allowed: string<br>Default:                               |
| sigma     | Estimate of the thermal noise. A value less than 0 means auto estimate.<br>allowed: double<br>Default: -1                                  |
| rmfg      | Foreground rotation measure in rad/m/m to subtract.<br>allowed: double<br>Default: 0.0                                                     |
| rmmax     | Maximum rotation measure in rad/m/m for which to solve. IMPORTANT TO SPECIFY.<br>allowed: double<br>Default: 0.0                           |
| maxpaerr  | Maximum input position angle error in degrees to allow in solution determination.<br>allowed: double<br>Default: 1e30                      |

## Returns

bool

## Example

### PARAMETER SUMMARY

|           |                                                                                                             |
|-----------|-------------------------------------------------------------------------------------------------------------|
| imagename | Name(s) of the input image(s).                                                                              |
| rm        | Output rotation measure image name. If not specified, no image is written.                                  |
| rmerr     | Output rotation measure error image name. If not specified, no image is written.                            |
| pa0       | Output position angle (degrees) at zero wavelength image name. If not specified, no image is written.       |
| pa0err    | Output position angle (degrees) at zero wavelength error image name. If not specified, no image is written. |
| nturns    | Output number of turns image name. If not specified, no image is written.                                   |
| chisq     | Output reduced chi squared image name. If not specified, no image is written.                               |
| sigma     | Estimate of the thermal noise. A value less than 0 means auto estimate.                                     |
| rmfg      | Foreground rotation measure in rad/m/m to subtract.                                                         |
| rmmx      | Maximum rotation measure in rad/m/m for which to solve. IMPORTANT TO SPECIFY                                |

This task generates the rotation measure image from stokes Q and U measurements at several different frequencies. You are required to specify the name of at least one image with a position angle axis containing stokes Q and U planes and with a frequency axis containing more than two pixels. Frequencies do not have to be equally spaced (ie the frequency coordinate can be a tabular array). It will work out the position angle images for you. You may also specify multiple image names. In this case these images will first be concatenated along the spectral axis using `ia.imageconcat()`. It is important that for all images, the axis order must be the same and the number of pixels along each axis must be identical, except for the spectral axis which may differ in length between images. The spectral axis does not have to be contiguous from one image to another.

See also the `fourierrotationmeasure` function for a new Fourier-based approach.

Rotation measure algorithms that work robustly are few. The main problem is in trying to account for the  $2\pi$  ambiguity (see Leahy et al, *Astronomy & Astrophysics*, 156, 234 or Killeen et al; <http://www.atnf.csiro.au/~nkilleen/rm.ps>).

The algorithm that this task uses is that of Leahy et al. in see Appendix A.1. But as in all these algorithms, the basic process is that for each spatial pixel, the position angle vs frequency data is fit to determine the rotation measure and the position angle at zero wavelength (and associated errors). An image containing the number of  $2\pi$  turns that were added to the data at each spatial pixel and for which the best fit was found can be written. The reduced chi-squared image for the fits can

also be written.

Note that no assessment of curvature (i.e. deviation from the simple linear position angle -  $\lambda^2$  functional form) is made.

Any combination of output images can be written.

The parameter sigma gives the thermal noise in Stokes Q and U. By default it is determined automatically using the image data. But if it proves to be inaccurate (maybe not many signal-free pixels), it may be specified. This is used for calculating the error in the position angles (via propagation of Gaussian errors).

The argument maxpaerr specifies the maximum allowable error in the position angle that is acceptable. The default is an infinite value. From the standard propagation of errors, the error in the linearly polarized position angle is determined from the Stokes Q and U images (at each directional pixel for each frequency). If the position angle error for any pixel exceeds the specified value, the position angle at that pixel is omitted from the fit. The process generates an error for the fit and this is used to compute the errors in the output images.

Note that maxpaerr is not used to mask pixels in the output images.

The argument rmfg is used to specify a foreground RM value. For example, you may know the mean RM in some direction out of the Galaxy, then including this can improve the algorithm by reducing ambiguity.

The parameter rmmax specifies the maximum absolute RM value that should be solved for. This quite an important parameter. If you leave it at the default, zero, no ambiguity handling will be used. So some apriori information should be supplied; this is the basic problem with rotation measure algorithms.

#### EXAMPLES

```
Calculate the rotation measure for a single polarization image
rmfit(imagename="mypol.im", rm="myrm.im", rmmax=50.0)
```

```
calculate the rotation measure using a set of polarization images from
different spectral windows or bands.
```

```
rmfit(imagename=["pol1.im", "pol2.im", "pol3.im", rm="myrm2.im", rmmax=50.0)
```



rmtables-task.html

## 0.1.89 rmtables

Requires:

### Synopsis

Remove tables cleanly, use this instead of `rm -rf`

### Description

This task removes tables if they are not being currently accessed via the casapy process. Note: if you have multiple sessions running bad things could happen if you remove a table being accessed by another process.

### Arguments

| Inputs     |                            |
|------------|----------------------------|
| tablenames | Name of the tables         |
|            | allowed:       stringArray |
|            | Default:                   |

### Example

Removes tables cleanly.

Arguments may contain `*` or `?`. Ranges `[]` are support but not `~` expansion.

sdaverage-task.html

## 0.1.90 sdaverage

Requires:

### Synopsis

ASAP SD task: averaging and smoothing of spectra

### Description

Task sdaverage performs averaging in time/polarization and smoothing of the single-dish spectra. When timeaverage=True, spectra are averaged in time. Spectra within each scan ID are averaged when scanaverage=True. When polaverage=True, spectra are averaged in polarization and time (Note time averaging with polaverage=True would be discarded in future). See examples in below for details of time/polarization average. When kernel is specified (!=”), each spectrum is smoothed by convolving the kernel after averaging of spectra.

If you give multiple IFs (spectral windows) in spw, then your scantable will have multiple IFs by default. Averaging of multi-resolution (multi-IFs) spectra can be achieved by setting a sub-parameter in timeaverage, averageall = True. It handles multi-IFs by merging IFs which have overlaps in frequency coverages and assigning new IFs in the output spectra.

Set plotlevel  $i=1$  to plot spectrum before and after smoothing, and verify=True to interactively select whether or not accept smoothing results. NOTE, so far, there is no mechanism to verify averaging of spectra in time and/or polarization.

### Arguments

|             |                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                                                         |
| infile      | name of input SD dataset<br>allowed: string<br>Default:                                                                                 |
| antenna     | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0                           |
| field       | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                                                  |
| spw         | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                                           |
| scan        | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                                        |
| pol         | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                                      |
| timeaverage | average spectra over time [True, False] (see examples in help)<br>allowed: bool<br>Default: False                                       |
| tweight     | weighting for time averaging<br>allowed: string<br>Default: tintsys                                                                     |
| scanaverage | average spectra within a scan number [True, False] (see examples in help)<br>allowed: bool<br>Default: False                            |
| averageall  | set True only when averaging spectra with different spectral resolutions<br>allowed: bool<br>Default: False                             |
| polaverage  | average spectra over polarizations [True, False]<br>allowed: bool<br>Default: False                                                     |
| pweight     | weighting for polarization averaging<br>allowed: string<br>Default: tsys                                                                |
| kernel      | type of spectral smoothing kernel ("=no smoothing)<br>allowed: string<br>Default:                                                       |
| kwidth      | width of smoothing kernel in channels<br>allowed: int<br>Default: 438                                                                   |
| chanwidth   | width of regridded channels<br>allowed: string<br>Default: 5                                                                            |
| verify      | interactively verify the results of smoothing for each spectrum. [not available for kernel="regrid"]<br>allowed: bool<br>Default: False |
| plotlevel   | plot and summarize results (0=none). See description                                                                                    |

## Returns

void

## Example

Keyword arguments:

infile -- name of input SD dataset

antenna -- select an antenna name or ID

default: 0

example: 'PM03'

NOTE this parameter is effective only for MS input

field -- select data by field IDs and names

default: '' (use all fields)

example: field='3C2\*' (all names starting with 3C2)

field='0,4,5~7' (field IDs 0,4,5,6,7)

field='0,3C273' (field ID 0 or field named 3C273)

this selection is in addition to the other selections to data

spw -- select data by IF IDs (spectral windows)

NOTE this task only supports IF ID selection and ignores channel

selection.

default: '' (use all IFs and channels)

example: spw='3,5,7' (IF IDs 3,5,7; all channels)

spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)

spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)

this selection is in addition to the other selections to data

scan -- select data by scan numbers

default: '' (use all scans)

example: scan='21~23' (scan IDs 21,22,23)

this selection is in addition to the other selections to data

pol -- select data by polarization IDs

default: '' (use all polarizations)

example: pol='0,1' (polarization IDs 0,1)

this selection is in addition to the other selections to data

timeaverage -- average spectra over time

options: (bool) True, False

default: False

>>>timeaverage expandable parameter

tweight -- weighting for time averaging

options: 'var' (1/var(spec) weighted)

'tsys' (1/Tsys\*\*2 weighted)



```

 'tint' (integration time weighted)
 'tintsys' (Tint/Tsys**2)
 'median' (median averaging)
 default: 'tintsys'
scanaverage -- average spectra within a scan number
 when True, spectra are NOT averaged over
 different scan numbers.
 options: (bool) True, False
 default: False
averageall -- average multi-resolution spectra
 spectra are averaged by referring
 their frequency coverage
 default: False

polaverage -- average spectra over polarizations
 options: (bool) True, False
 default: False

>>>polaverage expandable parameter
 pweight -- weighting for polarization averaging
 options: 'var' (1/var(spec) weighted)
 'tsys' (1/Tsys**2 weighted)
 default: 'tsys'

kernel -- type of spectral smoothing kernel
 options: '', 'hanning', 'gaussian', 'boxcar', 'regrid'
 default: '' (no smoothing)

>>>kernel expandable parameter
 kwidth -- width of spectral smoothing kernel
 options: (int) in channels
 default: 5
 example: 5 or 10 seem to be popular for boxcar
 ignored for hanning (fixed at 5 chans)
 (0 will turn off gaussian or boxcar)
 chanwidth -- channel width of regridded spectra
 default: '5' (in channels)
 example: '500MHz', '0.2km/s'
 verify -- interactively verify the results of smoothing for each
 spectrum. When verify = True, for each input spectrum,
 spectra before and after the operation are displayed
 in a plot window. At the prompt there are four choices
 of action: 'Y' (accept the operation and continue to
 the next input spectrum), 'N' (reject the operation
 and continue to the next input spectrum), 'A' (accept
 the current operation and continue non-interactively),

```

and 'R' (reject the current operation and exit from operation).  
 Note that when the operation is rejected by 'N' or 'R', no smoothing is done to the spectrum/spectra.  
 options: (bool) True,False  
 default: False

outfile -- name of output file  
 default: '' (<infile>\_sm)  
 outform -- output file format  
 options: 'ASAP','MS2', 'ASCII','SDFITS'  
 default: 'ASAP'  
 NOTE the ASAP format is easiest for further sd processing; use MS2 for CASA imaging.  
 If ASCII, then will append some stuff to the outfile name  
 overwrite -- overwrite the output file if already exists  
 options: (bool) True,False  
 default: False  
 NOTE this parameter is ignored when outform='ASCII'  
 plotlevel -- control for plotting and summary of smoothing results  
 options: (int) 0, 1, 2, and their negative counterparts  
 default: 0 (no plotting)  
 example: plotlevel=1; plot spectra before and after smoothing  
           plotlevel=2; additionally list data before and after operation.  
           plotlevel<0 as abs(plotlevel), e.g.  
           -1 => hardcopy of final plot (will be named  
           <outfile>\_smspec.eps)

# ----- AVERAGING OF SPECTRA -----

Task sdaverage has two modes of averaging spectra, i.e., time and polarization average.

When timeaverage=True, spectra are averaged over time for each IF (spectral window), polarization, and beam, independently. Note that, by default (scanaverage=False), timeaverage=True averages spectra irrespective of scan IDs.

It is possible to average spectra separately for each scan ID by setting a sub-parameter scanaverage=True.

For example, the combination of parameters: scan='0~2', timeaverage=True, and scanaverage=False: averages spectra in scan ID 0 through 2 all together to a spectrum,

scanaverage=True : averages spectra per scan ID and end up with three spectra from scan 0, 1, and 2.

When polaverage=True, spectra are averaged over polarization for each IF (spectral window) and beam. Note that, so far, time averaging is automatically switched on when polaverage is set to True. This behavior is not desirable and will be discarded in future.

-----  
WARNING  
-----

For the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

[sdbaseline-task.html](#)

### **0.1.91 sdbaseline**

Requires:

#### **Synopsis**

Fit/subtract a spectral baseline

#### **Description**

Task sdbaseline performs baseline fitting/removal for single-dish spectra. The fit parameters, terms and rms of base-line are saved to an ascii file, 'outfile<sub>i</sub>.blparam.txt'.

#### **Arguments**

|                |                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                                       |
| infile         | name of input SD dataset<br>allowed: string<br>Default:                                                               |
| antenna        | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0         |
| fluxunit       | units of the flux ("=current)<br>allowed: string<br>Default:                                                          |
| telescopeparam | parameters of telescope for flux conversion (see examples in help)<br>allowed: any<br>Default: variant                |
| field          | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                                |
| spw            | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                         |
| restfreq       | the rest frequency, e.g. '1.41GHz' (default unit: Hz) (see examples in help)<br>allowed: any<br>Default: variant      |
| frame          | frequency reference frame ("=current)<br>allowed: string<br>Default:                                                  |
| doppler        | doppler convention ("=current). Effective only when spw selection is in velocity unit.<br>allowed: string<br>Default: |
| timerange      | select data by time range, e.g. '09:14:0~09:54:0' ("=all) (see examples in help)<br>allowed: string<br>Default:       |
| scan           | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                      |
| pol            | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                    |
| tau            | the zenith atmospheric optical depth for correction<br>allowed: double<br>Default: 4440.0                             |
| maskmode       | mode of setting additional channel masks<br>allowed: string<br>Default:                                               |
| thresh         | S/N threshold for linefinder<br>allowed: double<br>Default: 5.0                                                       |
| avg_limit      | channel averaging for broad lines<br>allowed: int                                                                     |

## Returns

void

## Example

```

Keyword arguments

infile -- name of input SD dataset
antenna -- select an antenna name or ID
 default: 0
 example: 'PM03'
 NOTE this parameter is effective only for MS input
fluxunit -- units for line flux
 options: 'K','Jy',''
 default: '' (keep current fluxunit in data)
 WARNING: For GBT data, see description below.
>>> fluxunit expandable parameter
telescopeparam -- parameters of telescope for flux conversion
 options: (str) name or (list) list of gain info
 default: '' (none set)
 example: if telescopeparam='', it tries to get the telescope
 name from the data.
 Full antenna parameters (diameter,ap.eff.) known
 to ASAP are
 'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',
 'CEDUNA','HOBART'. For GBT, it fixes default fluxunit
 to 'K' first then convert to a new fluxunit.
 telescopeparam=[104.9,0.43] diameter(m), ap.eff.
 telescopeparam=[0.743] gain in Jy/K
 telescopeparam='FIX' to change default fluxunit
 see description below
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)/channels
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
```

```

 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all
 spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
>>> spw expandable parameter
 restfreq -- the rest frequency
 available type includes float, int, string, list of float,
 list of int, list of string, and list of dictionary. the
 default unit of restfreq in case of float, int, or string
 without unit is Hz. string input can be a value only
 (treated as Hz) or a value followed by unit for which 'GHz',
 'MHz', 'kHz', and 'Hz' are available.
 a list can be used to set different rest frequencies for
 each IF. the length of list input must be number of IFs.
 dictionary input should be a pair of line name and
 frequency with keys of 'name' and 'value', respectively.
 values in the dictionary input follows the same manner as
 as for single float or string input.
 example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]
 frame -- frequency reference frame
 options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
 default: '' (keep current frame in data)
 doppler -- doppler convention (effective only when spw is in
 velocity unit)
 options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
 default: '' (keep current doppler setting in data)
 timerange -- select data by time range
 default: '' (use all)
 example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:00~09:54:00' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
 this selection is in addition to the other selections to data
 scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
 pol -- select data by polarization IDs

```

```

 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
tau -- the zenith atmospheric optical depth for correction
 default: 0.0 (no correction)
maskmode -- mode of setting additional channel masks
 options: 'auto', 'list', or 'interact'
 default: 'auto'
 example: maskmode='auto' runs linefinder to detect line regions
 to be excluded from fitting. this mode requires three
 expandable parameters: thresh, avg_limit, and edge.
 USE WITH CARE! May need to tweak the expandable parameters.
 maskmode='list' uses the given masklist only: no additional
 masks applied.
 maskmode='interact' allows users to manually modify the
 mask regions by dragging mouse on the spectrum plotter GUI.
 use LEFT or RIGHT button to add or delete regions,
 respectively.

>>> maskmode expandable parameters
 thresh -- S/N threshold for linefinder. a single channel S/N ratio
 above which the channel is considered to be a detection.
 default: 5
 avg_limit -- channel averaging for broad lines. a number of
 consecutive channels not greater than this parameter
 can be averaged to search for broad lines.
 default: 4
 edge -- channels to drop at beginning and end of spectrum
 default: 0
 example: edge=[1000] drops 1000 channels at beginning AND end.
 edge=[1000,500] drops 1000 from beginning and 500
 from end.

 Note: For bad baselines threshold should be increased,
 and avg_limit decreased (or even switched off completely by
 setting this parameter to 1) to avoid detecting baseline
 undulations instead of real lines.
blfunc -- baseline model function
 options: 'poly', 'chebyshev', 'cspline', or 'sinusoid'
 default: 'poly'
 example: blfunc='poly' uses a single polynomial line of
 any order which should be given as an expandable
 parameter 'order' to fit baseline.
 blfunc='chebyshev' uses Chebyshev polynomials.
 blfunc='cspline' uses a cubic spline function, a piecewise
 cubic polynomial having C2-continuity (i.e., the second
 derivative is continuous at the joining points).

```



```

 blfunc='sinusoid' uses a combination of sinusoidal curves.
>>> blfunc expandable parameters
 order -- order of baseline model function
 options: (int) (<0 turns off baseline fitting)
 default: 5
 example: typically in range 2-9 (higher values
 seem to be needed for GBT)
 npiece -- number of the element polynomials of cubic spline curve
 options: (int) (<0 turns off baseline fitting)
 default: 2
 applyfft -- automatically set wave numbers of sinusoidal functions
 for fitting by applying some method like FFT.
 options: (bool) True, False
 default: True
 fftmethod -- method to be used when applyfft=True. Now only
 'fft' is available and it is the default.
 fftthresh -- threshold to select wave numbers to be used for
 sinusoidal fitting. both (float) and (str) accepted.
 given a float value, the unit is set to sigma.
 for string values, allowed formats include:
 'xsigma' or 'x' (= x-sigma level. e.g., '3sigma'), or
 'topx' (= the x strongest ones, e.g. 'top5').
 default is 3.0 (unit: sigma).
 addwn -- additional wave number(s) of sinusoids to be used
 for fitting.
 (list) and (int) are accepted to specify every
 wave numbers. also (str) can be used in case
 you need to specify wave numbers in a certain range.
 default: [0] (i.e., constant is subtracted at least)
 example: 0
 [0,1,2]
 'a-b' (= a, a+1, a+2, ..., b-1, b),
 '<a' (= 0,1,...,a-2,a-1),
 '>=a' (= a, a+1, ... up to the maximum wave
 number corresponding to the Nyquist
 frequency for the case of FFT).
 rejwn -- wave number(s) of sinusoid NOT to be used for fitting.
 can be set just as addwn but has higher priority:
 wave numbers which are specified both in addwn
 and rejwn will NOT be used.
 default: []
 clipthresh -- clipping threshold for iterative fitting
 default: 3
 clipniter -- maximum iteration number for iterative fitting
 default: 0 (no iteration, i.e., no clipping)
 verify -- interactively verify the results of operation for each spectrum.

```

When verify = True, for each input spectrum, spectra before and after the operation are displayed in a plot window. At the prompt there are four choices of action: 'Y' (accept the operation and continue to the next input spectrum), 'N' (reject the operation and continue to the next input spectrum), 'A' (accept the current operation and continue non-interactively), and 'R' (reject the current operation and exit from operation). Note that when the operation is rejected by 'N' or 'R', no operation is done to the spectrum/spectra.

options: (bool) True,False  
default: False

NOTE: Currently available only when blfunc='poly'

verbose -- output fitting results to logger. if False, the fitting results including coefficients, residual rms, etc., are not output to the CASA logger, while the processing speed gets faster.

options: (bool) True, False  
default: True

bloutput -- output fitting results to a text file. if False, the fitting results including coefficients, residual rms, etc., are not output to a text file (<outfile>\_blparam.txt), while the processing speed gets faster.

options: (bool) True, False  
default: True

blformat -- format of the logger output and text file specified with bloutput

options: '', 'csv'  
default: '' (same as in the past, easy to read but huge)

showprogress -- show progress status for large data

options: (bool) True, False  
default: True

>>> showprogress expandable parameter

minnrow -- minimum number of input spectra to show progress status

default: 1000

outfile -- name of output file

default: '' (<infile>\_bs)

outform -- output file format

options: 'ASAP', 'MS2', 'ASCII', 'SDFITS'  
default: 'ASAP'

NOTE the ASAP format is easiest for further sd processing; use MS2 for CASA imaging.  
If ASCII, then will append some stuff to the outfile name

overwrite -- overwrite the output file if already exists

options: (bool) True, False  
default: False

NOTE this parameter is ignored when outform='ASCII'

```

plotlevel -- control for plotting of results.
 options: 0, 1, 2, or <0
 default: 0 (no plotting)
 example: 0 (no plotting)
 1 (some)
 2 (more)
 <0 (hardcopy) as abs(plotlevel), e.g.
 -1 => hardcopy of final plot (will be named
 <outfile>_bspec.eps)

```

```

DESCRIPTION

```

Task sdbaseline performs baseline fitting/removal for single-dish spectra. The fit parameters, terms and rms of baseline are saved to an ascii file, '<outfile>\_blparam.txt' if bloutput is True.

```

BASELINE MODEL FUNCTION

```

The list of available model functions are shown above (see Keyword arguments section). In general 'cspline' or 'chebyshev' are recommended since they are more stable than others. 'poly' will work for lower order but will be unstable for higher order fitting. 'sinusoid' is kind of special mode that will be useful for the data that clearly shows standing wave in the spectral baseline.

```

SIGMA CLIPPING (ITERATIVE FITTING)

```

In general least square fitting is strongly affected by an extreme data so that the resulting fit makes worse. Sigma clipping is an iterative baseline fitting with data clipping based on a certain threshold. Threshold is set as a certain factor times rms of the resulting (baseline subtracted) spectra. If sigma clipping is on, baseline fit/removal is performed several times. After each baseline subtraction, the data whose absolute value is above threshold are detected and those data are excluded from the next round of fitting. By using sigma clipping, extreme data are excluded from the fit so that resulting fit is more robust.

The user is able to control a multiplication factor using parameter clipthresh for clipping threshold based on rms. Actual threshold for sigma clipping will be (clipthresh) x (rms of spectra). Also, the user can specify number of maximum iteration to the parameter clipniter.

In general, sigma clipping will lower the performance since it increases number of fits per spectra. However, it is strongly recommended to turn on sigma clipping unless you are sure that the data is free from any kind of extreme values that may affect the fit.

-----  
FLUX UNIT CONVERSION  
-----

The task is able to convert flux unit between K and Jy. To do that, fluxunit and its subparameter telescopeparam must be properly set. The fluxunit should be 'Jy' or 'K' depending on what unit input data is and what unit you want to convert. If given fluxunit is different from the unit of input data, unit conversion is performed. The telescopeparam is used to specify conversion factor. There are three ways to specify telescopeparam: 1) set Jy/K conversion factor, 2) set telescope diameter, D, and aperture efficiency, eta, separately, and 3) 'FIX' mode (only change the unit without converting spectral data). If you give telescopeparam as a list, then if the list has a single float it is assumed to be the gain in Jy/K (case 1), if two or more elements they are assumed to be telescope diameter (m) and aperture efficiency respectively (case 2). See the above parameter description as well as note on 'FIX' mode below for details.

There are two special cases that don't need telescopeparam for unit conversion. Telescope name is obtained from the data.

- 1) ASAP (sd tool) recognizes the conversion factor (actually D and eta) for the "AT" telescopes, namely ATNF MOPRA telescope, until 2004.
- 2) The task does know D and eta for GBT telescope.

If you wish to change the fluxunit, by leaving the sub-parameter telescopeparam unset (telescopeparam=''), it will use internal telescope parameters for flux conversion for the data from AT telescopes and it will use an approximate aperture efficiency conversion for the GBT data.

Note that sdbaseline assumes that the fluxunit is set correctly in the data already. If not, then set telescopeparam='FIX' and it will set the default units to fluxunit without conversion.

Note also that, if the data in infile is an ms from GBT and the default flux unit is missing, this task automatically fixes the default fluxunit to 'K' before the conversion.

-----  
WARNING  
-----

For the GBT raw SDFITS format data as input:

SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

`sdbaseline2-task.html`

### **0.1.92 sdbaseline2**

Requires:

#### **Synopsis**

Fit/subtract a spectral baseline

#### **Description**

Task `sdbaseline2` performs baseline fitting/removal for single-dish spectra.

#### **Arguments**

|           |                                                                                                                       |
|-----------|-----------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                       |
| infile    | name of input SD dataset<br>allowed: string<br>Default:                                                               |
| antenna   | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0         |
| row       | select data by row IDs, e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                                           |
| field     | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                                |
| spw       | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                         |
| restfreq  | the rest frequency, e.g. '1.41GHz' (default unit: Hz) (see examples in help)<br>allowed: any<br>Default: variant      |
| frame     | frequency reference frame ("=current)<br>allowed: string<br>Default:                                                  |
| doppler   | doppler convention ("=current). Effective only when spw selection is in velocity unit.<br>allowed: string<br>Default: |
| timerange | select data by time range, e.g. '09:14:0~09:54:0' ("=all) (see examples in help)<br>allowed: string<br>Default:       |
| scan      | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                      |
| pol       | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                    |
| blmode    | baselining mode ('subtract' or 'apply')<br>allowed: string<br>Default: subtract                                       |
| blparam   | per spectrum fit parameters<br>allowed: any<br>Default: variant                                                       |
| bltable   | name of baseline table<br>allowed: string<br>Default:                                                                 |
| outfile   | name of output file (See a WARNING in help)<br>allowed: string<br>Default:                                            |
| overwrite | overwrite the output file if already exists<br>allowed: bool<br>Default: False                                        |

## Returns

void

## Example

-----  
Keyword arguments  
-----

infile -- name of input SD dataset

antenna -- select an antenna name or ID

default: 0

example: 'PM03'

NOTE this parameter is effective only for MS input

field -- select data by field IDs and names

default: '' (use all fields)

example: field='3C2\*' (all names starting with 3C2)

field='0,4,5~7' (field IDs 0,4,5,6,7)

field='0,3C273' (field ID 0 or field named 3C273)

this selection is in addition to the other selections to data

spw -- select data by IF IDs (spectral windows)/channels

default: '' (use all IFs and channels)

example: spw='3,5,7' (IF IDs 3,5,7; all channels)

spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)

spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)

spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)

spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)

spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)

spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)

this selection is in addition to the other selections to data

>>> spw expandable parameter

restfreq -- the rest frequency

available type includes float, int, string, list of float,

list of int, list of string, and list of dictionary. the

default unit of restfreq in case of float, int, or string

without unit is Hz. string input can be a value only

(treated as Hz) or a value followed by unit for which 'GHz',

'MHz', 'kHz', and 'Hz' are available.

a list can be used to set different rest frequencies for

each IF. the length of list input must be number of IFs.

dictionary input should be a pair of line name and

frequency with keys of 'name' and 'value', respectively.

values in the dictionary input follows the same manner as



```

 as for single float or string input.
example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]
frame -- frequency reference frame
 options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
 default: '' (keep current frame in data)
doppler -- doppler convention (effective only when spw is in
 velocity unit)
 options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
 default: '' (keep current doppler setting in data)
timerange -- select data by time range
 default: '' (use all)
 example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:00~09:54:00' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
blmode -- 'subtract' or 'apply'
 default: 'subtract'
>>> blmode expandable parameter
blparam -- per spectrum fit parameters. it must be a list of
 dictionary. Each dictionary corresponds to each
 spectrum and must contain the following keys and values:
 'row': row number,
 'blfunc': function name. available ones include
 'poly', 'chebyshev', 'cspline' and 'sinusoid',
 'order': maximum order of polynomial. needed when
 blfunc='poly' or 'chebyshev',
 'npiece': number or piecewise polynomial.
 needed when blfunc='cspline' and
 'nwave': a list of sinusoidal wave numbers.
 needed when blfunc='sinusoid'.
 example: [{'row':0,'blfunc':'poly','order':5},

```

```

 {'row':1,'blfunc':'chebyshev','order':10},
 {'row':2,'blfunc':'cspline','npiece':4},
 {'row':3,'blfunc':'sinusoid','nwave':[0,1,2,3]}, ...]
bltable -- the name of baseline table (bltable='' would subtract baseline without
 creating bltable. mandatory when blmode='apply')
 default: ''
outfile -- name of output file
 default: '' (<infile>_bs)
overwrite -- overwrite existing outfile and bltable or not
 options: (bool) True, False
 default: False

```

-----  
DESCRIPTION  
-----

Task sdbaseline2 performs baseline fitting/removal for single-dish spectra.

-----  
BASELINE MODEL FUNCTION  
-----

The list of available model functions are shown above (see Keyword arguments section). In general 'cspline' or 'chebyshev' are recommended since they are more stable than others. 'poly' will work for lower order but will be unstable for higher order fitting. 'sinusoid' is kind of special mode that will be useful for the data that clearly shows standing wave in the spectral baseline.

-----  
SIGMA CLIPPING (ITERATIVE FITTING)  
-----

In general least square fitting is strongly affected by an extreme data so that the resulting fit makes worse. Sigma clipping is an iterative baseline fitting with data clipping based on a certain threshold. Threshold is set as a certain factor times rms of the resulting (baseline subtracted) spectra. If sigma clipping is on, baseline fit/removal is performed several times. After each baseline subtraction, the data whose absolute value is above threshold are detected and those data are excluded from the next round of fitting. By using sigma clipping, extreme data are excluded from the fit so that resulting fit is more robust.

The user is able to control a multiplication factor using parameter clipthresh for clipping threshold based on rms. Actual threshold for sigma clipping will be (clipthresh) x (rms of spectra). Also, the user can specify number of maximum iteration to the parameter clipniter.

In general, sigma clipping will lower the performance since it increases

number of fits per spectra. However, it is strongly recommended to turn on sigma clipping unless you are sure that the data is free from any kind of extreme values that may affect the fit.

[sdcal-task.html](#)

### **0.1.93 sdcal**

Requires:

#### **Synopsis**

ASAP SD calibration task

#### **Description**

Task sdcal performs calibration for single-dish spectra. The parameter, calmode, defines calibration mode. The available calibration modes are 'ps' (for position switching with explicit reference scans), 'otfraster' (for raster OTF scan without explicit reference scans), 'otf' (for non-raster OTF scan without explicit reference scans, e.g, Lissajous, double circle), 'fs' (for frequency switching), 'nod' (beam switching), and 'quotient' (for position switching scans by ATNF telescopes). The task selects appropriate calibration equation based on the value of calmode and telescope with which the data is taken. See below for details of calibration equation adopted in this task. By setting calmode='none', one can run sdcal on already calibrated data for atmospheric optical depth correction.

#### **Arguments**

|                |                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                               |
| infile         | name of input SD dataset<br>allowed: string<br>Default:                                                       |
| antenna        | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0 |
| fluxunit       | units of the flux ("=current)<br>allowed: string<br>Default:                                                  |
| telescopeparam | parameters of telescope for flux conversion (see examples in help)<br>allowed: any<br>Default: variant        |
| field          | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                        |
| spw            | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                 |
| scan           | select data by scan numbers, e.g. '21~23' (" = all)<br>allowed: string<br>Default:                            |
| pol            | select data by polarization IDs, e.g. '0,1' (" = all)<br>allowed: string<br>Default:                          |
| calmode        | SD calibration mode<br>allowed: string<br>Default: ps                                                         |
| fraction       | fraction of the OFF data to mark as OFF spectra, e.g., '10%'<br>allowed: any<br>Default: variant 10%          |
| noff           | number of the OFF data to mark (-1 = use fraction instead of number)<br>allowed: int<br>Default: -1           |
| width          | width of the pixel for edge detection<br>allowed: double<br>Default: 0.5                                      |
| elongated      | the observed area is elongated in one direction<br>allowed: bool<br>Default: False                            |
| markonly       | do calibration (False) or just mark OFF (True)<br>allowed: bool<br>Default: False                             |
| plotpointings  | plot pointing direction for ON and OFF<br>allowed: bool<br>Default: False                                     |
| tau            | the zenith atmospheric optical depth for correction (0. = no correction)<br>allowed: double<br>Default: 0.0   |

## Returns

void

## Example

Keyword arguments:

infile -- name of input SD dataset

antenna -- select an antenna name or ID

default: 0

example: 'PM03'

NOTE this parameter is effective only for MS input

fluxunit -- units for line flux

options: 'K','Jy',''

default: '' (keep current fluxunit in data)

WARNING: For GBT data, see description below.

>>> fluxunit expandable parameter

telescopeparam -- parameters of telescope for flux conversion

options: (str) name or (list) list of gain info

default: '' (none set)

example: if telescopeparam='', it tries to get the telescope name from the data.

Full antenna parameters (diameter,ap.eff.) known to ASAP are

'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',

'CEDUNA','HOBART'. For GBT, it fixes default fluxunit to 'K' first then convert to a new fluxunit.

telescopeparam=[104.9,0.43] diameter(m), ap.eff.

telescopeparam=[0.743] gain in Jy/K

telescopeparam='FIX' to change default fluxunit

see description below

field -- select data by field IDs and names

default: '' (use all fields)

example: field='3C2\*' (all names starting with 3C2)

field='0,4,5~7' (field IDs 0,4,5,6,7)

field='0,3C273' (field ID 0 or field named 3C273)

this selection is in addition to the other selections to data

spw -- select data by IF IDs (spectral windows)

NOTE this task only supports IF ID selection and ignores channel selection.

default: '' (use all IFs and channels)

```

example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)
this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
this selection is in addition to the other selections to data

calmode -- calibration mode
options: 'ps','nod','otf','otfraster',
 'fs','quotient','none'
default: 'ps'
example: choose mode 'none' if you have
 already calibrated and want to
 correct for atmospheric opacity defined by tau.
>>> calmode expandable parameter
 fraction -- edge marker parameter of 'otf' and 'otfraster'.
 Specify a number of OFF integrations (at each
 side of the raster rows in 'otfraster' mode)
 as a fraction of total number of integrations.
 In 'otfraster' mode, number of integrations
 to be marked as OFF, n_off, is determined by
 the following formula,

$$n_off = \text{floor}(\text{fraction} * n),$$

 where n is number of integrations per raster
 row. Note that n_off from both sides will be
 marked as OFF so that twice of specified
 fraction will be marked at most. For example,
 if you specify fraction='10%', resultant
 fraction of OFF integrations will be 20% at
 most.
 In 'otf' mode, n_off is given by,

$$n_off = \text{floor}(\text{fraction} * n),$$

 where n is number of total integrations.
 n_off is used as criterion of iterative marking
 process. Therefore, resulting total number of
 OFFs will be larger than n_off. In practice,

```

fraction is a geometrical fraction of edge region. Thus, if integrations are concentrated on edge region (e.g. some of Lissajous patterns), then resulting `n_off` may be unexpectedly large.

default: '10%'

options: '20%' in string style or float value less than 1.0 (e.g. 0.15).

'auto' is available only for 'otfraster'.

`noff` -- edge marking parameter for 'otfraster'.

It is used to specify a number of OFF scans near edge directly. Value of `noff` comes before setting by fraction. Note that `n_off` from both sides will be marked as OFF so that twice of specified `noff` will be marked at most.

default: -1 (use fraction)

options: any positive integer

`width` -- edge marking parameter for 'otf'.

Pixel width with respect to a median spatial separation between neighboring two data in time. Default will be fine in most cases.

default: 0.5

options: float value

`elongated` -- edge marking parameter for 'otf'.

Set True only if observed area is elongated in one direction.

options: (bool) True, False

default: False

`markonly` -- set True if you want to save data just after edge marking (i.e. uncalibrated data) to see how OFF scans are defined.

options: (bool) True, False

default: False

`plotpointings` -- load plotter and plot pointing directions of ON and OFF scans.

options: (bool) True, False

default: False

`tau` -- the zenith atmospheric optical depth for correction

default: 0.0 (no correction)

`verify` -- interactively verify the results of calibration.

When `verify` = True, for the first six on-source spectra (at max), spectra before and after the calibration are displayed in a plot window. At the prompt there are two choices of action: 'Y' (accept the operation for whole dataset), 'N' (reject the operation and finish task).



Note that when the operation is rejected by 'N',  
no operation is done to the spectrum/spectra.

options: (bool) True,False  
default: False

outfile -- name of output file  
default: '' (<infile>\_cal)

outform -- output file format  
options: 'ASAP','MS2', 'ASCII','SDFITS'  
default: 'ASAP'  
NOTE the ASAP format is easiest for further sd  
processing; use MS2 for CASA imaging.  
If ASCII, then will append some stuff to  
the outfile name

overwrite -- overwrite the output file if already exists  
options: (bool) True,False  
default: False  
NOTE this parameter is ignored when outform='ASCII'

plotlevel -- control for plotting of results  
options: (int) 0, 1, 2, and their negative counterparts  
default: 0 (no plotting)  
example: plotlevel=1; plot calibrated spectra  
plotlevel=2; additionally list data before and after operation.  
plotlevel<0 as abs(plotlevel), e.g.  
-1 => hardcopy of final plot (will be named  
<outfile>\_calspec.eps)

#### DESCRIPTION:

#### ----- HOW TO CHOOSE CALMODE -----

For position switching calibration, the user should choose appropriate calibration mode depending on the data. Use case for each mode is as follows:

'ps': position switch (including OTF) with explicit reference (OFF) scans  
'otf': non-raster OTF scan without explicit OFFs (e.g. Lissajous, double circle, etc.) intends to calibrate fast scan data  
'otfraster': raster OTF scan without explicit OFFs

So, if the data contains explicit reference scans, 'ps' should be used. Otherwise, 'otfraster' and 'otf' are appropriate for raster OTF and non-raster OTF, respectively. In 'otf' and 'otfraster' modes, the task first try to find several integrations near edge as OFF scans, then the

data are calibrated using those OFFs. If the observing pattern is raster, you should use the 'otfraster' mode to calibrate data. Otherwise, the 'otf' mode should be used. For detail about edge marking, see inline help of sd.edgemarker module and its methods. Those modes are designed for OTF observations without explicit OFF scans. However, these modes should work even if explicit reference scans exist. In this case, explicit reference scans will be ignored and scans near edges detected by edge marker will be used as reference.

Except for how to choose OFFs, the procedure to derive calibrated spectra is common for the above three modes. Selected (or preset) OFF integrations are separated by its continuity in time domain, averaged in each segment, then interpolated to timestamps for ON integrations. Effectively, it means that OFF integrations are averaged by each OFF scans for 'ps' mode, averaged by either ends of each raster row for 'otfraster' mode, averaged by each temporal segments of detected edges for 'otf' mode. The formula for calibrated spectrum is

$$T_{\text{sys}} * (\text{ON} - \text{OFF}) / \text{OFF}.$$

The 'fs' mode is for frequency switch calibration. Currently, only GBT frequency switch data is supported.

The 'quotient' mode is special mode for "AT" telescopes, namely ANNF MOPRA. It assumes that observing sequence looks like "target, reference, target, reference,..." and it derives calibrated spectrum by applying the above formula to the pair of "target" and "reference" in order. Reference spectra are not averaged in this mode.

#### ----- FLUX UNIT CONVERSION -----

The task is able to convert flux unit between K and Jy. To do that, fluxunit and its subparameter telescopeparam must be properly set. The fluxunit should be 'Jy' or 'K' depending on what unit input data is and what unit you want to convert. If given fluxunit is different from the unit of input data, unit conversion is performed. The telescopeparam is used to specify conversion factor. There are three ways to specify telescopeparam: 1) set Jy/K conversion factor, 2) set telescope diameter, D, and aperture efficiency, eta, separately, and 3) 'FIX' mode (only change the unit without converting spectral data). If you give telescopeparam as a list, then if the list has a single float it is assumed to be the gain in Jy/K (case 1), if two or more elements they are assumed to be telescope diameter (m) and aperture efficiency respectively (case 2).

See the above parameter description as well as note on 'FIX' mode below for details.

There are two special cases that don't need telescopeparam for unit conversion. Telescope name is obtained from the data.

1) ASAP (sd tool) recognizes the conversion factor (actually D and eta) for the "AT" telescopes, namely ATNF MOPRA telescope, until 2004.

2) The task does know D and eta for GBT telescope.

If you wish to change the fluxunit, by leaving the sub-parameter telescopeparam unset (telescopeparam=''), it will use internal telescope parameters for flux conversion for the data from AT telescopes and it will use an approximate aperture efficiency conversion for the GBT data.

Note that sdcal assumes that the fluxunit is set correctly in the data already. If not, then set telescopeparam='FIX' and it will set the default units to fluxunit without conversion.

Note also that, if the data in infile is an ms from GBT and the default flux unit is missing, this task automatically fixes the default fluxunit to 'K' before the conversion.

-----  
WARNING  
-----

For the GBT raw SDFITS format data as input:

SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

[sdcal2-task.html](#)

## 0.1.94 sdcal2

Requires:

### Synopsis

ASAP SD calibration task

### Description

Task `sdcal2` is an implementation of a calibration scheme like as interferometry, i.e., generate caltables and apply them. Available calibration modes are 'ps', 'otf', 'otfraster', and 'tsys'. Those modes generates caltables for sky or Tsys calibration. Those caltables can be applied to the data by using calmode 'apply'.

The above three calibration modes, 'ps', 'otf', and 'otfraster', generate sky calibration tables. The user should choose appropriate calibration mode depending on the data. Use case for each mode is as follows:

'ps': position switch (including OTF) with explicit reference (OFF) spectra

'otf': non-raster OTF scan without explicit OFFs (e.g. Lissajous, double circle, etc.) intends to calibrate fast scan data 'otfraster': raster OTF scan without explicit OFFs

So, if the data contains explicit reference spectra, 'ps' should be used.

Otherwise, 'otfraster' and 'otf' are appropriate for raster OTF and non-raster OTF, respectively. In 'otf' and 'otfraster' modes, the task first try to find several integrations near edge as OFF spectra, then the data are calibrated using those OFFs. If the observing pattern is raster, you should use the 'otfraster' mode to calibrate data. Otherwise, the 'otf' mode should be used.

For detail about edge marking, see inline help of `sd.edgemarker` module and its methods. Those modes are designed for OTF observations without explicit OFF spectra. However, these modes should work even if explicit reference spectra exist. In this case, these spectra will be ignored and spectra near edges detected by edge marker will be used as reference.

Except for how to choose OFFs, the procedure to derive calibrated spectra is common for the above three modes. Selected (or preset) OFF integrations are separated by its continuity in time domain, averaged in each segment, then interpolated to timestamps for ON integrations. Effectively, it means that OFF integrations are averaged by each OFF spectrum for 'ps' mode, averaged by either ends of each raster row for 'otfraster' mode, averaged by each temporal segments of detected edges for 'otf' mode. The formula for calibrated spectrum is

$$\text{Tsys} * (\text{ON} - \text{OFF}) / \text{OFF}.$$

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                                                                                                                                                                                                            |
| infile     | name of input SD dataset (must be in scantable format)<br>allowed: string<br>Default:                                                                                                                                                                                                                                                      |
| calmode    | SD calibration mode<br>allowed: string<br>Default: ps                                                                                                                                                                                                                                                                                      |
| fraction   | fraction of the OFF data to mark<br>allowed: any<br>Default: variant 10%                                                                                                                                                                                                                                                                   |
| noff       | number of the OFF data to mark<br>allowed: int<br>Default: -1                                                                                                                                                                                                                                                                              |
| width      | width of the pixel for edge detection<br>allowed: double<br>Default: 0.5                                                                                                                                                                                                                                                                   |
| elongated  | whether observed area is elongated in one direction or not<br>allowed: bool<br>Default: False                                                                                                                                                                                                                                              |
| tsysavg    | Whether Tsys is averaged in spectral axis or not<br>allowed: bool<br>Default: False                                                                                                                                                                                                                                                        |
| tsyspw     | list of IF IDs (spectral windows) and their channel ranges of averaging for Tsys calibration.<br>allowed: string<br>Default:                                                                                                                                                                                                               |
| applytable | (List of) sky and/or tsys tables<br>allowed: any<br>Default: variant                                                                                                                                                                                                                                                                       |
| interp     | Interpolation type in time[,freq]. Valid options are "nearest", "linear", "cspline", or any numeric string that indicates an order of polynomial interpolation. You can specify interpolation type for time and frequency separately by joining two of the above options by comma (e.g., "linear,cspline").<br>allowed: string<br>Default: |
| spwmap     | A dictionary indicating IFNO combinations to apply Tsys calibration to target. The key should be IFNO for Tsys calibration and its associated value must be a list of science IFNOs to be applied.<br>allowed: any<br>Default: variant                                                                                                     |
| field      | select data by field IDs and names, e.g. '3C2*' (" = all)<br>allowed: string<br>Default: 469                                                                                                                                                                                                                                               |
| spw        | select data by IF IDs (spectral windows), e.g., '3,5,7' (" = all)<br>allowed: string<br>Default:                                                                                                                                                                                                                                           |
| scan       | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                                                                                                                                                                                                                                           |
| pol        | select data by polarization IDs, e.g. '0,1' (" = all)                                                                                                                                                                                                                                                                                      |

## Returns

void

## Example

Keyword arguments:

infile -- Name of input SD dataset

calmode -- Calibration mode. If you want to generate calibration table or apply existing calibration tables, set calmode to simple string. On the other hand, if you want to calibrate data on-the-fly, you have to set calmode to a composite calmode string separated by comma. So far, sky calibration has three types, 'ps', 'otf', and 'otfraster'. If observation is configured to observe reference position, calmode must be 'ps'. Otherwise, 'otf' or 'otfraster' should be used depending on observing pattern. If observing pattern is raster scan, calmode must be 'otfraster' while 'otf' must be used when observing pattern is non-raster (e.g., Lissajous).

options: 'ps','otf','otfraster','tsys','apply'

default: 'ps'

example: Here is an example for composite calmode.

'ps,apply' (do sky cal and apply)

'ps,tsys,apply' (do sky and Tsys cal and apply)

>>> calmode expandable parameter

fraction -- Edge marker parameter of 'otf' and 'otfraster'. Specify a number of OFF integrations (at each side of the raster rows in 'otfraster' mode) as a fraction of total number of integrations. In 'otfraster' mode, number of integrations to be marked as OFF, n\_off, is determined by the following formula,

$$n_{\text{off}} = \text{floor}(\text{fraction} * n),$$

where n is number of integrations per raster row. Note that n\_off from both sides will be marked as OFF so that twice of specified fraction will be marked at most. For example, if you specify fraction='10%', resultant fraction of OFF integrations will be 20% at most.

In 'otf' mode, n\_off is given by,

$$n\_off = \text{floor}(\text{fraction} * n),$$

where n is number of total integrations.  
n\_off is used as criterion of iterative marking process. Therefore, resulting total number of OFFs will be larger than n\_off. In practice, fraction is a geometrical fraction of edge region. Thus, if integrations are concentrated on edge region (e.g. some of Lissajous patterns), then resulting n\_off may be unexpectedly large.

default: '10%'

options: '20%' in string style or float value less than 1.0 (e.g. 0.15).

'auto' is available only for 'otfraster'.

noff -- Edge marking parameter for 'otfraster'.

It is used to specify a number of OFF spectra near edge directly. Value of noff comes before setting by fraction. Note that n\_off from both sides will be marked as OFF so that twice of specified noff will be marked at most.

default: -1 (use fraction)

options: any positive integer

width -- Edge marking parameter for 'otf'.

Pixel width with respect to a median spatial separation between neighboring two data in time. Default will be fine in most cases.

default: 0.5

options: float value

elongated -- Edge marking parameter for 'otf'.

Set True only if observed area is elongated in one direction.

default: False

tsysavg -- Whether Tsys is averaged in spectral axis or not.

default: False

options: (bool) True, False

tsysppw -- list of IF IDs (spectral windows) and their channel ranges of averaging for Tsys calibration.

It does no effect if you don't want to do Tsys calibration. the user is able to specify channel range for averaging (effective if tsysavg is True).

default: '' (auto-detect tsys spws)

example: tsysppw='3,5,7' (IF IDs 3,5,7; all channels)

tsysppw='<2' (IF IDs less than 2; all channels)



```

 tsysspw='1:0~100' (IF ID1; between channels 0 and 100)
applytable -- List of sky/Tsys calibration tables you want to
 apply.
 default: ''
interp -- Interpolation method in time and frequency axis.
 Set comma separated method strings if you want
 to use different interpolation in time and
 frequency.
 options: 'linear', 'cspline', 'nearest',
 any numeric string indicating an order
 of polynomial.
 default: '' (linear in time and frequency)
 example: 'linear,cspline' (linear in time, cubic
 spline in frequency)
 'linear,3' (linear in time, third order
 polynomial in frequency)
 'nearest' (nearest in time and frequency)
spwmap -- Dictionary defining transfer of Tsys calibration.
 Key must be IFNO for Tsys and its value must be
 a list of IFNOs for science target.
 default: {}
 example: {1: [5,6], 3: [7,8]}
 Tsys in IFNO1 is transferred to IFNO5, 6
 while Tsys in IFNO3 is to IFNO7, 8.
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)
 NOTE this task only supports IF ID selection and ignores channel
 selection.
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all c
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data

```

```

outfile -- Name of output file
 NOTE if you omit, behavior of the task depends on calmode.
 If calmode includes 'apply', then omitting outfile indicates
 that infile is overwritten by the calibrated data. In this case,
 you have to set overwrite to True. If calmode doesn't include
 'apply', omitting outfile indicates that the task will use
 default outfile name based on infile and predefined suffix
 ('_sky' for sky, '_tsys' for Tsys).
 default: '' (<infile>_<suffix> for calibration
 while overwrite infile for apply mode)
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: False
 NOTE this parameter is ignored when outform='ASCII'

```

#### DESCRIPTION:

Task `sdcal2` is an implementation of a calibration scheme like as interferometry, i.e., generate caltables and apply them. Available calibration modes are 'ps', 'otf', 'otfraster', and 'tsys'. Those modes generates caltables for sky or Tsys calibration. Those caltables can be applied to the data by using calmode 'apply'.

The above three calibration modes, 'ps', 'otf', and 'otfraster', generate sky calibration tables. The user should choose appropriate calibration mode depending on the data. Use case for each mode is as follows:

```

'ps': position switch (including OTF) with explicit
 reference (OFF) spectra
'otf': non-raster OTF scan without explicit OFFs
 (e.g. Lissajous, double circle, etc.)
 intends to calibrate fast scan data
'otfraster': raster OTF scan without explicit OFFs

```

So, if the data contains explicit reference spectra, 'ps' should be used. Otherwise, 'otfraster' and 'otf' are appropriate for raster OTF and non-raster OTF, respectively. In 'otf' and 'otfraster' modes, the task first try to find several integrations near edge as OFF spectra, then the data are calibrated using those OFFs. If the observing pattern is raster, you should use the 'otfraster' mode to calibrate data. Otherwise, the 'otf' mode should be used. For detail about edge marking, see inline help of `sd.edgemarker` module and its methods. Those modes are designed for OTF observations without

explicit OFF spectra. However, these modes should work even if explicit reference spectra exist. In this case, these spectra will be ignored and spectra near edges detected by edge marker will be used as reference.

Except for how to choose OFFs, the procedure to derive calibrated spectra is common for the above three modes. Selected (or preset) OFF integrations are separated by its continuity in time domain, averaged in each segment, then interpolated to timestamps for ON integrations. Effectively, it means that OFF integrations are averaged by each OFF spectrum for 'ps' mode, averaged by either ends of each raster row for 'otfraster' mode, averaged by each temporal segments of detected edges for 'otf' mode. The formula for calibrated spectrum is

$$\text{Tsyz} * (\text{ON} - \text{OFF}) / \text{OFF}.$$

You can calibrate data on-the-fly like sdcal task by setting calmode to a composite calmode string separated by comma. For example, calmode='ps,apply' means doing sky calibration and apply it on-the-fly. In this case, caltable is generated as a temporary plain table and will be deleted at the end. Allowed calibration modes in this task is as follows:

```
ps
 generate sky caltable using 'ps' mode
otf
 generate sky caltable using 'otf' mode
otfraster
 generate sky caltable using 'otfraster' mode
tsys
 generate tsys caltable
apply
 apply caltables specified by applytable parameter
ps,apply
 generate temporary sky caltable using 'ps' mode and
 apply it. also apply caltables specified by applytable
ps,tsys,apply
 generate temporary sky caltable using 'ps' mode as well
 as temporary tsys caltable, and apply them.
otf,apply
 generate temporary sky caltable using 'otf' mode and
 apply it. also apply caltables specified by applytable
otf,tsys,apply
 generate temporary sky caltable using 'otf' mode as well
```

```

 as temporary tsys caltable, and apply them.
otfraster,apply
 generate temporary sky caltable using 'otfraster' mode
 and apply it. also apply caltables specified by applytable
otfraster,tsys,apply
 generate temporary sky caltable using 'otfraster' mode
 as well as temporary tsys caltable, and apply them.

```

There are several control parameters for sky/Tsys calibration and application of caltables. See the above parameter description.

In ALMA, Tsys measurement is usually done using different spectral setup from spectral windows for science target. In this case, `sdcal2` transfers Tsys values to science spectral windows in the application stage. To do that, the user has to give a list of spectral windows for Tsys measurement as well as mapping between spectral windows for Tsys measurement and science target. These can be specified by parameters `'tsysspw'` and `'spwmap'`, which are defined as subparameters of `'calmode'`. For example, suppose that Tsys measurements for science windows 17, 19, 21, and 23 are done in spw 9, 11, 13, and 15, respectively. In this case, `tsysspw` and `spwmap` should be specified as follows:

```

tsysspw = '9,11,13,15'
spwmap = {9:[17],11:[19],13:[21],15:[23]}

```

Below is an example of full specification of task parameters for `calmode` of `'ps,tsys,apply'`:

```

default(sdcal2)
infile = 'foo.asap'
calmode = 'ps,tsys,apply'
spw = ''
tsysspw = '9,11,13,15'
spwmap = {9:[17],11:[19],13:[21],15:[23]}
outfile = 'bar.asap'
sdcal2()

```

Note that, in contrast to `applycal` task, `spwmap` must be a dictionary with Tsys spectral window as key and a list of corresponding science spectral window as value. Note also that the parameter `'spw'` should not be used to specify a list of spectral windows for Tsys measurement. It is intended to select data to be calibrated so that the list should contain spectral windows for both science target and Tsys measurement. The task will fail if you use `'spw'` instead of `'tsysspw'`.

For Tsys calibration, the user is able to choose whether Tsys is

averaged in spectral axis or not. If `tsysavg` is `False` (default), resulting `Tsys` is spectral value. On the other hand, when `tsysavg` is `True`, `Tsys` is averaged in spectral axis before output. The channel range for averaging is whole channels by default. If channel range is specified by `tsysspw` string, it is used for averaging. The user can specify channel range with ms selection syntax. For example,

```
tsysspw = '1:0~100'
```

specifies spw 1 for `Tsys` calibration and channel range between channel 0 and 100 for averaging. You can specify more than one ranges per spw.

```
tsysspw = '1:0~100;200~400'
```

In this case, selected ranges are between 0 and 100 plus 200 and 400. Note that even if multiple ranges are selected, the task average whole ranges together and output single averaged value. You can specify multiple spws by separating comma.

```
tsysspw = '1:0~100,3:400~500'
```

Note that specified channel range is ignored if `tsysavg` is `False`.

sdcoadd-task.html

## **0.1.95 sdcoadd**

Requires:

### **Synopsis**

Coadd multiple scantables into one

### **Description**

Task sdcoadd performs co-add multiple single dish spectral data given by a list of spectral data file names in any of the following formats, ASAP, MS2,SDFITS, and ASCII. The units of line flux, the units of spectral axis, frame, and doppler are assumed to be those of the first one in the infiles. The task tries to combine spws according to a tolerance value specified by the parameter freqtol. Default tolerance is '0Hz', which means spws are combined only when spectral setup are the same. Note that, except for first data in the infiles, spw is ignored if there are no corresponding spectral data in the main table.

### **Arguments**

|           |                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                           |
| infile    | list of names of input SD dataset<br>allowed:       stringArray<br>Default:                                               |
| antenna   | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed:       any<br>Default:       variant 0 |
| freqtol   | Frequency shift tolerance for considering data as the same spwid<br>allowed:       any<br>Default:       variant          |
| outfile   | name of output file (See a WARNING in help)<br>allowed:       string<br>Default:                                          |
| outform   | output file format (See a WARNING in help)<br>allowed:       string<br>Default:       ASAP                                |
| overwrite | overwrite the output file if already exists (See a WARNING in help)<br>allowed:       bool<br>Default:       False        |

## Returns

void

## Example

```

Keyword arguments

infile -- list of names of input SD dataset
antenna -- select an antenna name or ID
 default: 0
 example: 'PM03'
 NOTE this parameter is effective only for MS input
freqtol -- Frequency shift tolerance for considering data to be in the same
 spwid. The number of channels must also be the same.
 default: '' == 0 Hz (combine spwid only when frequencies are the same)
 example: freqtol='10MHz' will not combine spwid unless they are

```

within 10 MHz.

Note: This option is useful to combine spectral windows with very slight frequency differences caused by Doppler tracking, for example.

outfile -- name of output file  
 default: '' (<infile>\_coadd)

outform -- format of output file  
 options: 'ASCII','SDFITS','MS','ASAP'  
 default: 'ASAP'  
 example: the ASAP format is easiest for further sd  
 processing; use MS for CASA imaging.  
 If ASCII, then will append some stuff to  
 the outfile name

overwrite -- overwrite the output file if already exists  
 options: (bool) True,False  
 default: False  
 NOTE this parameter is ignored when outform='ASCII'

-----  
 WARNING  
 -----

For the GBT raw SDFITS format data as input:  
 SDtasks are able to handle GBT raw SDFITS format data since the  
 data filler is available. However, the functionality is not well  
 tested yet, so that there may be unknown bugs.



`sdfit-task.html`

## **0.1.96   sdfit**

Requires:

### **Synopsis**

Fit a spectral line

### **Description**

Task `sdfit` is a basic line-fitter for single-dish spectra. It assumes that the spectra have been calibrated in `sdcal` or `sdreduce`.

### **Arguments**

|                |                                                                                                                                      |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Outputs        |                                                                                                                                      |
| xstat          | <p>RETURN ONLY: a Python dictionary of line statistics</p> <p>allowed: any</p> <p>Default: variant</p>                               |
| Inputs         |                                                                                                                                      |
| infile         | <p>name of input SD dataset</p> <p>allowed: string</p> <p>Default:</p>                                                               |
| antenna        | <p>select an antenna name or ID, e.g. 'PM03' (only effective for MS input)</p> <p>allowed: any</p> <p>Default: variant 0</p>         |
| fluxunit       | <p>units of the flux ("=current)</p> <p>allowed: string</p> <p>Default:</p>                                                          |
| telescopeparam | <p>parameters of telescope for flux conversion (see examples in help)</p> <p>allowed: any</p> <p>Default: variant</p>                |
| field          | <p>select data by field IDs and names, e.g. '3C2*' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                                |
| spw            | <p>select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                         |
| restfreq       | <p>the rest frequency, e.g. '1.41GHz' (default unit: Hz) (see examples in help)</p> <p>allowed: any</p> <p>Default: variant</p>      |
| frame          | <p>frequency reference frame ("=current)</p> <p>allowed: string</p> <p>Default:</p>                                                  |
| doppler        | <p>doppler convention ("=current). Effective only when spw selection is in velocity unit.</p> <p>allowed: string</p> <p>Default:</p> |
| scan           | <p>select data by scan numbers, e.g. '21~23' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                                      |
| pol            | <p>select data by polarization IDs, e.g. '0,1' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                                    |
| timeaverage    | <p>average spectra over time (see examples in help)</p> <p>allowed: bool</p> <p>Default: 481 False</p>                               |
| twight         | <p>weighting for time averaging</p> <p>allowed: string</p> <p>Default: tintsys</p>                                                   |
| scanaverage    | <p>average spectra within a scan number (see examples in help)</p> <p>allowed: bool</p> <p>Default: False</p>                        |
| polaverage     | <p>average spectra over polarizations</p>                                                                                            |

## Returns

variant

## Example

```

Keyword arguments

infile -- name of input SD dataset
antenna -- select an antenna name or ID
 default: 0
 example: 'PM03'
 NOTE this parameter is effective only for MS input
fluxunit -- units for line flux
 options: 'K','Jy',''
 default: '' (keep current fluxunit in data)
 WARNING: For GBT data, see description below.
>>> fluxunit expandable parameter
telescopeparam -- parameters of telescope for flux conversion
 options: (str) name or (list) list of gain info
 default: '' (none set)
 example: if telescopeparam='', it tries to get the telescope
 name from the data.
 Full antenna parameters (diameter,ap.eff.) known
 to ASAP are
 'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',
 'CEDUNA','HOBART'. For GBT, it fixes default fluxunit
 to 'K' first then convert to a new fluxunit.
 telescopeparam=[104.9,0.43] diameter(m), ap.eff.
 telescopeparam=[0.743] gain in Jy/K
 telescopeparam='FIX' to change default fluxunit
 see description below
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)/channels
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
```

```

 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all
 spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
>>> spw expandable parameter
 restfreq -- the rest frequency
 available type includes float, int, string, list of float,
 list of int, list of string, and list of dictionary. the
 default unit of restfreq in case of float, int, or string
 without unit is Hz. string input can be a value only
 (treated as Hz) or a value followed by unit for which 'GHz',
 'MHz', 'kHz', and 'Hz' are available.
 a list can be used to set different rest frequencies for
 each IF. the length of list input must be number of IFs.
 dictionary input should be a pair of line name and
 frequency with keys of 'name' and 'value', respectively.
 values in the dictionary input follows the same manner as
 as for single float or string input.
 example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO', 'value':345}]
 frame -- frequency reference frame
 options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
 default: '' (keep current frame in data)
 doppler -- doppler convention (effective only when spw is in
 velocity unit)
 options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
 default: '' (keep current doppler setting in data)
 scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
 pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
 fitfunc -- function for fitting
 options: 'gauss' (Gaussian), 'lorentz' (Lorentzian)
 default: 'gauss'
 fitmode -- mode for fitting
 options: 'auto', 'list', or 'interact'
 default: 'auto'

```

example: 'auto' will use the linefinder to fit for lines  
 using the following parameters  
 'list' will use maskline to define regions to  
 fit for lines with nfit in each  
 'interact' allows adding and deleting mask  
 regions by drawing rectangles on the plot  
 with mouse. Draw a rectangle with LEFT-mouse  
 to ADD the region to the mask and with RIGHT-mouse  
 to DELETE the region.

```
>>> fitmode expandable parameters
thresh -- S/N threshold for linefinder. a single channel S/N ratio
 above which the channel is considered to be a detection.
 default: 5
min_nchan -- minimum number of consecutive channels required to
 pass threshold
 default: 3
avg_limit -- channel averaging for broad lines. a number of
 consecutive channels not greater than this parameter
 can be averaged to search for broad lines.
 default: 4
box_size -- running mean box size specified as a fraction
 of the total spectrum length
 default: 0.2
edge -- channels to drop at beginning and end of spectrum
 default: 0
 example: edge=[1000] drops 1000 channels at beginning AND end.
 edge=[1000,500] drops 1000 from beginning and 500
 from end
```

Note: For bad baselines threshold should be increased,  
 and avg\_limit decreased (or even switched off completely by  
 setting this parameter to 1) to avoid detecting baseline  
 undulations instead of real lines.

```
nfit -- list of number of gaussian/lorentzian lines to fit in in maskline
 region (ignored when fitmode='auto')
 default: 0 (no fitting)
 example: nfit=[1] for single line in single region,
 nfit=[2] for two lines in single region,
 nfit=[1,1] for single lines in each of two regions, etc.
outfile -- name of output file
 default: no output fit file
 example: 'mysd.fit'
overwrite -- overwrite the output file if already exists
 options: (bool) True, False
```

```

 default: False
 plotlevel -- control for plotting of results
 options: 0, 1, 2
 default: 0 (no plotting)
 example: plotlevel=0 no plotting
 plotlevel=1 plots fit
 plotlevel=2 plots fit and residual
 no hardcopy available for fitter
 WARNING: be careful plotting OTF data with lots of fields

```

```

Returns

```

```

a Python dictionary of line statistics
 keys: 'peak', 'cent', 'fwhm', 'nfit'
 example: each value is a list of lists with one list of
 2 entries [fitvalue,error] per component.
 e.g. xstat['peak']=[[234.9, 4.8],[234.2, 5.3]]
 for 2 components.

```

```

DESCRIPTION

```

Task `sdffit` is a basic line-fitter for single-dish spectra. It assumes that the spectra have been calibrated in `sdcal` or `sdreduce`.

Furthermore, it assumes that any selection of scans, IFs, polarizations, and time and channel averaging/smoothing has also already been done (in other `sd` tasks) as there are no controls for these. Note that you can use `sdsave` to do selection, writing out a new scantable.

Note that multiple scans, IFs, and polarizations can in principle be handled, but we recommend that you use `scan`, `field`, `spw`, and `pol` to give a single selection for each fit.

Currently, you can choose Gaussian or Lorentzian profile as a fitting model.

For complicated spectra, `sdffit` does not do a good job of "auto-guessing" the starting model for the fit. We recommend you use `sd.fitter` in the toolkit which has more options, such as fixing components in the fit and supplying starting guesses by hand.

-----  
FITMODE  
-----

As described in the parameter description section, `sdfit` implements three fitting mode, 'auto', 'list', and 'interact'. Only difference between these modes are a method to set initial guess for line fitting. In 'auto' mode, initial guess is automatically set by executing line finder. On the other hand, 'list' and 'interact' allow to set initial guess manually. In these modes, only controllable parameter for the guess is range of the line region and number of lines per region. In 'list' mode, the user must give line region via `spw` parameter by using `ms` selection syntax while number of lines per region can be specified via `nfit` parameter. For example,

```
spw = '17:1500~2500'
nfit = [1]
```

will set line region between channels 1500 and 2500 for `spw` 17, and indicate that there is only one line in this region. Specifying single region with multiple line is also possible but is not recommended. In 'interact' mode, spectral data to be fitted will be displayed with pre-defined line region specified by `spw` parameter. The user is able to customize line region interactively.

-----  
FLUX UNIT CONVERSION  
-----

The task is able to convert flux unit between K and Jy. To do that, `fluxunit` and its subparameter `telescopeparam` must be properly set. The `fluxunit` should be 'Jy' or 'K' depending on what unit input data is and what unit you want to convert. If given `fluxunit` is different from the unit of input data, unit conversion is performed. The `telescopeparam` is used to specify conversion factor. There are three ways to specify `telescopeparam`: 1) set Jy/K conversion factor, 2) set telescope diameter, `D`, and aperture efficiency, `eta`, separately, and 3) 'FIX' mode (only change the unit without converting spectral data). If you give `telescopeparam` as a list, then if the list has a single float it is assumed to be the gain in Jy/K (case 1), if two or more elements they are assumed to be telescope diameter (m) and aperture efficiency respectively (case 2). See the above parameter description as well as note on 'FIX' mode below for details.

There are two special cases that don't need `telescopeparam` for unit conversion. Telescope name is obtained from the data.

- 1) ASAP (sd tool) recognizes the conversion factor (actually D and eta) for the "AT" telescopes, namely ATNF MOPRA telescope, until 2004.
- 2) The task does know D and eta for GBT telescope.  
If you wish to change the fluxunit, by leaving the sub-parameter telescopeparam unset (telescopeparam=''), it will use internal telescope parameters for flux conversion for the data from AT telescopes and it will use an approximate aperture efficiency conversion for the GBT data.

Note that sdcal assumes that the fluxunit is set correctly in the data already. If not, then set telescopeparam='FIX' and it will set the default units to fluxunit without conversion.  
Note also that, if the data in infile is an ms from GBT and the default flux unit is missing, this task automatically fixes the default fluxunit to 'K' before the conversion.

#### ----- AVERAGING OF SPECTRA -----

Task sdfit has two averaging modes, i.e., time and polarization average.

When timeaverage=True, spectra are averaged over time for each IF (spectral window), polarization, and beam, independently. Note that, by default (scanaverage=False), timeaverage=True averages spectra irrespective of scan IDs.

It is possible to average spectra separately for each scan ID by setting a sub-parameter scanaverage=True.

For example, the combination of parameters: scan='0~2', timeaverage=True, and scanaverage=False: averages spectra in scan ID 0 through 2 all together to a spectrum,

scanaverage=True : averages spectra per scan ID and end up with three spectra from scan 0, 1, and 2.

When polaverage=True, spectra are averaged over polarization for each IF (spectral window) and beam. Note that, so far, time averaging is automatically switched on when polaverage is set to True. This behavior is not desirable and will be discarded in future.

#### ----- WARNING -----

For the GBT raw SDFITS format data as input:

SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.



sdflag-task.html

## 0.1.97 sdflag

Requires:

### Synopsis

ASAP SD spectral spectral/row flagging task

### Description

Task sdflag performs either interactive or non-interactive channel/row based flagging on spectra.

Currently, there are three ways of non-interactive flagging available: (1) channel or row based flagging by selecting spectra by field, lists of scan numbers, IF numbers, and polarization indices in mode='manual', (2) channel based flagging by specifying a range of spectral values in mode='clip', and (3) row based flagging by specifying a list of row numbers in mode='rowid'. Note this is an EXPERT mode since it might not be straight forward for general users to select data by row IDs in scantable.

In mode='manual', the channel based flagging are invoked when spw parameter contains channel range selection. Otherwise, the whole channels are flagged for the selected spectra. Note channel range selection by spw parameter has effect only in mode='manual'.

Interactive flagging is available when mode='interactive'. The available ways of interactive flagging include: (1) row based flagging by selecting 'panel' and (2) channel based flagging by selecting 'region's of channels on Flag plotter. See the cookbook for details of how to select channel regions and spectra on the plotter.

NOTE the task sdflag only modifies flag information, FLAGROW and FLAGTRA, in the input scantable. This task keeps all records in input dataset. Data selection parameters are used for selecting data to modify flag information.

If plotlevel<sub>i</sub>=1, the task asks you if you really apply the flags before it is actually written to the data with a plot indicating flagged regions.

WARNING for overwrite option: Be sure 'outform' is the same as data format of input file when you overwrite it. The default value of the option 'overwrite' is True in this task, thereby the current dataset (infile) is overwritten unless a different file name is set to outfile. There is a known issue in overwriting infile. If 'outform' differs to the data format of infile, the data is overwritten with the new data format (specified by 'outform') and the data in the original format will be lost.

## Arguments

|             |                                                                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                                                                                     |
| infile      | name of input SD dataset<br>allowed: string<br>Default:                                                                                                             |
| antenna     | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0                                                       |
| mode        | mode of data selection and flag operation<br>allowed: string<br>Default: manual                                                                                     |
| unflag      | unflag selected data (False: flag, True: unflag)<br>allowed: bool<br>Default: False                                                                                 |
| field       | select data by field IDs and names, e.g. '3C2*' (" = all)<br>allowed: string<br>Default:                                                                            |
| spw         | select data by IF IDs (spectral windows), e.g., '3,5,7' (" = all)<br>allowed: string<br>Default:                                                                    |
| timerange   | select data by time range, e.g. '09:14:0~09:54:0' (" = all) (see examples in help)<br>allowed: string<br>Default:                                                   |
| scan        | select data by scan numbers, e.g. '21~23' (" = all)<br>allowed: string<br>Default:                                                                                  |
| pol         | select data by polarization IDs, e.g. '0,1' (" = all)<br>allowed: string<br>Default:                                                                                |
| beam        | select data by beam IDs, e.g. '0,1' (" = all)<br>allowed: string<br>Default:                                                                                        |
| restfreq    | the rest frequency, '1.41GHz' (default unit: Hz). Effective only when spw selection is in velocity unit. (see examples in help)<br>allowed: any<br>Default: variant |
| frame       | frequency reference frame ("=current) Effective only when spw selection is in velocity or frequency unit.<br>allowed: string<br>Default:                            |
| doppler     | doppler convention ("=current). Effective only when spw selection is in velocity unit.<br>allowed: string<br>Default: 490                                           |
| clipminmax  | range of data that will NOT be flagged<br>allowed: any<br>Default: variant                                                                                          |
| clipoutside | clip outside the range, or within it<br>allowed: any<br>Default: variant True                                                                                       |
| showflagged | show flagged data (in gray) on plots                                                                                                                                |

## Returns

void

## Example

Keyword arguments:

infile -- name of input SD dataset

antenna -- select an antenna name or ID

default: 0

example: 'PM03'

NOTE this parameter is effective only for MS input

mode -- type of flag operation

options: (str) 'manual', 'clip', 'interactive', 'rowid'

default: 'manual'

```
>>> common data selection parameters for all modes except mode='rowid'
```

field -- select data by field IDs and names

default: '' (use all fields)

example: field='3C2\*' (all names starting with 3C2)

field='0,4,5~7' (field IDs 0,4,5,6,7)

field='0,3C273' (field ID 0 or field named 3C273)

this selection is in addition to the other selections to data

spw -- select data by IF IDs (spectral windows)/channels

NOTE channel range selection is valid only in mode='manual'

default: '' (use all IFs and channels)

example: spw='3,5,7' (IF IDs 3,5,7; all channels)

spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)

spw='115GHz' (IF IDs with the center frequencies in range 30-45GHz)

spw='0:5~61' (IF ID 0; channels 5 to 61)

spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)

spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3

spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and

this selection is in addition to the other selections to data

timerange -- select data by time range

default: '' (use all)

example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'

Note: YYYY/MM/DD can be dropped as needed:

timerange='09:14:00~09:54:00' # this time range

timerange='09:44:00' # data within one integration of time

timerange='>10:24:00' # data after this time

timerange='09:44:00+00:13:00' #data 13 minutes after time

this selection is in addition to the other selections to data

```

scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
beam -- select data by beam IDs
 default: '' (use all beams)
 example: beam='0,1' (beam IDs 0,1)
 this selection is in addition to the other selections to data

>>> common data parameters for all modes except mode='interactive'
 unflag -- flag or unflag
 default: False (flag selected data)
 options: (bool) True, False

>>> mode='manual' expandable parameters
 restfreq -- the rest frequency (effective only when spw selection is in
 velocity unit.)
 available type includes float, int, string, list of float,
 list of int, list of string, and list of dictionary. the
 default unit of restfreq in case of float, int, or string
 without unit is Hz. string input can be a value only
 (treated as Hz) or a value followed by unit for which 'GHz',
 'MHz', 'kHz', and 'Hz' are available.
 a list can be used to set different rest frequencies for
 each IF. the length of list input must be number of IFs.
 dictionary input should be a pair of line name and
 frequency with keys of 'name' and 'value', respectively.
 values in the dictionary input follows the same manner as
 as for single float or string input.
 example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]

 frame -- frequency reference frame (effective only when spw selection is in
 velocity or frequency unit.)
 options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
 default: '' (keep current frame in data)

 doppler -- doppler convention (effective only when spw is in
 velocity unit)
 options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
 default: '' (keep current doppler setting in data)

```

```

>>> mode='clip' expandable parameters
clipminmax -- range of data that will NOT be flagged
 default: [] (means no clip operation)
 example: [0.0,1.5]
clipoutside -- clip OUTSIDE the range ?
 options: (bool)True,False
 default: True
 example: clipoutside=False means flag data WITHIN the range.
>>> mode='interactive' expandable parameters
showflagged -- show flagged data on plots
 options: (bool) True, False
 default: False
>>> mode='rowid' expandable parameters
row -- select data by row IDs to apply flag/unflag in the input scannable
 Note, this parameter is effective only when one or more row
 IDs are given explicitly
 default: '' (means no selection)
 example: '200~300,400~500' (rows 200 to 300 and 400 to 500)
outfile -- name of output file
 default: ''
 Note: by default (outfile=''), actual output file name is set as follows:
 (1) if overwrite=True (default), infile (input) will be overwritten.
 WARNING: If the formats of input and output files are different,
 this causes complete loss of input file.
 (2) if overwrite=False, outfile will be <infile>_f.
outform -- output file format
 options: 'ASAP','MS2', 'ASCII','SDFITS'
 default: 'ASAP'
 NOTE the ASAP format is easiest for further sd
 processing; use MS2 for CASA imaging.
 If ASCII, then will append some stuff to
 the outfile name
 WARNING: Be sure outform is same as the input file format when you
 overwrite the input file by overwrite=True and outfile='' (default).
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: True
 WARNING: input file is overwritten if overwrite=True and outfile='' (default).
 This causes the complete loss of input file if the formats of
 input and output files are different.
plotlevel -- control for plotting of results
 options: (int) 0, 1, 2, and their negative counterparts
 default: 0 (no plotting)
 example: plotlevel=1; plot spectra and flagged channels before and after
 current operation. asked if you accept the flag for each
 spw. also, the first spectrum after the operation is plotted.

```

plotlevel=2; additionally list scantable before and after operation.  
plotlevel<0 as abs(plotlevel), e.g.  
-1 => hardcopy of final plot (will be named  
<outfile>\_flag.eps)

sdflagmanager-task.html

## 0.1.98 sdflagmanager

Requires:

### Synopsis

ASAP SD task to manipulate flag version files

### Description

Task sdflagmanager enables users to save the current flag information (both channel and row flags) in the given SD dataset out to a separate 'flag version file'. In the current implementation, sdflagmanager calls flagmanager internally, so these flag version files are copies of the flag columns for a measurement set actually. They can be restored to the data set to obtain a previous flag version. Users can also list, delete and rename flag version files using sdflagmanager. It is wise to save a flagversion at the beginning or after serious editing.

### Arguments

| Inputs      |                                                                                                |
|-------------|------------------------------------------------------------------------------------------------|
| infile      | name of input SD dataset (ASAP scantable)<br>allowed: string<br>Default:                       |
| mode        | operation mode<br>allowed: string<br>Default: list                                             |
| versionname | Flag version name<br>allowed: string<br>Default:                                               |
| oldname     | Flag version to rename<br>allowed: string<br>Default:                                          |
| comment     | Short description of a versionname<br>allowed: string<br>Default:                              |
| merge       | Merge option: replace will save or over-write the flags<br>allowed: string<br>Default: replace |



## Returns

void

## Example

Keyword arguments:

```
infile -- name of input SD dataset
 default: ''
 example: infile='ngc5921.asap'
mode -- Flag version operation
 default: 'list'; to list existing flagtables
 'save' to copy flag columns of infile to a flag file
 'restore' to place the specified flag file into infile
 'delete' to delete the specified flag file
 'rename' to rename the specified flag file
>>> mode expandable parameters
 versionname -- Flag version name
 default: none; example: versionname='original_data'
 No imbedded blanks in the versionname
 comment -- Short description of a versionname, when mode is 'save'
 or 'rename'
 default: ''
 example: comment='Clip above 1.85'
 comment = versionname
 oldname -- When mode='rename', the flag file to rename
 merge -- merge operation
 options: 'or','and', but not recommended for now.
```

sdgrid-task.html

## **0.1.99 sdgrid**

Requires:

### **Synopsis**

SD gridding task

### **Description**

Task sdgrid performs spatial gridding according to the user specification of spatial grid, convolution function, etc. For grid configuration, the task supplements necessary information by referring input data if any of gridding parameter ('npix', 'cell', or 'center') is not specified by the user. If 'center' is default value (empty string), central position of the grid will be set to the center of observed area, i.e.  $x=0.5*(x_{\max}+x_{\min})$ ,  $y=0.5*(y_{\max}+y_{\min})$ . If either 'cell' or 'npix' is set, unspecified one will be calculated from the others. In that case, total extent of the grid will be set to cover all observed position. If neither 'cell' nor 'npix' is set, cell size will be set to 1.0 arcmin and number of pixel will be calculated based on that cell size. Currently, only J2000 frame is supported.

### **Arguments**

|              |                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------|
| Inputs       |                                                                                                                |
| infile       | a list of names of input SD datasets<br>allowed: any<br>Default: variant ""                                    |
| antenna      | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant -1 |
| spw          | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default: -1               |
| scan         | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                               |
| pol          | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                             |
| gridfunction | gridding function for imaging<br>allowed: string<br>Default: BOX                                               |
| convsupport  | truncate of convolution kernel<br>allowed: int<br>Default: -1                                                  |
| truncate     | truncation radius of convolution kernel<br>allowed: any<br>Default: variant -1                                 |
| gwidth       | HWHM for gaussian<br>allowed: any<br>Default: variant -1                                                       |
| jwidth       | c-parameter for jinc function<br>allowed: any<br>Default: variant -1                                           |
| weight       | weight type<br>allowed: string<br>Default: UNIFORM                                                             |
| clipminmax   | clip minimum and maximum values during gridding<br>allowed: bool<br>Default: False                             |
| outfile      | name of output file<br>allowed: string<br>Default:                                                             |
| overwrite    | overwrite the output file if already exists [True, False]<br>allowed: bool<br>Default: False                   |
| npix         | number of pixels in x and y, symmetric for single value<br>allowed: any<br>Default: variant -1                 |
| cell         | x and y cell size. default unit arcsec<br>allowed: any<br>Default: variant                                     |
| center       | Image center<br>allowed: any<br>Default: variant                                                               |

## Returns

void

## Example

Keyword arguments:

**infile** -- a list of names of input SD datasets. in case input is a single dataset, its name can be given as a string.

example: 'testimage.asap'  
['testimage1.asap', 'testimage2.asap']

**antenna** -- select an antenna name or ID

default: -1  
example: 'PM03'

NOTE this parameter is effective only for MS input

**spw** -- select data by IF IDs (spectral windows)

NOTE this task only supports IF ID selection and ignores channel selection.

default: '-1' (only process IFNO in the first row)  
example: spw='3,5,7' (IF IDs 3,5,7; all channels)  
spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)  
spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)

this selection is in addition to the other selections to data

**scan** -- select data by scan numbers

default: '' (use all scans)  
example: scan='21~23' (scan IDs 21,22,23)

this selection is in addition to the other selections to data

**pol** -- select data by polarization IDs

default: '' (use all polarizations)  
example: pol='0,1' (polarization IDs 0,1)

this selection is in addition to the other selections to data

**gridfunction** -- gridding function

options: 'BOX' (Box-car), 'SF' (Spheroidal),  
'GAUSS' (Gaussian), 'PB' (Primary-beam),  
'GJINC' (Gaussian\*Jinc)

default: 'BOX'  
example: 'SF'

>>> gridfunction expandable parameter:

**convsupport** -- convolution support for 'SF'  
default: -1 (use default for each gridfunction)  
example: 3

**truncate** -- truncation radius of convolution kernel.  
effective only for 'GAUSS' and 'GJINC'.

```

 default: '-1' (use default for each gridfunction)
 example: 3, '20arcsec', '3pixel'
gwidth -- HWHM for gaussian. Effective only for
 'GAUSS' and 'GJINC'.
 default: '-1' (use default for each gridfunction)
 example: 3, '20arcsec', '3pixel'
jwidth -- Width of jinc function. Effective only for
 'GJINC'.
 default: '-1' (use default for each gridfunction)
 example: 3, '20arcsec', '3pixel'
weight -- weight type (both lower-case and upper-case are acceptable)
 options: 'UNIFORM',
 'TSYS' (1/Tsys**2 weighted)
 'TINT' (integration time weighted)
 'TINTSYS' (Tint/Tsys**2)
 default: 'UNIFORM'
clipminmax -- do min/max clipping if True
 default: False
outfile -- name of output file
 default: '' (outfile will be set to infile[0]+'grid')
 example: 'mydata.asap.grid'
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: False
 NOTE this parameter is ignored when outform='ASCII'
npix -- x and y image size in pixels, symmetric for single value
 default: -1 (automatically calculated from cell size and
 the data)
 example: npix=200 (equivalent to [200,200])
cell -- x and y cell size. default unit arcsec
 default: '' (automatically calculated from npix if it is
 set, otherwise '1.0arcmin')
 example: cell=['0.2arcmin', '0.2arcmin']
 cell='0.2arcmin' (equivalent to example above)
 cell=12.0 (interpreted as '12.0arcsec'='0.2arcmin')
center -- grid center
 default: '' (automatically calculated from the data)
 example: 'J2000 13h44m00 -17d02m00'
 ['05:34:48.2', '-05.22.17.7'] (in J2000 frame)
 [1.46, -0.09] (interpreted as radian in J2000 frame)
plot -- Plot result or not
 default: False (not plot)
 example: if True, result will be plotted

```

DESCRIPTION:

The sdgrid task performs spatial gridding according to the user specification of spatial grid, convolution function, etc.

For grid configuration, the task supplements necessary information by referring input data if any of gridding parameter ('npix', 'cell', or 'center') is not specified by the user. If 'center' is default value (empty string), central position of the grid will be set to the center of observed area, i.e.  $x=0.5*(x_{\max}+x_{\min})$ ,  $y=0.5*(y_{\max}+y_{\min})$ . If either 'cell' or 'npix' is set, unspecified one will be calculated from the others. In that case, total extent of the grid will be set to cover all observed position. If neither 'cell' nor 'npix' is set, cell size will be set to 1.0 arcmin and number of pixel will be calculated based on that cell size.

Currently, only J2000 frame is supported.

The parameter gridfunction sets gridding function for imaging. Currently, the task supports 'BOX' (Box-car), 'SF' (Prolate Spheroidal Wave Function), 'GAUSS' (Gaussian), 'GJINC' (Gaussian\* Jinc), where  $Jinc(x) = J_1(\pi*x/c)/(\pi*x/c)$  with a first order Bessel function  $J_1$ , and 'PB' (Primary Beam, not implemented yet). For 'PB', correct antenna informations should be included in input file.

There are four subparameters for gridfunction: convsupport, truncate, gwidth, and jwidth. The convsupport is an integer specifying cut-off radius for 'SF' in units of pixel. By default (convsupport=-1), the cut-off radius is set to 3 pixels. The truncate is a cut-off radius for 'GAUSS' or 'GJINC'. It accepts integer, float, and string values of numeric plus unit. Allowed units are angular units such as 'deg', 'arcmin', 'arcsec', and 'pixel'. Default unit is 'pixel' so that string without unit or numerical values (integer or float) will be interpreted as radius in pixel. Default value for truncate, which is used when negative radius is set, is  $3*HWHM$  for 'GAUSS' and radius at first null for 'GJINC'. The gwidth is the HWHM of gaussian for 'GAUSS' and 'GJINC'. Default value is  $\sqrt{\log(2)}$  pixel for 'GAUSS' and  $2.52*\sqrt{\log(2)}$  pixel for 'GJINC'. The jwidth specifies width of the jinc function (parameter 'c' in the definition above). Default is 1.55 pixel. Both gwidth and jwidth allows integer, float, or string of numeric plus unit. Default values for gwidth and jwidth are taken from Mangum et al. (2007). Formula for 'GAUSS' and 'GJINC' are taken from Table 1 in the paper, and are written as below using gwidth and jwidth:

GAUSS:  $\exp[-(|r|/gwidth)**2]$

GJINC:  $J_1(\pi|r|/jwidth)/(\pi|r|/jwidth) * \exp[-(|r|/gwidth)^2]$

Boolean parameter 'plot' controls whether gridded result is plotted or not. If True, color map of gridded data will be shown. Pixel center and observed position are overlayed as blue dot and red dot, respectively. Currently, channel averaged value will be plotted.

Reference: Mangum, et al. 2007, A&A, 474, 679-687

sdimaging-task.html

### 0.1.100 sdimaging

Requires:

#### Synopsis

SD task: imaging for total power and spectral data

#### Description

Task sdimaging creates an image from input single-dish data sets. The input can be either total power and spectral data. Currently, this task directly accesses the Measurement Set data because of the data access efficiency. So it differs from other single-dish tasks that mostly operate on the ASAP scantable data format.

The coordinate of output image is defined by four axes, i.e., two spatial axes, frequency and polarization axes. By default, spatial coordinate of image is defined so that the all pointing directions in POINTING tables of input data sets are covered with the cell size, 1/3 of FWHM of primary beam of antennas in the first MS. Therefore, it is often easiest to leave spatial definitions at the default values. It is also possible to define spatial axes of the image by specifying the image center direction (phasecenter), number of image pixel (imsize) and size of the pixel (cell). The frequency coordinate of image is defined by three parameters, the number of channels (nchan), the channel id/frequency/velocity of the first channel (start), and channel width (width). There are three modes available to define unit of start and width, i.e., 'channel' (use channel indices), 'frequency' (use frequency unit, e.g., 'GHz'), and 'velocity' (use velocity unit, e.g., 'km/s'). By default, nchan, start, and width are defined so that all selected spectral windows are covered with the channel width equal to separation of first two channels selected. Finally, polarizations of image is defined by stokes parameter or polarization. For example, stokes='XXYY' produces an image cube with each plane contains the image of one of the polarizations, while stokes='I' produces a 'total intensity' or Stokes I image.

The task also supports various grid function (convolution kernel) to weight spectra. See description below for details of gridfunction available.

#### Arguments



|              |                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs       |                                                                                                                                          |
| infile       | <p>a list of names of input SD Measurementsets (only MS is allowed for this task)</p> <p>allowed:        stringArray</p> <p>Default:</p> |
| outfile      | <p>name of output image</p> <p>allowed:        string</p> <p>Default:</p>                                                                |
| overwrite    | <p>overwrite the output file if already exists [True, False]</p> <p>allowed:        bool</p> <p>Default:        False</p>                |
| field        | <p>select data by field IDs and names, e.g. '3C2*' ('=all)</p> <p>allowed:        any</p> <p>Default:        variant</p>                 |
| spw          | <p>select data by IF IDs (spectral windows), e.g. '3,5,7' ('=all)</p> <p>allowed:        any</p> <p>Default:        variant</p>          |
| antenna      | <p>select data by antenna names or IDs, e.g. 'PM03' ('= all antennas)</p> <p>allowed:        any</p> <p>Default:        variant</p>      |
| scan         | <p>select data by scan numbers, e.g. '21~23' ('=all)</p> <p>allowed:        any</p> <p>Default:        variant</p>                       |
| intent       | <p>select data by observational intent, e.g. '*ON_SOURCE*' ('=all)</p> <p>allowed:        any</p> <p>Default:        variant</p>         |
| mode         | <p>spectral gridding type</p> <p>allowed:        string</p> <p>Default:        channel</p>                                               |
| nchan        | <p>number of channels (planes) in output image (-1=all)</p> <p>allowed:        int</p> <p>Default:        -1</p>                         |
| start        | <p>start of output spectral dimension, e.g. '0', '110GHz', '-20km/s'</p> <p>allowed:        any</p> <p>Default:        variant 0</p>     |
| width        | <p>width of output spectral channels</p> <p>allowed:        any</p> <p>Default:        variant 1</p>                                     |
| veltype      | <p>velocity definition</p> <p>allowed:        string</p> <p>Default:        radio</p>                                                    |
| outframe     | <p>velocity frame of output image ('=current frame or LSRK for multiple-MS inputs)</p> <p>allowed:        string</p> <p>Default:</p>     |
| gridfunction | <p>gridding function for imaging (see description in help)</p> <p>allowed:        string</p>                                             |

## Returns

void

## Example

Keyword arguments:

infile -- a list of names of input SD Measurementsets

example: 'm100.PM01.ms'

['m100.PM01.ms','m100.PM03.ms']; multiple MSes

outfile -- name of output image

default: ''

example: 'mySDimage.im'

overwrite -- overwrite the output file if already exists

options: (bool) True,False

default: False (do NOT overwrite)

example: if True, existing file will be overwritten

field -- select data by field IDs and names

If field string is a non-negative integer, it is assumed to be a field index otherwise, it is assumed to be a field name

default: '' (use all fields)

example: field='3C2\*' (all names starting with 3C2)

field='0,4,5~7' (field IDs 0,4,5,6,7)

field='0,3C273' (field ID 3 or field named 3C273)

For multiple MS input, a list of field strings can be used:

field = ['0~2','0~4'] (field ids 0-2 for the first MS and 0-4 for the second)

field = '0~2' (field ids 0-2 for all input MSes)

this selection is in addition to the other selections to data

spw -- select data by spectral window IDs/channels

NOTE: channels de-selected here will contain all zeros if

selected by the parameter mode subparameters.

default: '' (use all IFs and channels)

example: spw='3,5,7' (IF IDs 3,5,7; all channels)

spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)

spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)

spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)

spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)

spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)

spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)

For multiple MS input, a list of spw strings can be used:

spw=['0','0~3'] (spw ids 0 for the first MS and 0-3 for the second)

```

 spw='0~3' (spw ids 0-3 for all input MSes)
 this selection is in addition to the other selections to data
antenna -- select data by antenna names or IDs
 If antenna string is a non-negative integer, it is
 assumed to be an antenna index, otherwise, it is
 considered an antenna name.
 default: '' (all baselines, i.e. all antenna in case of auto data)
 example: antenna='PM03'
 For multiple MS input, a list of antenna strings can be used:
 antenna=['5','6'] (antenna id5 for the first MS and 6 for the second)
 antenna='5' (antenna index 5 for all input MSes)
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 For multiple MS input, a list of scan strings can be used:
 scan=['0~100','10~200'] (scan ids 0-100 for the first MS
 and 10-200 for the second)
 scan='0~100' (scan ids 0-100 for all input MSes)
 this selection is in addition to the other selections to data
intent -- select data by observational intent, also referred to as 'scan intent'
 default: '' (use all scan intents)
 example: intent='*ON_SOURCE*' (any valid scan-intent expression accepted by the MSS)
 For multiple MS input, a list of scan-intent expressions can be used:
 intent=['ON_SOURCE','CALIBRATE_BANDPASS'] (scan intent ON_SOURCE for the first
 and CALIBRATE_BANDPASS for the second)
 this selection is in addition to the other selections to data
mode -- spectral gridding type
 options: 'channel', 'velocity', 'frequency'
 default: 'channel'
>>> mode expandable parameters
nchan -- Total number of channels in the output image.
 default: -1; Automatically selects enough channels to cover
 data selected by 'spw' consistent with 'start' and 'width'.
 It is often easiest to leave nchan at the default value.
 example: nchan=100
start -- First channel, velocity, or frequency.
 For mode='channel'; This selects the channel index number
 from the MS (0 based) that you want to correspond to the
 first channel of the output cube. The output cube will be
 in frequency space with the first channel having the
 frequency of the MS channel selected by start. start=0
 refers to the first channel in the first selected spw, even
 if that channel is de-selected in the spw parameter.
 Channels de-selected by the spw parameter will be filled with
 zeros if included by the start parameter. For example,

```

```

 spw=3~8:3~100 and start=2 will produce a cube that starts on
 the third channel (recall 0 based) of spw index 3, and the
 first channel will be blank.
 default: '' (the first input channel of first input spw)
 example: start=100 (mode='channel')
 start='22.3GHz' (mode='frequency')
 start='5.0km/s' (mode='velocity')
width -- Output channel width
 For mode='channel', default=1; width>1 indicates channel averaging
 example: width=4.
 For mode= 'velocity' or 'frequency', default=''; width of
 first input channel, or more precisely, the difference
 in frequencies between the first two selected channels.
 -- For example if channels 1 and 3 are selected with spw,
 then the default width will be the difference between their
 frequencies, and not the width of channel 1.
 -- Similarly, if the selected data has uneven channel-spacing,
 the default width will be picked from the first two selected
 channels. In this case, please specify the desired width.
 When specifying the width, one must give units
 examples: width='1.0km/s', or width='24.2kHz'.
 Setting width>0 gives channels of increasing frequency for
 mode='frequency', and increasing velocity for mode='velocity'.
veltype -- Velocity reference frame of output image
 Options: 'radio','optical','true','relativistic'
 default: 'radio'
outframe -- velocity reference frame of output image
 Options: '', 'LSRK', 'LSRD', 'BARY', 'GEO', 'TOPO', 'GALACTO',
 'LGROUP', 'CMB'
 default: ''; same as input data or 'LSRK' for multiple-MS inputs
 example: frame='bary' for Barycentric frame
gridfunction -- gridding function for imaging
 options: 'BOX' (Box-car), 'SF' (Spheroidal),
 'PB' (Primary-beam), 'GAUSS' (Gaussian),
 'GJINC' (Gaussian*Jinc)
 default: 'BOX'
 example: 'SF'
>>> gridfunction expandable parameter:
convsupport -- convolution support for 'SF'
 default: -1 (use default for each gridfunction)
 example: 3
truncate -- truncation radius of convolution kernel.
 effective only for 'GAUSS' and 'GJINC'.
 default: '-1' (use default for each gridfunction)
 example: 3, '20arcsec', '3pixel'
gwidth -- HWHM for gaussian. Effective only for

```

```

 'GAUSS' and 'GJINC'.
 default: '-1' (use default for each gridfunction)
 example: 3, '20arcsec', '3pixel'
jwidth -- Width of jinc function. Effective only for
 'GJINC'.
 default: '-1' (use default for each gridfunction)
 example: 3, '20arcsec', '3pixel'
imsize -- x and y image size in pixels, symmetric for single value
 default: [] (=cover all pointings in MS)
 example: imsize=200 (equivalent to [200,200])
cell -- x and y cell size. default unit arcmin
 default: '' (= 1/3 of FWHM of primary beam)
 example: cell=['0.2arcmin', '0.2arcmin']
 cell='0.2arcmin' (equivalent to example above)
phasecenter -- image phase center: direction measure or field ID
 default: '' (= the center of pointing directions in
 POINTING table of infiles)
 example: 6 (field id), 'J2000 13h44m00 -17d02m00',
 'AZEL -123d48m29 15d41m41'
ephemsrcname -- ephemeris source name for moving source (solar sytem objects)
 default: '' (none)
 if the source name in the data matches one of the solar system
 objects known by CASA, the imaging realigns the data by
 correcting shifts of the source during observation,
 so that the source appears to be fixed in the image.
 examples: 'MERCURY', 'VENUS', 'MARS', 'JUPITER', 'SATURN',
 'URANUS', 'NEPUTUNE', 'PLUTO', 'SUN', 'MOON'
pointingcolumn -- pointing data column to use
 option: 'direction', 'target', 'pointing_offset', 'source_offset', 'encoder'
 default: 'direction'
restfreq -- specify rest frequency to use for output image
 default: '' (refer input data)
 example: 1.0e11, '100GHz'
stokes -- stokes parameters or polarization types to image
 default: '' (use all polarizations)
 example: 'XX'
minweight -- Minimum weight ratio to the median of weight used in
 weight correction and weight based masking
 default: 0.1
 example: minweight = 0.

```

```

Gridding Kernel

```

The parameter gridfunction sets gridding function (convolution kernel)

for imaging. Currently, the task supports 'BOX' (Box-car), 'SF' (Prolate Spheroidal Wave Function), 'GAUSS' (Gaussian), 'GJINC' (Gaussian\*Jinc), where  $Jinc(x) = J_1(\pi*x/c)/(\pi*x/c)$  with a first order Bessel function  $J_1$ , and 'PB' (Primary Beam). For 'PB', correct antenna informations should be included in input file.

There are four subparameters for gridfunction: convsupport, truncate, gwidth, and jwidth. The convsupport is an integer specifying cut-off radius for 'SF' in units of pixel. By default (convsupport=-1), the cut-off radius is set to 3 pixels. The truncate is a cut-off radius for 'GAUSS' or 'GJINC'. It accepts integer, float, and string values of numeric plus unit. Allowed units are angular units such as 'deg', 'arcmin', 'arcsec', and 'pixel'. Default unit is 'pixel' so that string without unit or numerical values (integer or float) will be interpreted as radius in pixel. Default value for truncate, which is used when negative radius is set, is 3\*HWHM for 'GAUSS' and radius at first null for 'GJINC'. The gwidth is the HWHM of gaussian for 'GAUSS' and 'GJINC'. Default value is  $\sqrt{\log(2)}$  pixel for 'GAUSS' and  $2.52*\sqrt{\log(2)}$  pixel for 'GJINC'. The jwidth specifies width of the jinc function (parameter 'c' in the definition above). Default is 1.55 pixel. Both gwidth jwidth allows integer, float, or string of numeric plus unit. Default values for gwidth and jwidth are taken from Mangum et al. (2007). Formula for 'GAUSS' and 'GJINC' are taken from Table 1 in the paper, and are written as below using gwidth and jwidth:

GAUSS:  $\exp[-\log(2)*(|r|/gwidth)**2]$

GJINC:  $J_1(\pi*|r|/jwidth)/(\pi*|r|/jwidth)$   
 $* \exp[-\log(2)*(|r|/gwidth)^2]$

Reference: Mangum, et al. 2007, A&A, 474, 679-687

-----  
Mask in Output Image  
-----

The parameter minweight defines a threshold of weight values to mask. The pixels in outfile whose weight is smaller than minweight\*median(weight) are masked out. The task also creates a weight image with the name outfile.weight.

sdimprocess-task.html

### **0.1.101   sdimprocess**

Requires:

#### **Synopsis**

Task for single-dish image processing

#### **Description**

Task sdimprocess is used to remove a scanning noise that appears as a striped noise pattern along the scan direction in a raster scan data.

By default, the scanning noise is removed by using the FFT-based 'Basket-Weaving' method (Emerson & Grave 1988) that requires multiple images that observed exactly the same area with different scanning direction. If only one image is available, the 'Pressed-out' method (Sofue & Reich 1979) can be used to remove the scanning effect.

#### **Arguments**

|               |                                                                                            |
|---------------|--------------------------------------------------------------------------------------------|
| <b>Inputs</b> |                                                                                            |
| infile        | list of name of input SD images (FITS or CASA image)<br>allowed: any<br>Default: variant " |
| mode          | image processing mode<br>allowed: string<br>Default:                                       |
| numpoly       | order of polynomial fit for Pressed-out method<br>allowed: int<br>Default: 2               |
| beamsize      | beam size for Pressed-out method<br>allowed: double<br>Default: variant 0.0                |
| smoothsize    | size of smoothing beam for Pressed-out method<br>allowed: any<br>Default: variant 2.0      |
| direction     | scan direction of each image in unit of degree<br>allowed: any<br>Default: variant         |
| masklist      | mask width for Basket-Weaving (on percentage)<br>allowed: any<br>Default: variant 1.0      |
| tmax          | maximum threshold value for processing<br>allowed: double<br>Default: 0.0                  |
| tmin          | minimum threshold value for processing<br>allowed: double<br>Default: 0.0                  |
| outfile       | name of output file<br>allowed: string<br>Default:                                         |
| overwrite     | overwrite the output file if already exists<br>allowed: bool<br>Default: False             |

## Returns

void

## Example



-----  
Keyword arguments  
-----

infile -- name or list of names of input SD (FITS or CASA) image(s)  
mode -- image processing mode  
    options: 'basket' (FFT-based Basket-Weaving),  
            'press' (Pressed-out method)  
    default: 'basket'  
    >>>mode expandable parameter  
        direction -- scan direction of each input image in unit of degree  
            default: []  
            example: direction=[0.0, 90.0] means that the first image  
                    has scan direction along longitude axis while the  
                    second image is along latitude axis.  
        masklist -- mask width for Basket-Weaving on percentage  
            default: 1.0 (1.0% of map size)  
        numpoly -- order of polynomial fit in Presssed-out method  
            default: 2  
        beamsize -- beam size for Pressed-out method  
            default: 0.0  
            example: beamsize=10.0 is interpreted as '10arcsec'.  
                    beamsize='1arcmin' specifies beam size as  
                    quantity.  
        smoothsize -- smoothing beam size in Pressed-out method.  
            if numeric value is given, it is interpreted in unit  
            of beam size specified by the parameter beamsize  
            default: 2.0  
            example: smoothsize=2.0 means that smoothing beam size is  
                    2.0 \* beamsize.  
                    smoothsize='1arcmin' sets smoothsize directly.  
tmax -- maximum threshold value for processing  
    default: 0.0 (no threshold in maximum)  
    example: 10.0 (mask data larger value than 10.0)  
tmin -- minimum threshold value for processing  
    default: 0.0 (no threshold in minimum)  
    example: -10.0 (mask data smaller value than -10.0)  
outfile -- name of output file. output file is in CASA image format.  
    default: '' (use default name 'sdimprocess.out.im')  
    example: 'output.im'  
overwrite -- overwrite the output file if already exists  
    options: (bool) True, False  
    default: False

-----  
DESCRIPTION  
-----

Task `sdimprocess` is used to remove a scanning noise that appears as a striped noise pattern along the scan direction in a raster scan data.

By default, the scanning noise is removed by using the FFT-based 'Basket-Weaving' method (Emerson & Grave 1988) that requires multiple images that observed exactly the same area with different scanning direction. If only one image is available, the 'Pressed-out' method (Sofue & Reich 1979) can be used to remove the scanning effect.

For 'Basket-Weaving', scanning directions must have at least two different values. Normally, the scanning direction should be specified for each input image. Otherwise, specified scanning directions will be used iteratively. The `masklist` is a width of masking region in the Fourier plane. It is specified as a fraction (percentage) of the image size.

For 'Pressed-out', the scanning direction must be unique. There are two ways to specify a size of smoothing beam used for process. One is to specify smoothing size directly. To do this, `smoothsize` should be specified as string that consists of a numerical value and an unit (e.g. '10.0arcsec'). A value of `beamsize` will be ignored in this case. Another way to specify smoothing size is to set an observed beam size and indicate smoothing size as a scale factor of the observed beam size. In this case, the `beamsize` is interpreted as the observed beam size, and the `smoothsize` is the scale factor. If the `beamsize` is provided as float value, its unit is assumed to 'arcsec'. It is also possible to set the `beamsize` as string consisting of the numerical value and the unit. The `smoothsize` must be float value.

The `infile` only allows an image data (CASA or FITS), and not does not work with MS or Scantable. The `direction` is an angle with respect to the horizontal direction, and its unit is degree. Any value may be interpreted properly, but the value ranging from 0.0 to 180.0 will be secure. The `tmax` and the `tmin` is used to specify a threshold that defines a range of spectral values used for processing. The data point that has the value larger than `tmax` or smaller than `tmin` will be excluded from the processing. The default (0.0) is no threshold. The `outfile` specifies an output CASA image name. If the `outfile` is empty, the default name ('`sdimprocess.out.im`') will be used.

sdlist-task.html

### 0.1.102 sdlist

Requires:

#### Synopsis

list summary of single dish data

#### Description

Task sdlist lists the scan summary of the dataset after importing as a scantable into ASAP. It will optionally output this summary as file.

#### Arguments

| Inputs    |                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------|
| infile    | name of input SD dataset<br>allowed: string<br>Default:                                                       |
| antenna   | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0 |
| outfile   | name of output file (ASCII) for summary list<br>allowed: string<br>Default:                                   |
| overwrite | overwrite the output file if already exists [True, False]<br>allowed: bool<br>Default: False                  |

#### Returns

void

#### Example

-----  
Keyword arguments  
-----

infile -- name of input SD dataset  
          default: none - must input file name  
          example: 'mysd.asap'  
          See sdcalf for allowed formats.  
antenna -- select an antenna name or id (only effective for MS input)  
          default: 0  
          example: 'PM03'  
          NOTE this parameter is effective only for MS input  
outfile -- name of output file for summary list  
          default: '' (no output file)  
          example: 'mysd\_summary.txt'  
overwrite -- overwrite the output file if already exists  
          options: (bool) True,False  
          default: False

-----  
DESCRIPTION  
-----

Task sdlist lists the scan summary of the dataset after importing as a scantable into ASAP. It will optionally output this summary as file.

-----  
WARNING  
-----

For the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

sdmath-task.html

### **0.1.103 sdmath**

Requires:

#### **Synopsis**

ASAP SD task for simple arithmetic of spectra

#### **Description**

Task sdmath execute a simple arithmetic (i.e., subtraction, addition, multiplication, and division) expression for single dish spectra. The spectral data file can be any of the formats supported by ASAP (scantable, MS, rpfits, and SDFITS). In the expression, these file names should be put inside of single or double quotes.

You can use variables in the expression. If you want to use, you must define varnames dictionary. Name of variables should be simple, e.g. V0, V1, etc., to avoid unexpected error. Keys of varnames must be name of variables that you used in the expression, and their values will be substituted for variables in the expression. Allowed type for the value is numerical values, one- or two-dimensional lists (Python list or numpy.ndarray), and filename strings that indicate spectral data or ASCII text, which is space-separated list of numerical values consisting of adequate number of rows and columns. In case you give a list of file names in infiles, they are automatically referred to as IN0, IN1, etc. in expr and you can not use IN0, IN1, etc. as variable names in varnames.

#### **Arguments**

|                |                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                                           |
| infile         | a list of names of input SD datasets<br>allowed:       stringArray<br>Default:                                            |
| expr           | mathematical expression using spectra<br>allowed:       string<br>Default:                                                |
| varnames       | dictionary of variables and their values used in expr<br>allowed:       any<br>Default:       variant                     |
| antenna        | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed:       any<br>Default:       variant 0 |
| fluxunit       | units of the flux ("=current)<br>allowed:       string<br>Default:                                                        |
| telescopeparam | parameters of telescope for flux conversion (see examples in help)<br>allowed:       any<br>Default:       variant        |
| field          | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed:       string<br>Default:                              |
| spw            | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed:       string<br>Default:                       |
| scan           | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed:       string<br>Default:                                    |
| pol            | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed:       string<br>Default:                                  |
| outfile        | name of output file (must be specified)<br>allowed:       string<br>Default:                                              |
| outform        | output file format (See a WARNING in help)<br>allowed:       string<br>Default:       ASAP                                |
| overwrite      | overwrite the output file if already exists [True, False]<br>allowed:       bool<br>Default:       False                  |

## Returns

void

## Example

Keyword arguments:

```
infile -- a list of names of input SD datasets
 The file names will automatically replace the phrases
 IN0, IN1, ... in expr parameter.
expr -- mathematical expression using scantables
varnames -- a python dictionary of variables in expr and their values.
 Keys must be coincide with variables used in expr.
 Values are substituted in each value in expr.
antenna -- select an antenna name or ID
 default: 0
 example: 'PM03'
 NOTE this parameter is effective only for MS input
fluxunit -- units for line flux
 options: 'K','Jy',''
 default: '' (keep current fluxunit in data)
 WARNING: For GBT data, see description below.
>>> fluxunit expandable parameter
 telescopeparam -- parameters of telescope for flux conversion
 options: (str) name or (list) list of gain info
 default: '' (none set)
 example: if telescopeparam='', it tries to get the telescope
 name from the data.
 Full antenna parameters (diameter,ap.eff.) known
 to ASAP are
 'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',
 'CEDUNA','HOBART'. For GBT, it fixes default fluxunit
 to 'K' first then convert to a new fluxunit.
 telescopeparam=[104.9,0.43] diameter(m), ap.eff.
 telescopeparam=[0.743] gain in Jy/K
 telescopeparam='FIX' to change default fluxunit
 see description below
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
```

```

spw -- select data by IF IDs (spectral windows)
 NOTE this task only supports IF ID selection and ignores channel
 selection.
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
outfile -- name of output file
 default: '' (must be specified)
outform -- output file format
 options: 'ASAP', 'MS2', 'ASCII', 'SDFITS'
 default: 'ASAP'
 NOTE the ASAP format is easiest for further sd
 processing; use MS2 for CASA imaging.
 If ASCII, then will append some stuff to
 the outfile name
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: False
 NOTE this parameter is ignored when outform='ASCII'

```

#### DESCRIPTION:

Task `sdmath` execute a simple arithmetic (i.e., subtraction, addition, multiplication, and division) expression for single dish spectra. The spectral data file can be any of the formats supported by ASAP (scantable, MS, rpfits, and SDFITS). In the expression, these file names should be put inside of single or double quotes.

You can use variables in the expression. If you want to use, you must define varnames dictionary. Name of variables should be simple, e.g. V0, V1, etc., to avoid unexpected error. Keys of varnames must be name of variables that you used in the expression, and their values will be substituted for variables in the expression. Allowed type for the value is numerical values, one- or two-dimensional lists (Python list or `numpy.ndarray`), and filename strings that indicate



spectral data or ASCII text, which is space-separated list of numerical values consisting of adequate number of rows and columns. In case you give a list of file names in infiles, they are automatically referred to as IN0, IN1, etc. in expr and you can not use IN0, IN1, etc. as variable names in varnames.

The fluxunit can be set, otherwise, the current settings of the first spectral data in the expression are used.

Other selections (e.g. scan No, . IF, Pol) also apply to all the spectral data in the expression, so if any of the data does not contains selection, the task will produce no output.

WARNING for the GBT raw SDFITS format data as input:

SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

Example:

```
do on-off/off calculation
expr='("orion_on_data.asap"-orion_off_data.asap)/orion_off_data.asap"
outfile=orion_cal.asap'
sdmath()
```

```
do on-off/off calculation (using infiles)
infiles = ["orion_on_data.asap", "orion_off_data.asap"]
expr='(IN0-IN1)/IN1'
outfile=orion_cal.asap'
sdmath()
```

```
do on-off/off calculation using varnames
varnames={} (this can be skipped if you executed inp(sdmath) or
 default(sdmath).)
varnames['V0']="orion_on_data.asap"
varnames['V1']="orion_off_data.asap"
varnames['V2']=1.0
expr='V0/V1-V2'
outfile=orion_cal.asap'
sdmath()
```

```
do on-off/off calculation using varnames (in pythonic way)
sdmath(varnames={'V0':'orion_on_data.asap', 'V1':'orion_off_data.asap',
 'V2':1.0}, expr='V0/V1-V2', outfile=orion_cal.asap')
```

# interpretation of ASCII file value for varnames  
If the contents of input ASCII file is shown as,

```
0.5 0.3 0.2
1.0 0.2 0.9
```

it is interpreted as a list `[[0.5, 0.3, 0.2],[1.0, 0.2, 0.9]]`.

sdplot-task.html

### **0.1.104 sdplot**

Requires:

#### **Synopsis**

ASAP SD plotting task

#### **Description**

Task sdplot displays single-dish spectra, total power, or pointing direction of input data. It assumes that the spectra have been calibrated. It does allow selection of scans, spectral windows, polarizations, and some time and channel averaging/smoothing options also, but does not write out this data.

This task adds an additional toolbar to Matplotlib plotter. See the cookbook for details of its capability.

#### **Arguments**

|                |                                                                                                                      |
|----------------|----------------------------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                                      |
| infile         | name of input SD dataset<br>allowed: string<br>Default:                                                              |
| antenna        | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0        |
| fluxunit       | units of the flux ("=current)<br>allowed: string<br>Default:                                                         |
| telescopeparam | parameters of telescope for flux conversion (see examples in help)<br>allowed: any<br>Default: variant               |
| specunit       | units for spectral axis<br>allowed: string<br>Default:                                                               |
| restfreq       | rest frequency (default unit: Hz)<br>allowed: any<br>Default: variant                                                |
| frame          | frequency reference frame ("=current)<br>allowed: string<br>Default:                                                 |
| doppler        | doppler convention ("=current). Effective only when spw selection is in velocity unit<br>allowed: string<br>Default: |
| field          | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                               |
| spw            | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                        |
| scan           | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                     |
| pol            | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                   |
| beam           | select data by beam IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                           |
| rastermode     | mode of raster selection ['row', 'raster']<br>allowed: 523string<br>Default: row                                     |
| raster         | select data by raster scan row or map iteration e.g. '0~2' ("=all)<br>allowed: string<br>Default:                    |
| timeaverage    | average spectra over time [True, False] (see examples in help)<br>allowed: bool                                      |

## Returns

void

## Example

Keyword arguments:

infile -- name of input SD dataset

antenna -- select an antenna name or ID

default: 0

example: 'PM03'

NOTE this parameter is effective only for MS input

fluxunit -- units for line flux

options: 'K','Jy',''

default: '' (keep current fluxunit in data)

WARNING: For GBT data, see description below.

>>> fluxunit expandable parameter

telescopeparam -- parameters of telescope for flux conversion

options: (str) name or (list) list of gain info

default: '' (none set)

example: if telescopeparam='', it tries to get the telescope name from the data.

Full antenna parameters (diameter,ap.eff.) known to ASAP are

'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',

'CEDUNA','HOBART'. For GBT, it fixes default fluxunit to 'K' first then convert to a new fluxunit.

telescopeparam=[104.9,0.43] diameter(m), ap.eff.

telescopeparam=[0.743] gain in Jy/K

telescopeparam='FIX' to change default fluxunit

see description below

specunit -- units for spectral axis

options: (str) 'channel','km/s','GHz','MHz','kHz','Hz'

default: '' (=current)

example: this will be the units for masklist

>>> specunit expandable parameter

restfreq -- the rest frequency

available type includes float, int, string, list of float, list of int, list of string, and list of dictionary. the

default unit of restfreq in case of float, int, or string

without unit is Hz. string input can be a value only

(treated as Hz) or a value followed by unit for which 'GHz',

'MHz','kHz',and 'Hz' are available.

```

a list can be used to set different rest frequencies for
each IF. the length of list input must be number of IFs.
dictionary input should be a pair of line name and
frequency with keys of 'name' and 'value', respectively.
values in the dictionary input follows the same manner as
as for single float or string input.
example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]

frame -- frequency reference frame
options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
default: '' (keep current frame in data)

doppler -- doppler convention (effective only when spw is in
velocity unit)
options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
default: '' (keep current doppler setting in data)

field -- select data by field IDs and names
default: '' (use all fields)
example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data

spw -- select data by IF IDs (spectral windows)
NOTE this task only supports IF ID selction and ignores channel
selection.
default: '' (use all IFs and channels)
example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all c
 this selection is in addition to the other selections to data

scan -- select data by scan numbers
default: '' (use all scans)
example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data

pol -- select data by polarization IDs
default: '' (use all polarizations)
example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data

beam -- select data by beam IDs
default: '' (use all beams)
example: beam='0,1' (beam IDs 0,1)
 this selection is in addition to the other selections to data

rastermode -- mode of raster selection
options: 'row', 'raster'

```

```

 default: 'row'
>>> rasterrow expandable parameter
 raster -- select data by raster scan row or map iteration
 default: '' (use all data)
 example: raster='0~2'
timeaverage -- average spectra over time
 options: (bool) True, False
 default: False
>>>timeaverage expandable parameter
 tweight -- weighting for time averaging
 options: 'var' (1/var(spec) weighted)
 'tsys' (1/Tsys**2 weighted)
 'tint' (integration time weighted)
 'tintsys' (Tint/Tsys**2)
 'median' (median averaging)
 default: 'tintsys'
 scanaverage -- average spectra within a scan number
 when True, spectra are NOT averaged over
 different scan numbers.
 options: (bool) True, False
 default: False
polaverage -- average spectra over polarizations
 options: (bool) True, False
 default: False
>>>polaverage expandable parameter
 pweight -- weighting for polarization average
 options: 'var' (1/var(spec) weighted)
 'tsys' (1/Tsys**2 weighted)
 default: 'tsys'
kernel -- type of spectral smoothing
 options: 'hanning','gaussian','boxcar', 'none'
 default: '' (= no smoothing)
>>>kernel expandable parameter
 kwidth -- width of spectral smoothing kernel
 options: (int) in channels
 default: 5
 example: 5 or 10 seem to be popular for boxcar
 ignored for hanning (fixed at 5 chans)
 (0 will turn off gaussian or boxcar)
plottype -- type of plot
 options: 'spectra','totalpower','pointing','azel','grid'
 default: 'spectra'
>>> plottype expandable parameters
 stack -- code for stacking on single plot for spectral plotting
 options: 'p','b','i','t','s','r' or
 'pol', 'beam', 'if', 'time', 'scan', 'row'

```

```

default: 'p'
example: maximum of 16 stacked spectra
 stack by pol, beam, if, time, scan
Note stack selection is ignored when panel='r'.
Note behavior of stack='t' depends on plottype:
 * stack by time in plottype='spectra'
 * stack by source type in plottype='totalpower' and 'pointing'
panel -- code for splitting into multiple panels for spectral plotting
options: 'p','b','i','t','s','r' or
 'pol', 'beam', 'if', 'time', 'scan', 'row'
default: 'i'
example: maximum of 16 panels
 panel by pol, beam, if, time, scan
Note panel selection is ignored when stack='r'.
flrange -- range for flux axis of plot for spectral plotting
options: (list) [min,max]
default: [] (full range)
example: flrange=[-0.1,2.0] if 'K'
 assumes current fluxunit
sprange -- range for spectral axis of plot
options: (list) [min,max]
default: [] (full range)
example: sprange=[42.1,42.5] if 'GHz'
 assumes current specunit
linecat -- control for line catalog plotting for spectral plotting
options: (str) 'all','none' or by molecule
default: 'none' (no lines plotted)
example: linecat='SiO' for SiO lines
 linecat='*OH' for alcohols
 uses sprange to limit catalog
WARNING: specunit must be in frequency (*Hz)
 to plot from the line catalog!
 and must be 'GHz' or 'MHz' to use
 sprange to limit catalog
linedop -- doppler offset for line catalog plotting (spectral plotting)
options: (float) doppler velocity (km/s)
default: 0.0
example: linedop=-30.0
center -- the central direction of gridding
default: '' (map center)
example: 'J2000 19h30m00 -40d00m00'
Note currently only supports 'J2000' as direction frame
cell -- x and y cell size of gridding
default: [] (map extent divided by # of subplots in x and y)
example: cell=['1.0arcmin','1.0arcmin']
 cell='1.0arcmin' (equivalent to the example above)

```



Note default number of subplots is 1 x 1 in `plottype='grid'`.

`subplot` -- number of subplots (row and column) on a page  
 NOTICE plotter will slow down when a large number is specified  
 default: -1 (auto. for `plottype='spectra'`, 1x1 for `plottype='grid'`)  
 example: 23 (2 rows by 3 columns)

`colormap` -- the colours to be used for plot lines.  
 default: None  
 example: `colormap="green red black cyan magenta"` (html standard)  
`colormap="g r k c m"` (abbreviation)  
`colormap="#008000 #00FFFF #FF0090"` (RGB tuple)  
 The plotter will cycle through these colours  
 when lines are overlaid (stacking mode).

`linestyles` -- the linestyles to be used for plot lines.  
 default: None  
 example: `linestyles="line dashed dotted dashdot dashdotdot dashdashdot"`.  
 The plotter will cycle through these linestyles  
 when lines are overlaid (stacking mode).  
 WARNING: Linestyles can be specified only one color has been set.

`linewidth` -- width of plotted lines.  
 default: 1  
 example: `linewidth=1` (integer)  
`linewidth=0.75` (double)

`histogram` -- plot histogram  
 options: (bool) True, False  
 default: False

`scanpattern` -- plot additional lines on the plot to indicate scan patterns  
 when `plottype='pointing'`  
 options: (bool) True, False  
 default: False

`header` -- print header information on the plot  
 options: (bool) True, False  
 default: True  
 The header information is printed only on the logger when  
`plottype = 'azel'` and `'pointing'`.

>>> header expandable parameter  
`headsize` -- header font size  
 options: (int)  
 default: 9

`plotstyle` -- customise plot settings  
 options: (bool) True, False  
 default: False

>>> plotstyle expandable parameter  
`margin` -- a list of subplot margins in figure coordinate (0-1),  
 i.e., fraction of the figure width or height.  
 The order of elements should be:  
 [left, bottom, right, top, horizontal space btw panels,

```

 vertical space btw panels]
 example: margin = [0.125, 0.1, 0.9, 0.9, 0.2, 0.2]
legendloc -- legend location on the axes (0-10)
 options: (integer) 0 -10
 see help of "sd.plotter.set_legend" for
 the detail of location. Note that 0 ('best')
 is very slow.
 default: 1 ('upper right')
outfile -- file name for hardcopy output
 options: (str) filename.eps,.ps,.png
 default: '' (no hardcopy)
 example: 'specplot.eps','specplot.png'
 Note this autodetects the format from the suffix (.eps,.ps,.png).
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: False

```

#### DESCRIPTION:

Task sdplot displays single-dish spectra, total power, or pointing direction of input data. It assumes that the spectra have been calibrated. It does allow selection of scans, IFs, polarizations, and some time and channel averaging/smoothing options also, but does not write out this data.

This task adds an additional toolbar to Matplotlib plotter. See the cookbook for details of its capability.

#### \*\*\* Data selection \*\*\*

This task allows data selection via field name, scan, IF, polarization and beam IDs. Selection of field allows pattern matching using asterisk, e.g., 'FLS3a\*'. Selection of scans, IFs, polarizations, and beams, is possible by a CASA type selection syntax using a string of comma separated numbers with operators, i.e., '~', '>', and '<'.

For example, the following selection

```
scan = "<3,7~9,15"
```

is to select scan IDs 0, 1, 2, 7, 8, 9, and 15.

#### ----- AVERAGING OF SPECTRA -----

Task sdplot has two averaging modes, i.e., time and polarization average.

When `timeaverage=True`, spectra are averaged over time for each IF (spectral window), polarization, and beam, independently. Note that, by default (`scanaverage=False`), `timeaverage=True` averages spectra irrespective of scan IDs.

It is possible to average spectra separately for each scan ID by setting a sub-parameter `scanaverage=True`.

For example, the combination of parameters: `scan='0~2'`, `timeaverage=True`, and `scanaverage=False`: averages spectra in scan ID 0 through 2 all together to a spectrum,

`scanaverage=True` : averages spectra per scan ID and end up with three spectra from scan 0, 1, and 2.

When `polaverage=True`, spectra are averaged over polarization for each IF (spectral window) and beam. Note that, so far, time averaging is automatically switched on when `polaverage` is set to `True`. This behavior is not desirable and will be discarded in future.

\*\*\* available plottypes \*\*\*

- \* `plottype = 'spectra'` plots single dish spectra. Multiple scans, IFs, polarizations, and beams can be handles through stacking and panelling.  
This task uses the JPL line catalog as supplied by ASAP.  
If you wish to use a different catalog, or have it plot the line IDs from top or bottom (rather than alternating), then you will need to explore the sd toolkit also.
- \* `plottype = 'grid'` plots spectra based on their pointing direction. The spectra are gridded by direction before plotting.  
Multiple IFs and polarizations are not handled in this mode. Only the first IF and polarizaion is gridded and plotted if data includes multiple IDs after selections are applied. Hence, over plotting is not available

Currently most of the parameters are ignored in the following modes.

- \* `plottype='totalpower'` is used to plot the total power data.  
and only plot option is amplitude versus data row number.
- \* `plottype='azel'` plots azimuth and elevation tracks of the source.
- \* `plottype='pointing'` plots antenna poinitings.

\*\*\* control of plot lines in 'spectra' and 'grid' plottype \*\*\*

Note that `colormap` and `linestyles` cannot be controlled at a time. The `'linestyles'` is ignored if both of them are specified. Some plot options, like changing titles, legends, fonts, and the like are not supported in this task. You should use `sd.plotter` from the ASAP toolkit directly for this.

ASAP recognizes the data of the "AT" telescopes, but currently does not know about the GBT or any other telescope. This task does know about GBT. Telescope name is obtained from the data. If you wish to change the fluxunit (see below), and telescopeparam='', for the AT telescopes it will use internal telescope parameters for flux conversion. For GBT, it will use an approximate aperture efficiency conversion. If you give telescopeparam a list, then if the list has a single float it is assumed to be the gain in Jy/K, if two or more elements they are assumed to be telescope diameter (m) and aperture efficiency respectively.

WARNING: be careful plotting otf data with lots of fields!

WARNING for the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

sdreduce-task.html

### **0.1.105   sdreduce**

Requires:

#### **Synopsis**

ASAP SD task: do sdcal, sdaverage, and sdbaseline in one task

#### **Description**

Task sdreduce performs data selection, calibration, spectral averaging and/or baseline fitting for single-dish spectra. This task internally calls the tasks, sdcal, sdaverage, and sdbaseline and it can be used to run all the three steps in one task execution. This task has better performance than invoking the three tasks separately because it runs all three steps without writing intermediate data to disk.

It is possible to skip arbitrary operations by setting calmode = 'none' (for calibration), average=False (for time and polarization averaging), kernel = 'none' (for smoothing), and/or blfunc='none' (for baseline fitting).

Please take a look at descriptions of tasks, sdcal, sdaverage, and sdbaseline, for more information.

#### **Arguments**

|                |                                                                                                                          |
|----------------|--------------------------------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                                          |
| infile         | name of input SD dataset<br>allowed: string<br>Default:                                                                  |
| antenna        | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0            |
| fluxunit       | units of the flux ("=current)<br>allowed: string<br>Default:                                                             |
| telescopeparam | parameters of telescope for flux conversion (see description in help of sdcal)<br>allowed: string<br>Default:            |
| field          | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                                   |
| spw            | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                            |
| restfreq       | the rest frequency, e.g. '1.41GHz' (default unit: Hz) (see examples in help)<br>allowed: any<br>Default: variant         |
| frame          | frequency reference frame ("=current)<br>allowed: string<br>Default:                                                     |
| doppler        | doppler convention ("=current). Effective only when spw selection is in velocity unit.<br>allowed: string<br>Default:    |
| timerange      | select data by time range, e.g. '09:14:0~09:54:0' ("=all) (see examples in help of sdcal)<br>allowed: string<br>Default: |
| scan           | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                         |
| pol            | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                       |
| calmode        | SD calibration mode ('none' = skip calibration)<br>allowed: string<br>Default: none                                      |
| fraction       | fraction of the OFF data to mark as OFF spectra, e.g., '10%'<br>allowed: any<br>Default: variant 10%                     |
| noff           | number of the OFF data to mark (-1 = use fraction instead of number)<br>allowed: int<br>Default: -1                      |
| width          | width of the pixel for edge detection                                                                                    |

## Returns

void

## Example

Keyword arguments:

infile -- name of input SD dataset

antenna -- select an antenna name or ID

default: 0

example: 'PM03'

NOTE this parameter is effective only for MS input

fluxunit -- units for line flux

options: 'K','Jy',''

default: '' (keep current fluxunit in data)

WARNING: For GBT data, see description below.

>>> fluxunit expandable parameter

telescopeparam -- parameters of telescope for flux conversion

options: (str) name or (list) list of gain info

default: '' (none set)

example: if telescopeparam='', it tries to get the telescope name from the data.

Full antenna parameters (diameter,ap.eff.) known to ASAP are

'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',

'CEDUNA','HOBART'. For GBT, it fixes default fluxunit to 'K' first then convert to a new fluxunit.

telescopeparam=[104.9,0.43] diameter(m), ap.eff.

telescopeparam=[0.743] gain in Jy/K

telescopeparam='FIX' to change default fluxunit

see description below

field -- select data by field IDs and names

default: '' (use all fields)

example: field='3C2\*' (all names starting with 3C2)

field='0,4,5~7' (field IDs 0,4,5,6,7)

field='0,3C273' (field ID 0 or field named 3C273)

this selection is in addition to the other selections to data

spw -- select data by IF IDs (spectral windows)/channels

NOTE channel range selections are interpreted as mask regions to INCLUDE in BASELINE fit, and ignored in the other operations.

when maskmode is 'auto' or 'interact', the channel mask

will be applied first before fitting as base mask

```

default: '' (use all IFs and channels)
example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)
 spw='0:5~61' (IF ID 0; channels 5 to 61)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
this selection is in addition to the other selections to data
>>> spw expandable parameters
restfreq -- the rest frequency
 available type includes float, int, string, list of float,
 list of int, list of string, and list of dictionary. the
 default unit of restfreq in case of float, int, or string
 without unit is Hz. string input can be a value only
 (treated as Hz) or a value followed by unit for which 'GHz',
 'MHz', 'kHz', and 'Hz' are available.
 a list can be used to set different rest frequencies for
 each IF. the length of list input must be number of IFs.
 dictionary input should be a pair of line name and
 frequency with keys of 'name' and 'value', respectively.
 values in the dictionary input follows the same manner as
 as for single float or string input.
example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]
frame -- frequency reference frame
options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
default: '' (keep current frame in data)
doppler -- doppler convention (effective only when spw is in
 velocity unit)
options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
default: '' (keep current doppler setting in data)

timerange -- select data by time range
default: '' (use all)
example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
Note: YYYY/MM/DD can be dropped as needed:
timerange='09:14:00~09:54:00' # this time range
timerange='09:44:00' # data within one integration of time
timerange='>10:24:00' # data after this time
timerange='09:44:00+00:13:00' #data 13 minutes after time
this selection is in addition to the other selections to data
scan -- select data by scan numbers

```



```

 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data

calmode -- calibration mode
 options: 'ps','nod','otf','otfraster',
 'fs','quotient','none'
 default: 'none'
 example: choose mode 'none' if you have already calibrated
 and want to correct for atmospheric opacity defined
 by tau, subtract baseline or average/smooth spectra.
>>> calmode expandable parameter
 fraction -- edge marker parameter of 'otf' and 'otfraster'.
 Specify a number of OFF integrations (at each
 side of the raster rows in 'otfraster' mode)
 as a fraction of total number of integrations.
 In 'otfraster' mode, number of integrations
 to be marked as OFF, n_off, is determined by
 the following formula,

$$n_{\text{off}} = \text{floor}(\text{fraction} * n),$$

 where n is number of integrations per raster
 row. Note that n_off from both sides will be
 marked as OFF so that twice of specified
 fraction will be marked at most. For example,
 if you specify fraction='10%', resultant
 fraction of OFF integrations will be 20% at
 most.
 In 'otf' mode, n_off is given by,

$$n_{\text{off}} = \text{floor}(\text{fraction} * n),$$

 where n is number of total integrations.
 n_off is used as criterion of iterative marking
 process. Therefore, resulting total number of
 OFFs will be larger than n_off. In practice,
 fraction is a geometrical fraction of edge
 region. Thus, if integrations are concentrated
 on edge region (e.g. some of Lissajous
 patterns), then resulting n_off may be
 unexpectedly large.

```

```

 default: '10%'
 options: '20%' in string style or float value less
 than 1.0 (e.g. 0.15).
 'auto' is available only for 'otfraster'.
noff -- edge marking parameter for 'otfraster'.
 It is used to specify a number of OFF scans near
 edge directly. Value of noff comes before setting
 by fraction. Note that n_off from both sides will
 be marked as OFF so that twice of specified noff
 will be marked at most.
 default: -1 (use fraction)
 options: any positive integer
width -- edge marking parameter for 'otf'.
 Pixel width with respect to a median spatial
 separation between neighboring two data in time.
 Default will be fine in most cases.
 default: 0.5
 options: float value
elongated -- edge marking parameter for 'otf'.
 Set True only if observed area is elongated
 in one direction.
 options: (bool) True, False
 default: False
markonly -- set True if you want to save data just after
 edge marking (i.e. uncalibrated data) to see
 how OFF scans are defined.
 options: (bool) True, False
 default: False
plotpointings -- load plotter and plot pointing directions of
 ON and OFF scans.
 options: (bool) True, False
 default: False

tau -- the zenith atmospheric optical depth for correction
 default: 0.0 (no correction)
average -- averaging on spectral data
 options: (bool) True,False
 default: False

>>>average expandable parameter
timeaverage -- average spectra over time
 options: (bool) True, False
 default: False
 example: if True, this happens after calibration
tweight -- weighting for time averaging (effective only when
 timeaverage=True)

```

```

 options: 'var' (1/var(spec) weighted)
 'tsys' (1/Tsys**2 weighted)
 'tint' (integration time weighted)
 'tintsys' (Tint/Tsys**2)
 'median' (median averaging)
 default: 'tintsys'
scanaverage -- average spectra within a scan number (effective
 only when timeaverage=True)
 when True, spectra are NOT averaged over
 different scan numbers.
 options: (bool) True, False
 default: False
averageall -- average multi-resolution spectra (effective only
 when timeaverage=True)
 spectra are averaged by referring their frequency
 coverage
 default: False
polaverage -- average spectra over polarizations
 options: (bool) True, False
 default: False
pweight -- weighting for polarization averaging (effective only
 when polaverage=True)
 options: 'var' (1/var(spec) weighted)
 'tsys' (1/Tsys**2 weighted)
 default: 'tsys'

kernel -- type of spectral smoothing kernel
 options: 'none', 'hanning', 'gaussian', 'boxcar', 'regrid', ''(='none')
 default: 'none' (no smoothing)

>>>kernel expandable parameter
 kwidth -- width of spectral smoothing kernel
 options: (int) in channels
 default: 5
 example: 5 or 10 seem to be popular for boxcar
 ignored for hanning (fixed at 5 chans)
 (0 will turn off gaussian or boxcar)
 chanwidth -- channel width of regridded spectra
 default: '5' (in channels)
 example: '500MHz', '0.2km/s'

maskmode -- mode of setting additional channel masks
 options: 'auto', 'list', or 'interact'
 default: 'auto'
 example: maskmode='auto' runs linefinder to detect line regions
 to be excluded from fitting. this mode requires three

```

expandable parameters: thresh, avg\_limit, and edge.  
 USE WITH CARE! May need to tweak the expandable parameters.  
 maskmode='list' uses the given masklist only: no additional  
 masks applied.  
 maskmode='interact' allows users to manually modify the  
 mask regions by dragging mouse on the spectrum plotter GUI.  
 use LEFT or RIGHT button to add or delete regions,  
 respectively.

```
>>> maskmode expandable parameters
 thresh -- S/N threshold for linefinder. a single channel S/N ratio
 above which the channel is considered to be a detection.
 default: 5
 avg_limit -- channel averaging for broad lines. a number of
 consecutive channels not greater than this parameter
 can be averaged to search for broad lines.
 default: 4
 edge -- channels to drop at beginning and end of spectrum
 default: 0
 example: edge=[1000] drops 1000 channels at beginning AND end.
 edge=[1000,500] drops 1000 from beginning and 500
 from end.
```

Note: For bad baselines threshold should be increased,  
 and avg\_limit decreased (or even switched off completely by  
 setting this parameter to 1) to avoid detecting baseline  
 undulations instead of real lines.

```
blfunc -- baseline model function
options: 'poly', 'chebyshev', 'cspline', or 'sinusoid'
default: 'none' (no baseline fit)
example: blfunc='poly' uses a single polynomial line of
 any order which should be given as an expandable
 parameter 'order' to fit baseline.
 blfunc='chebyshev' uses Chebyshev polynomials.
 blfunc='cspline' uses a cubic spline function, a piecewise
 cubic polynomial having C2-continuity (i.e., the second
 derivative is continuous at the joining points).
 blfunc='sinusoid' uses a combination of sinusoidal curves.
>>> blfunc expandable parameters
 order -- order of baseline model function
 options: (int) (<0 turns off baseline fitting)
 default: 5
 example: typically in range 2-9 (higher values
 seem to be needed for GBT)
 npiece -- number of the element polynomials of cubic spline curve
 options: (int) (<0 turns off baseline fitting)
```

```

 default: 2
applyfft -- automatically set wave numbers of sinusoidal functions
 for fitting by applying some method like FFT.
 options: (bool) True, False
 default: True
fftmethod -- method to be used when applyfft=True. Now only
 'fft' is available and it is the default.
fftthresh -- threshold to select wave numbers to be used for
 sinusoidal fitting. both (float) and (str) accepted.
 given a float value, the unit is set to sigma.
 for string values, allowed formats include:
 'xsigma' or 'x' (= x-sigma level. e.g., '3sigma'), or
 'topx' (= the x strongest ones, e.g. 'top5').
 default is 3.0 (unit: sigma).
addwn -- additional wave number(s) of sinusoids to be used
 for fitting.
 (list) and (int) are accepted to specify every
 wave numbers. also (str) can be used in case
 you need to specify wave numbers in a certain range.
 default: [0] (i.e., constant is subtracted at least)
 example: 0
 [0,1,2]
 'a-b' (= a, a+1, a+2, ..., b-1, b),
 '<a' (= 0,1,...,a-2,a-1),
 '>=a' (= a, a+1, ... up to the maximum wave
 number corresponding to the Nyquist
 frequency for the case of FFT).
rejwn -- wave number(s) of sinusoid NOT to be used for fitting.
 can be set just as addwn but has higher priority:
 wave numbers which are specified both in addwn
 and rejwn will NOT be used.
 default: []
clipthresh -- clipping threshold for iterative fitting
 default: 3
clipniter -- maximum iteration number for iterative fitting
 default: 0 (no iteration, i.e., no clipping)

verifycal -- interactively verify the results of calibration
 See description of verify parameter in the task, sdcal,
 for details.
 options: (bool) True,False
 default: False
verifysm -- interactively verify the results of smoothing for each
 spectrum.
 See description of verify parameter in the task, sdaverage,
 for details.

```

```

options: (bool) True,False
default: False
Note: verification is not yet available for kernel='regrid'
verifybl -- interactively verify the results of baseline fitting for
each spectrum.
See description of verify parameter in the task, sdbaseline,
for details.
options: (bool) True,False
default: False
NOTE: Currently available only when blfunc='poly'
verbosebl -- output fitting results to logger. If False, the fitting results
including coefficients, residual rms, etc., are not output to
the CASA logger, while the processing speed gets faster.
options: (bool) True, False
default: True
bloutput -- output fitting results to a text file. if False, the fitting
results including coefficients, residual rms, etc., are not
output to a text file (<outfile>_blparam.txt), while
the processing speed gets faster.
options: (bool) True, False
default: True
blformat -- format of the logger output and text file specified with bloutput
options: '', 'csv'
default: '' (same as in the past, easy to read but huge)
showprogress -- show progress status for large data
options: (bool) True, False
default: True
>>> showprogress expandable parameter
minnrow -- minimum number of input spectra to show progress status
default: 1000
outfile -- name of output file
default: '' (<infile>_cal)
outform -- output file format
options: 'ASAP', 'MS2', 'ASCII', 'SDFITS'
default: 'ASAP'
NOTE the ASAP format is easiest for further sd
processing; use MS2 for CASA imaging.
If ASCII, then will append some stuff to
the outfile name
overwrite -- overwrite the output file if already exists
options: (bool) True,False
default: False
NOTE this parameter is ignored when outform='ASCII'
plotlevel -- control for plotting and summary of results
options: (int) 0, 1, 2, and their negative counterparts
default: 0 (no plotting)

```

example: `plotlevel=1`; show plot of spectra (see description of  
the parameter in `sdcal`, `sdaverage`, and `sdbaseline`)  
`plotlevel=2`; additionally list data before and after operation.  
`plotlevel<0` as `abs(plotlevel)`, e.g.  
`-1` => hardcopy of final plot at each step, i.e.,  
`<outfile>_calspec.eps`, `<outfile>_smspec.eps`, and/or  
`<outfile>_bsspec.eps`

-----  
WARNING  
-----

For the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the  
data filler is available. However, the functionality is not well  
tested yet, so that there may be unknown bugs.

sdsave-task.html

### **0.1.106 sdsave**

Requires:

#### **Synopsis**

Save the sd spectra in various format

#### **Description**

Task sdsave writes the single dish data to a disk file in specified format (ASAP, MS2, SDFITS, ASCII). It is possible to save the subset of the data by selecting field names, spw ids, time ranges, scan numbers, and polarization ids. The ASAP (scantable) format is recommended for further analysis using Sd tool or tasks except imaging. For further imaging using imager or sdimage, save the data to the Measurement Set (MS2).

#### **Arguments**



|            |                                                                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                       |
| infile     | name of input SD dataset<br>allowed: string<br>Default:                                                                                               |
| splitant   | split output file by antenna (only effective for MS input)<br>allowed: bool<br>Default: False                                                         |
| antenna    | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0                                         |
| getpt      | fill DIRECTION column properly (True), or reuse POINTING table in original MS (False) (only effective for MS input)<br>allowed: bool<br>Default: True |
| field      | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                                                                |
| spw        | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                                                         |
| timerange  | select data by time range, e.g. '09:14:0~09:54:0' ("=all) (see examples in help)<br>allowed: string<br>Default:                                       |
| scan       | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                                                      |
| pol        | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                                                    |
| beam       | select data by beam IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                                                            |
| restfreq   | the rest frequency, e.g. '1.41GHz' (default unit: Hz) (see examples in help)<br>allowed: any<br>Default: variant                                      |
| outfile    | name of output file (See a WARNING in help)<br>allowed: string<br>Default:                                                                            |
| outform    | output file format (See a WARNING in help)<br>allowed: string<br>Default: ASAP                                                                        |
| fillweight | fill the WEIGHT <sub>501</sub> and SIGMA columns for output MS<br>allowed: bool<br>Default: False                                                     |
| overwrite  | overwrite the output file if already exists<br>allowed: bool<br>Default: False                                                                        |

## Returns

void

## Example

-----  
Keyword arguments  
-----

```
infile -- name of input SD dataset
splitant -- split output file by antenna. this parameter is only
 effective for MS input.
 options: (bool) True, False
 default: False
>>>splitant expandable parameter
 antenna -- select an antenna name or ID. this parameter is
 effective only for MS input.
 default: 0
 example: antenna=0 specifies antenna by id
 antenna='PM03' specifies antenna by name
getpt -- fill DIRECTION column properly (True), or reuse POINTING
 table in original MS (False). this parameter is only
 effective for MS input.
 options: (bool) True, False
 default: True
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)/channels
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)
 spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
timerange -- select data by time range
 default: '' (use all)
```

```

example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
Note: YYYY/MM/DD can be dropped as needed:
timerange='09:14:00~09:54:00' # this time range
timerange='09:44:00' # data within one integration of time
timerange='>10:24:00' # data after this time
timerange='09:44:00+00:13:00' #data 13 minutes after time
this selection is in addition to the other selections to data
scan -- select data by scan numbers
default: '' (use all scans)
example: scan='21~23' (scan IDs 21,22,23)
this selection is in addition to the other selections to data
pol -- select data by polarization IDs
default: '' (use all polarizations)
example: pol='0,1' (polarization IDs 0,1)
this selection is in addition to the other selections to data
beam -- select data by beam IDs
default: '' (use all beams)
example: beam='0,1' (beam IDs 0,1)
this selection is in addition to the other selections to data
restfreq -- the rest frequency
available type includes float, int, string, list of float,
list of int, list of string, and list of dictionary. the
default unit of restfreq in case of float, int, or string
without unit is Hz. string input can be a value only
(treated as Hz) or a value followed by unit for which 'GHz',
'MHz','kHz',and 'Hz' are available.
a list can be used to set different rest frequencies for
each IF. the length of list input must be number of IFs.
dictionary input should be a pair of line name and
frequency with keys of 'name' and 'value', respectively.
values in the dictionary input follows the same manner as
as for single float or string input.
example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]
outfile -- name of output file
default: '' (<infile>_saved)
NOTE actual output file name(s) will be modified if splitant
is True as antenna names are to be included. If outfile has a
suffix '.asap' or '.ASAP', antenna name will be inserted before
the suffix like 'out.antName.asap', otherwise, antenna name
will be simply appended to outfile like 'out.sdfits.antName'.
outform -- output file format
options: 'ASAP','MS2', 'ASCII','SDFITS'

```

```

 default: 'ASAP'
 NOTE the ASAP format is easiest for further sd
 processing; use MS2 for CASA imaging.
 If ASCII, then will append some stuff to
 the outfile name
fillweight -- fill WEIGHT and SIGMA column for output MS
 default: True
 options: True, False

```

#### ----- DESCRIPTION -----

Task sdsave writes the single dish data to a disk file in specified format (ASAP, MS2, SDFITS, ASCII). It is possible to save the subset of the data by selecting field names, spw ids, time ranges, scan numbers, and polarization ids. The ASAP (scantable) format is recommended for further analysis using Sd tool or tasks except imaging. For further imaging using imager or sdimaging, save the data to the Measurement Set (MS2).

Note that setting getpt=False needs a lot of attention. If you set getpt=False, the task retrieves pointing direction from MS's FIELD table, which might not be correct for single dish observation, instead to check MS's POINTING table, which is the default behavior of the task (getpt=True). To compensate this, absolute path to MS's POINTING table is stored, and it will be used for POINTING table when the data is converted back to MS format. In general, getpt=False is faster especially for large data. However, MS created from Scantable cannot have correct POINTING table if original MS's POINTING table doesn't exist. Such situation will happen when original MS is removed or renamed, or imported Scantable is moved to other computer alone.

#### ----- WARNING -----

For the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

sdscale-task.html

### 0.1.107 sdscale

Requires:

#### Synopsis

Scale the sd spectra

#### Description

Task sdscale performs scaling of single-dish spectra by scaling factor given by parameter named factor. By setting scaletsys = True, associated Tsys is also scaled.

#### Arguments

| Inputs    |                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------|
| infile    | name of input SD dataset<br>allowed: string<br>Default:                                                       |
| antenna   | select an antenna name or ID, e.g. 'PM03' (only effective for MS input)<br>allowed: any<br>Default: variant 0 |
| factor    | scaling factor (float or float list)<br>allowed: any<br>Default: variant 1.0                                  |
| scaletsys | scaling of associated Tsys<br>allowed: bool<br>Default: True                                                  |
| outfile   | name of output file (See a WARNING in help)<br>allowed: string<br>Default:                                    |
| overwrite | overwrite the output file if already exists<br>allowed: bool<br>Default: False                                |
| verbose   | Print verbose log output<br>allowed: bool<br>Default: True                                                    |

## Returns

void

## Example

```

Keyword arguments

infile -- name of input SD dataset
antenna -- select an antenna name or ID
 default: 0
 example: 'PM03'
 NOTE this parameter is effective only for MS input
factor -- scaling factor. float, one- or two-dimensional float list,
 or filename storing scaling factor are acceptable
 default: 1.0 (no scaling)
 example: see description below
scaletsys -- scaling of associated Tsys
 options: (bool) True, False
 default: True
outfile -- name of output file
 default: outfile='' (<infile>_scaled<factor>)
 example: 'scaled.asap'
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: False
 NOTE this parameter is ignored when outform='ASCII'
verbose -- Print verbose log messages. If True, Tsys values before
 (and after) scaling are printed to logger.
 options: (bool) True, False
 default: True
```

## DESCRIPTION

```

Task sdscale performs scaling of single-dish spectra.
Associated Tsys is also scaled if scaletsys is True.
Tsys informations are written in the casalogger and they are
automatically stored in 'casapy.log'.
The infile can be any of ASAP, MS, SDFITS, or RPFITS format.
If outfile name is given or outfile=''(default), the scaled data
is written to a new file with the same format as the input data
```

(Note: in case of the RPFITS format input data, it will be written to SDFITS format).

The scaling factor, `factor`, accepts both scalar type and list type value. The list must be one or two dimensional. If `factor` is one dimensional, its length must coincide with a number of spectral channel. If `factor` is two dimensional, its shape must be  $(n, 1)$  or  $(n, m)$ , where  $n$  is a number of spectrum, while  $m$  is a number of channel for each spectrum.  $m$  can be variable for each spectrum. In addition, the `factor` can be an ASCII filename that stores a space-separated list of scaling factor consisting of adequate number of rows and columns. For example, if the contents of input ASCII file is shown as,

```
0.5 0.3 0.2
1.0 0.2 0.9
```

it is interpreted as a list `[[0.5, 0.3, 0.2],[1.0, 0.2, 0.9]]`.

```

WARNING

```

For the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the data filler is available. However, the functionality is not well tested yet, so that there may be unknown bugs.

sdstat-task.html

### **0.1.108 sdstat**

Requires:

#### **Synopsis**

list statistics of spectral

#### **Description**

Task sdstat computes basic statistics for each of single-dish spectrum. This task returns a Python dictionary of statistics. The return value contains the maximum and minimum intensity and their channels ('max', 'max\_abscissa', 'min', and 'min\_abscissa'), RMS ('rms'), mean ('mean'), sum ('sum'), median ('median'), standard deviation ('stddev'), total intensity ('totint'), and equivalent width ('eqw'). If you do have multiple scantable rows, then the return values will be lists.

It is possible to select channel regions to calculate spectra either non-interactively by spw parameter or interactively on a plotter by setting interactive=True.

If one of averaging parameters is set True, the spectra are averaged before calculating the statistics.

#### **Arguments**



|                |                                                                                                                                      |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Outputs        |                                                                                                                                      |
| xstat          | <p>RETURN ONLY: a Python dictionary of line statistics</p> <p>allowed: any</p> <p>Default: variant</p>                               |
| Inputs         |                                                                                                                                      |
| infile         | <p>name of input SD dataset</p> <p>allowed: string</p> <p>Default:</p>                                                               |
| antenna        | <p>select an antenna name or ID, e.g. 'PM03' (only effective for MS input)</p> <p>allowed: any</p> <p>Default: variant 0</p>         |
| fluxunit       | <p>units of the flux ("=current)</p> <p>allowed: string</p> <p>Default:</p>                                                          |
| telescopeparam | <p>parameters of telescope for flux conversion (see examples in help)</p> <p>allowed: any</p> <p>Default: variant</p>                |
| field          | <p>select data by field IDs and names, e.g. '3C2*' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                                |
| spw            | <p>select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                         |
| restfreq       | <p>the rest frequency, e.g. '1.41GHz' (default unit: Hz) (see examples in help)</p> <p>allowed: any</p> <p>Default: variant</p>      |
| frame          | <p>frequency reference frame ("=current)</p> <p>allowed: string</p> <p>Default:</p>                                                  |
| doppler        | <p>doppler convention ("=current). Effective only when spw selection is in velocity unit.</p> <p>allowed: string</p> <p>Default:</p> |
| timerange      | <p>select data by time range, e.g. '09:14:0~09:54:0' ("=all) (see examples in help)</p> <p>allowed: string</p> <p>Default:</p>       |
| scan           | <p>select data by scan numbers, e.g. '21~23' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                                      |
| pol            | <p>select data by polarization IDs, e.g. '0,1' ("=all)</p> <p>allowed: 552string</p> <p>Default:</p>                                 |
| beam           | <p>select data by beam IDs, e.g. '0,1' ("=all)</p> <p>allowed: string</p> <p>Default:</p>                                            |
| timeaverage    | <p>average spectra over time [True, False] (see examples in help)</p> <p>allowed: bool</p> <p>Default: False</p>                     |

## Returns

void

## Example

```

How to use return values

```

```
xstat = sdstat();
print "rms = ",xstat['rms']
```

these can be used for testing in scripts or for regression

'max\_abscissa' and 'min\_abscissa' refer to the channel of max and min intensity.

'totint' is the integrated intensity (sum\*channel).

'eqw' is equivalent width (totint/mag) where mag is either max or min depending on which has greater magnitude.

Note that 'max\_abscissa', 'min\_abscissa', 'totint' and 'eqw' are quantities (python dictionaries with keys, 'unit' and 'value').

## ----- AVERAGING OF SPECTRA -----

Task sdstat has two averaging modes, i.e., time and polarization average.

When timeaverage=True, spectra are averaged over time for each IF (spectral window), polarization, and beam, independently. Note that, by default (scanaverage=False), timeaverage=True averages spectra irrespective of scan IDs.

It is possible to average spectra separately for each scan ID by setting a sub-parameter scanaverage=True.

For example, the combination of parameters: scan='0~2', timeaverage=True, and scanaverage=False: averages spectra in scan ID 0 through 2 all together to a spectrum,

scanaverage=True : averages spectra per scan ID and end up with three spectra from scan 0, 1, and 2.

When polaverage=True, spectra are averaged over polarization for each IF (spectral window) and beam. Note that, so far, time averaging is automatically switched on when polaverage is set to True. This behavior is not desirable and will be discarded in future.

-----  
FLUX UNIT CONVERSION  
-----

The task is able to convert flux unit between K and Jy. To do that, fluxunit and its subparameter telescopeparam must be properly set. The fluxunit should be 'Jy' or 'K' depending on what unit input data is and what unit you want to convert. If given fluxunit is different from the unit of input data, unit conversion is performed. The telescopeparam is used to specify conversion factor. There are three ways to specify telescopeparam: 1) set Jy/K conversion factor, 2) set telescope diameter, D, and aperture efficiency, eta, separately, and 3) 'FIX' mode (only change the unit without converting spectral data). If you give telescopeparam as a list, then if the list has a single float it is assumed to be the gain in Jy/K (case 1), if two or more elements they are assumed to be telescope diameter (m) and aperture efficiency respectively (case 2). See the above parameter description as well as note on 'FIX' mode below for details.

There are two special cases that don't need telescopeparam for unit conversion. Telescope name is obtained from the data.

- 1) ASAP (sd tool) recognizes the conversion factor (actually D and eta) for the "AT" telescopes, namely ATNF MOPRA telescope, until 2004.
- 2) The task does know D and eta for GBT telescope.  
If you wish to change the fluxunit, by leaving the sub-parameter telescopeparam unset (telescopeparam=''), it will use internal telescope parameters for flux conversion for the data from AT telescopes and it will use an approximate aperture efficiency conversion for the GBT data.

Note that xxx assumes that the fluxunit is set correctly in the data already. If not, then set telescopeparam='FIX' and it will set the default units to fluxunit without conversion.

Note also that, if the data in infile is an ms from GBT and the default flux unit is missing, this task automatically fixes the default fluxunit to 'K' before the conversion.

-----  
Keyword arguments  
-----

infile -- name of input SD dataset  
          default: none - must input file name  
          example: 'mysd.asap'  
          See sdcal for allowed formats.  
antenna -- select an antenna name or ID

```

 default: 0
 example: 'PM03'
 NOTE this parameter is effective only for MS input
fluxunit -- units for line flux
 options: 'K','Jy',''
 default: '' (keep current fluxunit in data)
 WARNING: For GBT data, see description below.
>>> fluxunit expandable parameter
 telescopeparam -- parameters of telescope for flux conversion
 options: (str) name or (list) list of gain info
 default: '' (none set)
 example: if telescopeparam='', it tries to get the telescope
 name from the data.
 Full antenna parameters (diameter,ap.eff.) known
 to ASAP are
 'ATPKSMB', 'ATPKSHOH', 'ATMOPRA', 'DSS-43',
 'CEDUNA','HOBART'. For GBT, it fixes default fluxunit
 to 'K' first then convert to a new fluxunit.
 telescopeparam=[104.9,0.43] diameter(m), ap.eff.
 telescopeparam=[0.743] gain in Jy/K
 telescopeparam='FIX' to change default fluxunit
 see description below

field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)/channels
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all c
 spw='0:5~61' (IF ID 0; channels 5 to 61)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
>>> spw expandable parameter
 restfreq -- the rest frequency
 available type includes float, int, string, list of float,
 list of int, list of string, and list of dictionary. the
 default unit of restfreq in case of float, int, or string
 without unit is Hz. string input can be a value only
 (treated as Hz) or a value followed by unit for which 'GHz',

```

```

'MHz','kHz',and 'Hz' are available.
a list can be used to set different rest frequencies for
each IF. the length of list input must be number of IFs.
dictionary input should be a pair of line name and
frequency with keys of 'name' and 'value', respectively.
values in the dictionary input follows the same manner as
as for single float or string input.
example: 345.796
 '1420MHz'
 [345.8, 347.0, 356.7]
 ['345.8MHz', '347.0MHz', '356.7MHz']
 [{'name':'CO','value':345}]
frame -- frequency reference frame
options: 'LSRK', 'TOPO', 'LSRD', 'BARY', 'GALACTO', 'LGROUP', 'CMB'
default: '' (keep current frame in data)
doppler -- doppler convention (effective only when spw is in
velocity unit)
options: 'RADIO', 'OPTICAL', 'Z', 'BETA', or 'GAMMA'
default: '' (keep current doppler setting in data)
timerange -- select data by time range
default: '' (use all)
example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
Note: YYYY/MM/DD can be dropped as needed:
timerange='09:14:00~09:54:00' # this time range
timerange='09:44:00' # data within one integration of time
timerange='>10:24:00' # data after this time
timerange='09:44:00+00:13:00' #data 13 minutes after time
this selection is in addition to the other selections to data
scan -- select data by scan numbers
default: '' (use all scans)
example: scan='21~23' (scan IDs 21,22,23)
this selection is in addition to the other selections to data
pol -- select data by polarization IDs
default: '' (use all polarizations)
example: pol='0,1' (polarization IDs 0,1)
this selection is in addition to the other selections to data
beam -- select data by beam IDs
default: '' (use all beams)
example: beam='0,1' (beam IDs 0,1)
this selection is in addition to the other selections to data
timeaverage -- average spectra over time
options: (bool) True, False
default: False
>>> timeaverage expandable parameter
tweight -- weighting for time averaging
options: 'var' (1/var(spec) weighted)

```

```

 'tsys' (1/Tsys**2 weighted)
 'tint' (integration time weighted)
 'tintsys' (Tint/Tsys**2)
 'median' (median averaging)
 default: 'tintsys'
 scanaverage -- average spectra within a scan number
 when True, spectra are NOT averaged over
 different scan numbers.
 options: (bool) True, False
 default: False
 polaverage -- average spectra over polarizations
 options: (bool) True, False
 default: False
 >>> polaverage expandable parameter
 pweight -- weighting for polarization averaging
 options: 'var' (1/var(spec) weighted)
 'tsys' (1/Tsys**2 weighted)
 default: 'tsys'
 interactive -- determines interactive masking
 options: (bool) True,False
 default: False
 example: interactive=True allows adding and deleting mask
 regions by drawing rectangles on the plot with mouse.
 Draw a rectangle with LEFT-mouse to ADD the region to
 the mask and with RIGHT-mouse to DELETE the region.
 outfile -- name of output file (ASCII) to save statistics
 default: '' (no output statistics file)
 example: 'stat.txt'
 format -- format string to print statistic values
 default: '3.3f'
 overwrite -- overwrite the statistics file if already exists
 options: (bool) True,False
 default: False

```

```

Returns: a Python dictionary of line statistics
 keys: 'rms','stddev','max','min','max_abscissa',
 'min_abscissa','sum','median','mean','totint','eqw'

```

#### ----- WARNING -----

For the GBT raw SDFITS format data as input:  
SDtasks are able to handle GBT raw SDFITS format data since the  
data filler is available. However, the functionality is not well

tested yet, so that there may be unknown bugs.

setjy-task.html

### **0.1.109 setjy**

Requires:

#### **Synopsis**

Fills the model column with the visibilities of a calibrator

#### **Description**

This task places the model visibility amp and phase associated with a specified clean components image into the model column of the data set. The flux density (I,Q,U,V) for a point source calibrator can be entered explicitly. Models are available for 3C48, 3C138, and 3C286 between 1.4 and 43 GHz. 3C147 is available above 13 GHz. These models are scaled to the precise frequency of the data. Only I models are presently available.

The location of the models is system dependent: At the AOC, the models are in the directory::/usr/lib/casapy/data/nrao/VLA/CalModels/ 3C286.L.im (egs)

setjy need only be run on the calibrator sources with a known flux density and/or model.

For Solar System Objects, model determination was updated and it is available via the 'Butler-JPL-Horizons 2012' standard.

Currently they are modeled as uniform temperature disks based on their ephemeris at the time of observation (note that this may oversimplify objects, in particular asteroids). Specify the name of the object in the 'field' parameter.

#### **Arguments**



|             |                                                                                                                           |
|-------------|---------------------------------------------------------------------------------------------------------------------------|
| Outputs     |                                                                                                                           |
| fluxd       | Dictionary containing flux densities and their errors.<br>allowed: any<br>Default: variant                                |
| Inputs      |                                                                                                                           |
| vis         | Name of input visibility file<br>allowed: string<br>Default:                                                              |
| field       | Field name(s)<br>allowed: string<br>Default:                                                                              |
| spw         | Spectral window identifier (list)<br>allowed: string<br>Default:                                                          |
| selectdata  | Other data selection parameters<br>allowed: bool<br>Default: False                                                        |
| timerange   | Time range to operate on (for usescratch=T)<br>allowed: any<br>Default: variant                                           |
| scan        | Scan number range (for usescratch=T)<br>allowed: any<br>Default: variant                                                  |
| intent      | Observation intent<br>allowed: string<br>Default:                                                                         |
| observation | Observation ID range (for usescratch=T)<br>allowed: any<br>Default: variant                                               |
| scalebychan | scale the flux density on a per channel basis or else on a per spw basis<br>allowed: bool<br>Default: True                |
| standard    | Flux density standard<br>allowed: string<br>Default: Perley-Butler 2013                                                   |
| model       | File location for field model<br>allowed: string<br>Default:                                                              |
| modimage    | File location for field model<br>allowed: string<br>Default:                                                              |
| listmodels  | List the available modimages for VLA calibrators or Tb models for Solar System objects<br>allowed: bool<br>Default: False |
| fluxdensity | Specified flux density [I,Q,U,V]; (-1 will lookup values)<br>allowed: any<br>Default: variant -1                          |
| spix        | Spectral index (including higher terms) of I fluxdensity<br>allowed: any<br>Default: variant 0.0                          |
| reffreq     | Reference frequency for spix                                                                                              |

## Returns

void

## Example

The task sets the model visibility amp and phase of a specified source (generally a calibrator). The simplest way is to enter the flux density (I,Q,U,V) explicitly, but this is valid only for a point source.

For an extended source, the clean model (image.model) can be specified and the model visibilities associated with this clean model is placed in the visibility model column.

Models are available for 3C48, 3C138, 3C286 between 1.4 and 43 GHz. 3C147 is available above 4 GHz. These models are scaled to the precise frequency of the data. Only I models are presently available.

The location of the models is system dependent: At the AOC and CV, the models are in the directory: `:/usr/lib/casapy/data/nrao/VLA/CalModels` or `:/usr/lib64/casapy/data/nrao/VLA/CalModels` (depending on whether 32 or 64 bit CASA was installed on the machine being used). In general (using Python), the stock models should be in `casa['dirs']['data'] + '/nrao/VLA/CalModels'` `setjy` also looks for models in the current directory before trying `casa['dirs']['data'] + '/nrao/VLA/CalModels'`.

`setjy` need only be run on the calibrator sources with a known flux density and/or model.

Solar System Objects are supported via the 'Butler-JPL-Horizons 2012' standard. This uses new brightness temperature models and a new flux calculation code that replace the 'Butler-JPL-Horizons 2010' standard. The older 'Butler-JPL-Horizons 2010' standard is still available for comparison. Users may want to use `predictcomp` task to see the differences. Currently they are modeled as uniform temperature disks based on their ephemerides at the time of observation (note that this may oversimplify objects, in particular asteroids). The object name is obtained from the 'field' parameter. Recognized objects are listed below, under 'standard'.

With standard='manual', flux densities and spectral index can be manually specified. As in the previous CASA versions, if fluxdensity[0] (Stokes I) is < 0, the default standard will be used to calculate flux density as a function of frequency.

The calculated flux densities are reported in the logger but also will be returned as a dictionary if you run as,  
fluxds = setjy(vis='ngc5921.ms', ...).

The dictionary have the structure,

```
{field name, {spw Id: {'fluxd': [I,Q,U,V] (flux densities in Jy)}}}
```

and the description is also in fluxds['format'].

Keyword arguments:

vis -- Name of input visibility file

default: none. example: vis='ngc5921.ms'

field -- Select field using field id(s) or field name(s).

default: ''=all fields, but run setjy one field at a time.

[run listobs to obtain the list id's or names of calibrators]

If field is a non-negative integer, it is assumed to be a field

index. Otherwise, it is taken to be a field name (case sensitive

- it must match the name as listed by listobs).

field='0~2'; field ids 0,1,2

field='0,4,5~7'; field ids 0,4,5,6,7

field='3C286,3C295'; field named 3C286 and 3C295

field = '3,4C\*'; field id 3, all names starting with 4C

spw -- Spectral window selection string.

default: '' = all spectral windows

Note that setjy only selects by spectral window, and ignores

channel selections. Fine-grained control could be achieved using

(and possibly constructing) a cube for modimage.

selectdata -- Other parameters for selecting part(s) of the MS

to operate on.

(Currently all time-oriented and most likely only of

interest when using a Solar System object as a calibrator.)

default: False

>>> selectdata=True expandable parameters

See help par.selectdata for more on these.

Note: for usescratch=False, timerange, scan, and observation

are ignored (i.e. time-specific virtual model is not possible.).

timerange -- Select data based on time range (when usescratch=T):

default: '' (all); examples,

timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'

Note: if YYYY/MM/DD is missing date defaults to first day in data set

timerange='09:14:0~09:54:0' picks 40 min on first day

timerange='25:00:00~27:30:00' picks 1 hr to 3 hr

30min on NEXT day

timerange='09:44:00' pick data within one integration

of time

timerange='>10:24:00' data after this time

For multiple MS input, a list of timerange strings can be used:

timerange=['09:14:0~09:54:0', '>10:24:00']

timerange='09:14:0~09:54:0'; apply the same timerange for all input MSes

scan -- Scan number range (when usescratch=T).

default: '' (all)

example: scan='1~5'

For multiple MS input, a list of scan strings can be used:

scan=['0~100', '10~200']

scan='0~100; scan ids 0-100 for all input MSes

Check 'go listobs' to insure the scan numbers are in order.

observation -- Observation ID range (when usescratch=T).

default: '' (all)

example: observation='1~5'

intent -- observation intent.

default: '' (all)

example: using wildcard characters,

intent="\*CALIBRATE\_AMPLI\*"

will match field(s) contains CALIBRATE\_AMPLI in a list of intent

WARNING: If a source with a specific field id has scans that can be distributed with intent selection, one should set usescratch=True. Otherwise, model of the source may be cleared and overwritten even if the p not selected by intent.

scalebychan -- This determines whether the fluxdensity set in the model is calculated on a per channel basis. If False then it only one fluxdensity value is calculated per spw. (Either way, all channels in spw are modified.) It is effectively True if fluxdensity[0] > 0.0.

default: True

standard -- Flux density standard, used if fluxdensity[0] < 0.0

default: 'Perley-Butler 2013'; example: standard='Baars'

Options: 'Baars',

'Perley 90',

```

'Perley-Taylor 95',
'Perley-Taylor 99',
'Perley-Butler 2010',
'Perley-Butler 2013',
'Scaife-Heald 2012',
'Butler-JPL-Horizons 2010',
'Butler-JPL-Horizons 2012',
'manual'
'fluxscale'

```

All but the last two are for extragalactic calibrators, and the final two are for Solar System objects. Note that Scaife-Heald 2012 is for the low frequencies (mostly valid for the frequency range, 30-300MHz).

#### Extragalactic calibrators:

Following source names and their common aliases are recognized. The last column shows which standards support for each source. Note that the task does not do exact matching of the name (also case insensitive) and it recognizes as long as the field name contains the string listed below (e.g. 'PKS 1934-638' works). For 3C Name, a space or an underscore between 3C and the number (e.g. '3C 286' and '3C\_286') also works. If the matching by the field name fails, the task tries to match by its position to the known calibrator list stored in the data directory (~/.data/nrao/VLA/standards/fluxscalibrator.data).

| 3C Name | B1950 Name | J2000 Name | Alt. J2000 Name | standards*    |
|---------|------------|------------|-----------------|---------------|
| 3C48    | 0134+329   | 0137+331   | J0137+3309      | 1,2,3,4,5,6,7 |
| 3C123   | 0433+295   | 0437+296   | J0437+2940      | 2             |
| 3C138   | 0518+165   | 0521+166   | J0521+1638      | 1,2,3,4,5,6   |
| 3C147   | 0538+498   | 0542+498   | J0542+4951      | 1,2,3,4,5,6,7 |
| 3C196   | 0809+483   | 0813+482   | J0813+4813      | 1,2,7         |
| 3C286   | 1328+307   | 1331+305   | J1331+3030      | 1,2,3,4,5,6,7 |
| 3C295   | 1409+524   | 1411+522   | J1411+5212      | 1,2,3,4,5,6,7 |
| -       | 1934-638   | -          | J1939-6342      | 1,3,4,5,6     |
| 3C380   | 1828+487   | 1829+487   | J1829+4845      | 7             |

\* supported in: 1 - Perley-Butler 2010, 2 - Perley-Butler 2013 (ref. Perley and Butler 2013, ApJS 204, 19), 3 - Perley-Taylor 99, 4 - Perley-Taylor 95, 5 - Perley 90, 6 - Baars, 7 - Scaife-Heald 2012

#### Solar system objects:

The 'Butler-JPL-Horizons 2012' standard is recommended over 'Butler-JPL-Horizons 2010' as the former uses updated models. Recognized Solar System objects (for 'Butler-JPL-Horizons 2012') are:

Planets: Venus, Mars, Jupiter, Uranus, Neptune

Moons: Jupiter: Io, Europa, Ganymede, Callisto  
Saturn: Titan

Asteroids: Ceres, Pallas\*\*, Vesta\*\*, Juno\*\*

- \* Venus: model for ~300MHz to 350GHz, no atmospheric lines (CO, H2O, HD0, etc)
- \* Mars: tabulated as a function of time and frequency (30 - 1000GHz) based on Rudy et al (1988), no atmospheric lines (CO, H2O, H2O2, HD0, etc)
- \* Jupiter: model for 30-1020GHz, does not include synchrotron emission
- \* Uranus: model for 60-1800GHz, contains no rings or synchrotron.
- \* Neptune: model for 2-2000GHz, the broad CO absorption line is included, but contains no rings or synchrotron.
- \* Titan: model for 53.3-1024.1GHz, include many spectral lines
  
- \*\* not recommended (The temperature is not yet adjusted for varying distance from the Sun. The model data can be scaled after running setjy, but it is an involved process.)

The 'field' parameter must match the case of the field name(s) in vis (as shown by listobs).

Flux density calculation with Solar System objects depends on ephemerides. The setjy task looks for the data in

```
os.getenv('CASAPATH').split()[0] + '/data/ephemerides/JPL-Horizons'.
```

If no ephemeris for the right object at the right time is present, the calculation will fail. Ask the helpdesk to make an ephemeris. The very adventurous and well versed in python can try it using CASA's recipes.ephemerides package:

```
import recipes.ephemerides as eph
help eph
```

CASA comes with ephemerides for several more objects, but they are intended for use with me.framecomet(), and are not (yet) suitable flux density calibrators. It is up to the observer to pick a good flux density calibrator (bright, spherical and featureless, on a circular orbit, in the right part of the sky, and not too resolved). Even some of the objects listed above may prove to require more sophisticated flux density models than are currently implemented in CASA. For many objects running casalog.filter('INF01') before running setjy will send more

information to the logger. The cookbook also has an appendix with descriptions of the models used by setjy (both extragalactic and Solar System).

```
>>> standard="Perley-Butler 2010" or "Perley-Butler 2013 expandable parameter
 model -- Model image (I only) for setting the model visibilities.
```

```

* Previously, this parameter is called 'modimage', now modimage *
* is deprecated. The setjy still accepts modimage but will be *
* removed in future releases. Please use the parameter, 'model' *
* instead. *

```

The model can be a cube, and its channels do not have to exactly match those of vis. It is recommended to use modimage for sources that are resolved by the observation, but the Butler-JPL-Horizons standard supplies a basic model of what several Solar System objects look like. default: '': do not use a model image.

Each field must be done separately when using a model image.

Both the amplitude and phase are calculated. At the AOC or CV, the models are located in casa['dirs']['data']  
+ '/nrao/VLA/CalModels/', e.g.  
/usr/lib/casapy/data/nrao/VLA/CalModels/3C286\_L.im  
lib64

If model does not start with '/', setjy will look for a match in '.', './CalModels', and any CalModels directories within the casa['dirs']['data'] tree (excluding certain branches).

Note that model should be deconvolved, i.e. a set of clean components instead of an image that has been convolved with a clean beam.

```
listmodels -- If True, do nothing but list candidates for model
 (for extragalactic calibrators) that are present on the system.
It looks for *.im* *.mod* in . including its sub-directories but skipping
any directory name start with ".",
CalModels, and CalModels directories in the casa['dirs']['data'] tree.
It does not check whether they are appropriate for the MS!
If standard='Butler-JPL-Horizons 2012', Tb models (frequency-depended
brightness temperature models) for Solar System objects used in the
standard. For standard='Butler-JPL-Horizons 2010', the recognized
Solar System objects are listed.
```

```
>>> standard="Perley-Butler 2013" expandable parameter
 interpolation -- method for interpolation ('nearest', 'linear', 'cubic',
 or 'spline') in time for the time variable sources (3C48,3C138,3C147).
 This parameter is ignored for other non-variable sources in the standard.
 default:'nearest'
```

```
>>> standard="Butler-JPL-Horizons 2012" expandable parameter
```

```
 useephemdir -- If True: use the direction from the ephemeris table for
 the solar system object.
 default: False -use the direction information in the MS(i.e. Field table)
```

```
>>> standard="manual" expandable parameters
 fluxdensity -- Specified flux density [I,Q,U,V] in Jy
 default: -1, uses [1,0,0,0] flux density for unrecognized sources,
 and standard flux densities for ones recognized by the default
 standard (Perley-Butler 2010).
 setjy will try to use the standard if fluxdensity is not
 positive.
```

Only one flux density can be specified at a time. The phases are set to zero.

```
example fluxdensity=-1 will use the default standard for recognized
 calibrators (like 3C286, 3C147 and 3C48) and insert 1.0
 for selected fields with unrecognized sources.
example field = '1'; fluxdensity=[3.2,0,0,0] will put in
 a flux density of I=3.2 for field='1'
```

At present (June 2000), this is the only method to insert a polarized flux density model.

```
example: fluxdensity=[2.63,0.21,-0.33,0.02]
 will put in I,Q,U,V flux densities of 2.63,0.21,-0.33,
 and 0.02, respectively, in the model column.
```

```
 spix -- Spectral index for I flux density (a float or a list of float values):
 where S = fluxdensity * (freq/reffreq)**(spix[0]+spix[1]*log(freq/reffreq)+..)
 Default: [] =>0.0 (no effect)
 Only used if fluxdensity is being used.
 N.B.: If fluxdensity is positive, and spix is nonzero, then reffreq
 must be set too! (See below)
```

It is applied in the same way to all polarizations, and does not account for Faraday rotation or depolarization.

Example: [-0.7, -0.15] for alpha and a curvature term



refffreq -- The reference frequency for spix, given with units.  
 Default: '1GHz'; this is only here to prevent division by 0!  
 N.B.: If the flux density is being scaled by spectral index,  
 then refffreq must be set to whatever reference frequency is  
 correct for the given fluxdensity and spix. It cannot be  
 determined from vis. On the other hand, if spix is 0, then any  
 positive frequency can be used (and ignored).

Examples: '86.0GHz', '4.65e9Hz'

polindex -- Coefficients of the frequency-dependent linear polarization index ( $P$ ),  
 expressed as,  

$$\text{pol. index} = \sqrt{Q^2 + U^2} / I = c_0 + c_1 * ((\text{freq} - \text{refffreq}) / \text{refffreq}) + c_2 * ((\text{freq} - \text{refffreq}) / \text{refffreq})^2$$
  
 When Q and U flux densities are given fluxdensity, c0 is determined  
 these flux densities and the entry for c0 in polindex is ignored. On  
 fluxdensity can be set to 0.0 and then polindex[0] and polangle[0] a  
 at refffreq.

Default: []

Example: [0.2, -0.01] (= [c0, c1])

polangle -- Coefficients of the frequency-dependent linear polarization angle ( $\theta$ )  

$$\text{pol. angle} = 0.5 * \arctan(U/Q) = d_0 + d_1 * ((\text{freq} - \text{refffreq}) / \text{refffreq}) + d_2 * ((\text{freq} - \text{refffreq}) / \text{refffreq})^2$$
  
 When Q and U flux densities are given fluxdensity, d0 is determined  
 these flux densities and the entry for d0 in polangle is ignored. On  
 fluxdensity can be set to 0.0 and then polindex[0] and polangle[0] a  
 at refffreq.

Default: []

Example: [0.57, 0.2] (= [d0, d1])

rotmeas -- rotation measure (in rad/m<sup>2</sup>)

>>> standard="fluxscale" expandable parameters

fluxdict -- Output dictionary from fluxscale

Using the flexibly results, the flux density, spectral index,  
 and reference frequency are extracted and set to fluxdensity,  
 spix, and refffreq parameters, respectively.

The field and spw selections can be used to specify subset of  
 the fluxdict to be used to set the model. If they are left as  
 default (field="", spw="") all fields and/or spws in  
 the fluxdict (but those spws with fluxd=-1 will be skipped) are  
 used.

default: {}

usescratch -- If False: 'virtual' model is created. The model information is saved

either in the SOURCE\_MODEL column in the SOURCE table (if one exists) or in the key of the main table in the MS and model visibilities are evaluated on the fly when calibration or plotting in plotms.

If True: the model visibility will be evaluated and saved on disk in the MODEL\_DATA column. This will increase your ms in size by a factor of 1.5 (w.r.t. the case when you only have the DATA and the CORRECTED\_DATA column). Use True if you need to interact with the MODEL\_DATA in python, say. Also, use True if you need finer than field and selections using scans/time (and when use with intent selection, please see WARNING intent parameter description).

\*By running usescratch=T, it will remove the existing virtual model from previous runs. usescratch=F will not remove the existing MODEL\_DATA but in subsequent process the virtual model with matching field and spw combination will be used if it exists regardless of the presence of the MODEL\_DATA column.

default: False

Returned dictionary:

When the setjy task is executed as setjy(vis='', ..), the flux densities used to set the model are returned as a Python dictionary with the format, {field Id: {spw Id: {fluxd: [I,Q,U,V] in Jy}, 'fieldName':field name }}, where field Id and spw Id are in string type.

`simalma-task.html`

### **0.1.110 simalma**

Requires:

#### **Synopsis**

Simulation task for ALMA

#### **Description**

This task simulates ALMA observation including 12-m, ACA 7-m and total power arrays, and images and analyzes simulated data.

This task makes multiple calls to `simobserve` (to calculate visibilities and total power spectra), followed by gridding of total power spectra (if total power is requested), concatenation of the simulated visibilities, calls to the `simanalyze` task for visibility inversion and deconvolution and calculation of difference and fidelity images, and feathering of single dish and interferometric data.

These steps may not all be familiar to new users, so the `simalma` task runs by default in a "dryrun" mode, in which it assesses the user's input parameters and sky model, and prints an informational report including the required calls to other CASA tasks, both to the screen and to a text file in the project directory (defined below).

The user can modify their parameters based on the information, then either run with `dryrun=False` to actually call the other tasks to create the simulated data, or run the other tasks individually one at a time to better understand and control the process.

NOTE The ALMA project is refining the optimal method of combining the three types of data. If that best practice is changed after this release of CASA, the user can control the process by modifying the calls to the other CASA tasks.

More information is available at

[http://casaguides.nrao.edu/index.php?title=Simulating\\_Observations\\_in\\_CASA](http://casaguides.nrao.edu/index.php?title=Simulating_Observations_in_CASA)

Please contact the CASA helpdesk with any questions.

#### **Arguments**

|              |                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| Inputs       |                                                                                                                       |
| project      | root prefix for output file names<br>allowed: string<br>Default: sim                                                  |
| dryrun       | dryrun=True will only produce the informative report,<br>not run simobserve/analyze<br>allowed: bool<br>Default: True |
| skymodel     | model image to observe<br>allowed: string<br>Default:                                                                 |
| inbright     | scale surface brightness of brightest pixel e.g.<br>"1.2Jy/pixel"<br>allowed: string<br>Default:                      |
| indirection  | set new direction e.g. "J2000 19h00m00 -40d00m00"<br>allowed: string<br>Default:                                      |
| incell       | set new cell/pixel size e.g. "0.1arcsec"<br>allowed: string<br>Default:                                               |
| incenter     | set new frequency of center channel e.g. "89GHz" (re-<br>quired even for 2D model)<br>allowed: string<br>Default:     |
| inwidth      | set new channel width e.g. "10MHz" (required even for<br>2D model)<br>allowed: string<br>Default:                     |
| complist     | componentlist to observe<br>allowed: string<br>Default:                                                               |
| compwidth    | bandwidth of components<br>allowed: string<br>Default: "8GHz"                                                         |
| setpointings | allowed: bool<br>Default: True                                                                                        |
| ptgfile      | list of pointing positions<br>allowed: string<br>Default: \$project.ptg.txt                                           |
| integration  | integration (sampling) time<br>allowed: string<br>Default: 10s                                                        |
| direction    | "J2000 19h00m00 -40d00m00" or "" to center on model<br>allowed: stringArray<br>Default:                               |
| mapsize      | angular size of map or "" to cover model<br>allowed: stringArray<br>Default:                                          |
| antennalist  | antenna position files of ALMA 12m and 7m arrays<br>allowed: stringArray<br>Default: alma.cycle1.1.cfg aca.cycle1.cfg |
| hourangle    | hour angle of observation center e.g. -3:00:00, or "tran-<br>sit"                                                     |

## Returns

bool

## Example

```

Parameters:

project -- root filename for all output files. A subdirectory will be
 created, and all created files will be placed in that subdirectory
 including the informational report.

skymodel -- input image (used as a model of the sky)
 * simalma requires a CASA or fits image. If you merely have a grid of
 numbers, you will need to write them out as fits or write a
 CASA script to read them in and use the ia tool to create an image
 and insert the data.

 * simalma does NOT require a coordinate system in the header. If the
 coordinate information is incomplete, missing, or you would like to
 override it, set the appropriate "in" parameters. NOTE that setting
 those parameters simply changes the header values, ignoring
 any values already in the image. No regridding is performed.

 * If you have a proper Coordinate System, simalma will do its best to
 generate visibilities from that, and then create a synthesis image
 according to the specified user parameters.

 * You can manipulate an image header manually with the "imhead" task.

inbright -- peak brightness in Jy/pixel, or "" for unchanged
 * NOTE: "unchanged" will take the numerical values in your image
 and assume they are in Jy/pixel, even if it says some other unit
 in the header.
indirection -- central direction, or "" for unchanged
incell -- spatial pixel size, or "" for unchanged
incenter -- frequency of center channel e.g. "89GHz", or "" for unchanged
inwidth -- width of channels, or "" for unchanged - this should be a
 string representing a quantity with units e.g. "10MHz"
 * NOTE: only works reliably with frequencies, not velocities
 * NOTE: it is not possible to change the number of spectral planes
 of the sky model, only to relabel them with different frequencies
```

That kind of regridding can be accomplished with the CASA toolkit.

```

complist -- component list model of the sky, added to or instead of skymodel
compwidth -- bandwidth of components; if simulating from components only,
 this defines the bandwidth of the MS and output images

```

```

setpointings -- calculate a map of pointings, or if false, the user should
 provide a ptgfile
* if graphics are on, display the pointings shown on the model image
* if a list of directions is not specified, observations with the
 ALMA 12m and ACA 7m arrays will observe a region of size "mapsize"
 using the same hexagonal algorithm as the ALMA OT, with Nyquist
 sampling.
* The total power array maps a slightly (+1 primary beam) larger area
 than the 12m array does, to improve later image combination.
 It samples the region with lattice grids of spacing 0.33 lambda/D.

```

```
ptgfile -- a text file specifying directions in the same
 format as the example, and optional integration times, e.g.
 #Epoch RA DEC TIME(optional)
 J2000 23h59m28.10 -019d52m12.35 10.0
* if the time column is not present in the file, it will use
 "integration" for all pointings.
* NOTE: at this time the file should contain only science pointings:
 simalma will observe these until totaltime is used up.
integration --- Time interval for each integration e.g. '10s'
* NOTE: to simulate a "scan" longer than one integration, use
 setpointings to generate a pointing file, and then edit the
 file to increase the time at each point to be larger than
 the parameter integration time.
direction -- mosaic center direction e.g. 'J2000 19h00m00 -40d00m00'
* can optionally be a list of pointings
* otherwise simalma will pack mapsize with grids proper for the
 array (see below).
mapsize -- angular size of map e.g. '40arcsec' or ['1arcmin','30arcsec']
* set to "" to span the model image

```

```
antennalist -- vector of ascii files containing antenna positions,
 one for each configuration of 7m or 12m dishes.
* NOTE: In this task, it should be an ALMA configuration.
* standard arrays are found in your CASA data repository,
 os.getenv("CASAPATH").split()[0]+"/data/alma/simmos/"
* a string of the form "alma;0.5arcsec" will be parsed into a
 12m ALMA configuration - see casaguides.nrao.edu
```

```

 * examples: ['alma.cycle2.5.cfg','aca.cycle2.i.cfg']
 ['alma.cycle1;0.3arcsec','alma.cycle1.1.cfg','aca.i.cfg']
hourangle -- hour angle of observation e.g. '-3h'
 * note that if you don't add a unit, it will assume seconds.
totaltime --- total time of observations. This should either be a scalar
 time quantity expressed as a string e.g. '1h', '3600sec', '10min',
 or a vector of such quantities, corresponding to the elements of
 the antennalist vector, e.g. ['5min','20min','3h']. If you
 specify a scalar, that will be used for the highest resolution
 12m configuration in antennalist, and any lower resolution 12m
 configurations, any 7m configurations, and any TP configurations
 will have observing times relative to totaltime of 0.5, 2, and 4,
 respectively.

tpnant -- the number of total power antennas to use in simulation.
tptime -- if tpnant>0, the user must specify the observing time for
 total power as a CASA quantity e.g. '4h'.
 * NOTE: in CASA 4.2 this is not broken up among multiple days -
 a 20h track will include observations below the horizon,
 which is probably not what is desired.

pwv -- precipitable water vapor if constructing an atmospheric model.
 Set 0 for noise-free simulation. When pwv>0, thermal noise is
 applied to the simulated data.
 * J. Pardo's ATM library will be used to construct an atmospheric
 profile for the ALMA site:
 altitude 5000m, ground pressure 650mbar, relhum=20%,
 a water layer of pwv at altitude of 2km,
 the sky brightness temperature returned by ATM, and internally
 tabulated receiver temperatures.
 See the documents of simobserve for more details.

image -- option to invert and deconvolve the simulated measurement set(s)
 * NOTE: interactive clean or more parameters than the subset visible
 here are available by simply running the clean task directly.
 * if graphics turned on, display the clean image and residual image
 * uses Cotton-Schwab clean for single fields and Mosaic gridding
 for multiple fields (with Clark PSF calculation in minor cycles).
imsize -- image size in spatial pixels (x,y)
 0 or -1 will use the model image size; example: imsize=[500,500]
imdirection -- phase center for synthesized image. default is to
 center on the sky model.
cell -- cell size e.g. '10arcsec'. "" defaults to the skymodel cell
niter -- number of clean/deconvolution iterations, 0 for no cleaning
threshold -- flux level to stop cleaning

```

graphics -- view plots on the screen, saved to file, both, or neither  
verbose -- print extra information to the logger and terminal  
overwrite -- overwrite existing files in the project subdirectory  
-----

Please see the documents of simobserve and simanalyze for  
the list of outputs produced.



### 0.1.111 simobserve

Requires:

#### Synopsis

visibility simulation task

#### Description

This task simulates interferometric or total power measurement sets. It is currently optimized for JVLA and ALMA, although many observatories are included, and adding your own is simply a matter of providing an antenna location file (see below).

simobserve is meant to work in conjunction with the simanalyze task - Calling simobserve one more times will produce simulated measurement set(s), which are then gridded, inverted and deconvolved into output simulated images using simanalyze.

ALMA users are encouraged to use the simalma task, which provides additional information on the multiple simobserve and simanalyze calls required to simulate an ALMA observation which may consist of 12m interferometric, 7m interferometric, and 12m total power data.

More information and examples are available at

[http://casaguides.nrao.edu/index.php?title=Simulating\\_Observations\\_in\\_CASA](http://casaguides.nrao.edu/index.php?title=Simulating_Observations_in_CASA)

Please contact CASA experts with any questions.

#### Arguments

|                 |                                                                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs          |                                                                                                                                                |
| project         | root prefix for output file names<br>allowed: string<br>Default: sim                                                                           |
| skymodel        | model image to observe<br>allowed: string<br>Default:                                                                                          |
| inbright        | scale surface brightness of brightest pixel e.g. "1.2Jy/pixel"<br>allowed: string<br>Default:                                                  |
| indirection     | set new direction e.g. "J2000 19h00m00 -40d00m00"<br>allowed: string<br>Default:                                                               |
| incell          | set new cell/pixel size e.g. "0.1arcsec"<br>allowed: string<br>Default:                                                                        |
| incenter        | set new frequency of center channel e.g. "89GHz" (required even for 2D model)<br>allowed: string<br>Default:                                   |
| inwidth         | set new channel width e.g. "10MHz" (required even for 2D model)<br>allowed: string<br>Default:                                                 |
| complist        | componentlist to observe<br>allowed: string<br>Default:                                                                                        |
| compwidth       | bandwidth of components<br>allowed: string<br>Default: "8GHz"                                                                                  |
| setpointings    | allowed: bool<br>Default: True                                                                                                                 |
| ptgfile         | list of pointing positions<br>allowed: string<br>Default: \$project.ptg.txt                                                                    |
| integration     | integration (sampling) time<br>allowed: string<br>Default: 10s                                                                                 |
| direction       | "J2000 19h00m00 -40d00m00" or "" to center on model<br>allowed: stringArray<br>Default:                                                        |
| mapsize         | angular size of map or "" to cover model<br>allowed: stringArray<br>Default:                                                                   |
| maptype         | hexagonal, square (raster), ALMA, etc<br>allowed: 577 string<br>Default: hexagonal                                                             |
| pointingspacing | spacing in between pointings or "0.25PB" or "" for ALMA default $INT = \lambda/D/\sqrt{3}$ , $SD = \lambda/D/3$<br>allowed: string<br>Default: |
| caldirection    | pt source calibrator [experimental]<br>allowed: string                                                                                         |

## Returns

bool

## Example

```

project -- the root filename for all output files.

skymodel -- input image (used as a model of the sky)
 * simalma requires a CASA or fits image. If you merely have a grid of
 numbers, you will need to write them out as fits or write a
 CASA script to read them in and use the ia tool to create an image
 and insert the data.

 * simalma does NOT require a coordinate system in the header. If the
 coordinate information is incomplete, missing, or you would like to
 override it, set the appropriate "in" parameters. NOTE that setting
 those parameters simply changes the header values, ignoring
 any values already in the image. No regridding is performed.

 * If you have a proper Coordinate System, simalma will do its best to
 generate visibilities from that, and then create a synthesis image
 according to the specified user parameters.

 * You can manipulate an image header manually with the "imhead" task.

inbright -- peak brightness in Jy/pixel, or "" for unchanged
 * NOTE: "unchanged" will take the numerical values in your image
 and assume they are in Jy/pixel, even if it says some other unit
 in the header.

indirection -- central direction, or "" for unchanged
incell -- spatial pixel size, or "" for unchanged
incenter -- frequency of center channel e.g. "89GHz", or "" for unchanged
inwidth -- width of channels, or "" for unchanged - this should be a
 string representing a quantity with units e.g. "10MHz"
 * NOTE: only works reliably with frequencies, not velocities
 * NOTE: it is not possible to change the number of spectral planes
 of the sky model, only to relabel them with different frequencies
 That kind of regridding can be accomplished with the CASA toolkit.

complist -- component list model of the sky, added to or instead of skymodel
 see http://casaguides.nrao.edu/index.php?title=Simulation_Guide_Component_Lists_%28
```

```

compwidth -- bandwidth of components; if simulating from components only,
 this defines the bandwidth of the MS and output images

setpointings -- calculate a map of pointings, or if false, provide ptgfile
 * if graphics are on, display the pointings shown on the model image
ptgfile -- a text file specifying directions in the following
 format, with optional integration times, e.g.
 #Epoch RA DEC TIME(optional)
 J2000 23h59m28.10 -019d52m12.35 10.0
 * if the time column is not present in the file, it will use
 "integration" for all pointings.
 * NOTE: at this time the file should contain only science pointings:
 simobserve will observe these, then optionally the calibrator,
 then the list of science pointings again, etc, until totaltime
 is used up.
integration --- Time interval for each integration e.g. '10s'
 * NOTE: to simulate a "scan" longer than one integration, use
 setpointings to generate a pointing file, and then edit the
 file to increase the time at each point to be larger than
 the parameter integration time.
direction -- mosaic center direction e.g. 'J2000 19h00m00 -40d00m00'
 * can optionally be a list of pointings
 * otherwise simobserve will pack mapsize according to maptype
mapsize -- angular size of map
 * set to "" to span the model image
maptype -- hexagonal, square (rectangular raster),
 "ALMA" for the same hex algorithm as the ALMA Cycle 1 OT
 or "ALMA2012" for the algorithm used in the Cycle 0 OT
pointingspacing -- spacing in between beams e.g. '1arcsec'
 "0.25PB" to use 1/4 of the primary beam FWHM,
 "nyquist" will use $\lambda/d/2$,
 "" will use $\lambda/d/\sqrt{3}$ for INT, $\lambda/d/3$ for SD

obsmode -- observation mode to calculate visibilities from a skymodel image
 (which may have been modified above), an optional component list,
 and a pointing file (which also may have been generated above)
 * this parameter takes two possible values:
 - interferometer (or i)
 - singledish (or s)
 * if graphics are on, this observe step will display the array
 (similar to plotants), the uv coverage, the synthesized (dirty) beam,
 and ephemeris information
 * if simulating from component list, you should specify "compwidth",
 the desired bandwidth. There is not currently a way to specify the
 spectrum of a component, so simulations from a componentlist only
 will be continuum (1 chan)

```

```

refdate -- date of simulated observation eg: '2014/05/21'
hourangle -- hour angle of observation e.g. '-3h'
 * note that if you don't add a unit, it will assume hours
totaltime --- total time of observation e.g. '7200s' or if a number without
 units, interpreted as the number of times to repeat the map
antennalist -- ascii file containing antenna positions.
 each row has x y z coordinates and antenna diameter;
 header lines are required to specify the observatory name
 and coordinate system e.g.
 # observatory=ALMA
 # coordsys=UTM
 # datum=WGS84
 # zone=19
 * standard arrays are found in your CASA data repository,
 os.getenv("CASAPATH").split()[0]+"/data/alma/simmos/"
 * if "", simobserve will not produce an interferometric MS
 * a string of the form "alma;0.5arcsec" will be parsed into a full
 12m ALMA configuration. This only works for full ALMA and may fail
 to find the standard configuration files on some systems
 - see casaguides.nrao.edu for more information.
caldirection -- an unresolved calibrator can be observed
 interleaved with the science pointings.
 * The calibrator is implemented as a point source clean component
 with this specified direction and flux=calflux
sdant -- the index of the antenna in the list to use for total
 power. defaults to the first antenna on the list.

thermalnoise -- add thermal noise
 * this parameter takes two possible values:
 - tsys-atm: J. Pardo's ATM library will be used to construct an
 atmospheric profile for the ALMA site:
 altitude 5000m, ground pressure 650mbar, relhum=20%,
 a water layer of user_pwv at altitude of 2km,
 the sky brightness temperature returned by ATM,
 and internally tabulated receiver temperatures
 - tsys-manual: instead of using the ATM model, specify the zenith
 sky brightness and opacity manually. Noise is added and then
 the visibility flux scale is referenced above the atmosphere.
 * In either mode, noise is calculated using an antenna spillover
 efficiency of 0.96, taper of 0.86,
 surface accuracy of 25 and 300 microns for ALMA and EVLA
 respectively (using the Ruze formula for surface efficiency),
 correlator efficiencies of 0.95 and 0.91 for ALMA and EVLA,
 receiver temperatures for ALMA of
 17, 30, 37, 51, 65, 83,147,196,175,230 K interpolated between
 35, 75,110,145,185,230,345,409,675,867 GHz,

```

```

 for EVLA of
 500, 70, 60, 55, 100, 130, 350 K interpolated between
 0.33,1.47,4.89,8.44,22.5,33.5,43.3 GHz,
 for SMA of
 67, 116, 134, 500 K interpolated between
 212.,310.,383.,660. GHz
 * These are only approximate numbers and do not take into account
 performance at edges of receiver bands, neither are they guaranteed
 to reflect the most recent measurements. Caveat emptor and use the
 sm tool to add noise if you want more precise control.
t_ground -- ground/spillover temperature in K
user_pwv -- precipitable water vapor if constructing an atmospheric model
t_sky -- atmospheric temperature in K [for tsys-manual]
tau0 -- zenith opacity at observing frequency [for tsys-manual]
 * see casaguides.nrao.edu for more information on noise,
 in particular how to add a phase screen using the toolkit
seed -- random number seed for noise generation

leakage -- add cross polarization corruption of this fractional magnitude

graphics -- view plots on the screen, saved to file, both, or neither
verbose -- print extra information to the logger and terminal
overwrite -- overwrite existing files in the project subdirectory

Output produced: (not all will always exist, depending on input parameters)
To support different runs with different arrays, the names have the
configuration name from antennalist appended.

project.[cfg].skymodel = 4d input sky model image (optionally) scaled
project.[cfg].skymodel.flat.regrid.conv = input sky regridded to match the
 output image, and convolved with the output clean beam
project.[cfg].skymodel.png = diagnostic figure of sky model with pointings

project.[cfg].ptg.txt = list of mosaic pointings
project.[cfg].quick.psf = psf calculated from uv coverage
project.[cfg].ms = noise-free measurement set
project.[cfg].noisy.ms = corrupted measurement set
project.[cfg].observe.png = diagnostic figure of uv coverage and
 visibilities

project.[cfg].simobserve.last = saved input parameters for simobserve task

```



### **0.1.112   simanalyze**

Requires:

#### **Synopsis**

image and analyze measurement sets created with simobserve

#### **Description**

This task is for imaging and analyzing measurement sets (MSs) simulated with simobserve or simalma.

#### **Arguments**



|             |                                                                                                                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                                                                                                                                                                                 |
| project     | <p>root prefix for output file names</p> <p>allowed: string</p> <p>Default: sim</p>                                                                                                                                                                             |
| image       | <p>(re)image <i>project.*.mstop</i>project.image</p> <p>allowed: bool</p> <p>Default: True</p>                                                                                                                                                                  |
| imagename   | <p>simulation output image to analyze (default = first \$project/*.image found)</p> <p>allowed: string</p> <p>Default: default</p>                                                                                                                              |
| skymodel    | <p>skymodel image to analyze (the .skymodel image created by simobserve or simalma and used by one of those tasks to create an MS; if unspecified, will try to find one similar to your specified output image name)</p> <p>allowed: string</p> <p>Default:</p> |
| vis         | <p>Measurement Set(s) to image</p> <p>allowed: string</p> <p>Default: default</p>                                                                                                                                                                               |
| modelimage  | <p>lower resolution prior image to use in clean e.g. existing total power image</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                      |
| imsize      | <p>output image size in pixels (x,y) or 0 to match model</p> <p>allowed: intArray</p> <p>Default: 00</p>                                                                                                                                                        |
| imdirection | <p>set output image direction, (otherwise center on the model)</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                       |
| cell        | <p>cell size with units e.g. "10arcsec" or "" to equal model</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                         |
| interactive | <p>interactive clean? (make sure to set niter&gt;0 also)</p> <p>allowed: bool</p> <p>Default: False</p>                                                                                                                                                         |
| niter       | <p>maximum number of iterations (0 for dirty image)</p> <p>allowed: int</p> <p>Default: 0</p>                                                                                                                                                                   |
| threshold   | <p>flux level (+units) to stop cleaning</p> <p>allowed: string</p> <p>Default: 0.1mJy</p>                                                                                                                                                                       |
| weighting   | <p>weighting to apply to visibilities. briggs will use robust=0.5</p> <p>allowed: string</p> <p>Default: natural</p>                                                                                                                                            |
| mask        | <p>Cleanbox(es), mask image(s), region(s), or a level</p> <p>allowed: any</p> <p>Default: variant</p>                                                                                                                                                           |
| outertaper  | <p>uv-taper on outer baselines in uv-plane</p> <p>allowed: stringArray</p> <p>Default:</p>                                                                                                                                                                      |
| pbcor       | <p>correct the output of synthesis images for primary beam response?</p>                                                                                                                                                                                        |

## Returns

void

## Example

```
* "project" needs to be the directory of results generated by running
simobserve or simalma. In particular $project/$project.skymodel
will be required in order to compare output and input images.
```

```

```

```
mode image=True:
```

```
* One should input one or more simulated MSs using the "vis" parameter.
These can include a total power MS.
Simanalyze will grid any total power MS,
clean (invert and deconvolve) any interferometric MSs,
and feather the results.
```

```
* the "vis" parameter:
```

- example: single MS: vis="mysim.alma.out03.ms"
- example: multiple MSs: vis=["mysim.alma.out03.ms","mysim.aca.tp.ms"]
- one can use '\$project' and let the task automatically replace it with the project name, e.g., vis="\$project.noisy.ms,\$project.noisy.sd.ms". However, note that if you created measurement set(s) using simobserve, MS names will include the configuration, e.g. \$project.alma\_out20.noisy.ms
- setting "vis" to "default" will find and attempt to image all measurement sets (interferometric and single dish) in the project directory

```
* Sometimes it is preferable to grid the single dish MS using the
sdimaging task for more control. In that case one can input
the resulting single dish imaging under "featherimage", only
put interferometric MSs in "vis", and simanalyze will clean the
interferometric and feather with your "featherimage".
```

```
* Sometimes it is preferable to use a low resolution (single dish or
synthesis) image as a prior model during clean deconvolution
of a higher resolution interferometric MS. That is accomplished
by putting the low-resolution image in "modelimage" and the MS
to be deconvolved in "vis". NOTE: This is not the original skymodel
that was used in simobserve or simalma. It is recommended to
leave this blank unless the user is familiar with using a prior
```

in clean deconvolution. (see casaguides) NOTE 2: modelimage will not be used if the MS to be imaged is total power.

- \* uses Cotton-Schwab clean for single fields and Mosaic gridding for multiple fields (with Clark PSF calculation in minor cycles).
- \* interactive clean or use of more parameters than the subset visible here are available by simply running the clean task directly, then using simanalyze in the mode image=False (see below).
- \* if graphics are turned on, this step will display the clean image and residual image
- \* the "mask" parameter:  
Specification of cleanbox(es), mask image(s), primary beam coverage level, and/or region(s) to be used for cleaning. clean tends to perform better, and is less likely to diverge, if the clean component placement is limited by a mask to where real emission is expected to be. e.g. pixel ranges mask=[110,110,150,145], filename of mask image mask='myimage.mask', or a file with mask regions -- see help for the clean task for more information.

-----  
mode image=False:

- \* Sometimes the user has created a synthesized image themselves, most likely using the clean task, perhaps along with sdimagging and feather, or a previous call to simanalyze with image=True
- \* The user should input that simulated image as "imagenname". It will have suffix .image if created by clean, simanalyze, or simalma
- \* simanalyze will attempt to find an appropriate skymodel image - this is the \*.skymodel image created by simobserve or simalma, the (optionally rescaled) original sky model, which was used to create the measurement set.  
simanalyze will look in the project directory, but if there are multiple skymodels present it may not find the right one, so the "skymodel" parameter allows explicit specification.

-----  
mode analyze=True is used to create an image of the difference between the input skymodel and the simulated output image (whether that output image is being generated in the same call to simanalyze, with image=True, or has already been generated, and simanalyze is being called with image=False).

showuv -- display uv coverage

showpsf -- display synthesized (dirty) beam (ignored in single dish simulation)

```

showmodel -- display sky model at original resolution
showconvolved -- display sky model convolved with output beam
showclean -- display the synthesized image
showresidual -- display the clean residual image (ignored in single dish simulation)
showdifference -- display difference between output cleaned image and
 input model sky image convolved with output clean beam
showfidelity -- display fidelity image
 fidelity = abs(input) / max[abs(input-output), 0.7*rms(output)]

```

Note that the RMS is calculated in the lower quarter of the image.  
This is likely not the best choice, so you are encouraged to  
measure RMS yourself in an off-source region using the viewer.

dryrun=True is an advanced technical mode only useful for interferometric  
(not single dish) data.

-----  
Output produced: (not all will always exist, depending on input parameters)  
To support different runs with different arrays, the names have the  
configuration name from antennalist appended.  
-----

```

project.[cfg].skymodel.flat.regrid.conv = input sky regridded to match
 the output image, and convolved with the output clean beam

project.[cfg].image = synthesized image
project.[cfg].flux.pbcoverage = primary beam correction for mosaic image
project.[cfg].residual = residual image after cleaning
project.[cfg].clean.last = parameter file of what parameters were used in
 the clean task
project.[cfg].psf = synthesized (dirty) beam calculated from weighted uv
 distribution
project.[cfg].image.png = diagnostic figure of clean image and residual

project.[cfg].fidelity = fidelity image
project.[cfg].analysis.png = diagnostic figure of difference and fidelity

project.[cfg].simanalyze.last = saved input parameters for simanalyze task

```

-----  
Please see <http://casaguides.nrao.edu>, and contact the CASA helpdesk  
with questions.

slsearch-task.html

### **0.1.113 slsearch**

Requires:

#### **Synopsis**

Search a spectral line table.

#### **Arguments**

|            |                                                                                                                                 |
|------------|---------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                 |
| tablename  | Input spectral line table name to search. If not specified, use the default table in the system.<br>allowed: string<br>Default: |
| outfile    | Results table name. Blank means do not write the table to disk.<br>allowed: string<br>Default:                                  |
| freqrange  | Frequency range in GHz.<br>allowed: doubleArray<br>Default: 84,90                                                               |
| species    | Species to search for.<br>allowed: stringArray<br>Default:                                                                      |
| reconly    | List only NRAO recommended frequencies.<br>allowed: bool<br>Default: False                                                      |
| chemnames  | Chemical names to search for.<br>allowed: stringArray<br>Default:                                                               |
| qns        | Resolved quantum numbers to search for.<br>allowed: stringArray<br>Default:                                                     |
| intensity  | CDMS/JPL intensity range. -1 -> do not use an intensity range.<br>allowed: doubleArray<br>Default: -1                           |
| smu2       | $S^*\mu^*\mu$ range in Debye**2. -1 -> do not use an $S^*\mu^*\mu$ range.<br>allowed: doubleArray<br>Default: -1                |
| loga       | log(A) (Einstein coefficient) range. -1 -> do not use a loga range.<br>allowed: doubleArray<br>Default: -1                      |
| el         | Lower energy state range in Kelvin. -1 -> do not use an el range.<br>allowed: doubleArray<br>Default: -1                        |
| eu         | Upper energy state range in Kelvin. -1 -> do not use an eu range.<br>allowed: doubleArray<br>Default: -1                        |
| rrlinclude | Include RRLs in the result set?<br>allowed: bool<br>Default: True                                                               |
| rrlonly    | Include only RRLs in the result set?<br>allowed: bool<br>Default: False                                                         |
| verbose    | List result set to logger (and optionally logfile)?<br>allowed: bool<br>Default: False                                          |
| logfile    | List result set to this logfile (only used if verbose=True).<br>allowed: string<br>Default: ""                                  |

## Returns

bool

## Example

### PARAMETER SUMMARY

|            |                                                                                           |
|------------|-------------------------------------------------------------------------------------------|
| tablename  | Input spectral line table name to search. If not specified, use the default table.        |
| outfile    | Results table name. Blank means do not write the table to disk.                           |
| freqrange  | Frequency range in GHz.                                                                   |
| species    | Species to search for.                                                                    |
| reconly    | List only NRAO recommended frequencies.                                                   |
| chemnames  | Chemical names to search for.                                                             |
| qns        | Resolved quantum numbers to search for.                                                   |
| intensity  | CDMS/JPL intensity range. -1 -> do not use an intensity range.                            |
| smu2       | S*mu*mu range in Debye**2. -1 -> do not use an S*mu*mu range.                             |
| loga       | log(A) (Einstein coefficient) range. -1 -> do not use a loga range.                       |
| el         | Lower energy state range in Kelvin. -1 -> do not use an el range.                         |
| eu         | Upper energy state range in Kelvin. -1 -> do not use an eu range.                         |
| rrlinclude | Include RRLs in the result set?                                                           |
| rrlonly    | Include only RRLs in the result set?                                                      |
| verbose    | List result set to logger (and optionally logfile)?                                       |
| logfile    | List result set to this logfile (only used if verbose=True).                              |
| append     | If true, append to logfile if it already exists, if false overwrite logfile if it exists. |

Search the specified spectral line table. The results table can be written to disk by specifying outfile. If outfile is not specified (ie outfile=""), no table is created. Because Splatalogue does not include RRLs for loga, eu, and el for radio recombination lines (rrls), one must specify to include RRLs in the search. If rrlinclude is set to True, RRLs will be included ignoring any filters on intensity, smu2, loga, eu, and el. If rrlonly is set to True, only RRLs will be included. If verbose is set to True, the search results will be listed to the logger. If logfile is specified, the logger output is listed. If verbose=True, one can also specify that the results be listed to the logfile. If append is set to True and the logfile exists, one can specify that the results be appended to it or to overwrite it with the new results.

```
put search results in a table but do not list to the logger
slsearch("myspectrallines.tbl", verbose=False)
```

smoothcal-task.html

### 0.1.114 smoothcal

Requires:

#### Synopsis

Smooth calibration solution(s) derived from one or more sources:

#### Description

A G- or T-type gain calibration can be smoothed. Amplitude and phase are currently smoothed with the same time. Calibration values will be smoothed over all fields.

#### Arguments

| Inputs     |                                                                                            |
|------------|--------------------------------------------------------------------------------------------|
| vis        | Name of input visibility file (MS)<br>allowed: string<br>Default:                          |
| tablein    | Input calibration table<br>allowed: string<br>Default:                                     |
| caltable   | Output calibration table (overwrite tablein if unspecified)<br>allowed: string<br>Default: |
| field      | Field name list<br>allowed: stringArray<br>Default:                                        |
| smoothtype | Smoothing filter to use<br>allowed: string<br>Default: median                              |
| smoothtime | Smoothing time (sec)<br>allowed: any<br>Default: variant 60.0                              |

#### Returns

void



## Example

A G- or T-type gain calibration can be smoothed. The amplitude and phase smoothing times are currently the same. Calibration values will be smoothed for only the specified fields. Smoothing is performed independently per field, per spw, and per antenna.

Keyword arguments:

```
vis -- Name of input visibility file
 default: none; example: vis='ngc5921.ms'
tablein -- Input calibration table (G or T)
 default: none; example: tablein='ngc5921.gcal'
caltable -- Output calibration table (smoothed)
 default: '' (will overwrite tablein);
 example: caltable='ngc5921_smooth.gcal'
field -- subset of fields to select and smooth
 default: '' means all; example: field='0319_415_1,3C286'
smoothtype -- The smoothing filter to be used for both amp and phase
 default: 'median'; example: smoothtype='mean'
 Options: 'median','mean'
smoothtime -- Smoothing filter time (sec)
 default: 300.0; example: smoothtime=60.
```

`specfit-task.html`

### **0.1.115 specfit**

Requires:

#### **Synopsis**

Fit 1-dimensional gaussians and/or polynomial models to an image or image region

#### **Description**

#### **Arguments**

|           |                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                                                                                                                                   |
| imagename | Name of the input image<br>allowed: string<br>Default:                                                                                                                                                                            |
| box       | Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.<br>allowed: string<br>Default:                                                                     |
| region    | Region selection. See "help par.region" for details. Default is to use the full image.<br>allowed: string<br>Default:                                                                                                             |
| chans     | Channels to use. See "help par.chans" for details. Default is to use all channels.<br>allowed: string<br>Default:                                                                                                                 |
| stokes    | Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.<br>allowed: string<br>Default:                                                                                                      |
| axis      | The profile axis. Default: use the spectral axis if one exists, axis 0 otherwise (<0).<br>allowed: int<br>Default: -1                                                                                                             |
| mask      | Mask to use. See help par.mask. Default is none..<br>allowed: string<br>Default:                                                                                                                                                  |
| ngauss    | Number of Gaussian elements. Default: 1.<br>allowed: int<br>Default: 1                                                                                                                                                            |
| poly      | Order of polynomial element. Default: do not fit a polynomial (<0).<br>allowed: int<br>Default: -1                                                                                                                                |
| estimates | Name of file containing initial estimates. Default: No initial estimates ("").<br>allowed: string<br>Default:                                                                                                                     |
| minpts    | Minimum number of unmasked points necessary to attempt fit.<br>allowed: int<br>Default: 1                                                                                                                                         |
| multifit  | If true, fit a profile along the desired axis at each pixel in the specified region. If false, average the non-fit axis pixels and do a single fit to that average profile. Default False.<br>allowed: 594 bool<br>Default: False |
| model     | Name of model image. Default: do not write the model image ("").<br>allowed: string<br>Default:                                                                                                                                   |
| residual  | Name of residual image. Default: do not write the residual image ("").<br>allowed: string                                                                                                                                         |

## Returns

record

## Example

This task simultaneously fits one or more gaussian singlets lorentzian singlets, gaussian mu

### PARAMETER SUMMARY

|             |                                                                                                                                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------|
| imagename   | Name of the input (CASA, FITS, MIRIAD) image                                                                                   |
| box         | Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane. |
| region      | Region selection. See "help par.region" for details. Default is to use the full image.                                         |
| chans       | Channels to use. See "help par.chans" for details. Default is to use all channels.                                             |
| stokes      | Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.                                  |
| axis        | Axis along which to do the fit(s). <0 means use the spectral axis or the zeroth axis if a spectral axis is not present.        |
| mask        | Mask to use. See help par.mask. Default is none.                                                                               |
| stretch     | Stretch the input mask if necessary and possible? Only used if a mask is sp<br>See help par.stretch.                           |
| ngauss      | Maximum number of gaussians to fit.                                                                                            |
| poly        | Order of polynomial to fit. <0 means do not fit a polynomial.                                                                  |
| estimates   | Name of file containing initial gaussian estimates.                                                                            |
| minpts      | Minimum number of points necessary to attempt a fit.                                                                           |
| multifit    | Fit models at each pixel in region (true) or average profiles and fit a sing                                                   |
| model       | Name of model image to write.                                                                                                  |
| residual    | Name of residual image to write.                                                                                               |
| amp         | Name of amplitude solution image. Default: do not write the image ("")                                                         |
| amperr      | Name of amplitude solution error image. Default: do not write the image ("")                                                   |
| center      | Name of center solution image. Default: do not write the image ("")                                                            |
| centererr   | Name of center solution error image. Default: do not write the image ("")                                                      |
| fwhm        | Name of fwhm solution image. Default: do not write the image ("")                                                              |
| fwhmerr     | Name of fwhm solution error image. Default: do not write the image ("")                                                        |
| integral    | Name of integral solution image. Default: do not write the image ("")                                                          |
| integralerr | Name of integral solution error image. Default: do not write the image ("")                                                    |
| wantreturn  | If true, return a record summarizing the fit results, if false, return false                                                   |
| stretch     | Stretch the mask if necessary and possible? See help par.stretch                                                               |
| logresults  | Output results to logger?                                                                                                      |
| pampest     | Initial estimate of PCF profile (gaussian or lorentzian) amplitudes.                                                           |

|                |                                                                                                           |
|----------------|-----------------------------------------------------------------------------------------------------------|
| pcenterest     | Initial estimate PCF profile centers, in pixels.                                                          |
| pfwhmest       | Initial estimate PCF profile FWHMs, in pixels.                                                            |
| pfix           | PCF profile parameters to fix during fit.                                                                 |
| pfunc          | PCF singlet functions to fit. "gaussian" or "lorentzian" (minimal match support).                         |
| gmncomps       | Number of components in each Gaussian multiplet to fit.                                                   |
| gmampcon       | The amplitude ratio constraints for non-reference components to reference components.                     |
| gmcentercon    | The center offset constraints (in pixels) for non-reference components to reference components.           |
| gmfwhmcon      | The FWHM ratio constraints for non-reference components to reference components.                          |
| gmampest       | Initial estimate of individual gaussian amplitudes in gaussian multiplets.                                |
| gmcenterest    | Initial estimate of individual gaussian centers in gaussian multiplets, in pixels.                        |
| gmfwhmest      | Initial estimate of individual gaussian FWHMs in gaussian multiplets, in pixels.                          |
| gmfix          | Parameters of individual gaussians in gaussian multiplets to fix during fit.                              |
| logfile        | File in which to log results. Default is not to write a logfile.                                          |
| append         | Append results to logfile? Logfile must be specified. Default is to append.                               |
| goodamprange   | Acceptable amplitude solution range. 0 => all amplitude solutions are acceptable.                         |
| goodcenterange | Acceptable center solution range in pixels relative to region start. [0.0] => all centers are acceptable. |
| goodfwhmrange  | Acceptable FWHM solution range in pixels. [0.0] => all FWHM solutions are acceptable.                     |
| sigma          | Standard deviation array or image name.                                                                   |
| outsigma       | Name of output image used for standard deviation. Ignored if sigma is empty.                              |

This task simultaneously performs a non-linear, least squares fit using the Levenberg-Marquardt algorithm. It can fit one or more lorentzian singlets, one or more gaussian multiplets, and/or a polynomial to one or more profiles. The fitting algorithm may be found in AIPS++ Note 224 (<http://www.astron.nl/casacore/trunk/casacore/notes/note224.html>) by W.H. Press et al., Cambridge University Press. A gaussian/lorentzian singlet is a gaussian profile with a fixed center position, and width) are all independent from any other feature that may be simultaneously fit. A gaussian multiplet is a set of one or more gaussian lines in which at least one (and possibly two or three) parameter of each line (amplitude, center position, and width) is fixed to a reference profile. For example, one can specify a doublet in which the amplitude of the zeroth line and/or the center of the first line is 20 pixels from the center of the zeroth line (in pixels) to that of the zeroth line. There is no limit to the number of lines in a multiplet (except of course that the number of parameters to be fit should be significantly less than the number of data points). A single reference profile in a multiplet to which to tie constraints of parameters of the other lines.

#### AXIS

The axis parameter indicates on which axis profiles should be fit; a value <0 indicates the zeroth axis should be used.

#### MINIMUM NUMBER OF PIXELS

The minpts parameter indicates the minimum number of unmasked pixels that must be present in the region to be attempted. When multifit=T, positions with too few good points will be masked in any subsequent fits.

#### ONE FIT OF REGION AVERAGE OR PIXEL BY PIXEL FIT

The multifit parameter indicates if profiles should be fit at each pixel in the selected region, or if the region is averaged and the fit done to that average profile (false).

#### POLYNOMIAL FITTING

The order of the polynomial to fit is specified only via the poly parameter. If poly<0, no polynomial fit is attempted.

coefficients can be specified; these are determined automatically.

#### GAUSSIAN SINGLET FITTING

In the absence of an estimates file and no estimates being specified by the p\*est parameters, n\*est indicates the maximum number of gaussian singlets that should be fit. The initial estimates are determined automatically in this case. If it deems appropriate, the fitter will fit fewer than this number. If n\*est is zero, n\*est is ignored (see below). n\*est is also ignored if the p\*est parameters are specified. If n\*est is greater than zero. If estimates is not specified or the p\*est parameters are not specified, an error will occur as this indicates there is nothing to fit.

One can specify initial estimates of gaussian singlet parameters via an estimates file or the p\*est parameters. The latter is the recommended way to specify these estimates as support for the p\*est option is used, an amplitude initial estimate must always be nonzero. A negative fwhm

#### SPECIFYING INITIAL ESTIMATES FOR GAUSSIAN AND LORENTZIAN SINGLETS (RECOMMENDED METHOD)

One may specify initial estimates via the pampest, pcenterest, and pfwhmest parameters. In the case of these parameters can be numbers. pampest must be specified in image brightness units, pcenterest is the zeroth pixel, and pfwhmest must be given in pixels. Optionally pfix can be specified and in the case of a string. In it is coded which parameters should be held constant during the fit. Any parameter (fwhm) is allowed; eg pfix="pc" means fix both the amplitude and center during the fit. In the case of singlets, these parameters must be specified as arrays of numbers. The length of the arrays must be the same for all the p\*est parameters.

If no parameters are to be fixed for any of the singlets, pfix can be set to the empty string. If a parameter is to be fixed, pfix must be an array of strings and have a length equal to the p\*est arrays. The empty string should be represented as an empty string in the pfix array. So, for example, if one desires to fix the amplitude of one, one must specify pfix=["", "f", ""], the empty strings indicating no parameters of the other singlets.

In the case of multifit=True, the initial estimates, whether from the p\*est parameters or from the estimates file, are of the first fit. This is normally the bottom left corner of the region selected. If masked, the first attempted fit fails, the fitting proceeds to the next pixel with the pixel value of the lowest intensity. If a successful fit has been performed, subsequent fits will use the results of a fit for a nearby pixel as the initial estimate for the parameters at the current location. The fixed parameter string will be used for all subsequent fits.

One specifies what type of PCF profile to fit via the pfunc parameter. A PCF function is one that can be fit as both gaussian and lorentzian singlets can. If all singlets to be fit are gaussians, one can specify pfunc="G". If at least one lorentzian is to be fit, pfunc must be specified as an array of strings (in the case of multiple singlets). The position of each string corresponds to the position of the p\*est and pfix arrays. Minimal match ("g", "G", "l", or "L") is supported. So, if one wanted to fit both gaussian and lorentzian singlets, the zeroth and last of which were lorentzians, one would specify pfunc=["L", "G", "L"], the first and last of which were gaussians.

#### ESTIMATES FILE FOR GAUSSIAN SINGLETS (NONRECOMMENDED METHOD)

Initial estimates for gaussian singlets can be specified in an estimates file. Estimates file is only used if p\*est parameters, so it is recommended users use those parameters instead. If an estimates file is specified, n\*est must be 0 or empty and mgncomps must be 0 or empty. Only gaussian singlets can be specified in the estimates file. If more gaussian multiplets and/or one or more lorentzian singlets simultaneously, the p\*est parameters must be specified.

of all gaussian singlets to fit; one cannot use an estimates file in this case. If an estimate can be fit simultaneously by specifying the poly parameter. The estimates file must contain for all gaussian singlets to be fit. The number of gaussian singlets to fit is gotten from the comments which are indicated by a "#" at the beginning of a line. All non-comment lines will be in the format of such a line is

```
[peak intensity], [center], [fwhm], [optional fixed parameter string]
```

The first three values are required and must be numerical values. The peak intensity must be specified, the center must be specified in pixels offset from the zeroth pixel, and fwhm must be specified in pixels. The optional parameter(s) represents the parameter(s) that should be held constant during the fit. Any combination of parameters is permitted, eg "fc" means hold the fwhm and the center constant during the fit. Fixed parameters are indicated by a "f" at the end of the line. An example file:

```
estimates file indicating that two gaussians should be fit
first gaussian estimate, peak=40, center at pixel number 10.5, fwhm = 5.8 pixels, all parameters
fit
40, 10.5, 5.8
second gaussian, peak = 4, center at pixel number 90.2, fwhm = 7.2 pixels, hold fwhm constant
4, 90.2, 7.2, f
end file
```

#### GAUSSIAN MULTIPLY FITTING

Any number of gaussian multiplets, each containing any number of two or more components, can be fit simultaneously by specifying the gmncomps parameter. The gmncomps parameter indicates the number of components in each multiplet. In the case of a single multiplet, an integer (>1) can be specified. In the case of 2 or more multiplets, an array of integers can be specified. For example, gmncomps=[2, 4, 3] means 3 separate multiplets are to be fit, the zeroth being a doublet, the first being a quadruplet of gaussians, and the second being a triplet.

Initial estimates of all gaussians in all multiplets are specified via the gm\*est parameters. The gm\*est parameters start with the zeroth component of the zeroth multiplet to the last component of the zeroth multiplet, then the first multiplet to the last component of the first multiplet, etc to the zeroth component of the last multiplet. The zeroth element of a multiplet is defined as the reference profile. The significance that it is the profile to which all constraints of all other profiles in that multiplet are relative. In our example of gmncomps=[2, 4, 3], gmampest, gmcenterest, and gmfwhest must each be nine element arrays (the zeroth element of each array is the sum of all profiles summed over all multiplets) element arrays. The zeroth, second, and sixth elements of each array are the zeroth, first, and second multiplet, respectively.

The fixed relationships between the non-reference profile(s) and the reference profile of a multiplet are specified via the gm\*con parameters. At least one, and any combination, of constraints can be specified for each component of a multiplet. The amplitude ratio of a non-reference line to that of the reference line is set in gmamcon. The ratio of the fwhm of a non-reference line to that of the reference line is set in gmfwcon. The offset of a non-reference line to that of the reference line is set in gmcentercon. In the case where a non-reference line of any multiplet, the value of the associated parameter must be 0. In the

a single doublet, a constraint may be specified as a number or an array of a single number and `gmcentercon=[32.4]` means there is a single doublet to fit where the amplitude ratio of the doublet is to be 0.65 and the center of the first line is constrained to be offset by 32.4 pixels from the center of a total of three or more gaussians, the constraints parameters must be specified as an array of gaussians summed over all multiplets minus the number of reference lines (one per multiplet minus the number of reference lines cannot be constrained by themselves). In the cases where an array must be specified, 0 should be specified. Here's an example

```
gmncmps=[2, 4, 3]
gmampcon= [0 , 0.2, 0 , 0.1, 4.5, 0]
gmcentercon=[24.2, 45.6, 92.7, 0 , -22.8, -33.5]
gmfwmmcon=""
```

In this case we have our previous example of one doublet, one quadruplet, and one triplet. The first component of the doublet that its center is offset by 24.2 pixels from the zeroth (reference) component. The first component of the quadruplet has an amplitude of 0.2 times that of the quadruplet's zeroth component and its center is constrained to be offset by 45.6 pixels from the reference component. The second component of the quadruplet is constrained to have its center offset by 92.7 pixels from the reference component and the third component is constrained to have an amplitude of 0.1 times that of the quadruplet's zeroth component. The first component of the triplet is constrained to have an amplitude of 4.5 times that of the triplet's zeroth component and its center is constrained to be offset by -22.8 pixels from the reference component's center. The second component of the triplet is constrained to be offset by -33.5 pixels from the center of the reference component. No lines have constraints for that parameter. Note that using 0 to indicate no constraint for line center means that the component is at the same position as the reference component but having a different FWHM from the reference component. To specify a constraint for FWHM, try using a very small positive (or even negative) value for the center constraint.

Note that when a parameter for a line is constrained, the corresponding value for that component is ignored and the value of the constrained parameter is automatically used instead. So let's see the following estimates:

```
gmampest = [1, .2, 2, .1, .1, .5, 3, 2, 5]
gmcenterest = [20, 10 , 30, 45.2, 609 , -233, 30, -859, 1]
```

Before any fitting is done, the constraints would be taken into account and these arrays would be:

```
gmampest = [1, .2, 2, .4, .1, .2, 3, 13.5, 5]
gmcenterest = [20, 44.2, 30, 75.6, 127.7, -233, 30, 7.2, -3.5]
```

The value of `gmfwmmest` would be unchanged since there are no FWHM constraints in this example.

In addition to being constrained by values of the reference component, parameters of individual components are specified via the `gmfix` parameter. If no parameters are to be fixed, `gmfix` can be specified as an empty array. In the case where any parameter is to be fixed, `gmfix` must be specified as an array of strings. The strings encode which parameters to be fixed for each component. If a component is to have no parameters fixed, an empty string is used. In other cases one or more parameters are to be fixed using "p", "c", and/or "f" described above for fixing singlet parameters. There are a few other things to be aware of. In the case where a non-reference component parameter is constrained and the



set as fixed, that parameter in the non-reference parameter will automatically be fixed even if it is not in the `gmfix` array. This is the only way the constraint can be honored afterall. In the converse, if a parameter in a non-reference component is specified as fixed, but the corresponding parameter in the reference component is not, an error will occur. Fixing an unconstrained parameter in a non-reference component is always allowed. Fixing parameters in a reference component (with the above caveat that corresponding constrained parameters in non-reference components be silently held fixed as well).

The same rules that apply to singlets when `multifit=True` apply to multiplets.

#### LIMITING RANGES FOR SOLUTION PARAMETERS

In cases of low (or no) signal to noise spectra, it is still possible for the fit to converge to a nonsensical solution. The astronomer can use her knowledge of the source to filter out obviously invalid solutions. Any solution which contains a NaN value as a value or error in any one of its parameters is considered invalid.

One can also limit the ranges of solution parameters to known "good" values via the `goodamprange` parameters. Any combination can be specified and the limit constraints will be ANDed together. Choosing ranges that might be fit; choosing ranges on a component by component basis is not supported. If specifying a range, an array of exactly two numerical values must be given to indicate the range of acceptable values for that parameter. `goodamprange` is expressed in terms of image brightness units. `goodcenterrange` is expressed in terms of pixels from the zeroth pixel in the specified region. `goodfwhmrange` is expressed in terms of pixels (the endpoints are given for FWHM range endpoints). In the case of a multiple-PCF fit, if any of the corresponding ranges, the entire solution is considered to be invalid.

In addition, solutions for which the absolute value of the ratio of the amplitude error to the FWHM error exceeds 100 are automatically marked as invalid.

#### INCLUDING STANDARD DEVIATIONS OF PIXEL VALUES

If the standard deviations of the pixel values in the input image are known and they vary in a regular pattern (near the edge of the band), they can be included in the `sigma` parameter. This parameter takes an array or image must have one of three shapes: 1. the shape of the input image, 2. the same shape as the input image of all axes being one except for the fit axis which must have length corresponding to its 1D length, 3. a 1D array with length equal to the length of the fit axis in the input image. In cases 2 and 3, the array must be replicated such that the image that is ultimately used is the same shape as the input image. It is only the relative values that are important. A value of 0 means that pixel should not be included. If pixel A has a higher standard deviation than pixel B, then pixel A is noisier than pixel B and will have a lower weight. The weight of a pixel is the usual

$$\text{weight} = 1/(\text{sigma}*\text{sigma})$$

In the case of `multifit=F`, the `sigma` values at each pixel along the fit axis in the hyperplane are averaged and the resultant averaged standard deviation spectrum is the one used. The `sigma` values are normalized such that the maximum value is 1. This mitigates a known overflow issue.

One can write the normalized standard deviation image used in the fit but specifying its name as `sigma` for subsequent runs.

#### RETURNED DICTIONARY STRUCTURE

The dictionary returned (if wantreturn=True) has a (necessarily) complex structure. First, the abscissa unit and the ordinate unit described by simple strings. Next there are arrays of fit quality. These arrays have the shape of the specified region collapsed along the fit axis, axis having length of 1:

attempted: a boolean array indicating which fits were attempted (eg if too few unmasked points)  
converged: a boolean array indicating which fits converged. False if the fit was not attempted  
valid: a boolean array indicating which solutions fall within the specified valid ranges (see section LIMITING RANGES FOR SOLUTION PARAMETERS for details).  
niter: an int array indicating the number of iterations for each profile, <0 if the fit did not converge  
ncomps: the number of components (gaussian singlets + lorentzian singlets + gaussian multiplets)  
direction: a string array containing the world direction coordinate for each profile

There is a "type" array having number of dimensions equal to the number of dimensions in the data. The first n-1 dimensions is the same as the shape of the above arrays. The length of the last dimension is the number of components fit. The values of this array are strings describing the components that were fit: "GAUSSIAN" in the case of gaussian singlets, "LORENTZIAN" in the case of lorentzian singlets, and "MULTI" for multiplets.

If any gaussian singlets were fit, there will be a subdictionary accessible via the "gs" key. It will have keys "centerErr", "fwhm", "fwhmErr", "integral", and "integralErr". Each of these arrays will have the same shape as the "type" array above. The shape of the first n-1 dimensions will be the same as the shape of the arrays described above. The last dimension will have length equal to the maximum number of gaussian singlets that were fit. Along this axis, the values will be the corresponding fit result or associated error (depending on the array's associated key) of the fit. If the fit did not converge, or that particular component was excluded from the fit, a value of 0 will be returned.

If any lorentzian singlets were fit, their solutions will be accessible via the "ls" key. The structure will be the same as the "gs" arrays described above.

If any gaussian multiplets were fit, there will be subdictionaries accessible by keys "gm0", "gm1", etc. Each of these dictionaries will have the same arrays described above. The shape of the first n-1 dimensions will be the same as the shape of the arrays described above. The last dimension will have length equal to the number of components in that particular multiplet. Each pixel value or error for that component number in the multiplet, eg the zeroth pixel along that axis, will be the parameter solution or error for the reference component of the multiplet.

The polynomial coefficient solutions and errors are not returned, although they are logged.

#### OUTPUT IMAGES

In addition to the returned dictionary, optionally one or more of any combination of output images can be written. The model and residual parameters indicate the names of the model and residual images to be written. If the parameter is None, the image should not be written.

One can also write none, any or all of the solution and error images for Gaussian singlet, Lorentzian singlet, or Gaussian multiplet via the parameters amp, amperr, center, centererr, fwhm, fwhmerr, integral, and integralerr.

contain the arrays described for the associated parameter solutions or errors described in p  
singlets, "\_ls" is appended to the image names, in the case of Gaussian multiplets, "\_gm" is  
is a linear axis and represents component number (and is named accordingly). In the case where  
Lorentzians are fitted, the image names are further appended with an underscore and the rele  
In the case of Gaussian multiplets, the image names are appended with an underscore, followed  
multiplet group, followed by an underscore, followed by the number of the component in that  
component number 4 of multiplet group number 3). Pixels for which fits were not attempted, o  
have values of NaN (not a number) or INF (infinity) will be masked as bad.

Writing analogous images for polynomial coefficients is not supported.

#### EXAMPLE

```
res = specif(imagename="myspectrum.im", ngauss=2, box="3,3,4,5", poly=2, multifit=true, want
```

specsmooth-task.html

### **0.1.116 specsmooth**

Requires:

#### **Synopsis**

Smooth an image region in one dimension

#### **Description**

#### **Arguments**

|           |                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                                                                   |
| imagename | Name of the input image<br>allowed: string<br>Default:                                                                                                            |
| outfile   | Output image name.<br>allowed: string<br>Default:                                                                                                                 |
| box       | Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.<br>allowed: string<br>Default:     |
| chans     | Channels to use. See "help par.chans" for details. Channels must be contiguous. Default is to use all channels.<br>allowed: string<br>Default:                    |
| stokes    | Stokes planes to use. See "help par.stokes" for details. Planes specified must be contiguous. Default is to use all Stokes planes.<br>allowed: string<br>Default: |
| region    | Region selection. See "help par.region" for details. Default is to use the full image.<br>allowed: any<br>Default: variant                                        |
| mask      | Mask to use. See help par.mask. Default is none..<br>allowed: string<br>Default:                                                                                  |
| overwrite | Overwrite the output if it exists?<br>allowed: bool<br>Default: False                                                                                             |
| stretch   | Stretch the mask if necessary and possible? See help par.stretch. Default False<br>allowed: bool<br>Default: False                                                |
| axis      | The profile axis. Default: use the spectral axis if one exists, axis 0 otherwise (<0).<br>allowed: int<br>Default: -1                                             |
| function  | Convolution function. hanning and boxcar are supported functions. Minimum match is supported.<br>allowed: string<br>Default: boxcar                               |
| width     | Width of boxcar, in pixels.<br>allowed: int<br>Default: 2                                                                                                         |
| dmethod   | Decimation method. "" means no decimation, "copy" and "mean" are also supported (minimum match).<br>allowed: string<br>Default: copy                              |

## Returns

record

## Example

Smooth an image region in one dimension.

### PARAMETER SUMMARY

|           |                                                                                                                                                                       |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| imagename | Name of the input (CASA, FITS, MIRIAD) image                                                                                                                          |
| box       | Rectangular region to select in direction plane. See "help par.box" for details.<br>Default is to use the entire direction plane. Only a single box may be specified. |
| chans     | Channels to use. See "help par.chans" for details. Channels must be contiguous.<br>Default is to use all channels.                                                    |
| stokes    | Stokes planes to use. See "help par.stokes" for details. Planes specified must be contiguous. Default is to use all Stokes planes.                                    |
| region    | Region selection. See "help par.region" for details. Default is to use the full region.                                                                               |
| mask      | Mask to use. See help par.mask. Default is none.                                                                                                                      |
| overwrite | If the specified outfile already exists, overwrite it if True.                                                                                                        |
| stretch   | Stretch the input mask if necessary and possible? Only used if a mask is specified.<br>See help par.stretch.                                                          |
| axis      | Pixel axis along which to do the convolution <0 means use the spectral axis.                                                                                          |
| function  | Convolution function to use. Supported values are "boxcar" and "hanning". Minimal match is supported.                                                                 |
| width     | Width of boxcar in pixels. Used only if function parameter minimally matches.                                                                                         |
| dmethod   | Plane decimation method. "" means no decimation should be performed. Other supported values are "copy" and "mean". Minimal match is supported. See below for details. |

This application performs one dimensional convolution along a specified axis of an image or selected region of an image. Hanning smoothing and boxcar smoothing are supported. Both float valued and complex valued images are supported. Masked pixel values are set to zero prior to convolution. All nondefault pixel masks are ignored during the calculation. The convolution is done in the image domain (i.e., not with an FFT).

### BOXCAR SMOOTHING

One dimensional boxcar convolution is defined by

$$z[i] = (y[i] + y[i+1] + \dots + y[i+w])/w$$

where  $z[i]$  is the value at pixel  $i$  in the box car smoothed image,  $y[k]$

is the pixel value of the input image at pixel  $k$ , and  $w$  is a positive integer representing the width of the boxcar in pixels. The length of the axis along which the convolution is to occur must be at least  $w$  pixels in the selected region, unless decimation using the mean function is chosen in which case the axis length must be at least  $2*w$  (see below).

If `dmethod=""` (no decimation), the length of the output axis will be equal to the length of the input axis -  $w + 1$ . The pixel mask, ORed with the OTF mask if specified, is copied from the selected region of the input image to the output image. Thus for example, if the selected region in the input image has six planes along the convolution axis, if the specified boxcar width is 2, and if the pixel values, which are all unmasked, on a slice along this axis are [1, 2, 5, 10, 17, 26], then the corresponding output slice will be of length five pixels and the output pixel values will be [1.5, 3.5, 7.5, 13.5, 21.5].

If `dmethod="copy"`, the output image is the image calculated if `dmethod=""`, except that only every  $w$ th plane is kept. Both the pixel and mask values of these planes are copied directly to the output image, without further processing. Thus for example, if the selected region in the input image has six planes along the convolution axis, the boxcar width is chosen to be 2, and if the pixel values, which are all unmasked, on a slice along this axis are [1, 2, 5, 10, 17, 26], the corresponding output pixel values will be [1.5, 7.5, 21.5].

If `dmethod="mean"`, first the image described in the `dmethod=""` case is calculated. Then, the  $i$ th plane of the output image is calculated by averaging the  $i*w$  to the  $(i+1)*w-1$  planes of this intermediate image. Thus, for example, if the selected region in the input image has six planes along the convolution axis, the boxcar width is chosen to be 2, and if the pixel values, which are all unmasked, on a slice along this axis are [1, 2, 5, 10, 17, 26], then the corresponding output pixel values will be [2.5, 10.5]. Any pixels at the end of the convolution axis of the intermediate image that do not fall into a complete bin of width  $w$  are ignored. Masked values are taken into consideration when forming this average, so if one of the values is masked, it is not used in the average. If at least one of the values in the intermediate image bin is not masked, the corresponding output pixel will not be masked.

#### HANNING SMOOTHING

Hanning convolution of one axis of an image is defined by

$$z[i] = 0.25*y[i-1] + 0.5*y[i] + 0.25*y[i+1] \quad (\text{equation 1})$$

where  $z[i]$  is the value at pixel  $i$  in the hanning smoothed image, and  $y[i-1]$ ,  $y[i]$ , and  $y[i+1]$  are the values of the input image at pixels  $i-1$ ,  $i$ , and  $i+1$  respectively. The length of the axis along which the convolution is to occur must be at least three pixels in the selected region.

If `dmethod=""` (no decimation of image planes), the length of the output axis will be the same as that of the input axis. The output pixel values along the convolution axis will be related to those of the input values according to equation 1, except the first and last pixels. In that case,

$$z[0] = 0.5*(y[0] + y[1])$$

and,

$$z[N-1] = 0.5*(y[N-2] + y[N-1])$$

where `N` is the number of pixels along the convolution axis.

The pixel mask, ORed with the OTF mask if specified, is copied from the selected region of the input image to the output image. Thus for example, if the selected region in the input image has six planes along the convolution axis, and if the pixel values, which are all unmasked, on a slice along this axis are [1, 2, 5, 10, 17, 26], the corresponding output pixel values will be [1.5, 2.5, 5.5, 10.5, 17.5, 21.5].

If `dmethod="copy"`, the output image is the image calculated if `dmethod=""`, except that only the odd-numbered planes are kept. Furthermore, if the number of planes along the convolution axis in the selected region of the input image is even, the last odd number plane is also discarded. Thus, if the selected region has `N` pixels along the convolution axis in the input image, along the convolution axis the output image will have  $(N-1)/2$  planes if `N` is odd, or  $(N-2)/2$  planes if `N` is even. The pixel and mask values are copied directly, without further processing. Thus for example, if the selected region in the input image has six planes along the convolution axis, and if the pixel values, which are all unmasked, on a slice along this axis are [1, 2, 5, 10, 17, 26], the corresponding output pixel values will be [2.5, 10.5].

If `dmethod="mean"`, first the image described in the `dmethod=""` case is calculated. The first plane and last plane(s) of that image are then discarded as described in the `dmethod="copy"` case. Then, the `i`th plane of the output image is calculated by averaging the  $(2*i)$ th and  $(2*i + 1)$ th planes of the intermediate image. Thus for example, if the selected region in the input image has six planes along the convolution axis, and if the pixel values, which are all unmasked, on a slice along this axis are [1, 2, 5, 10, 17, 26], the corresponding output pixel values will be [4.0, 14.0]. Masked values are taken into consideration when forming this average, so if one of the values is masked, it is not used in the average. If at least one of the values in the input pair is not masked, the corresponding output pixel will not be masked.

## EXAMPLES

```
boxcar smooth the spectral axis by 3 pixels, say it's axis 2 and only
```



```
write every other pixel
specsmooth(imagename="mynonsmoothed.im", outfile="myboxcarsmoothed.im",
axis=2, function="boxcar", dmethod="copy", width=3, overwrite=True)

hanning smooth the spectral axis, say it's axis 2 and do not perform decimation
of image planes
specsmooth(imagename="mynonsmoothed.im", outfile="myhanningsmoothed.im",
axis=2, dmethod="", overwrite=True)
```

splattotable-task.html

### 0.1.117 splattotable

Requires:

#### Synopsis

Convert a downloaded Splatalogue spectral line list to a casa table.

#### Arguments

| Inputs    |                                                                               |
|-----------|-------------------------------------------------------------------------------|
| filenames | Files containing Splatalogue lists.<br>allowed:       stringArray<br>Default: |
| table     | Output table name. Must be specified.<br>allowed:       string<br>Default:    |

#### Returns

bool

#### Example

##### PARAMETER SUMMARY

filenames   Files containing Splatalogue lists.  
table       Output table name. Must be specified

This task reads a spectral line list(s) downloaded from Splatalogue ([www.splatalogue.net](http://www.splatalogue.net)) and can be queried via eg the slsearch task.

##### REQUIRMENTS OF THE DOWNLOADED FILES

The downloaded files must be in a specific format for this task to succeed. The columns are important things; one can filter the results as one desires using Splatalogue, but the parts below. The columns which must be present in the downloaded file in this exact order are: "Sp", "Freq in GHz", "Resolved QNs", "CDMS/JPL Intensity", "Sijmu2 (D2)", "Log10 (Aij)", "EL (K)", "EU (K)", "Linelist". In order to get these columns in this order,

select exactly the following items on the Splatalogue search interface. Under "Specify Range" select exactly "CDMS/JPL Intensity", "Sij  $\mu^2$ ", and "Aij". Under "Energy Levels", one should select exactly "Elower (K)" and "Eupper (K)". Under "Display Options", one should select exactly "Display Ordered Frequency ONLY" and "Display NRAO Recommended Frequencies". On the resulting page, one should select the Tab Field Separator and then export the list. The resulting file is in the format for importing into CASA.

```
splattotable("mysplatlist.txt", "mynews1.tbl", true)
```

split-task.html

### 0.1.118 **split**

Requires:

#### **Synopsis**

Create a visibility subset from an existing visibility set

#### **Description**

---

This is the new implementation of split.

\* The old implementation is available for a short time as oldsplit. \* Task split2 has been renamed to split. The split2 alias will be removed soon. \* Please, update your scripts to call split instead.

---

This new split task uses the MSTransform framework underneath. Split is the general purpose program to make a new data set that is a subset or averaged form of an existing data set. General selection parameters are included, and one or all of the various data columns (DATA, LAG\_DATA and/or FLOAT\_DATA, MODEL\_DATA and/or CORRECTED\_DATA) can be selected.

Split is often used after the initial calibration of the data to make a smaller Measurement Set with only the data that will be used in further flagging, imaging and/or self-calibration. Split can average over frequency (channels) and time (integrations).

Split also supports the Multi-MS (MMS) format as input. For more information about MMS, see the help of partition and mstransform.

#### **Arguments**

|             |                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                  |
| vis         | Name of input Measurement set or Multi-MS<br>allowed: string<br>Default:                         |
| outputvis   | Name of output Measurement set or Multi-MS<br>allowed: string<br>Default:                        |
| keepmms     | If the input is a Multi-MS the output will also be a Multi-MS.<br>allowed: bool<br>Default: True |
| field       | Select field using ID(s) or name(s).<br>allowed: any<br>Default: variant                         |
| spw         | Select spectral window/channels.<br>allowed: any<br>Default: variant                             |
| scan        | Select data by scan numbers.<br>allowed: any<br>Default: variant                                 |
| antenna     | Select data based on antenna/baseline.<br>allowed: any<br>Default: variant                       |
| correlation | Correlation: " ==> all, correlation='XX,YY'.<br>allowed: any<br>Default: variant                 |
| timerange   | Select data by time range.<br>allowed: any<br>Default: variant                                   |
| intent      | Select data by scan intent.<br>allowed: any<br>Default: variant                                  |
| array       | Select (sub)array(s) by array ID number.<br>allowed: any<br>Default: variant                     |
| uvrange     | Select data by baseline length.<br>allowed: any<br>Default: variant                              |
| observation | Select by observation ID(s).<br>allowed: any<br>Default: variant                                 |
| feed        | Multi-feed numbers: Not yet implemented.<br>allowed: any<br>Default: variant                     |
| datacolumn  | Which data column(s) to process.                                                                 |

## Example

Detailed Keyword arguments:

```
vis -- Name of input Measurement set or Multi-MS.
 default: none;
 example: vis='ngc5921.ms'
```

```
outputvis -- Name of output Measurement set or Multi-MS (MMS).
 default: none;
 example: outputvis='ngc5921_src.ms'
```

IMPORTANT: if a .flagversions file with the name of the output MS exist, this task will exit with an error. The user needs to rename or remove the existing flagversions file or choose a different output name for the MS.

```
keepmms -- Create a Multi-MS as the output if the input is a Multi-MS.
 default: True
```

By default it will create a Multi-MS when the input is a Multi-MS. The output Multi-MS will have the same partition axis of the input MMS. See 'help partition' for more information on the MMS format.

NOTE: It is not possible to do time average with combine='scan' if the input MMS was partitioned with separationaxis='scan' or 'auto'. In this case, the task will abort with an error.

--- Data Selection ---

```
field -- Select field using field id(s) or field name(s).
 [run listobs to obtain the list of d's or names]
 default: ''=all fields If field string is a non-negative
 integer, it is assumed to be a field index
 otherwise, it is assumed to be a field name
 field='0~2'; field ids 0,1,2
 field='0,4,5~7'; field ids 0,4,5,6,7
 field='3C286,3C295'; fields named 3C286 and 3C295
 field = '3,4C*'; field id 3, all names starting with 4C
```

```
spw -- Select spectral window/channels
 default: ''=all spectral windows and channels
```

```

spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
spw='<2'; spectral windows less than 2 (i.e. 0,1)
spw='0:5~61'; spw 0, channels 5 to 61
spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.
spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
spw = '*:3~64' channels 3 through 64 for all sp id's
 spw = ' :3~64' will NOT work.

```

NOTE: mstransform does not support multiple channel ranges per spectral window (';').

```

scan -- Scan number range
 default: '' = all

```

```

antenna -- Select data based on antenna/baseline
 default: '' (all)
 Non-negative integers are assumed to be antenna indices, and
 anything else is taken as an antenna name.

```

examples:

```

antenna='5&6': baseline between antenna index 5 and index 6.
antenna='VA05&VA06': baseline between VLA antenna 5 and 6.
antenna='5&6;7&8': baselines 5-6 and 7-8
antenna='5': all baselines with antenna 5
antenna='5,6,10': all baselines including antennas 5, 6, or 10
antenna='5,6,10&': all baselines with *only* antennas 5, 6, or
 10. (cross-correlations only. Use &&
 to include autocorrelations, and &&&
 to get only autocorrelations.)
antenna='!ea03,ea12,ea17': all baselines except those that
 include EVLA antennas ea03, ea12, or
 ea17.

```

```

correlation -- Correlation types or expression.
 default: '' (all correlations)
 example: correlation='XX,YY'

```

```

timerange -- Select data based on time range:
 default: '' (all); examples,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: if YYYY/MM/DD is missing date, timerange defaults to the
 first day in the dataset
 timerange='09:14:0~09:54:0' picks 40 min on first day
 timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min
 on next day
 timerange='09:44:00' data within one integration of time

```

```

 timerange='>10:24:00' data after this time

array -- (Sub)array number range
 default: '' = all

uvrange -- Select data within uvrange (default units meters)
 default: ''=all; example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
 uvrange='0~1000km'; uvrange in kilometers

observation -- Select by observation ID(s)
 default: '' = all

feed -- Selection based on the feed - NOT IMPLEMENTED YET
 default: '' = all

datacolumn -- Which data column to use for processing (case-insensitive).
 default: 'corrected'; example: datacolumn='data'
 options: 'data', 'model', 'corrected', 'all', 'float_data', 'lag_data',
 'float_data,data', 'lag_data,data'.

 NOTE: 'all' = whichever of the above that are present. If the requested
 column does not exist, the task will exit with an error.

keepflags -- Keep completely flagged rows in the output or drop them. This has no
 effect on partially flagged rows. All of the channels and correlations
 of a row must be flagged for it to be droppable, and a row must be
 well defined to be keepable.

 IMPORTANT: Regardless of this parameter, flagged data is never included in
 channel averaging. On the other hand, partially flagged rows will
 always be included in time averaging. The average value of the
 flagged data for averages containing ONLY flagged data in the relevant
 output channel will be written to the output with the corresponding
 flag set to True, while only unflagged data is used on averages where
 there is some unflagged data with the flag set to False.

 default: True (keep completely flagged rows in the output)

--- Channel averaging parameter ---

width -- Number of input channels to average to create an output
 channel. If a list is given, each bin will apply to one spw in

```



the selection.  
 default: 1 => no channel averaging.  
 options: (int) or [int]  
  
 example: chanbin=[2,3] => average 2 channels of 1st selected  
 spectral window and 3 in the second one.

### --- Time averaging parameters ---

timebin -- Bin width for time averaging. When timebin is greater than 0s,  
 the task will average data in time. Flagged data will be included  
 in the average calculation, unless the parameter keepflags is set to False.  
 In this case only partially flagged rows will be used in the average.  
 default: '0s'

combine -- Let the timebin span across scan, state or both.  
 State is equivalent to sub-scans. One scan may have several  
 state ids. For ALMA MSs, the sub-scans are limited to about  
 30s duration each. In these cases, the task will automatically  
 add state to the combine parameter. To see the number of states  
 in an MS, use the msmd tool. See help msmd.

NOTE: It is not possible to do time average with combine='scan'  
 if the input MMS was partitioned with separationaxis='scan'  
 or 'auto'. In this case, the task will abort with an error.

default: '' (separate time bins by both of the above)  
 options: 'scan', 'state', 'state,scan'

#### examples:

combine = 'scan'; can be useful when the scan number  
 goes up with each integration as in many WSRT MSs.  
 combine = ['scan', 'state']: disregard scan and state  
 numbers when time averaging.  
 combine = 'state,scan'; same as above.

[spxfit-task.html](#)

### **0.1.119   spxfit**

Requires:

#### **Synopsis**

Fit a 1-dimensional model(s) to an image(s) or region for determination of spectral index.

#### **Description**

#### **Arguments**

|            |                                                                                                                                                                                                                                                                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                                                                                                                                                                                                                                       |
| imagenname | <p>Name of the input image(s)</p> <p>allowed: any</p> <p>Default: variant</p>                                                                                                                                                                                                                                                                                         |
| box        | <p>Rectangular region to select in direction plane. See "help par.box" for details. Default is to use the entire direction plane.</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                                                          |
| region     | <p>Region selection. See "help par.region" for details. Default is to use the full image.</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                                                                                                  |
| chans      | <p>Channels to use. See "help par.chans" for details. Default is to use all channels.</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                                                                                                      |
| stokes     | <p>Stokes planes to use. See "help par.stokes" for details. Default is to use all Stokes planes.</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                                                                                           |
| axis       | <p>The profile axis. Default: use the spectral axis if one exists, axis 0 otherwise (&lt;0).</p> <p>allowed: int</p> <p>Default: -1</p>                                                                                                                                                                                                                               |
| mask       | <p>Mask to use. See help par.mask. Default is none.</p> <p>allowed: string</p> <p>Default:</p>                                                                                                                                                                                                                                                                        |
| minpts     | <p>Minimum number of unmasked points necessary to attempt fit.</p> <p>allowed: int</p> <p>Default: 1</p>                                                                                                                                                                                                                                                              |
| multifit   | <p>If true, fit a profile along the desired axis at each pixel in the specified region. If false, average the non-fit axis pixels and do a single fit to that average profile. Default False.</p> <p>allowed: bool</p> <p>Default: False</p>                                                                                                                          |
| spxtype    | <p>Type of function to fit. "plp" = power logarithmic polynomial, "ltp" = logarithmic transformed polynomial.</p> <p>allowed: string</p> <p>Default: plp</p>                                                                                                                                                                                                          |
| spxest     | <p>REQUIRED. Initial estimates as array of numerical values for the spectral index function coefficients. eg [1.5, -0.8] if fitting a plp function thought to be close to <math>1.5*(x/div)^{-0.8}</math> or [0.4055, -0.8] if fitting an ltp function thought to be close to <math>\ln(1.5) - 0.8*\ln(x/div)</math>.</p> <p>allowed: doubleArray</p> <p>Default:</p> |
| spxfix     | <p>Fix the corresponding spectral index function coefficients during the fit. True means hold fixed.</p> <p>allowed: boolArray</p> <p>Default:</p>                                                                                                                                                                                                                    |
| div        | <p>Divisor (numerical value or quantity) to use in the logarithmic terms of the plp or ltp function. 0 means cal-</p>                                                                                                                                                                                                                                                 |

## Returns

record

## Example

This task fits a power logarithmic polynomial or a logarithmic transformed polynomial to one

### PARAMETER SUMMARY

|            |                                                                                                                                                                                                                                                                                                     |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| imagename  | Name of the input image(s). More than one image name can be given as an array, in which case the images are concatenated along the specified axis and the resultant image is what is used by the fitter. In this case, all images must have the same dimensions along all axes other than the fit a |
| box        | Rectangular region to select in direction plane. See "help par.box" for deta<br>Default is to use the entire direction plane.                                                                                                                                                                       |
| region     | Region selection. See "help par.region" for details. Default is to use the f                                                                                                                                                                                                                        |
| chans      | Channels to use. See "help par.chans" for details. Default is to use all cha                                                                                                                                                                                                                        |
| stokes     | Stokes planes to use. See "help par.stokes" for details. Default is to use a                                                                                                                                                                                                                        |
| axis       | Axis along which to do the fit(s). <0 means use the spectral axis or the<br>zeroth axis if a spectral axis is not present.                                                                                                                                                                          |
| mask       | Mask to use. See help par.mask. Default is none.                                                                                                                                                                                                                                                    |
| stretch    | Stretch the input mask if necessary and possible? Only used if a mask is spe<br>See help par.stretch.                                                                                                                                                                                               |
| minpts     | Minimum number of points necessary to attempt a fit.                                                                                                                                                                                                                                                |
| multifit   | Fit models at each pixel in region (true) or average profiles and fit a sing                                                                                                                                                                                                                        |
| spxtype    | Type of function to fit. "plp" => power logarithmic polynomial, "ltp" => log<br>transformed polynomial.                                                                                                                                                                                             |
| spxest     | REQUIRED. Initial estimates as array of numerical values for the spectral in<br>function coefficients. eg [1.5, -0.8] if fitting a plp function thought to b<br>1.5*(x/div)**(-0.8), or [0.4055, -0.8] if fitting an lpt function thought to<br>$\ln(1.5) - 0.8*\ln(x/div)$ .                       |
| spxfix     | Fix the corresponding spx function coefficients during the fit. True=>hold f                                                                                                                                                                                                                        |
| div        | Divisor (numerical value or quantity) to use in the logarithmic terms of the<br>function. 0 => calculate a useful value on the fly.                                                                                                                                                                 |
| spxsol     | Name of the function coefficients solution image to write.                                                                                                                                                                                                                                          |
| spxerr     | Name of the function coefficients error image to write.                                                                                                                                                                                                                                             |
| model      | Name of model image to write.                                                                                                                                                                                                                                                                       |
| residual   | Name of residual image to write.                                                                                                                                                                                                                                                                    |
| wantreturn | If true, return a record summarizing the fit results, if false, return false                                                                                                                                                                                                                        |
| stretch    | Stretch the mask if necessary and possible? See help par.stretch                                                                                                                                                                                                                                    |
| logresults | Output results to logger?                                                                                                                                                                                                                                                                           |
| logfile    | File in which to log results. Default is not to write a logfile.                                                                                                                                                                                                                                    |

append            Append results to logfile? Logfile must be specified. Default is to append  
sigma             Standard deviation numerical array, image name (string), or string array of  
                 concatenated to create the sigma image that is used by the fitter.  
outsigma          Name of output image used for standard deviation. Ignored if sigma is empty.

This application performs a non-linear, least squares fits using the Levenberg-Marquardt algorithm. It fits a logarithmic transformed polynomial to pixel values along a specified axis of an image or image array. This function is described in AIPS++ Note 224 (<http://www.astron.nl/casacore/trunk/casacore/doc/notes/224.html>) and in the AIPS++ User's Guide, Cambridge University Press. These functions are most often used for fitting the spectral index and high frequency turnover of a power law polynomial (plp) has the form

$$y = c0*(x/div)**(c1 + c2*\ln(x/div) + c3*\ln(x/div)**2 + \dots + cn*\ln(x/div)**(n - 1))$$

and a logarithmic transformed polynomial (ltp) is simply the result of this equation after taking the natural logarithm of both sides

$$\ln(y) = c0 + c1*\ln(x/div) + c2*\ln(x/div)**2 + \dots + cn*\ln(x/div)**n$$

Because the logarithm of the ordinate values must be taken before fitting a logarithmic transformed polynomial, all non-positive pixel values are effectively masked for the purposes of fitting.

The coefficients of the two forms are equal to each other except that c0 in the second equation is equal to  $\ln(c0)$  of the first. In the case of fitting a spectral index, which is traditionally represented by a power law, c0 is equal to c1.

In both cases, div is a numerical value used to scale abscissa values so they are closer to 1. This parameter improves the probability that the fit will converge. This parameter may be specified via the `div` parameter (the default) indicates that the application should determine a reasonable value for div, which is based on the range of the data.

$$div = 10**int(log10(sqrt(min(x)*max(x))))$$

where `min(x)` and `max(x)` are the minimum and maximum abscissa values, respectively.

So, for example, if  $S(\nu)$  is proportional to  $\nu^{\alpha}$  and you expect  $\alpha$  to be near -0.8, and your image(s) have spectral units of Hz, you would specify `spxest=[1.5, -0.8]` and `div=1e9` if fitting an ltp function close to  $\ln(1.5) - 0.8*\ln(x/1e9)$ .

#### A CAUTIONARY NOTE

Note that the likelihood of getting a reliable solution increases with the number of good data points and the quality of the initial estimate. It is possible that the first solution found might not be the best, so, if a solution is found, it is recommended that the fit be repeated using the solution of the first fit as the initial estimate for the new fit. This process should be repeated until the solutions from consecutive fits are very close. The convergent solution is very likely the best solution.

#### AXIS

The axis parameter indicates on which axis profiles should be fit; a value <0 indicates the other axis.

or if one does not exist, that the zeroth axis should be used.

#### MINIMUM NUMBER OF PIXELS

The minpts parameter indicates the minimum number of unmasked pixels that must be present in the region to be attempted. When multifit=T, positions with too few good points will be masked in any case.

#### ONE FIT OF REGION AVERAGE OR PIXEL BY PIXEL FIT

The multifit parameter indicates if profiles should be fit at each pixel in the selected region or if the region is averaged and the fit done to that average profile (false).

#### FUNCTION TYPE

Which function to fit is specified in the spxtype parameter. Only two values (case insensitive) are allowed: "plp" indicates that a power logarithmic polynomial should be fit. A value of "ltp" indicates that a linear polynomial should be fit.

#### INCLUDING STANDARD DEVIATIONS OF PIXEL VALUES

If the standard deviations of the pixel values in the input image are known and they vary in the region (near the edge of the band), they can be included in the sigma parameter. This parameter takes a 1D array or image must have one of three shapes: 1. the shape of the input image, 2. the same shape as the input image of all axes being one except for the fit axis which must have length corresponding to its length, 3. 1D array of length equal to the length of the fit axis in the input image. In cases 2 and 3, the array must be replicated such that the image that is ultimately used is the same shape as the input image. It is only the relative values that are important. A value of 0 means that pixel should not be included. If A has a higher standard deviation than pixel B, then pixel A is noisier than pixel B and will be given less weight. The weight of a pixel is the usual

$$\text{weight} = 1/(\text{sigma}*\text{sigma})$$

In the case of multifit=F, the sigma values at each pixel along the fit axis in the hyperplane are averaged and the resultant averaged standard deviation spectrum is the one used. The array is normalized such that the maximum value is 1. This mitigates a known overflow issue.

One can write the normalized standard deviation image used in the fit by specifying its name in the sigma parameter. It is used as sigma for subsequent runs.

#### RETURNED DICTIONARY STRUCTURE

The returned dictionary has a (necessarily) complex structure. First, there are keys "xUnit" and "yUnit" for the abscissa unit and the ordinate unit described by simple strings. Next there are arrays for the fit quality. These arrays have the shape of the specified region collapsed along the fit axis. The arrays are: attempted, converged, valid, niter. The attempted array has length of 1:

attempted: a boolean array indicating which fits were attempted (eg if too few unmasked points were present)  
converged: a boolean array indicating which fits converged. False if the fit was not attempted or did not converge  
valid: a boolean array indicating which solutions fall within the specified valid ranges. If a value or error is NaN is automatically marked as invalid.  
niter: an int array indicating the number of iterations for each profile, <0 if the fit was not attempted

direction: a string array containing the world direction coordinate for each profile

There is a "type" array having number of dimensions equal to the number of dimensions in the first n-1 dimensions is the same as the shape of the above arrays. The length of the last component is fit. The values of this array are all "POWER LOGARITHMIC POLYNOMIAL" or "LOGARITHMIC" on which type function was fit.

There will be a subdictionary accessible via the "plp" or "ltp" key (depending on which type) with subkeys "solution" and "error" which will each have an array of values. Each of these arrays is described above. The shape of the first n-1 dimensions will be the same as the shape of the input. The final dimension will have length equal to the number of parameters that were fit. Along this dimension is the corresponding fit result or associated error (depending on the array's associated key) of the fit. If the fit was not attempted or did not converge, a value of NAN will be present.

#### OUTPUT IMAGES

In addition to the returned dictionary, optionally one or more of any combination of output images can be written. The model and residual parameters indicate the names of the model and residual images to be written. Images that should not be written.

The parameters spxsol and spxerr are the names of the solution and error images to write, respectively. If more than one coefficient are fit, the image names will be appended with an underscore and a coefficient number ("\_0", "\_1", etc). These images contain the arrays for the associated parameters described in previous sections. Pixels for which fits were not attempted, did not converge, or resulted in NaN (not a number) or INF (infinity) will be masked as bad.

#### LPT vs PLP

Ultimately, the choice of which functional form to use in determining the spectral index is left to the user on the scientific goals. However, below is a summary of one user's experience and preferences.

If the weights are known or can be determined from the images (eg. the source-free image rms), I generally favor a weighted fit using the non-linear (power-law) model. An unweighted fit using the non-linear model has too much leverage to large flux values.

If the weights are unknown or will not be considered by the fitting algorithm, then I prefer the linear model. This does not work well in low signal-to-noise regions. A conservative mask could be created to mask these regions but this could hinder many science objectives.

#### EXAMPLES

```
res = spxfit(imagename=["im0.im","im1.im"], multifit=true, spxtype="plp", spxest=[0.5,2,0.1])
```

statwt-task.html

### **0.1.120 statwt**

Requires:

#### **Synopsis**

Reweight visibilities according to their scatter (Experimental)

#### **Arguments**



|           |                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                   |
| vis       | Name of measurement set<br>allowed: string<br>Default:                                                            |
| dorms     | Use rms instead of stddev?<br>allowed: bool<br>Default: False                                                     |
| byantenna | Estimate the noise per antenna -not implemented (vs. per baseline)<br>allowed: bool<br>Default: False             |
| sepacs    | If solving by antenna, treat autocorrs separately (not implemented)<br>allowed: bool<br>Default: True             |
| fitspw    | The signal-free spectral window:channels to estimate the scatter from<br>allowed: any<br>Default: variant         |
| fitcorr   | The signal-free correlation(s) to estimate the scatter from (not implemented)<br>allowed: any<br>Default: variant |
| combine   | Let estimates span changes in spw, corr, scan and/or state<br>allowed: any<br>Default: variant                    |
| timebin   | Bin length for estimates (not implemented)<br>allowed: string<br>Default: 0s                                      |
| minsamp   | Minimum number of unflagged visibilities for estimating the scatter<br>allowed: int<br>Default: 2                 |
| field     | Select field using ID(s) or name(s)<br>allowed: any<br>Default: variant                                           |
| spw       | Select spectral window/channels<br>allowed: any<br>Default: variant                                               |
| antenna   | Select data based on antenna/baseline<br>allowed: any<br>Default: variant                                         |
| timerange | Select data by time range<br>allowed: string<br>Default:                                                          |
| scan      | Select data by scan numbers<br>allowed: string<br>Default:                                                        |
| intent    | Select data by scan intents                                                                                       |

## Example

The WEIGHT and SIGMA columns of measurement sets are often set to arbitrary values (e.g. 1), or theoretically estimated from poorly known antenna and receiver properties. Many tasks (e.g. clean) are insensitive to an overall scale error in WEIGHT, but are affected by errors in the relative weights between visibilities. Other tasks, such as uvmodelfit, or anything which depends on theoretical estimates of the noise, require (reasonably) correct weights and sigmas. statwt empirically measures the visibility scatter (typically as a function of time, antenna, and/or baseline) and uses that to set WEIGHT and SIGMA. It is important that all necessary calibrations are applied to the data prior to running this task for correct determination of weights and sigmas.

Note: Some of the parameters (byantenna, sepacs, fitcorr, and timebin) are not fully implemented for CASA 3.4.

Keyword arguments:

vis -- Name of the measurement set.

default: none; example: vis='ngc5921.ms'

dorms -- Estimate the scatter using rms instead of the standard deviation?

Ideally the visibilities used to estimate the scatter, as selected by fitspw and fitcorr, should be pure noise. If you know for certain that they are, then setting dorms to True will give the best result. Otherwise, use False (standard sample standard deviation). More robust scatter estimates like the interquartile range or median absolute deviation from the median are not offered because they require sorting by value, which is not possible for complex numbers.

default: False

byantenna -- Assume that the noise is factorable by antenna (feed).

If false, treat it separately for each baseline  
(recommended if there is strong signal).

default: False (\*\* byantenna=True is not yet implemented)

sepacs -- If solving by antenna, treat autocorrelations separately.

(Acknowledge that what autocorrelations "see" is very  
different from what crosscorrelations see.)

```

 default: True (** not yet implemented)

--- Data Selection (see help par.selectdata for more detailed
 information)

fitspw -- The (ideally) signal-free spectral window:channels to
 estimate the scatter from.
 default: '' (All)

fitcorr -- The (ideally) signal-free correlations to
 estimate the scatter from.
 default: '' (All)
 *** not yet implemented

combine -- Let samples span multiple spws, corrs, scans, and/or states.
 combine = 'spw': Recommended when a line spans an entire spw
 - set fitspw to the neighboring spws and
 apply their weight to the line spw(s).
 However, the effect of the line signal per
 visibility may be relatively harmless
 compared to the noise difference between
 spws.
 combine = 'scan': Can be useful when the scan number
 goes up with each integration,
 as in many WSRT MSes.
 combine = ['scan', 'spw']: disregard scan and spw
 numbers when gathering samples.
 combine = 'spw,scan': Same as above.
 default: '' (None)

timebin -- Sample interval.
 default: '0s' or '-1s' (1 integration at a time)
 example: timebin='30s'
 '10' means '10s'
 *** not yet implemented

minsamp -- Minimum number of unflagged visibilities for estimating the
 scatter. Selected visibilities for which the weight cannot
 be estimated will be flagged. Note that minsamp is
 effectively at least 2 if dorms is False, and 1 if it is
 True.

field -- Select field using field id(s) or field name(s).
 [run listobs to obtain the list id's or names]
 default: ''=all fields If field string is a non-negative

```

integer, it is assumed to be a field index  
otherwise, it is assumed to be a field name  
field='0~2'; field ids 0,1,2  
field='0,4,5~7'; field ids 0,4,5,6,7  
field='3C286,3C295'; fields named 3C286 and 3C295  
field = '3,4C\*'; field id 3, all names starting with 4C

spw -- Select spectral window/channels for changing WEIGHT and SIGMA.  
default: ''=all spectral windows and channels  
spw='0~2,4'; spectral windows 0,1,2,4 (all channels)  
spw='<2'; spectral windows less than 2 (i.e. 0,1)  
spw='0:5~61'; spw 0, channels 5 to 61  
spw='0,10,3:3~45'; spw 0,10 all channels, spw 3 - chans 3 to 45.  
spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.  
spw = '\*:3~64' channels 3 through 64 for all sp id's  
spw = ' :3~64' will NOT work.  
statwt does not support multiple channel ranges per spectral  
window (',' ) because it is not clear whether to keep the ranges  
in the original spectral window or make a new spectral window  
for each additional range.

antenna -- Select antennas/baselines for changing WEIGHT and SIGMA.  
default: '' (all)  
Non-negative integers are assumed to be antenna indices, and  
anything else is taken as an antenna name.

Examples:

antenna='5&6': baseline between antenna index 5 and index 6.  
antenna='VA05&VA06': baseline between VLA antenna 5 and 6.  
antenna='5&6;7&8': baselines 5-6 and 7-8  
antenna='5': all baselines with antenna 5  
antenna='5,6,10': all baselines including antennas 5, 6, or 10  
antenna='5,6,10&': all baselines with *only* antennas 5, 6, or  
10. (cross-correlations only. Use &&  
to include autocorrelations, and &&&  
to get only autocorrelations.)  
antenna='!ea03,ea12,ea17': all baselines except those that  
include EVLA antennas ea03, ea12, or  
ea17.

timerange -- Select data based on time range:  
default = '' (all); examples,  
timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'  
Note: if YYYY/MM/DD is missing date, timerange defaults to the  
first day in the dataset  
timerange='09:14:0~09:54:0' picks 40 min on first day  
timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min

```

 on next day
 timerange='09:44:00' data within one integration of time
 timerange='>10:24:00' data after this time
scan -- Scan number range
 default: ''=all
intent -- Select by scan intent (state). Case sensitive.
 default: '' = all
Examples:
intent = 'CALIBRATE_ATMOSPHERE_REFERENCE'
intent = 'calibrate_atmosphere_reference'.upper() # same as above
Select states that include one or both of CALIBRATE_WVR.REFERENCE
or OBSERVE_TARGET_ON_SOURCE.
intent = 'CALIBRATE_WVR.REFERENCE, OBSERVE_TARGET_ON_SOURCE'
array -- (Sub)array number range
 default: ''=all
correlation -- Select correlations, e.g. 'rr, ll' or ['XY', 'YX'].
 default '' (all).
 NB: In CASA v4.5, non-trivial correlation selection has
 been disabled since it was not working correctly, and
 it is likely undesirable to set the weights in a
 correlation-dependent way.

observation -- Select by observation ID(s).
 default: '' = all
datacolumn -- Which data column to calculate the scatter from
 default='corrected'; example: datacolumn='data'
 Options: 'data', 'corrected', 'model', 'float_data'
 note: 'corrected' will fall back to DATA if CORRECTED_DATA
 is absent.

```

`tclean-task.html`

### **0.1.121    `tclean`**

Requires:

#### **Synopsis**

Radio Interferometric Image Reconstruction

#### **Description**

Form images from visibilities and reconstruct a sky model. This task handles continuum images and spectral line cubes, supports outlier fields, contains standard clean based algorithms along with algorithms for multi-scale and wideband image reconstruction, widefield imaging correcting for the w-term, full primary-beam imaging and joint mosaic imaging (with heterogeneous array support for ALMA).

#### **Arguments**

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| vis        | <p>Name(s) of input visibility file(s) default: none; example: vis='ngc5921.ms' vis=['ngc5921a.ms','ngc5921b.ms']; multiple MSes allowed: any Default: variant</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| selectdata | <p>Enable data selection parameters.<br/>allowed: bool<br/>Default: True</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| field      | <p>Select fields to image or mosaic. Use field id(s) or name(s). [go listobs' to obtain the list id's or names] default: " = all fields If field string is a non-negative integer, it is assumed to be a field index otherwise, it is assumed to be a field name field='0~2'; field ids 0,1,2 field='0,4,5~7'; field ids 0,4,5,6,7 field='3C286,3C295'; field named 3C286 and 3C295 field = '3,4C*'; field id 3, all names starting with 4C For multiple MS input, a list of field strings can be used: field = ['0~2','0~4']; field ids 0-2 for the first MS and 0-4 for the second field = '0~2'; field ids 0-2 for all input MSes</p> <p>allowed: any<br/>Default: variant</p>                                                                                                                                                                                                                                                                                                                                     |
| spw        | <p>Select spectral window/channels NOTE: channels de-selected here will contain all zeros if selected by the parameter mode subparameters. default: " =all spectral windows and channels spw='0~2,4'; spectral windows 0,1,2,4 (all channels) spw='0:5~61'; spw 0, channels 5 to 61 spw='&lt;2'; spectral windows less than 2 (i.e. 0,1) spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45. spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each. For multiple MS input, a list of spw strings can be used: spw=['0','0~3']; spw ids 0 for the first MS and 0-3 for the second spw='0~3' spw ids 0-3 for all input MS spw='3:10~20;50~60' for multiple channel ranges within spw id 3 spw='3:10~20;50~60,4:0~30' for different channel ranges for spw ids 3 and 4 spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10, spw 1, channels 20-30, and spw 2, channels, 1,2 and 3 spw='1~4;6:15~48' for channels 15 through 48 for spw ids 1,2,3,4 and 6</p> <p>allowed: any<br/>Default: variant</p> |
| timerange  | <p>Range of time to select from data<br/>default: " (all); examples, timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'<br/>Note: if YYYY/MM/DD is missing date defaults to first day in data set timerange='09:14:0~09:54:0' picks 40 min on first day timerange='25:00:00~27:30:00' picks 1 hr to 3 hr 30min on NEXT day timerange='09:44:00' pick data within one integration of time timerange='&gt; 10:24:00' data after this</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Returns**

void

**Example**

This is the first release of our refactored imager code. Although most features have been used and validated, there are many details that have not been thoroughly tested. Feedback will be much appreciated.

Usage Examples :

-----

- (A) A suite of test programs that demo all usable modes of tclean on small test datasets  
[https://svn.cv.nrao.edu/svn/casa/branches/release-4\\_5/gcwrap/python/scripts/test](https://svn.cv.nrao.edu/svn/casa/branches/release-4_5/gcwrap/python/scripts/test)
- (B) A set of demo examples for ALMA imaging  
[https://casaguides.nrao.edu/index.php/TCLEAN\\_and\\_ALMA](https://casaguides.nrao.edu/index.php/TCLEAN_and_ALMA)



testconcat-task.html

### 0.1.122 testconcat

Requires:

#### Synopsis

Concatenate the subtables of several visibility data sets, not the MAIN bulk data.

#### Arguments

| Inputs        |                                                                                                                  |
|---------------|------------------------------------------------------------------------------------------------------------------|
| vis           | Name(s) of input visibility files to be test-concatenated<br>allowed:       stringArray<br>Default:              |
| testconcatvis | Name of output MS containing the merged subtables<br>allowed:       string<br>Default:                           |
| frequitol     | Frequency shift tolerance for considering data as the same spwid<br>allowed:       any<br>Default:       variant |
| dirtol        | Direction shift tolerance for considering data as the same field<br>allowed:       any<br>Default:       variant |
| copypointing  | Copy all rows of the POINTING table.<br>allowed:       bool<br>Default:       True                               |

#### Example

The list of data sets given in the vis argument are concatenated into an output data set in testconcatvis without the bulk data of the MAIN table. This is useful for obtaining the information in the merged subtables without actually performing a time-consuming concatenation of the MAIN tables on disk.

Keyword arguments:

vis -- Name of input visibility files for which the subtables are to be combined  
default: none; example: vis = 'mydata.ms',  
vis=['src2.ms','ngc5921.ms','ngc315.ms']

testconcatvis -- Name of MS that will contain the concatenated subtables  
default: none; example: testconcatvis='test.ms'

freqtol -- Frequency shift tolerance for considering data to be in the same  
spwid. The number of channels must also be the same.  
default: '' do not combine unless frequencies are equal  
example: freqtol='10MHz' will not combine spwid unless they are  
within 10 MHz.

Note: This option is useful to combine spectral windows with very slight  
frequency differences caused by Doppler tracking, for example.

dirtol -- Direction shift tolerance for considering data as the same field  
default: '' means always combine.  
example: dirtol='1.arcsec' will not combine data for a field unless  
their phase center differ by less than 1 arcsec. If the field names  
are different in the input data sets, the name in the output data  
set will be the first relevant data set in the list.

copypointing -- Make a proper copy of the POINTING subtable (can be time consuming).  
If False, the result is an empty POINTING table.  
default: True

`tsdbaseline-task.html`

### **0.1.123   tsdbaseline**

Requires:

#### **Synopsis**

Fit/subtract a spectral baseline

#### **Description**

Task `tsdbaseline` fits and/or subtracts baseline from single-dish spectra. Given baseline parameters (baseline type, order, etc.), `tsdbaseline` computes the best-fit baseline for each spectrum by least-square fitting method and, if you want, subtracts it. The best-fit baseline parameters (including baseline type, coefficients of basis functions, etc.) and other values such as residual rms can be saved in various formats including ascii text (in human-readable format or CSV format) or baseline table (a CASA table). `Tsdbaseline` has another mode to 'apply' a baseline table to a MS data; for each spectrum in MS, the best-fit baseline is reproduced from the baseline parameters stored in the given baseline table and subtracted. Putting 'fit' and 'subtract' into separate processes can be useful for pipeline processing for huge dataset.

#### **Arguments**

|            |                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                    |
| infile     | name of input SD dataset<br>allowed: string<br>Default:                                                            |
| datacolumn | name of data column to be used ['data', 'float_data', or 'corrected']<br>allowed: string<br>Default: data          |
| antenna    | select data by antenna name or ID, e.g. 'PM03'<br>allowed: string<br>Default:                                      |
| field      | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                             |
| spw        | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                      |
| timerange  | select data by time range, e.g. '09:14:0~09:54:0' ("=all) (see examples in help)<br>allowed: string<br>Default:    |
| scan       | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                   |
| pol        | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                 |
| intent     | select data by observational intent, e.g. '*ON_SOURCE*' ("=all)<br>allowed: string<br>Default:                     |
| maskmode   | mode of setting additional channel masks. 'list' and 'auto' are available now.<br>allowed: string<br>Default: list |
| thresh     | S/N threshold for linefinder<br>allowed: double<br>Default: 5.0                                                    |
| avg_limit  | channel averaging for broad lines<br>allowed: int<br>Default: 4                                                    |
| minwidth   | the minimum channel width to detect as a line<br>allowed: int<br>Default: 4                                        |
| edge       | channels to drop at beginning and end of spectrum<br>allowed: intArray<br>Default: 639                             |
| blmode     | baselining mode ['fit' or 'apply']<br>allowed: string<br>Default: fit                                              |
| dosubtract | subtract baseline from input data [True, False]<br>allowed: bool<br>Default: True                                  |
| blformat   | format(s) of file(s) in which best-fit parameters are written ['text', 'csv', 'table' or ""]                       |

## Returns

void

## Example

```

Keyword arguments

```

```
infile -- name of input SD dataset
datacolumn -- name of data column to be used
 options: 'data', 'float_data', or 'corrected'
 default: 'data'
antenna -- select data by antenna name or ID
 default: '' (use all antennas)
 example: 'PM03'
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)/channels
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)
 spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
timerange -- select data by time range
 default: '' (use all)
 example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:00~09:54:00' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
```

```

example: scan='21~23' (scan IDs 21,22,23)
this selection is in addition to the other selections to data
pol -- select data by polarization IDs
default: '' (use all polarizations)
example: pol='0,1' (polarization IDs 0,1)
this selection is in addition to the other selections to data
intent -- select data by observational intent, also referred to as 'scan intent'
default: '' (use all scan intents)
example: intent='*ON_SOURCE*' (any valid scan-intent expression accepted by the MSS)
this selection is in addition to the other selections to data
maskmode -- mode of setting additional channel masks. When blmode='apply'
and/or blfunc='variable', maskmode and its subparameters
are ignored.
options: 'list' and 'auto' ('interact' will be available later)
default: 'list'
example: maskmode='auto' runs linefinder to detect line regions
to be excluded from fitting. this mode requires three
expandable parameters: thresh, avg_limit, minwidth, and edge.
NOTE maskmode='auto' is EXPERIMENTAL.
USE WITH CARE! May need to tweak the expandable parameters.
maskmode='list' uses the given masklist only: no additional
masks applied.
maskmode='interact' allows users to manually modify the
mask regions by dragging mouse on the spectrum plotter GUI.
use LEFT or RIGHT button to add or delete regions,
respectively.
>>> maskmode expandable parameters
thresh -- S/N threshold for linefinder. a single channel S/N ratio
above which the channel is considered to be a detection.
default: 5
avg_limit -- channel averaging for broad lines. a number of
consecutive channels not greater than this parameter
can be averaged to search for broad lines.
default: 4
minwidth -- the minimum channel width to detect as a line.
a line with number of consecutive channels less
than this parameter will not be detected as a line.
default: 4
edge -- channels to drop at beginning and end of spectrum
default: 0
example: edge=[1000] drops 1000 channels at beginning AND end.
edge=[1000,500] drops 1000 from beginning and 500
from end.
Note: For bad baselines threshold should be increased,
and avg_limit decreased (r even switched off completely by
setting this parameter to 1) to avoid detecting baseline

```

undulations instead of real lines.

`blmode` -- baselining mode.

options: 'fit', 'apply'

default: 'fit'

example: `blmode='fit'` calculates the best-fit baseline based on given baseline type, then (if you set `dosubtract=True`) subtract it from each spectrum. The information about best-fit baselines (baseline type, order, coefficients, etc.) can be stored in various formats (cf. `blformat`). `blmode='apply'` reads a baseline table as well as input MS, reproduces the best-fit baseline via info written in the baseline table, then subtracts it from each spectrum.

>>> `blmode` expandable parameters

`dosubtract` -- execute baseline subtraction in addition to fitting.

Note that `dosubtract=False` will be ignored if `bloutput` is given, that is, baseline subtraction will be always executed for the input MS in case `bloutput` is not specified.

options: (bool) True, False

default: True

`blformat` -- format(s) of file(s) in which best-fit parameters are written.

options: 'text', 'csv', 'table', and '' can be set for a single output. In case you want to output fitting results in multiple formats, a list containing the above keywords is accepted as well.

default: 'text'

example: (1) `blformat='text'` outputs an ascii text file with the best-fit baseline parameters written in human-readable format. It may be good to read, but you should mind it might be huge.

(2) `blformat='csv'` outputs a CSV file. For example, output of csv with `blfunc='poly'` is as below:

```
#scan, beam, spw, pol, MJD[s], fitrange (i.e. inverse mask), blfunc
4,0,17,0,4915973292.23,[[252;3828]],poly,1,767.647,-0.00956208,26.3
... .
```

(3) `blformat='table'` outputs a baseline table which can be used to apply afterwards.

(4) `blformat=''` doesn't output any parameter file.

(5) `blformat=['csv','table']` outputs both a CSV file and a baseline table.

(6) If one or more ''s appear in `blformat`, they are all ignored. For example, if `blformat=['','text','']` is given, only 'text' will be output.

(7) Elements of `blformat` other than '' must not

be duplicated. For example, `blformat=['text','', 'text']` is not accepted.

`bloutput` -- name(s) of file(s) in which best-fit parameters are written. If `bloutput` is a null string `''`, name(s) of baseline parameter file(s) will be set as follows:  
`<outfile>_blparam.txt` for `blformat='text'`,  
`<outfile>_blparam.csv` for `blformat='csv'`, and  
`<outfile>_blparam.bltable` for `blformat='table'`.  
Otherwise, `blformat` and `bloutput` must have the same length, and one-to-one correspondence is assumed between them. If there are `''` elements in `bloutput`, output file names will be set by following the above rules. If there are `''` elements in `blformat`, the corresponding `bloutput` elements will be ignored. Also, non-`''` `bloutput` elements corresponding to non-`''` `blformat` elements must not be duplicated.

default: `''`

example: (1) `bloutput=''` and `blformat=['csv','table']`: outputs a csv file `'<outfile>_blparam.csv'` and a baseline table `'<outfile>_blparam.bltable'`.  
(2) `bloutput=['foo.csv','']` and `blformat=['csv','table']`: outputs a csv file `'foo.csv'` and a baseline table `'<outfile>_blparam.bltable'`.  
(3) `bloutput=['foo.csv','bar.bltable']` and `blformat=['csv','']`: outputs a csv file `'foo.csv'` only.  
(4) `bloutput=['foo.csv','foo.csv','bar.bltable']` and `blformat=['csv','', 'table']`: the second `'foo.csv'` is ignored because it corresponds to the `blformat` element `''`, and thus outputs a csv file `'foo.csv'` and a baseline table `'bar.bltable'`.  
(5) `bloutput=['foo.csv','foo.csv','bar.bltable']` and `blformat=['csv','text','table']`: will be error since `'foo.csv'` is duplicated.  
(6) `bloutput=['foo.csv','bar.bltable']` and `blformat=['csv','', 'table']`: will be error since `bloutput` and `blformat` have different lengths.

`bltable` -- name of baseline table to apply  
default: `''`

`blfunc` -- baseline model function. In cases `blmode='apply'` or `blparam` is set, `blfunc` and its subparameters are ignored.  
options: `'poly'`, `'chebyshev'`, `'cspline'`, or `'variable'`  
default: `'poly'`

example: `blfunc='poly'` uses a single polynomial line of any order which should be given as an expandable parameter `'order'` to fit baseline.  
`blfunc='chebyshev'` uses Chebyshev polynomials.



blfunc='cspline' uses a cubic spline function, a piecewise cubic polynomial having C2-continuity (i.e., the second derivative is continuous at the joining points).

NOTE blfunc='variable' IS EXPERT MODE!!!

```
>>> blfunc expandable parameters
 order -- order of baseline model function
 options: (int) (<0 turns off baseline fitting)
 default: 5
 example: typically in range 2-9 (higher values
 seem to be needed for GBT)
 npiece -- number of the element polynomials of cubic spline curve
 options: (int) (<0 turns off baseline fitting)
 default: 2

 clipthresh -- clipping threshold for iterative fitting
 default: 3
 clipniter -- maximum iteration number for iterative fitting
 default: 0 (no iteration, i.e., no clipping)
 blparam -- the name of text file that stores per spectrum fit
 parameters. See below for details of format.
verbose -- (NOT SUPPORTED YET) output fitting results to logger. if False, the fitting results
 including coefficients, residual rms, etc., are not output to
 the CASA logger, while the processing speed gets faster.
 options: (bool) False
 default: False (verbose=True is currently unavailable)
showprogress -- (NOT SUPPORTED YET) show progress status for large data
 options: (bool) False (this capability is currently unavailable.)
 default: False
>>> showprogress expandable parameter
 minnrow -- (NOT SUPPORTED YET) minimum number of input spectra to show progress status
 default: 1000
outfile -- name of output file
 default: '' (<infile>_bs)
overwrite -- overwrite the output file if already exists
 options: (bool) True, False
 default: False
 NOTE this parameter is ignored when outform='ASCII'
```

-----  
DESCRIPTION  
-----

Task tsdbaseline performs baseline fitting/subtraction for single-dish spectra. The fit parameters, terms and rms of baseline can be saved into an ascii file

or baseline table. Subtracting baseline from data in input MS using existing baseline table is also possible.

#### ----- BASELINE MODEL FUNCTION -----

The list of available model functions are shown above (see Keyword arguments section). In general 'cspline' or 'chebyshev' are recommended since they are more stable than others. 'poly' will work for lower order but will be unstable for higher order fitting.

#### ----- SIGMA CLIPPING (ITERATIVE FITTING) -----

In general least square fitting is strongly affected by an extreme data so that the resulting fit makes worse. Sigma clipping is an iterative baseline fitting with data clipping based on a certain threshold. Threshold is set as a certain factor times rms of the resulting (baseline subtracted) spectra. If sigma clipping is on, baseline fit/removal is performed several times. After each baseline subtraction, the data whose absolute value is above threshold are detected and those data are excluded from the next round of fitting. By using sigma clipping, extreme data are excluded from the fit so that resulting fit is more robust.

The user is able to control a multiplication factor using parameter clipthresh for clipping threshold based on rms. Actual threshold for sigma clipping will be (clipthresh) x (rms of spectra). Also, the user can specify number of maximum iteration to the parameter clipniter.

In general, sigma clipping will lower the performance since it increases number of fits per spectra. However, it is strongly recommended to turn on sigma clipping unless you are sure that the data is free from any kind of extreme values that may affect the fit.

#### ----- PER SPECTRUM FIT PARAMETERS -----

Per spectrum baseline fitting parameter is accepted in blfunc='variable'. Note this is an expert mode. The fitting parameters should be defined in a text file for each spectrum in the input MS. The text file should store comma separated values in order of:  
row ID, polarization ID, masklist, blfunc, order, npiece, nwave, clipthresh, clipniter, use\_linefinder, thresh, edge, chan\_avg\_limit.  
Each row in the text file must contain the following keys and values:  
\* 'row': row number,  
\* 'blfunc': function name.

```

 available ones include, 'poly', 'chebyshev', 'cspline',
* 'order': maximum order of polynomial. needed when blfunc='poly'
 or 'chebyshev',
* 'npiece': number or piecewise polynomial. needed when blfunc='cspline',
and

```

example:

```

#row,pol,masklist,blfunc,order,npiece,nwave,clipthresh,clipniter,use_lf,thres,edge,avg_limit
0,0,[[500,1600]],poly,5,,,,,,,,,
0,1,,chebyshev,10,,,,,,,,,
1,0,,cspline,,4,,,,,,,,,

```

tsdcal-task.html

### 0.1.124 tsdcal

Requires:

#### Synopsis

MS SD calibration task

#### Description

Task tsdcal is an implementation of a calibration scheme like as interferometry, i.e., generate caltables and apply them. Available calibration modes are 'ps', 'otfraster', and 'tsys'. Those modes generates caltables for sky or Tsys calibration. The caltables can be applied to the data by using calmode 'apply'. First two calibration modes, 'ps' and 'otfraster', generate sky calibration tables. The user should choose appropriate calibration mode depending on the data. Use case for each mode is as follows:

'ps': position switch (including OTF) with explicit reference (OFF) spectra

'otfraster': raster OTF scan without explicit OFFs

So, if the data contains explicit reference spectra, 'ps' should be used.

Otherwise, 'otfraster' is appropriate for raster OTF data. Non-raster OTF data is not supported yet. In 'otfraster' mode, the task first try to find several integrations near edge as OFF spectra, then the data are calibrated using those OFFs. If the observing pattern is raster, you should use the 'otfraster' mode to calibrate data. The 'otfraster' mode is designed for OTF observations without explicit OFF spectra. However, these modes should work even if explicit reference spectra exist. In this case, these spectra will be ignored and spectra near edges detected by edge marker will be used as reference.

Except for how to choose OFFs, the procedure to derive calibrated spectra is common for the above two modes. Selected (or preset) OFF integrations are separated by its continuity in time domain, averaged in each segment, then interpolated to timestamps for ON integrations. Effectively, it means that OFF integrations are averaged by each OFF spectrum for 'ps' mode, averaged by either ends of each raster row for 'otfraster' mode. The formula for calibrated spectrum is

$$\text{Tsys} * (\text{ON} - \text{OFF}) / \text{OFF}.$$

#### Arguments

|            |                                                                                                                                                                                                                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                                                                                                                                                                                                            |
| infile     | name of input SD dataset (must be MS)<br>allowed: string<br>Default:                                                                                                                                                                                                                                                                       |
| calmode    | SD calibration mode<br>allowed: string<br>Default: ps                                                                                                                                                                                                                                                                                      |
| fraction   | fraction of the OFF data to mark<br>allowed: any<br>Default: variant 10%                                                                                                                                                                                                                                                                   |
| noff       | number of the OFF data to mark<br>allowed: int<br>Default: -1                                                                                                                                                                                                                                                                              |
| width      | width of the pixel for edge detection<br>allowed: double<br>Default: 0.5                                                                                                                                                                                                                                                                   |
| elongated  | whether observed area is elongated in one direction or not<br>allowed: bool<br>Default: False                                                                                                                                                                                                                                              |
| applytable | (List of) sky and/or tsys tables<br>allowed: any<br>Default: variant                                                                                                                                                                                                                                                                       |
| interp     | Interpolation type in time[,freq]. Valid options are "nearest", "linear", "cspline", or any numeric string that indicates an order of polynomial interpolation. You can specify interpolation type for time and frequency separately by joining two of the above options by comma (e.g., "linear,cspline").<br>allowed: string<br>Default: |
| spwmap     | A dictionary indicating spw combinations to apply Tsys calibration to target. The key should be spw for Tsys calibration and its associated value must be a list of science spws to be applied.<br>allowed: any<br>Default: variant                                                                                                        |
| outfile    | name of output file (See a WARNING in help)<br>allowed: string<br>Default:                                                                                                                                                                                                                                                                 |
| overwrite  | overwrite the output file if already exists<br>allowed: bool<br>Default: False                                                                                                                                                                                                                                                             |
| field      | select data by field IDs and names, e.g. '3C2*' (" = all)<br>allowed: string<br>Default:                                                                                                                                                                                                                                                   |
| spw        | select data by spw IDs (spectral windows), e.g., '3,5,7' (" = all)<br>allowed: string<br>Default:                                                                                                                                                                                                                                          |
| scan       | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                                                                                                                                                                                                                                           |

## Returns

void

## Example

Keyword arguments:

infile -- Name of input SD dataset

calmode -- Calibration mode. If you want to generate calibration table or apply existing calibration tables, set calmode to simple string. On the other hand, if you want to calibrate data on-the-fly, you have to set calmode to a composite calmode string separated by comma. So far, sky calibration has two types, 'ps' and 'otfraster'. If observation is configured to observe reference position, calmode must be 'ps'. Otherwise, 'otfraster' should be used. Non-raster observing pattern is not supported yet (e.g., Lissajous).

options: 'ps','otfraster','tsys','apply'

default: 'ps'

example: Here is an example for composite calmode.

'ps,apply' (do sky cal and apply)

'ps,tsys,apply' (do sky and Tsys cal and apply)

>>> calmode expandable parameter

fraction -- Edge marker parameter of 'otfraster'.

Specify a number of OFF integrations (at each side of the raster rows in 'otfraster' mode) as a fraction of total number of integrations. In 'otfraster' mode, number of integrations to be marked as OFF, n\_off, is determined by the following formula,

$$n\_off = \text{floor}(\text{fraction} * n),$$

where n is number of integrations per raster row. Note that n\_off from both sides will be marked as OFF so that twice of specified fraction will be marked at most. For example, if you specify fraction='10%', resultant fraction of OFF integrations will be 20% at most.

default: '10%'

options: '20%' in string style or float value less than 1.0 (e.g. 0.15).

'auto' is available only for 'otfraster'.

noff -- Edge marking parameter for 'otfraster'.  
It is used to specify a number of OFF spectra near edge directly. Value of noff comes before setting by fraction. Note that n\_off from both sides will be marked as OFF so that twice of specified noff will be marked at most.  
default: -1 (use fraction)  
options: any positive integer

applytable -- List of sky/Tsys calibration tables you want to apply.  
default: ''

interp -- Interpolation method in time and frequency axis.  
Set comma separated method strings if you want to use different interpolation in time and frequency.  
options: 'linear', 'cspline', 'nearest',  
any numeric string indicating an order of polynomial.  
default: '' (linear in time and frequency)  
example: 'linear,cspline' (linear in time, cubic spline in frequency)  
'linear,3' (linear in time, third order polynomial in frequency)  
'nearest' (nearest in time and frequency)

spwmap -- Dictionary defining transfer of Tsys calibration.  
Key must be spw for Tsys and its value must be a list of spws for science target.  
default: {}  
example: {1: [5,6], 3: [7,8]}  
Tsys in spw 1 is transferred to spws 5 and 6 while Tsys in spw 3 is to spws 7 and 8.

field -- select data by field IDs and names  
default: '' (use all fields)  
example: field='3C2\*' (all names starting with 3C2)  
field='0,4,5~7' (field IDs 0,4,5,6,7)  
field='0,3C273' (field ID 0 or field named 3C273)  
this selection is in addition to the other selections to data

spw -- select data by spw IDs (spectral windows)  
NOTE this task only supports spw ID selection and ignores channel selection.  
default: '' (use all spws and channels)  
example: spw='3,5,7' (spw IDs 3,5,7; all channels)  
spw='<2' (spw IDs less than 2, i.e., 0,1; all channels)  
spw='30~45GHz' (spw IDs with the center frequencies in range 30-45GHz; all

```

 this selection is in addition to the other selections to data
 NOTE spw input must be '' (''= all) in calmode='tsys'.
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
 NOTE scan input must be '' (''= all) in calmode='tsys'.
outfile -- Name of output file
 NOTE if you omit, behavior of the task depends on calmode.
 If calmode includes 'apply', then omitting outfile indicates
 that infile is overwritten by the calibrated data. In this case,
 you have to set overwrite to True. If calmode doesn't include
 'apply', omitting outfile indicates that the task will use
 default outfile name based on infile and predefined suffix
 ('_sky' for sky, '_tsys' for Tsys).
 default: '' (<infile>_<suffix> for calibration
 while overwrite infile for apply mode)
overwrite -- overwrite the output file if already exists
 options: (bool) True,False
 default: False
 NOTE this parameter is ignored when outform='ASCII'

```

#### DESCRIPTION:

Task tsdcal is an implementation of a calibration scheme like as interferometry, i.e., generate caltables and apply them. Available calibration modes are 'ps', 'otfraster', and 'tsys'. Those modes generates caltables for sky or Tsys calibration. The caltables can be applied to the data by using calmode 'apply'.

First two calibration modes, 'ps', and 'otfraster', generate sky calibration tables. The user should choose appropriate calibration mode depending on the data. Use case for each mode is as follows:

```

'ps': position switch (including OTF) with explicit
 reference (OFF) spectra
'otfraster': raster OTF scan without explicit OFFs

```

So, if the data contains explicit reference spectra, 'ps' should be used. Otherwise, 'otfraster' is appropriate for raster OTF, respectively. Non raster OTF data is not supported yet. In 'otfraster' mode, the task first try to find several integrations near edge as OFF spectra, then the data are calibrated using those OFFs. If the observing pattern is raster, you should use the 'otfraster' mode to calibrate data. The 'otfraster' mode is designed for OTF observations



without explicit OFF spectra. However, these modes should work even if explicit reference spectra exist. In this case, these spectra will be ignored and spectra near edges detected by edge marker will be used as reference.

Except for how to choose OFFs, the procedure to derive calibrated spectra is common for the above two modes. Selected (or preset) OFF integrations are separated by its continuity in time domain, averaged in each segment, then interpolated to timestamps for ON integrations. Effectively, it means that OFF integrations are averaged by each OFF spectrum for 'ps' mode, averaged by either ends of each raster row for 'otfraster' mode. The formula for calibrated spectrum is

$$\text{Tsyz} * (\text{ON} - \text{OFF}) / \text{OFF}.$$

You can calibrate data on-the-fly like sdcal task by setting calmode to a composite calmode string separated by comma. For example, calmode='ps,apply' means doing sky calibration and apply it on-the-fly. In this case, caltable is generated as a temporary plain table and will be deleted at the end. Allowed calibration modes in this task is as follows:

```
ps
 generate sky caltable using 'ps' mode
otfraster
 generate sky caltable using 'otfraster' mode
tsys
 generate tsys caltable
apply
 apply caltables specified by applytable parameter
ps,apply
 generate temporary sky caltable using 'ps' mode and
 apply it. also apply caltables specified by applytable
ps,tsys,apply
 generate temporary sky caltable using 'ps' mode as well
 as temporary tsys caltable, and apply them.
otfraster,apply
 generate temporary sky caltable using 'otfraster' mode
 and apply it. also apply caltables specified by applytable
otfraster,tsys,apply
 generate temporary sky caltable using 'otfraster' mode
 as well as temporary tsys caltable, and apply them.
```

There are several control parameters for sky/Tsys calibration and application of caltables. See the above parameter description.

In ALMA, Tsys measurement is usually done using different spectral setup from spectral windows for science target. In this case, `tsdcal` transfers Tsys values to science spectral windows in the application stage. To do that, the user has to give a list of spectral windows for Tsys measurement as well as mapping between spectral windows for Tsys measurement and science target. These can be specified by parameters `'tsysspw'` and `'spwmap'`, which are defined as subparameters of `'calmode'`. For example, suppose that Tsys measurements for science windows 17, 19, 21, and 23 are done in spw 9, 11, 13, and 15, respectively. In this case, `tsysspw` and `spwmap` should be specified as follows:

```
tsysspw = '9,11,13,15'
spwmap = {9:[17],11:[19],13:[21],15:[23]}
```

Below is an example of full specification of task parameters for `calmode` of `'ps,tsys,apply'`:

```
default(tsdcal)
infile = 'foo.asap'
calmode = 'ps,tsys,apply'
spw = ''
tsysspw = '9,11,13,15'
spwmap = {9:[17],11:[19],13:[21],15:[23]}
outfile = 'bar.asap'
tsdcal()
```

Note that, in contrast to `applycal` task, `spwmap` must be a dictionary with Tsys spectral window as key and a list of corresponding science spectral window as value. Note also that the parameter `'spw'` should not be used to specify a list of spectral windows for Tsys measurement. It is intended to select data to be calibrated so that the list should contain spectral windows for both science target and Tsys measurement. The task will fail if you use `'spw'` instead of `'tsysspw'`.

For Tsys calibration, the user is able to choose whether Tsys is averaged in spectral axis or not. If `tsysavg` is `False` (default), resulting Tsys is spectral value. On the other hand, when `tsysavg` is `True`, Tsys is averaged in spectral axis before output. The channel range for averaging is whole channels by default. If channel range is specified by `tsysspw` string, it is used for averaging. The user can specify channel range with ms selection syntax. For example,

```
tsysspw = '1:0~100'
```

specifies spw 1 for Tsys calibration and channel range between channel 0 and 100 for averaging. You can specify more than one ranges per spw.

```
tsysspw = '1:0~100;200~400'
```

In this case, selected ranges are between 0 and 100 plus 200 and 400. Note that even if multiple ranges are selected, the task average whole ranges together and output single averaged value. You can specify multiple spws by separating comma.

```
tsysspw = '1:0~100,3:400~500'
```

Note that specified channel range is ignored if `tsysavg` is `False`.

`tsdfit-task.html`

### **0.1.125   tsdfit**

Requires:

#### **Synopsis**

Fit a spectral line

#### **Description**

Task tsdfit is a basic line-fitter for single-dish spectra. It assumes that the spectra have been calibrated in tsdcal or sdreduce.

#### **Arguments**

|            |                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------|
| Outputs    |                                                                                                                    |
| xstat      | RETURN ONLY: a Python dictionary of line statistics<br>allowed: any<br>Default: variant                            |
| Inputs     |                                                                                                                    |
| infile     | name of input SD dataset<br>allowed: string<br>Default:                                                            |
| datacolumn | name of data column to be used ['data', 'float_data', or 'corrected_data']<br>allowed: string<br>Default: data     |
| antenna    | select data by antenna name or ID, e.g. 'PM03'<br>allowed: string<br>Default:                                      |
| field      | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                             |
| spw        | select data by IF IDs (spectral windows), e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                      |
| timerange  | select data by time range, e.g. '09:14:0~09:54:0' ("=all)<br>(see examples in help)<br>allowed: string<br>Default: |
| scan       | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                   |
| pol        | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                 |
| intent     | select data by observational intent, e.g. '*ON_SOURCE*' ("=all)<br>allowed: string<br>Default:                     |
| fitfunc    | function for fitting<br>allowed: string<br>Default: gaussian                                                       |
| fitmode    | mode for fitting<br>allowed: string<br>Default: list                                                               |
| nfit       | list of number of lines to fit in maskline region.<br>allowed: intArray<br>Default: 0                              |
| thresh     | S/N threshold for linefinder<br>allowed: double<br>Default: 5.5                                                    |
| min_nchan  | minimum number of consecutive channels for linefinder<br>allowed: int<br>Default: 3                                |
| avg_limit  | channel averaging for broad lines<br>allowed: int<br>Default: 4                                                    |
| box_size   | running mean box size<br>allowed: double                                                                           |

## Returns

variant

## Example

```

Keyword arguments

```

```
infile -- name of input SD dataset
datacolumn -- name of data column to be used
 options: 'data', 'float_data', or 'corrected_data'
 default: 'data'
antenna -- select data by antenna name or ID
 default: '' (use all antennas)
 example: 'PM03'
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by IF IDs (spectral windows)/channels
 default: '' (use all IFs and channels)
 example: spw='3,5,7' (IF IDs 3,5,7; all channels)
 spw='<2' (IF IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (IF IDs with the center frequencies in range 30-45GHz; all channels)
 spw='0:5~61' (IF ID 0; channels 5 to 61; all channels)
 spw='3:10~20;50~60' (select multiple channel ranges within IF ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for IF IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for IF IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
timerange -- select data by time range
 default: '' (use all)
 example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:00~09:54:00' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
```

```

 example: scan='21~23' (scan IDs 21,22,23)
 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
intent -- select data by observational intent, also referred to as 'scan intent'
 default: '' (use all scan intents)
 example: intent='*ON_SOURCE*' (any valid scan-intent expression accepted by the MSS)
 this selection is in addition to the other selections to data
fitfunc -- function for fitting
 options: 'gaussian', 'lorentzian'
 default: 'gaussian'
fitmode -- mode for fitting
 options: 'list' ('auto' and 'interact' will be available later)
 default: 'list'
 example: 'list' will use channel ranges specified in the parameter
 spw to fit for lines
 'auto' will use the linefinder to fit for lines
 using the following parameters
 'interact' allows adding and deleting mask
 regions by drawing rectangles on the plot
 with mouse. Draw a rectangle with LEFT-mouse
 to ADD the region to the mask and with RIGHT-mouse
 to DELETE the region.
>>> fitmode expandable parameters
 thresh -- S/N threshold for linefinder. a single channel S/N ratio
 above which the channel is considered to be a detection.
 default: 5
 min_nchan -- minimum number of consecutive channels required to
 pass threshold
 default: 3
 avg_limit -- channel averaging for broad lines. a number of
 consecutive channels not greater than this parameter
 can be averaged to search for broad lines.
 default: 4
 box_size -- running mean box size specified as a fraction
 of the total spectrum length
 default: 0.2
 edge -- channels to drop at beginning and end of spectrum
 default: 0
 example: edge=[1000] drops 1000 channels at beginning AND end.
 edge=[1000,500] drops 1000 from beginning and 500
 from end

```

Note: For bad baselines threshold should be increased,

and avg\_limit decreased (or even switched off completely by setting this parameter to 1) to avoid detecting baseline undulations instead of real lines.

nfit -- list of number of lines to fit in each region specified by the parameter spw  
 default: [0] (no fitting)  
 example: nfit=[1] for single line in single region,  
           nfit=[2] for two lines in single region,  
           nfit=[1,1] for single lines in each of two regions, etc.

outfile -- name of output file  
 default: no output fit file  
 example: 'mysd.fit'

overwrite -- overwrite the output file if already exists  
 options: (bool) True, False  
 default: False

-----  
 Returns  
 -----

a Python dictionary of line statistics  
 keys: 'peak', 'cent', 'fwhm', 'nfit'  
 example: each value except for 'nfit' is a list of lists with  
           a list of 2 entries [fitvalue,error] per component.  
           e.g. xstat['peak']=[[234.9, 4.8],[234.2, 5.3]]  
           for 2 components.

-----  
 DESCRIPTION  
 -----

Task tsdfit is a basic line-fitter for single-dish spectra. It assumes that the spectra have been calibrated in tsdcal or sdreduce.

Note that multiple scans, IFs, and polarizations can in principle be handled, but we recommend that you use scan, field, spw, and pol to give a single selection for each fit.

-----  
 FITMODE  
 -----

As described in the parameter description section, tsdfit implements a fitting mode 'list' so far. The 'list' mode allows users to set initial guess manually. Only controllable parameter for the guess is range of the line region and number of lines per region. In 'list' mode, users must give line region via spw parameter by using ms selection syntax while number of lines per region can be



specified via `nfit` parameter. For example,

```
spw = '17:1500~2500'
nfit = [1]
```

will set line region between channels 1500 and 2500 for spw 17, and indicate that there is only one line in this region. Specifying single region with multiple line is also possible but is not recommended.

[tsdsmoother-task.html](#)

### **0.1.126   tdsmsmooth**

Requires:

#### **Synopsis**

Smooth spectral data

#### **Description**

Task tdsmsmooth performs smoothing along spectral axis using user-specified smoothing kernel. Currently gaussian and boxcar kernels are supported.

#### **Arguments**

|            |                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                    |
| infile     | name of input SD dataset<br>allowed: string<br>Default:                                                            |
| datacolumn | name of data column to be used ['data', 'float_data', or 'corrected']<br>allowed: string<br>Default: data          |
| antenna    | select data by antenna name or ID, e.g. 'PM03'<br>allowed: string<br>Default:                                      |
| field      | select data by field IDs and names, e.g. '3C2*' ("=all)<br>allowed: string<br>Default:                             |
| spw        | select data by spectral window IDs, e.g. '3,5,7' ("=all)<br>allowed: string<br>Default:                            |
| timerange  | select data by time range, e.g. '09:14:0~09:54:0' ("=all)<br>(see examples in help)<br>allowed: string<br>Default: |
| scan       | select data by scan numbers, e.g. '21~23' ("=all)<br>allowed: string<br>Default:                                   |
| pol        | select data by polarization IDs, e.g. '0,1' ("=all)<br>allowed: string<br>Default:                                 |
| intent     | select data by observational intent, e.g. '*ON_SOURCE*' ("=all)<br>allowed: string<br>Default:                     |
| kernel     | spectral smoothing kernel type<br>allowed: string<br>Default: gaussian                                             |
| kwidth     | smoothing kernel width in channel<br>allowed: int<br>Default: 5                                                    |
| outfile    | name of output file<br>allowed: string<br>Default:                                                                 |
| overwrite  | overwrite the output file if already exists<br>allowed: bool<br>Default: False                                     |

## Returns

void

## Example

```

Keyword arguments

infile -- name of input SD dataset
datacolumn -- name of data column to be used
 options: 'data', 'float_data', or 'corrected'
 default: 'data'
antenna -- select data by antenna name or ID
 default: '' (use all antennas)
 example: 'PM03'
field -- select data by field IDs and names
 default: '' (use all fields)
 example: field='3C2*' (all names starting with 3C2)
 field='0,4,5~7' (field IDs 0,4,5,6,7)
 field='0,3C273' (field ID 0 or field named 3C273)
 this selection is in addition to the other selections to data
spw -- select data by spectral window IDs/channels
 default: '' (use all spws and channels)
 example: spw='3,5,7' (spw IDs 3,5,7; all channels)
 spw='<2' (spw IDs less than 2, i.e., 0,1; all channels)
 spw='30~45GHz' (spw IDs with the center frequencies in range 30-45GHz; all
 spw='0:5~61' (spw ID 0; channels 5 to 61; all channels)
 spw='3:10~20;50~60' (select multiple channel ranges within spw ID 3)
 spw='3:10~20,4:0~30' (select different channel ranges for spw IDs 3 and 4)
 spw='1~4;6:15~48' (for channels 15 through 48 for spw IDs 1,2,3,4 and 6)
 this selection is in addition to the other selections to data
timerange -- select data by time range
 default: '' (use all)
 example: timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:00~09:54:00' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
 this selection is in addition to the other selections to data
scan -- select data by scan numbers
 default: '' (use all scans)
 example: scan='21~23' (scan IDs 21,22,23)
```

```

 this selection is in addition to the other selections to data
pol -- select data by polarization IDs
 default: '' (use all polarizations)
 example: pol='0,1' (polarization IDs 0,1)
 this selection is in addition to the other selections to data
intent -- select data by observational intent, also referred to as 'scan intent'
 default: '' (use all scan intents)
 example: intent='*ON_SOURCE*' (any valid scan-intent expression accepted by the MSS)
 this selection is in addition to the other selections to data
kernel -- type of spectral smoothing kernel
 options: 'gaussian', 'boxcar'
 default: 'gaussian' (no smoothing)

>>>kernel expandable parameter
 kwidth -- width of spectral smoothing kernel
 options: (int) in channels
 default: 5
outfile -- name of output file
 default: '' (<infile>_bs)
overwrite -- overwrite the output file if already exists
 options: (bool) True, False
 default: False
 NOTE this parameter is ignored when outform='ASCII'

```

```

DESCRIPTION

```

Task `tsdsmooth` performs smoothing along spectral axis using user-specified smoothing kernel. Currently gaussian and boxcar kernels are supported.

uvcontsub-task.html

### **0.1.127 uvcontsub**

Requires:

#### **Synopsis**

Continuum fitting and subtraction in the uv plane

#### **Description**

#### **Arguments**

|              |                                                                                                                             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------|
| Inputs       |                                                                                                                             |
| vis          | Name of input MS. Output goes to vis + ".contsub"<br>(will be overwritten if already exists)<br>allowed: string<br>Default: |
| field        | Select field(s) using id(s) or name(s)<br>allowed: any<br>Default: variant                                                  |
| fitspw       | Spectral window:channel selection for fitting the continuum<br>allowed: string<br>Default:                                  |
| excludechans | exclude Spectral window:channel selection in fitspw for fitting<br>allowed: bool<br>Default: False                          |
| combine      | Data axes to combine for the continuum estimation<br>(none, or spw and/or scan)<br>allowed: string<br>Default:              |
| solint       | Continuum fit timescale (int recommended!)<br>allowed: any<br>Default: variant int                                          |
| fitorder     | Polynomial order for the fits<br>allowed: int<br>Default: 0                                                                 |
| spw          | Spectral window selection for output<br>allowed: string<br>Default:                                                         |
| want_cont    | Create vis + ".cont" to hold the continuum estimate.<br>allowed: bool<br>Default: False                                     |

## Example

Continuum fitting and subtraction in the uv plane:

This task estimates the continuum emission by fitting polynomials to the real and imaginary parts of the spectral windows and channels selected by fitspw. This fit represents a model of the continuum in all channels.

The fitted continuum spectrum is subtracted from all channels selected in spw, and the result (presumably only line emission) is stored in a new MS (vis + ".contsub"). If an MS with the output name already exists, it will be overwritten. It will read from the CORRECTED\_DATA column of vis if it is present, or DATA if it is not. Whichever column is read is presumed to have already been calibrated.

If want\_cont is True, the continuum fit is placed in a second new MS (vis + '.cont', also overwritten if it already exists). N.B. because the continuum model is necessarily a smoothed fit, images made with it are liable to have their field of view reduced in some strange way. Images of the continuum should be made by simply excluding the line channels (and probably averaging the remaining ones) in clean.

Keyword arguments:

vis -- Name of input visibility file

default: none; example: vis='ngc5921.ms'

field -- Field selection for continuum estimation and subtraction.

The estimation and subtraction is done for each selected field in turn. (Run listobs to get lists of the ID and names.)

default: field = '' means select all fields

field = 1 # will get field\_id=1 (if you give it an integer, it will retrieve the source with that index.

field = '1328+307' specifies source '1328+307'

field = '13\*' will retrieve '1328+307' and any other fields beginning with '13'

fitspw -- Selection of spectral windows and channels to use in the fit for the continuum, using general spw:chan syntax.

The ranges of channels also can be specified by frequencies as in the MS selection syntax (spw ids are required but '\*' can be used, see the example below).

See the note under combine.

default: '' (all)

example: fitspw='0:5~30;40~55'

--> select the ranges by channels in the spw id 0

fitspw='0:5~30;40~55,1:10~25;45~58,2'

--> select channel ranges 5-30 and 40-55 for the spw id 0, 10-25 and 45-58 for spwid 1, and use all channels for spwid 2

fitspw='0:113.767~114.528GHz;114.744~115.447GHz'

--> select the ranges by frequencies in the spw id 0

fitspw='0:113.767~114.528GHz;114.744~115.447GHz,1:111.892~112.654GHz'

--> select the different ranges by frequencies for the spw id 0

fitspw='\*:113.767~114.528GHz;114.744~115.447GHz'



```

--> select the same frequency ranges for all the relevant s
>>> expandable parameter for fitspw
 excludechans - if True, it will exclude the spws:channels specified in fitspw
 for the fit
 default: False (use fitspw for the fit)
 example: fitspw='0:114.528GHz~114.744GHz'; excludechans=True
 --> exclude the frequency range, 114.528GHz - 114.744GHz in the spw
combine -- Data axes to combine for the continuum estimate.
It must include 'spw' if spw contains spws that are not in
fitspw!
 default: '' --> solutions will break at scan, field, and spw
 boundaries according to solint
 Options: '', 'spw', 'scan', or 'spw, scan'
 example: combine='spw' --> form spw-merged continuum estimate
solint -- Timescale for per-baseline fit (units optional)
default (recommended): 'int' --> no time averaging, do a
 fit for each integration and let the
 noisy fits average out in the image.

example: solint='10s' --> average to 10s before fitting
 10 or '10' --> '10s' (unitless: assumes seconds)
 options: 'int' --> per integration
 'inf' --> per scan

 If solint is longer than 'int', the continuum estimate can be
 corrupted by time smearing!

fitorder -- Polynomial order for the fits of the continuum w.r.t.
 frequency. fitorders > 1 are strongly discouraged
 because high order polynomials have more flexibility, may
 absorb line emission, and tend to go wild at the edges of
 fitspw, which is not what you want.

default: 0 (constant); example: fitorder=1

spw -- Optional per spectral window selection of channels to include
 in the output. See the note under combine.

 The spectral windows will be renumbered to start from 0, as in
 split.
want_cont -- Create vis + '.cont' to hold the continuum estimate.
default: 'False'; example: want_cont=True
The continuum estimate will be placed in vis + '.cont'
 async -- Run task in a separate process (return CASA prompt)
 default: False; example: async=True

```



uvcontsub3-task.html

### **0.1.128 uvcontsub3**

Requires:

#### **Synopsis**

An experimental clone of uvcontsub

#### **Description**

#### **Arguments**

|             |                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------|
| Inputs      |                                                                                                     |
| vis         | Name of input MS. Output goes to vis + ".contsub"<br>allowed: string<br>Default:                    |
| fitspw      | Spectral window:channel selection for fitting the continuum<br>allowed: string<br>Default:          |
| combine     | Data axes to combine for the continuum estimation (none ("") or spw)<br>allowed: string<br>Default: |
| fitorder    | Polynomial order for the fits<br>allowed: int<br>Default: 0                                         |
| field       | Select field(s) using id(s) or name(s)<br>allowed: any<br>Default: variant                          |
| spw         | Spectral window selection for output<br>allowed: string<br>Default:                                 |
| scan        | Select data by scan numbers<br>allowed: string<br>Default:                                          |
| intent      | Select data by scan intents<br>allowed: string<br>Default:                                          |
| correlation | Select correlations<br>allowed: any<br>Default: variant                                             |
| observation | Select by observation ID(s)<br>allowed: any<br>Default: variant                                     |

## Example

uvcontsub3 is an experimental clone of uvcontsub with the goal of taking less time and temporary disk space.

Continuum fitting and subtraction in the uv plane:

This task estimates the continuum emission by fitting polynomials to the real and imaginary parts of the spectral windows and channels selected by fitspw. This fit represents a model of the continuum in all channels.

The fitted continuum spectrum is subtracted from all channels selected in spw, and the result (presumably only line emission) is stored in a new MS (vis + ".contsub").

It will read from the CORRECTED\_DATA column of vis if it is present, or DATA if it is not. Whichever column is read is presumed to have already been calibrated.

Keyword arguments:

vis -- Name of input visibility file  
default: none; example: vis='ngc5921.ms'

fitspw -- Selection of spectral windows and channels to use in the fit for the continuum, using general spw:chan syntax.  
See the note under combine.

default: '' (all)  
example: fitspw='0:5~30;40~55'

combine -- Let the continuum estimation span multiple spectral windows.  
default = '' (Make separate estimates for each spw.)  
combine = 'spw': Necessary when one or more of the spws are completely blanketed by lines, so the estimate must be made in different spws.

fitorder -- Polynomial order for the fits of the continuum w.r.t. frequency. fitorders > 1 are strongly discouraged because high order polynomials have more flexibility, may absorb line emission, and tend go wild at the edges of fitspw, which is not what you want.

default: 0 (constant); example: fitorder=1

field -- Field selection for continuum estimation and subtraction.  
The estimation and subtraction is done for each selected field in turn. (Run listobs to get lists of the ID and names.)  
default: ''=all fields. If the field string is a non-negative integer, it is assumed to be a field index  
otherwise, it is assumed to be a field name  
field='0~2'; field ids 0,1,2  
field='0,4,5~7'; field ids 0,4,5,6,7

```

 field='3C286,3C295'; fields named 3C286 and 3C295
 field = '3,4C*'; field id 3, all names starting with 4C

spw -- Select spectral windows for the output.
 default: ''=all spectral windows
 N.B. uvcontsub3 does not yet support exclusion by channels for
 the output. Meanwhile, use split to further reduce the size
 of the output MS if desired.
 spw='0~2,4'; spectral windows 0,1,2,4
 spw='<2'; spectral windows less than 2 (i.e. 0,1)

scan -- Scan number range
 default: ''=all

intent -- Select by scan intent (state). Case sensitive.
 default: '' = all
 Examples:
 intent = 'CALIBRATE_ATMOSPHERE_REFERENCE'
 intent = 'calibrate_atmosphere_reference'.upper() # same as above
 # Select states that include one or both of CALIBRATE_WVR.REFERENCE
 # or OBSERVE_TARGET_ON_SOURCE.
 intent = 'CALIBRATE_WVR.REFERENCE, OBSERVE_TARGET_ON_SOURCE'

correlation -- Select correlations, e.g. 'rr, ll' or ['XY', 'YX'].
 default '' (all).

observation -- Select by observation ID(s).
 default: '' = all

```

[uvmodelfit-task.html](#)

### **0.1.129 uvmodelfit**

Requires:

#### **Synopsis**

Fit a single component source model to the uv data

#### **Description**

Fit a single component source model to the uv data

#### **Arguments**

|            |                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                              |
| vis        | Name of input visibility file<br>allowed: string<br>Default:                                                                 |
| field      | Select field using field id(s) or field name(s)<br>allowed: string<br>Default:                                               |
| spw        | Select spectral window/channels<br>allowed: string<br>Default:                                                               |
| selectdata | Other data selection parameters<br>allowed: bool<br>Default: True                                                            |
| timerange  | Select data based on time range<br>allowed: string<br>Default:                                                               |
| uvrange    | Select data within uvrange (default units meters)<br>allowed: any<br>Default: variant                                        |
| antenna    | Select data based on antenna/baseline<br>allowed: string<br>Default:                                                         |
| scan       | Scan number range<br>allowed: string<br>Default:                                                                             |
| msselect   | Optional complex data selection (ignore for now)<br>allowed: string<br>Default:                                              |
| niter      | Number of fitting iterations to execute<br>allowed: int<br>Default: 5                                                        |
| comptype   | component model type: P(oint), G(aussian), or D(isk)<br>allowed: string<br>Default: P                                        |
| sourcepar  | Starting guess for component parameters (3 values for type P, 5 for G and D)<br>allowed: doubleArray<br>Default: 1.0 0.0 0.0 |
| varypar    | Control which parameters to let vary in the fit<br>allowed: boolArray<br>Default:                                            |
| outfile    | Optional output component list table<br>allowed: string<br>Default:                                                          |



## Example

Fit a single component source model to the uv data. Three models are available: P=point; G=Gaussian; D=Disk. Fitting parameters can be held fixed. The results are given in the log and placed in a components file.

Keyword arguments:

vis -- Name of input visibility file  
default: none; example: vis='ngc5921.ms'

--- Data Selection

```
field -- Select data based on field id(s) or name(s)
 default: '' (all); example: field='1'
 field='0~2' # field ids inclusive from 0 to 2
 field='3C*' # all field names starting with 3C
spw -- Select data based on spectral window
 default: '' (all); example: spw='1'
 spw='<2' #spectral windows less than 2
 spw='>1' #spectral windows greater than 1
selectdata -- Select a subset of the visibility using MSSelection
 default: False; example: selectdata=True
timerange -- Select data based on time range:
 default = '' (all); example,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:0~09:54:0' # this time range
 timerange='09:44:00' # data within one integration of time
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
uvrange -- Select data within uvrange (default units kilo-lambda)
 default: '' (all); example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo lambda
 uvrange='0~1000km'; uvrange in kilometers
antenna -- Select data based on antenna/baseline
 default: '' (all); example: antenna='5&6' baseline 5-6
 antenna='5&6;7&8' #baseline 5-6 and 7-8
 antenna='5' # all baselines with antenna 5
 antenna='5,6' # all baselines with antennas 5 and 6
scan -- Select data based on scan number - New, under developement
 default: '' (all); example: scan='>3'
msselect -- Optional data selection (field,spw,time,etc)
```

default:'' means select all; example:msselect='FIELD\_ID==0',  
msselect='FIELD\_ID IN [0,1,2]' means select fields 0,1 and 2  
msselect='FIELD\_ID <= 1' means select fields 0, 1  
msselect='FIELD\_ID==0 && ANTENNA1 IN [0] && ANTENNA2 IN [2:26]'  
means select field 0 and antennas 0 to 26, except antenna 1.  
Other msselect fields are: 'DATA\_DESC\_ID', 'SPECTRAL\_WINDOW\_ID',  
'POLARIZATION\_ID', 'SCAN\_NUMBER', 'TIME', 'UVW'  
See cookbook for more details

niter -- Number of fitting iterations to execute  
default: 5; example: niter=20  
comptype -- component model type  
default: 'P';  
Options: 'P' (point source), 'G' (elliptical gaussian),  
'D' (elliptical disk)  
sourcepar -- Starting guess for component parameters  
default: [1,0,0]; (for comptype='P')  
IF comptype = 'P' then  
sourcepar = [flux,xoff,yoff] where  
flux = Jy, xoff = offset east (arcsec), yoff = offset north (arcsec).  
IF comptype = 'G' or 'D', then  
sourcepar = [flux,xoff,yoff,majax,axrat,pos] where  
majax = FWHM along the major axis (arcsec), axrat < 1 is  
the ratio of minor to major axis, pos=angle in deg  
varypar -- Control which parameters to let vary in the fit  
default: [] (all vary);  
example: vary=[F,T,T]

examples:

fit a point:  
comptype = 'P'  
sourcepar = [0.4,0.2,-0.3];  
varypar = [T,T,T]

fit a circular Gaussian:  
comptype = 'G'  
sourcepar = [1.4,0.3,-0.2,0.3, 1, 0]  
varypar = [ T , T , T , T , F, F]

outfile -- Optional output component list table  
default: ''; example: outfile='componentlist.cl'

How to get the output values:

```

cl.open('componentlist.cl')
fit = cl.getcompoent() stores component information
fit to see the whole mess
flux = fit['flux']['value'] to store the I,Q,U,V, flux
print flux

ra = fit['shape']['direction']['m0']['value']
dec = fit['shape']['direction']['m1']['value']
print ra, dec

bmaj = fit['shape']['majoraxis']['value'] to get major axis
bmin = fit['shape']['minoraxis']['value'] to get minor axis

```

uvsub-task.html

### 0.1.130 uvsub

Requires:

#### Synopsis

Subtract/add model from/to the corrected visibility data.

#### Description

This function subtracts model visibility data (MODEL\_DATA column) from corrected visibility data (CORRECTED\_DATA column) leaving the residuals in the corrected data column. If the parameter 'reverse' is set true, the process is reversed. Note the DATA column is left untouched. If the ms has no CORRECTED\_DATA column, one is made, copying DATA column, ahead of doing the uvsub process

#### Arguments

| Inputs  |                                                                                     |
|---------|-------------------------------------------------------------------------------------|
| vis     | Name of input visibility file (MS)<br>allowed: string<br>Default:                   |
| reverse | reverse the operation (add rather than subtract)<br>allowed: bool<br>Default: False |

#### Returns

void

#### Example

Help for uvsub task

This function subtracts model visibility data from corrected visibility

data leaving the residuals in the corrected data column. If the parameter 'reverse' is set true, the process is reversed. Please note the model visibility used is the one that has been saved in the MODEL\_DATA of the CORRECTED\_DATA column is the one that is modified. If no CORRECTED\_DATA column exists in the a copy of the DATA column is saved in it before the uvsub operation selected is performed.

Keyword arguments:

vis -- Name of input visibility file (MS)

default: none; example: vis='ngc5921.ms'

reverse -- Reverse the operation (add rather than subtract)

default: False; example: reverse=true

uvsub(vis='ngc5921.ms', reverse=False)

viewer-task.html

### **0.1.131 viewer**

Requires:

#### **Synopsis**

View an image or visibility data set

#### **Description**

The viewer will display images in raster, contour, vector or marker form. Images can be blinked, and movies are available for spectral-line image cubes. For measurement sets, many display and editing options are available.

The viewer can be run outside of casapy by typing `jasaviewer`.

Executing viewer `viewer` will bring up a display panel window, which can be resized. If no data file was specified, a Load Data window will also appear.

Click on the desired data file and choose the display type; the rendered data should appear on the display panel.

A Data Display Options window will also appear. It has drop-down subsections for related options, most of which are self-explanatory.

The state of the viewer – loaded data and related display options – can be saved in a 'restore' file for later use. You can provide the restore filename on the command line or select it from the Load Data window.

See the cookbook for more details on using the viewer.

#### **Arguments**

|               |                                                                                                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Inputs</b> |                                                                                                                                                                                                                |
| infile        | (Optional) Name of file to visualize.<br>allowed: string<br>Default:                                                                                                                                           |
| displaytype   | (Optional) Type of visual rendering (raster, contour, vector or marker). <code>lel</code> if an <code>lel</code> expression is given for <code>infile</code> (advanced).<br>allowed: string<br>Default: raster |
| channel       | (Optional) access a specific channel in the image cube<br>allowed: int<br>Default: 0                                                                                                                           |
| zoom          | (Optional) zoom in/out by increments<br>allowed: int<br>Default: 1                                                                                                                                             |
| outfile       | (Optional) name of the output file to generate<br>allowed: string<br>Default:                                                                                                                                  |
| outscale      | (Optional) amount to scale output bitmap formats (non-PS, non-PDF)<br>allowed: double<br>Default: 1.0                                                                                                          |
| outdpi        | (Optional) output DPI for PS/PDF<br>allowed: int<br>Default: 300                                                                                                                                               |
| outformat     | (Optional) format of the output e.g. jpg or pdf (this is overridden by the output files extension)<br>allowed: string<br>Default: jpg                                                                          |
| outlandscape  | (Optional) should the output mode be landscape (PS or PDF)<br>allowed: bool<br>Default: False                                                                                                                  |
| gui           | (Optional) Display the panel in a GUI.<br>allowed: bool<br>Default: True                                                                                                                                       |

## Returns

void

## Example

examples of usage:

```
viewer
viewer "myimage.im"
viewer "mymeasurementset.ms"
viewer "myrestorefile.rstr"

viewer "myimage.im", "contour"

viewer "'myimage1.im' - 2 * 'myimage2.im'", "l1"
```

Keyword arguments:

```
infile -- Name of file to visualize
default: ''
example: infile='ngc5921.image'
If no infile is specified the Load Data window
will appear for selecting data.
displaytype -- (optional): method of rendering data
visually (raster, contour, vector or marker).
You can also set this parameter to 'l1' and
provide an l1 expression for infile (advanced).
default: 'raster'
example: displaytype='contour'
```

Note: the filetype parameter is optional; typing of data files is now inferred:

```
example: viewer infile='my.im'
obsolete: viewer infile='my.im', filetype='raster'
the filetype is still used to load contours, etc.
```



wvrgcal-task.html

### 0.1.132 wvrgcal

Requires:

#### Synopsis

Generate a gain table based on Water Vapour Radiometer data

#### Description

Information about the observation and the performance of WVRGCAL is written to the CASA logger and also returned in a dictionary; see the CASA cookbook for a more detailed description of these parameters. The dictionary element 'success' is True if no errors occurred.

Of particular note is the discrepancy parameter (Disc): high values ( $\geq$  a few hundred microns) may indicate some levels of cloud contamination and the effect of applying the WVRGCAL correction should be checked; values  $\geq$  1000  $\mu$ m in all antennas have currently been found to indicate that WVRGCAL correction should not be used.

vis – Name of input visibility file default: none; example: vis='ngc5921.ms'

caltable – Name of output gain calibration table default: none; example: caltable='ngc5921.wvr'

toffset – Time offset (sec) between interferometric and WVR data default: 0 (ALMA default for cycle 1, for cycle 0, i.e. up to Jan 2013 it was -1)

segsource – Do a new coefficient calculation for each source default: True

tie – Prioritise tying the phase of these sources as well as possible (requires segsource=True) default: [] example: ['3C273,NGC253', 'IC433,3C279']

sourceflag – Flag the WVR data for these source(s) as bad and do not produce corrections for it (requires segsource=True) default: [] (none) example: ['3C273']

nsol – Number of solutions for phase correction coefficients during this observation. By default only one set of coefficients is generated for the entire observation. If more sets are requested, then they will be evenly distributed in time throughout the observation. Values  $\geq$  1 require segsource=False. default: 1

disperse – Apply correction for dispersion default: False

wvrflag – Regard the WVR data for these antenna(s) as bad and use interpolated values instead default: [] (none) example: ['DV03','DA05','PM02']

statfield – Compute the statistics (Phase RMS, Disc) on this field only default: '' (all)

statsource – Compute the statistics (Phase RMS, Disc) on this source only default: '' (all)

smooth – Smooth the calibration solution on the given timescale default: " (no smoothing), example: '3s' smooth on a timescale of 3 seconds  
 scale – Scale the entire phase correction by this factor default: 1. (no scaling)  
 spw – List of the spectral window IDs for which solutions should be saved into the caltable default: [] (all spectral windows), example [17,19,21,23]  
 wvrspw – List of the spectral window IDs from which the WVR data should be taken default: [] (all WVR spectral windows), example [0]  
 reversespw – Reverse the sign of the correction for the listed SPWs (only needed for early ALMA data before Cycle 0) default: " (none), example: reversespw='0~2,4'; spectral windows 0,1,2,4  
 cont – Estimate the continuum (e.g., due to clouds) default: False  
 maxdistm – maximum distance (m) an antenna may have to be considered for being part of the antenna set (minimum 3 antennas) for the interpolation of a solution for a flagged antenna default: 500.  
 minnumants – minimum number of near antennas required for interpolation default: 2  
 mingoodfrac – If the fraction of unflagged data for an antenna is below this value (0. to 1.), the antenna is flagged. default: 0.8  
 usefieldtab – derive the antenna AZ/EL values from the FIELD rather than the POINTING table default: False  
 refant – use the WVR data from this antenna for calculating the dT/dL parameters (can give ranked list) default: " (use the first good or interpolatable antenna), examples: 'DA45' - use DA45 ['DA45','DV51'] - use DA45 and if that is not good, use DV51 instead  
 offsetstable – (experimental) subtract the temperature offsets in this table from the WVR measurements before using them to calculate the phase corrections default: " (do not apply any offsets) examples: 'uid\_\_\_A002\_Xabd867\_X2277.cloud\_offsets' use the given table

## Arguments

|            |                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs     |                                                                                                                                                              |
| vis        | Name of input visibility file<br>allowed: string<br>Default:                                                                                                 |
| caltable   | Name of output gain calibration table<br>allowed: string<br>Default:                                                                                         |
| toffset    | Time offset (sec) between interferometric and WVR data<br>allowed: double<br>Default: 0                                                                      |
| segsource  | Do a new coefficient calculation for each source<br>allowed: bool<br>Default: True                                                                           |
| sourceflag | Regard the WVR data for these source(s) as bad and do not produce corrections for it (requires segsource=True)<br>allowed: stringArray<br>Default:           |
| tie        | Prioritise tying the phase of these sources as well as possible (requires segsource=True)<br>allowed: stringArray<br>Default:                                |
| nsol       | Number of solutions for phase correction coefficients (nsol>1 requires segsource=False)<br>allowed: int<br>Default: 1                                        |
| disperse   | Apply correction for dispersion<br>allowed: bool<br>Default: False                                                                                           |
| wvrflag    | Regard the WVR data for these antenna(s) as bad and replace its data with interpolated values from neighbouring antennas<br>allowed: stringArray<br>Default: |
| statfield  | Compute the statistics (Phase RMS, Disc) on this field only<br>allowed: string<br>Default:                                                                   |
| statsource | Compute the statistics (Phase RMS, Disc) on this source only<br>allowed: string<br>Default:                                                                  |
| smooth     | Smooth calibration solution on the given timescale<br>allowed: string<br>Default:                                                                            |
| scale      | Scale the entire phase correction by this factor<br>allowed: double<br>Default: 1.                                                                           |
| spw        | List of the spectral window IDs for which solutions should be saved into the caltable<br>allowed: intArray<br>Default:                                       |
| wvrspw     | List of the spectral window IDs from which the WVR data should be taken<br>allowed: intArray<br>Default:                                                     |
| reversespw | Reverse the sign of the correction for the listed SPWs                                                                                                       |

**Returns**

variant

**Example**

```
wvrgcal(vis='uid__A002_X1d54a1_X5.ms', caltable='cal-wvr-uid__A002_X1d54a1_X5.W',
 toffset=-1, segsource=True, tie=['Titan,1037-295,NGC3256'], statsource='1037-295
```

virtualconcat-task.html

### 0.1.133 virtualconcat

Requires:

#### Synopsis

Concatenate several visibility data sets into a multi-MS

#### Arguments

|                |                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Inputs         |                                                                                                                                   |
| vis            | List of names of input visibility files to be concatenated<br>allowed:       stringArray<br>Default:                              |
| concatvis      | Name of the output visibility file (a multi-MS)<br>allowed:       string<br>Default:                                              |
| frequitol      | Frequency shift tolerance for considering data as the same spwid<br>allowed:       any<br>Default:       variant                  |
| dirtol         | Direction shift tolerance for considering data as the same field<br>allowed:       any<br>Default:       variant                  |
| respectname    | If true, fields with a different name are not merged even if their direction agrees<br>allowed:       bool<br>Default:       True |
| visweightscale | List of the weight scaling factors to be applied to the individual MSs<br>allowed:       doubleArray<br>Default:                  |
| keepcopy       | If true, a copy of the input MSs is kept in their original place.<br>allowed:       bool<br>Default:       False                  |
| copypointing   | If true, keep the POINTING table information in the output MMS. If false, don't.<br>allowed:       bool<br>Default:       True    |

## Example

The list of data sets given in the vis argument are moved into an output multi-MS data set concatvis and virtually concatenated.

NOTE: This task will modify the input datasets by moving them and reindexing them. If you want to keep a copy of your original data, please set the parameter keepcopy to True.

There is no limit to the number of input data sets.

If none of the input data sets have any scratch columns (model and corrected columns), none are created in the concatvis. Otherwise these columns are created on output and initialized to their default value (1 in model column, data in corrected column) for those data with no input columns.

Spectral windows for each data set with the same channelization, and within a specified frequency tolerance of another data set will be combined into one spectral window.

A field position in one data set that is within a specified direction tolerance of another field position in any other data set will be combined into one field. The field names need not be the same---only their position is used.

Each appended dataset is assigned a new observation id if the corresponding rows in the observation table are not the same.

Keyword arguments:

vis -- Name of input visibility files to be combined

default: none; example: vis = ['src2.ms', 'ngc5921.ms', 'ngc315.ms']

concatvis -- Name of visibility file that will contain the concatenated data

note: if this file exists on disk then the input files are

added to this file. Otherwise the new file contains

the concatenated data. Be careful here when concatenating to

an existing file.

default: none; example: concatvis='src2.ms'

example: concatvis='outvis.ms'

other examples:

virtualconcat(vis=['src2.ms', 'ngc5921.ms'], concatvis='out.mms')

will concatenate 'ngc5921.ms' and 'src2.ms' into a file named

'out.mms'; the original 'ngc5921.ms' and 'src2.ms' are gone.

'out.mms' is a multims. As most of the data is only moved, not

copied, this is faster and subsequent tasks can run in parallel on the subMSs of out.mms.

```
virtualconcat(vis=['src2.ms','ngc5921.ms'], concatvis='out.mms', keepcopy=True)
```

will concatenate 'ngc5921.ms' and 'src2.ms' into a file named 'out.mms'; the original 'ngc5921.ms' and 'src2.ms' are as before but you consume more disk space and time for the copy.

Note: run flagmanager to save flags in the concatvis

freqtol -- Frequency shift tolerance for considering data to be in the same spwid. The number of channels must also be the same.  
 default: '' do not combine unless frequencies are equal  
 example: freqtol='10MHz' will not combine spwid unless they are within 10 MHz.

Note: This option is useful to combine spectral windows with very slight frequency differences caused by Doppler tracking, for example.

dirtol -- Direction shift tolerance for considering data as the same field  
 default: '' means always combine.  
 example: dirtol='1.arcsec' will not combine data for a field unless their phase center differ by less than 1 arcsec. If the field names are different in the input data sets, the name in the output data set will be the first relevant data set in the list.

respectname -- If true, fields with a different name are not merged even if their direction agrees (within dirtol).  
 default: True

visweightscale -- The weights of the individual MSs will be scaled in the concatenated output MS by the factors in this list. Useful for handling heterogeneous arrays. Use plotms to inspect the "Wt" column as a reference for determining the scaling factors. See the cookbook for more details.  
 example: [1.,3.,3.] - scale the weights of the second and third MS by a factor 3.  
 default: [] (empty list) - no scaling

keepcopy -- If true, a copy of the input MSs is kept in their original place.  
 default: false

copypointing -- If true, the POINTING table information will be present in the output. If false, the result is an empty POINTING table.  
 default: true

vishead-task.html

### 0.1.134 vishead

Requires:

#### Synopsis

List, summary, get, and put metadata in a measurement set

#### Description

List, summary, get, and put "header" information in a measurement set.

#### Arguments

|           |                                                                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inputs    |                                                                                                                                                                      |
| vis       | Name of input visibility file<br>allowed: string<br>Default:                                                                                                         |
| mode      | options: list, summary, get, put<br>allowed: string<br>Default: summary                                                                                              |
| listitems | items to list ([] for all)<br>allowed: stringArray<br>Default: telescope observer project field<br>freq_group_name spw_name sched-<br>ule schedule_type release_date |
| hdkey     | keyword to get/put<br>allowed: string<br>Default:                                                                                                                    |
| hdindex   | keyword index to get/put, counting from zero. "==">all<br>allowed: string<br>Default:                                                                                |
| hdvalue   | value of hdkey<br>allowed: any<br>Default: variant                                                                                                                   |

#### Example



This task allows the user to manipulate some meta-data parameters in a measurement set. The mode='list' shows those keywords that are presently implemented, with their values. The contents associated with the keywords can be obtained (get) and changed (put).

The modes that are available are:

```
list --- List all keywords that are recognized, and list the
 value(s) for each. Only these keywords can be obtained
 (get) or changed (put)
summary --- Equivalent to running taskname='listobs'; verbose=F
get --- Get the specified keyword value(s) from the ms
put --- Put the specified keyword value(s) into the ms
```

Parameters currently implemented are (June 1, 2009):

|                 |                                       |
|-----------------|---------------------------------------|
| cal_grp         |                                       |
| field           | Field names                           |
| fld_code        | Field Observing codes                 |
| freq_group_name |                                       |
| log             |                                       |
| observer        | Observer name                         |
| project         | Project name                          |
| ptcs            | Phase tracking centers for each field |
| release_date    |                                       |
| schedule        |                                       |
| schedule_type   |                                       |
| spw_name        | Spectral parameters?                  |
| source_name     | Source Names (=Field Names?)          |
| telescope       | Telescope Name                        |

Keyword arguments:

```
vis --- Name of input visibility file
 default: none, example: vis='my.ms'

mode --- Mode of operation for vishead
 default = 'list'; example: mode='get'

hdkey--- keyword to get or put from the ms (used in get/put mode only)
 ex: hdkey='telescope'

hdindex--- index (counting from 0) if keyword is an array (used in get/put
 mode only)
```

```

ex: hdindex='2'; hdindex=''->put/get full array;

hdvalue --- value to be put in the MS (used in put mode only)
ex: hdvalue=array(['MyTelescope'])

```

#### Examples:

To transfer the parameters to useful python items requires some care.

```

taskname = 'vishead'
default()
vis = '3C84C.ms'
mode = 'get'

```

```

to get a field name (string),
 hdkey = 'field'; hdindex = '2'; hdvalue=vishead();
 print hdvalue[0] = the name for field='2'

```

```

to get an phase center (number)
 hdkey = 'ptcs'; hdindex = '1'; hdvalue = vishead();
 hdvalue[0][0] gives the ra, hdvalue[0][1] gives the dec in field '1'

```

```

taskname = 'vishead'
default()
vis = '3C84C.ms'
mode = 'put'

```

To change a string,

```

 hdkey = 'field'; hdindex = '2'; hdvalue = 'junk'; vishead()
 field='2' is renamed 'junk'

```

To change a number, (egs. ra of field=1 to 0.5 radian)  
is too complicated to figure out!

visstat-task.html

### **0.1.135 visstat**

Requires:

#### **Synopsis**

Displays statistical information from a Measurement Set, or from a Multi-MS

#### **Arguments**

|             |                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------|
| Outputs     |                                                                                                   |
| xstat       | Statistical information for the selected measurement set<br>allowed: any<br>Default: variant      |
| Inputs      |                                                                                                   |
| vis         | Name of Measurement Set or Multi-MS<br>allowed: string<br>Default:                                |
| axis        | Which values to use<br>allowed: string<br>Default: amplitude                                      |
| datacolumn  | Which data column to use (data, corrected, model)<br>allowed: string<br>Default: data             |
| useflags    | Take flagging into account?<br>allowed: bool<br>Default: True                                     |
| spw         | spectral-window/frequency/channel<br>allowed: string<br>Default:                                  |
| field       | Field names or field index numbers: "==">all, field='0~2,3C286'<br>allowed: string<br>Default:    |
| selectdata  | More data selection parameters (antenna, timerange etc)<br>allowed: bool<br>Default: True         |
| antenna     | antenna/baselines: "==">all, antenna = '3,VA04'<br>allowed: string<br>Default:                    |
| uvrange     | uv range: "==">all; uvrange = '0~100klambda', default units=meters<br>allowed: string<br>Default: |
| timerange   | time range: "==">all, timerange='09:14:0~09:54:0'<br>allowed: string<br>Default:                  |
| correlation | Select data based on correlation<br>allowed: string<br>Default:                                   |
| scan        | scan numbers: "==">all<br>allowed: string<br>Default:                                             |
| array       | (sub)array numbers: "==">all<br>allowed: string<br>Default:                                       |
| observation | observation ID number(s): " = all<br>allowed: <del>any</del><br>Default: variant                  |

## Returns

void

## Example

This task returns statistical information about data in a Measurement Set or Multi-MS.

The following values are computed: mean value, sum of values, sum of squared values, median, median absolute deviation, quartile, minimum, maximum, variance, standard deviation, and root mean square.

The following axes are supported: uvw, flag, weight, sigma, antenna1, antenna2, feed1, feed2, field\_id, array\_id, data\_desc\_id, flag\_row, interval, scan, scan\_number, time, weight\_spectrum, amp, amplitude, phase, real, imag, imaginary, and uvrangle.

Optionally, the statistical information can be computed based only on a given subset of the measurement set.

Note: If the MS consists of inhomogeneous data, for example several spectral windows each having a different number of channels, it may be necessary to use selection parameters to select a homogeneous subset of the MS, e.g. `spw='2'`.

Keyword arguments:

```
vis --- Name of input Measurement Set or Multi-MS
 default: '', example: vis='my.ms'
```

```
axis -- Which data to analyze.
```

```
 default: 'amplitude'
 axis='phase'
 axis='imag'
 axis='scan_number'
 axis='flag'
```

The phase of a complex number is in radians in the range  $[-\pi; \pi[$ .

```

datacolumn -- Which data column to use for complex data.
 default: 'data'
 datacolumn='data'
 datacolumn='corrected'
 datacolumn='model'

useflags -- Take MS flags into account?
 default: True
 useflag=False
 useflag=True
If useflags=False, flagged values are included in the statistics.
If useflags=True, any flagged values are not used in the statistics.

spw -- Select data based on spectral window and channels
 default: '' (all); example: spw='1'
 spw='<2' #spectral windows less than 2
 spw='>1' #spectral windows greater than 1
 spw='0:0~10' # first 10 channels from spw 0
 spw='0:0~5;56~60' # multiple separated channel chunks.

field -- Select data based on field id(s) or name(s)
 default: '' (all); example: field='1'
 field='0~2' # field ids inclusive from 0 to 2
 field='3C*' # all field names starting with 3C

selectdata -- Other data selection parameters
 default: True

antenna -- Select data based on baseline
 default: '' (all); example: antenna='5&6' baseline 5-6
 antenna='5&6;7&8' #baseline 5-6 and 7-8
 antenna='5' # all baselines with antenna 5
 antenna='5,6' # all baselines with antennas 5 and 6

correlation -- Correlation types
 default: '' (all);
 example: correlation='RR LL'

uvrange -- Select data within uvrange (default units meters)
 default: '' (all); example:
 uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
 uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
 uvrange='0~1000km'; uvrange in kilometers

timerange -- Select data based on time range:
 default = '' (all); example,
 timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
 Note: YYYY/MM/DD can be dropped as needed:
 timerange='09:14:0~09:54:0' # this time range
 timerange='09:44:00' # data within one integration of time

```

```
 timerange='>10:24:00' # data after this time
 timerange='09:44:00+00:13:00' #data 13 minutes after time
scan -- Select data based on scan number
 default: '' (all); example: scan='>3'
array -- Selection based on the antenna array
observation -- Selection by observation ID(s).
 default: '' (all); example: observation='1~3'
```

visstat2-task.html

### **0.1.136 visstat2**

Requires:

#### **Synopsis**

Displays statistical information from a Measurement Set, or from a Multi-MS

#### **Arguments**



|             |                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------|
| Outputs     |                                                                                                   |
| xstat       | Statistical information for the selected measurement set<br>allowed: any<br>Default: variant      |
| Inputs      |                                                                                                   |
| vis         | Name of Measurement Set or Multi-MS<br>allowed: string<br>Default:                                |
| axis        | Which values to use<br>allowed: string<br>Default: amplitude                                      |
| datacolumn  | Which data column to use (data, corrected, model, float_data)<br>allowed: string<br>Default: data |
| useflags    | Take flagging into account?<br>allowed: bool<br>Default: True                                     |
| spw         | spectral-window/frequency/channel<br>allowed: string<br>Default:                                  |
| field       | Field names or field index numbers: "==">all, field='0~2,3C286'<br>allowed: string<br>Default:    |
| selectdata  | More data selection parameters (antenna, timerange etc)<br>allowed: bool<br>Default: True         |
| antenna     | antenna/baselines: "==">all, antenna = '3,VA04'<br>allowed: string<br>Default:                    |
| uvrange     | uv range: "==">all; uvrange = '0~100klambda', default units=meters<br>allowed: string<br>Default: |
| timerange   | time range: "==">all, timerange='09:14:0~09:54:0'<br>allowed: string<br>Default:                  |
| correlation | Select data based on correlation<br>allowed: string<br>Default:                                   |
| scan        | scan numbers: "==">all<br>allowed: string<br>Default:                                             |
| array       | (sub)array numbers: "==">all<br>allowed: string<br>Default:                                       |
| observation | observation ID number(s): " = all<br>allowed: any<br>Default: variant                             |
| timeaverage | Average data in time.<br>allowed: bool<br>Default: False                                          |
| timebin     | Bin width for time averaging.<br>allowed: string                                                  |

## Returns

void

## Example

This task returns statistical information about data in a Measurement Set or Multi-MS.

The following values are computed: mean value, sum of values, sum of squared values, median, median absolute deviation, quartile, minimum, maximum, variance, standard deviation, and root mean square.

The following axes are supported: uvw, flag, weight, sigma, antenna1, antenna2, feed1, feed2, field\_id, array\_id, data\_desc\_id, flag\_row, interval, scan, scan\_number, time, weight\_spectrum, amp, amplitude, phase, real, imag, imaginary, and uvrangle.

Optionally, the statistical information can be computed based only on a given subset of the measurement set.

Note: If the MS consists of inhomogeneous data, for example several spectral windows each having a different number of channels, it may be necessary to use selection parameters to select a homogeneous subset of the MS, e.g. spw='2'.

Keyword arguments:

```
vis --- Name of input Measurement Set or Multi-MS
default: '', example: vis='my.ms'
```

```
axis -- Which data to analyze.
```

```
default: 'amplitude'
axis='phase'
axis='imag'
axis='scan_number'
axis='flag'
```

The phase of a complex number is in radians in the range  $[-\pi; \pi]$ .

```

datacolumn -- Which data column to use for complex data.
default: 'data'
datacolumn='data'
datacolumn='corrected'
datacolumn='model'
datacolumn='float_data'

useflags -- Take MS flags into account?
default: True
useflag=False
useflag=True
If useflags=False, flagged values are included in the statistics.
If useflags=True, any flagged values are not used in the statistics.

spw -- Select data based on spectral window and channels
default: '' (all); example: spw='1'
spw='<2' #spectral windows less than 2
spw='>1' #spectral windows greater than 1
spw='0:0~10' # first 10 channels from spw 0
spw='0:0~5;56~60' # multiple separated channel chunks.

field -- Select data based on field id(s) or name(s)
default: '' (all); example: field='1'
field='0~2' # field ids inclusive from 0 to 2
field='3C*' # all field names starting with 3C

selectdata -- Other data selection parameters
default: True
antenna -- Select data based on baseline
default: '' (all); example: antenna='5&6' baseline 5-6
antenna='5&6;7&8' #baseline 5-6 and 7-8
antenna='5' # all baselines with antenna 5
antenna='5,6' # all baselines with antennas 5 and 6
correlation -- Correlation types
default: '' (all);
example: correlation='RR LL'
uvrange -- Select data within uvrange (default units meters)
default: '' (all); example:
uvrange='0~1000klambda'; uvrange from 0-1000 kilo-lambda
uvrange='>4klambda'; uvranges greater than 4 kilo-lambda
uvrange='0~1000km'; uvrange in kilometers
timerange -- Select data based on time range:
default = '' (all); example,
timerange = 'YYYY/MM/DD/hh:mm:ss~YYYY/MM/DD/hh:mm:ss'
Note: YYYY/MM/DD can be dropped as needed:
timerange='09:14:0~09:54:0' # this time range

```

```

timerange='09:44:00' # data within one integration of time
timerange='>10:24:00' # data after this time
timerange='09:44:00+00:13:00' #data 13 minutes after time
scan -- Select data based on scan number
default: '' (all); example: scan='>3'
array -- Selection based on the antenna array
observation -- Selection by observation ID(s).
default: '' (all); example: observation='1~3'

```

--- Time averaging parameters ---

```

timeaverage -- Average data in time. Flagged data will be included in the average
calculation, unless the parameter keepflags is set to False. In this
case only partially flagged rows will be used in the average.
WEIGHT_SPECTRUM/SIGMA_SPECTRUM will be used (if present) in
addition to the flags to compute a weighted average. The calculation is done
in the same way as for the channel average case but across the time axis.
Otherwise (if WEIGHT_SPECTRUM/SIGMA_SPECTRUM are not present)
mstransform will use WEIGH/SIGMA instead, as in split.
default: False

```

```

timebin -- Bin width for time averaging.
default: '0s'

```

```

timespan -- Let the timebin span across scan, state or both.
State is equivalent to sub-scans. One scan may have several
state ids. For ALMA MSs, the sub-scans are limited to about
30s duration each. In these cases, the task will automatically
add state to the timespan parameter. To see the number of states
in an MS, use the msmd tool. See help msmd.

```

```

default: '' (separate time bins by both of the above)
options: 'scan', 'state', 'state,scan'

```

examples:

```

timespan = 'scan'; can be useful when the scan number
goes up with each integration as in many WSRT MSs.
timespan = ['scan', 'state']: disregard scan and state
numbers when time averaging.
timespan = 'state,scan'; same as above.

```

```

maxuvwdistance -- Provide a maximum separation of start-to-end baselines
that can be included in an average. (int)
default: 0.0 (given in meters)

```



[widebandpbcor-task.html](#)

### **0.1.137 widebandpbcor**

Requires:

#### **Synopsis**

Wideband PB-correction on the output of the MS-MFS algorithm

#### **Description**

WideBand Primary-beam correction. It computes a set of PBs at the specified frequencies, calculates Taylor-coefficient images that represent the PB spectrum, performs a polynomial division to PB-correct the output Taylor-coefficient images from `clean(nterms,1)`, and recompute spectral index (and curvature) using the PB-corrected Taylor-coefficient images

#### **Arguments**

|               |                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------|
| <b>Inputs</b> |                                                                                                    |
| vis           | Name of measurement set.<br>allowed: string<br>Default:                                            |
| imagename     | Name-prefix of multi-term images to operate on.<br>allowed: string<br>Default:                     |
| nterms        | Number of taylor terms to use<br>allowed: int<br>Default: 2                                        |
| threshold     | Intensity above which to re-calculate spectral index<br>allowed: string<br>Default:                |
| action        | PB-correction (pbcor) or only calc spectral-index (cal-alpha)<br>allowed: string<br>Default: pbcor |
| reffreq       | Reference frequency (if specified in clean)<br>allowed: string<br>Default:                         |
| pbmin         | PB threshold below which to not correct<br>allowed: double<br>Default: 0.2                         |
| field         | Fields to include in the PB calculation<br>allowed: string<br>Default:                             |
| spwlist       | List of N spw ids<br>allowed: intArray<br>Default:                                                 |
| chanlist      | List of N channel ids<br>allowed: intArray<br>Default:                                             |
| weightlist    | List of N weights (relative)<br>allowed: doubleArray<br>Default:                                   |

## Returns

void

## Example

## Wide-band Primary-beam correction

- (1) Compute a set of Primary Beams at the specified frequencies
- (2) Calculate Taylor-coefficient images that represent the PB spectrum
- (3) Perform a polynomial division to PB-correct the output Taylor-coefficient images from the MS-MFS algorithm ( `clean(nterms>1)` )
- (4) Recompute spectral index (and curvature) using the corrected Taylor-coefficient images

[ Optionally, skip PB-correction, and only recalculate spectral index with a different threshold ]

This is a temporary task, meant for use until projection-based gridding algorithms are available via the 'clean' task.

An output directory named `imagename.pbcor.workdirectory` is created, and filled with an image-cube of the evaluated primary beams at all specified frequencies, Taylor-coefficients, and a 'spectral index' due to the primary beam. Note that for the actual pb-correction, only the Taylor-coefficient images are used.

### Task parameters :

`vis` -- Name of input visibility file

example : `vis = 'ngc5921.ms'`

Only one MS can be specified here, and it must contain at-least one timestep of data at all frequencies required to calculate the PB spectrum. ( In case of multiple MSs with different spectral windows, for now, please split/concat a small fraction of the data to form such an MS )

`imagename` -- Pre-name of input and output images. Same as in the clean task.

example : `imagename = 'run1'`

Restored-images ( `run1.image.tt0`,etc) and residual images ( `run1.residual.tt0` ) must be available on disk.

`nterms` -- Number of Taylor terms to be used to model the frequency-dependence of the primary beam.

example : `nterms = 2`

`nterms` must be less than or equal to the number of frequencies specified via `spwlist`, `chanlist` and `weightlist`.  
`nterms=1` will do a standard division by the average PB computed over all specified frequencies.

`threshold` -- Flux level in the restored intensity map, below which to not recalculate spectral index.

example : `threshold = '0.1Jy'`



action -- Choice of PB-correction with spectral-index recalculation  
           or only spectral-index recalculation (using the specified threshold)  
 example : action='pbcor' or action='calcalpha'

With action='pbcor', the following output images are created/overwritten.

- imagename.pbcor.workdirectory : This directory contains an image cube with  
 PBs at the list of specified frequencies, and Taylor-coefficient images that  
 describe the PB spectrum.
  - imagename.pb.cube : Concatenated cube of PBs
  - imagename.pb.tt0, tt1, ... : Taylor coefficients describing the PB sp
  - imagename.pb.alpha : Spectral index of the PB (for information only)
- imagename.image.pbcor.tt0,tt1,... : Corrected Taylor coefficients
- imagename.pbcor.image.alpha : Corrected Spectral Index
- imagename.pbcor.image.alpha.error : New error map.

With action='calcalpha', the following output images are created/overwritten

- imagename.image.alpha : Corrected Spectral Index
- imagename.image.alpha.error : New error map.

reffreq -- Reference frequency about which the Taylor-expansion is defined.

example : reffreq = '1.5GHz'

If left unspecified, it is picked from the input restored image.

Note : If reffreq was specified during task clean to produce the images  
 it must be specified here.

pbmin -- PB gain level below which to not compute Taylor-coefficients or

apply PB-corrections.

example : pbmin = 0.1

field -- Field selection for the Primary Beam calculation.

example : field = '3C291'

This field selection must be identical to that used in 'clean'

spwlist -- List of SPW ids for which to make separate Primary Beams

chanlist -- List of channel ids, within the above SPW ids, at which to make PBs.

example : spwlist=[0,1,2], chanlist=[32,32,32]

Make PBs at frequencies corresponding to channel 32 of  
 spws 0,1 and 2.

example : spwlist=[0,0,0], chanlist=[0,10,20]

Make PBs at frequencies corresponding to channels 0,10,20  
 of spw 0

Primary beams are computed at these specified frequencies and  
 for pointings selected by 'field'. Taylor-coefficients that represent

the PB spectrum are computed from these images.

`weightlist` -- List of relative weights to apply to the PBs selected via the  
                  `spwlist,chanlist` parameters. Weights should approximately represent the  
                  sum-of-weights applicable during imaging each of these frequencies.

example : `weightlist=[0.5,1.0,1.0]`

                  The first frequency had less usable data due to flagged RFI  
                  but the other two had relatively equal weight.

                  These weights are applied to the PB spectrum while computing  
                  PB Taylor-coefficients. Setting weights to anything other than 1.0  
                  makes a difference only with very lop-sided weights.

widefield-task.html

### **0.1.138 widefield**

Requires:

#### **Synopsis**

Wide-field imaging and deconvolution with selected algorithm

#### **Description**

This is the main wide-field imaging/deconvolution task. It uses the wprojection method for a large field of view, can make many facets, and can include outlier fields. Several deconvolution algorithms are supported. Interactive cleaning is also supported

#### **Arguments**

|             |                                                                                           |
|-------------|-------------------------------------------------------------------------------------------|
| Inputs      |                                                                                           |
| vis         | name of input visibility file<br>allowed: stringArray<br>Default:                         |
| imagename   | Pre-name of output images<br>allowed: any<br>Default: variant                             |
| outlierfile | Text file with image names, sizes, centers<br>allowed: string<br>Default:                 |
| field       | Field Name<br>allowed: string<br>Default:                                                 |
| spw         | Spectral windows:channels: " is all<br>allowed: any<br>Default: variant                   |
| selectdata  | Other data selection parameters<br>allowed: bool<br>Default: False                        |
| timerange   | Range of time to select from data<br>allowed: string<br>Default:                          |
| uvrange     | Select data within uvrange<br>allowed: string<br>Default:                                 |
| antenna     | Select data based on antenna/baseline<br>allowed: string<br>Default:                      |
| scan        | scan number range<br>allowed: string<br>Default:                                          |
| mode        | Type of selection (mfs, channel, velocity, frequency)<br>allowed: string<br>Default: mfs  |
| niter       | Maximum number of iterations<br>allowed: int<br>Default: 500                              |
| gain        | Loop gain for cleaning<br>allowed: double<br>Default: 0.1                                 |
| threshold   | Flux level to stop cleaning. Must include units<br>allowed: any<br>Default: variant 0.0Jy |
| psfmode     | Algorithm to use (clark, hogbom)<br>allowed: string<br>Default: 70%clark                  |
| ftmachine   | Gridding method for the image (wproject, ft)<br>allowed: string<br>Default:               |
| facets      | Number of facets along each axis in main image only<br>allowed: int<br>Default: 3         |
| wprojplanes | Number of planes to use in wprojection convolutioun function                              |

## Returns

void

## Example

Wide-field imaging and deconvolution with selected algorithm:

This is the main wide-field imaging/deconvolution task. It uses the wprojection method for a large field of view, can make many facets, and can include outlier fields. Several deconvolution algorithms are supported. Interactive cleaning is also supported.

For making large images (>2000 on a size), see hints at the end of the descriptions. For making images larger than about 5000x5000, the available memory must be larger than 2 Gbytes. For such images therefore a computer with a 64-bit operating system may be needed.

Keyword arguments:

```
vis -- Name of all input visibility files
 default: none; example: vis='ngc5921.ms'
 example: vis=['data01.ms', 'data02.ms']
imasename -- Pre-name of output images:
 default: none; example: imasename='n5921'
 if outlier fields are included, then
 imasename=['n5921', 'outlier1', outlier2']
 and the first imasename is the wide-field image
 output images names are: n5921.clean, n5921.residual,
 n5921.model, n5921.interactive.mask
mode -- Type of selection
 default: 'mfs'; example: mode='channel';
 Options: 'mfs', channel, velocity, frequency'
alg -- Algorithm to use
 default: 'clark';
 Options: 'clark', 'hogbom', 'multiscale', 'entropy'
 Strongly advise 'clark'. multiscale and entropy
 well-tested.
imsize -- Image pixel size (x,y)
 default = [256,256]; example: imsize=[500,500], or imsize=500
```

```

 example for multiple fields: imsize=[(1000, 1000), (100, 100)]
cell -- Cell size (x,y)
 default=['1arcsec','1arcsec']
 example: cell=['0.5arcsec','0.5arcsec'], or cell='0.5arcsec'
phasecenter -- direction position or the field for the image center
 A list of the above is needed for multiple-fields
 default: '' -->field='0' as center; example: phasecenter='6'
 phasecenter='J2000 19h30m00 -40d00m00'
 phasecenter=['J2000 19h30m00 -40d00m00', 'J2000 19h57m00 40d00m00']
 for wide-field, plus one outlier field.
stokes -- Stokes parameters to image
 default='I'; example: stokes='IQUV';
 Options: 'I','IV','IQU','IQUV'
niter -- Number iterations, set to zero for no CLEANing
 default: 500; example: niter=500
gain -- Loop gain for CLEANing
 default: 0.1; example: gain=0.1
threshold -- Flux level at which to stop CLEANing (units=mJy)
 default: 0.0; example: threshold=0.0
mask -- Name(s) of mask image(s) used for CLEANing
 default: '' example: mask='orion.mask'
 Number of mask fields must equal number of imaged fields
cleanbox -- List of [blc-x,blc-y,trc-x,trc-y] values
 default: []; example: cleanbox=[110,110,150,145]
 Note: This can also be a filename with clean values:
 fieldindex blc-x blc-y trc-x trc-y
 cleanbox = 'interactive' is very useful.
--- Data Selection
nchan -- Number of channels to select
 default: 1; example: nchan=45
start -- Start channel, 0-relative
 default=0; example: start=5
 if mode='frequency' then a frequency value e.g start='1.4GHz'
width -- Channel width (value > 1 indicates channel averaging)
 default=1; example: width=5
 if mode='frequency' then a frequency value e.g width='10kHz'
step -- Step in channel number
 default=1; example: step=2
field -- Select field using field id(s) or field name(s).
 [run listobs to obtain the list id's or names]
 default: ''=all fields
 If field string is a non-negative integer, it is assumed a field index
 otherwise, it is assumed a field name
 field='0~2'; field ids 0,1,2
 field='0,4,5~7'; field ids 0,4,5,6,7
 field='3C286,3C295'; field named 3C286 adn 3C295

```

```

 field = '3,4C*'; field id 3, all names starting with 4C
 example for multiple ms in vis parameter:
 field=['0~2', '1,2']
spw -- Select spectral window/channels
 default: ''=all spectral windows and channels
 spw='0~2,4'; spectral windows 0,1,2,4 (all channels)
 spw='<2'; spectral windows less than 2 (i.e. 0,1)
 spw='0:5~61'; spw 0, channels 5 to 61
 spw='0,10,3:3~45'; spw 0,10 all channels, spw 3, channels 3 to 45.
 spw='0~2:2~6'; spw 0,1,2 with channels 2 through 6 in each.
 spw='0:0~10;15~60'; spectral window 0 with channels 0-10,15-60
 spw='0:0~10,1:20~30,2:1;2;3'; spw 0, channels 0-10,
 spw 1, channels 20-30, and spw 2, channels, 1,2 and 3
 For multiple ms in vis parameter:
 spw=['0,10,3:3~45', '<2']
timerange -- Select time range subset of data (not implemented yet)
 default='' meaning no time selection
 example: timerange='YYYY/MM/DD/HH:MM:SS.sss'
 timerange='< YYYY/MM/DD/HH:MM:SS.sss'
 timerange='> YYYY/MM/DD/HH:MM:SS.sss'
 timerange='ddd/HH:MM:SS.sss'
 timerange='< ddd/HH:MM:SS.sss'
 timerange='> ddd/HH:MM:SS.sss'
restfreq -- Specify rest frequency to use for image
 default='' (i.e., try to use the restfreq specified in the visibility data)

--- Weighting
weighting -- Weighting to apply to visibilities
 default='natural'; example: weighting='uniform';
 Options: 'natural','uniform','briggs','briggsabs','radial', 'superuniform'
robust -- 'briggs' and 'briggsabs' robustness parameter
 default=0.0; example: robust=0.5;
 Options: -2.0 to 2.0; -2 (uniform)/+2 (natural)
npixels -- number of pixels to determine uv-cell size for weight calculation
 -- Used with superuniform or briggs weighting schemes
 example: npixels=3

--- widefield controls
ftmachine -- Gridding method for the image;
 ft (standard interferometric gridding).
 wproject (wprojection algorithm for gridding)
 default: wproject
wprojplanes -- Number w-projection planes to use for gridding
 default: 256
 example: wprojplanes=64
Good value = BMAX(klambda) * Map width(arcmin)^2 / 600

```

```

 facets -- Number of facets along one axis on central image
 image is divided in facets x facets rectangles.
 default: 1
 example: facets=3 makes 3x3 images to cover the field
if ftmachine = 'ft', only faceting is used
 if ftmachine = 'wproject', both wplanes and faceting
 can be used (see below).

cyclefactor -- Change the threshold at which the deconvolution cycle will
 stop and degrid and subtract from the visibilities. For bad PSFs,
 reconcile often (cyclefactor=4 or 5); For good PSFs, use
 cyclefactor 1.5 to 2.0.
 default: 2.5; example: cyclefactor=4, but decreases speed considerably.
 <cycle threshold = cyclefactor * max sidelobe * max residual>
cyclespeedup -- Cycle threshold doubles in this number of iterations
 default: -1; example: cyclespeedup=500

--- MEM parameters (Experimental, not well-tested)
sigma -- Target image sigma
 default: '0.001Jy'; example: sigma='0.1Jy'
targetflux -- Target flux for final image
 default: '1.0Jy'; example: targetflux='200Jy'
constrainflux -- Constrain image to match target flux;
 otherwise, targetflux is used to initialize model only.
 default: False; example: constrainflux=True
prior -- Name of MEM prior images
 default: ['']; example: prior='source_mem.image'

--- Multi-scale parameters (Experimental, not well-tested)
negcomponent -- Stop component search when the largest scale has found this
 number of negative components; -1 means continue component search
 even if the largest component is negative.
 default: 2; example: negcomponent=-1
scales -- Used for alg='multiscale'; set a number of scales or a vector
 default: [0,3,10]; example: scales=[0.0,3.0,10.0, 30]
-- interactive masking
npercycle -- when cleanbox is set to 'interactive',
 this is the number of iterations between each clean to update mask
 interactively. However, this number can be adjusted during execution.

uvtaper -- Apply additional uv tapering of the visibilities.
 default: uvtaper=False; example: uvtaper=True
 uvtaper=True expandable parameters
 outertaper -- uv-taper on outer baselines in uv-plane
 [bmaj, bmin, bpa] taper Gaussian scale in uv or
 angular units. NOTE: uv taper in (klambda) is

```



```

 roughly on-sky FWHM(arcsec/200)
 default: outertaper=[]; no outer taper applied
 example: outertaper=['5klambda'] circular taper
 FWHM=5 kilo-lambda
 outertaper=['5klambda','3klambda','45.0deg']
 outertaper=['10arcsec'] on-sky FWHM 10"
 outertaper=['300.0'] default units are meters
 in aperture plane
 innertaper -- uv-taper in center of uv-plane
 NOT YET IMPLEMENTED

```

```

restoringbeam -- Output Gaussian restoring beam for CLEAN image
 [bmaj, bmin, bpa] elliptical Gaussian restoring beam
 default units are in arc-seconds for bmaj,bmin, degrees
 for bpa default: restoringbeam=[]; Use PSF calculated
 from dirty beam.
 example: restoringbeam=['10arcsec'] circular Gaussian
 FWHM 10" example:
 restoringbeam=['10.0','5.0','45.0deg'] 10"x5"
 at 45 degrees

```

```

calready -- if True will create scratch columns if they are
 not there. And after clean completes the predicted model
 visibility is from the clean components are
 written to the ms.

```

```

async -- Run asynchronously
 default = False; do not run asynchronously

```

```

=====

```

#### HINTS ON RUNNING WIDEFIELD

1. Decide if the images will be specified directly in the inputs or with an outlier file. For more than a few fields, an outlier file more convenient.

Direct Method:

```

cell = ['1.0arcsec', '1.0arcsec']
imagename = ['M1_0','M1_1','M1_2']
imsize = [[1024,1024],[128,128],[128,128]]
phasecenter = ['J2000 13h27m20.98 43d26m28.0',
 'J2000 13h30m52.159 43d23m08.02', 'J2000 13h24m08.16 43d09m48.0']

```

Text file method (in outlier.txt)

```

imagenname = 'M1'
outlierfile = 'outlier.txt'
[phasecenter, imsize ignored]

```

Contents of outlier.txt

```

C 0 1024 1024 13 27 20.98 43 26 28.0
C 1 128 128 13 30 52.158 43 23 08.00
C 2 128 128 13 24 08.163 43 09 48.00

```

In both cases the following images will be made:

```

M1_0.image, M1_1.image, M1_2.image cleaned images
M1_0.model, M1_1.model, M1_2.model model images
M1_0.residual, M1_1.residual, M1_2.residual residual images

```

2. Wprojection: It is fastest to use wprojection without faceting.  
ftmachine = 'wproject'  
wprojplane = NN

The value of NN should be chosen as small as possible to reduce execution time. The algorithm

NN = BMAX(klambda) \* imagewidth (arcmin)<sup>2</sup> / 600, with a minimum of 16, should be adequate.

3. Depending on the memory of the computer, a limit of about 5000x5000 may occur for example if a computer has 2Gbyte of RAM. Also a 32-bit computer has a maximum limit of 2Gbyte memory usable per process, irrespective of how much physical RAM is present. Hence it is recommended to move to a 64-bit computer with more than 2 GByte of RAM for >5000x5000 images

4. For data with extremely large 'w' values, i.e low frequency, long baseline and very widefield image, the wprojection convolution can be very large and either not fit in memory or slow for processing. In these cases you should consider using both ftmachine='wproject' and facets=xx where is 3.

---

## Permissions

- Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

- Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.
- Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by AUI.