

CASA Alpha patch 20070417

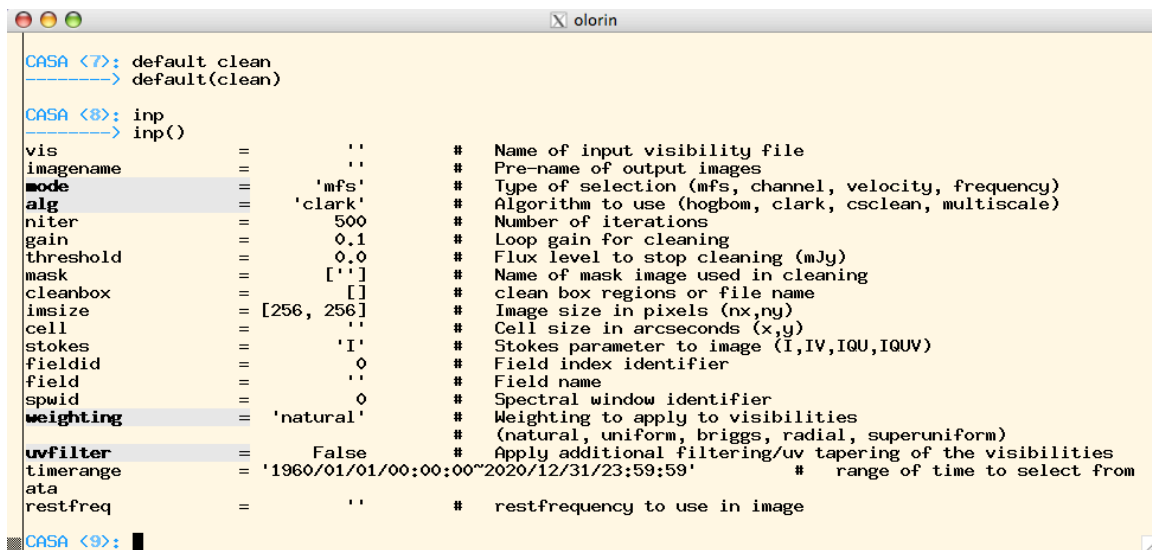
Key changes:

- *Task interface enhancement*
- *Data selection enhancement*
- *XY Display enhancement*
- *Flag versions have been enabled*
- *Reassignment of CASA tool/task names disabled*
- *ALMATST5 stability issues*
- *Matplotlib profiling*
- *Summary interface changes*

Task interface enhancement

The task interface was improved in the following ways:

- Error handling: the input file is tested for existence in all cases (avoiding throwing more obscure messages when problems arise); a non-existent file will be marked in red by 'inp' command to help signal the user (see below).
- Default values for parameters were improved based on task context; single line help can also change based on the use of the parameter in a task.
- Task parameter lists can expand and contract based on the user-specified values of other parameters. This allows generally concise parameter lists for a task unless parameters are needed. See figure below for an illustration. Formatting of parameter lists in interactive mode is used to highlight the type of parameters available:
 - Key: parameter = value
 - ♣ parameter
 - Plain black font = standard parameter
 - Bold black font + grey highlight = expandable parameter; setting this parameter to different values will open up access to relevant sub parameters.
 - Indented green font = sub parameter
 - ♣ Value
 - Black font = parameter is set to default value
 - Blue font = parameter is set to non-default value
 - Red font = parameter is outside of the menu choice options, or is the wrong type.



```
CASA <7>: default clean
-----> default(clean)

CASA <8>: inp
-----> inp()

vis = '' # Name of input visibility file
imagenam = '' # Pre-name of output images
mode = 'mfs' # Type of selection (mfs, channel, velocity, frequency)
alg = 'clark' # Algorithm to use (hogbom, clark, csclean, multiscale)
niter = 500 # Number of iterations
gain = 0.1 # Loop gain for cleaning
threshold = 0.0 # Flux level to stop cleaning (mJy)
mask = [''] # Name of mask image used in cleaning
cleanbox = [] # clean box regions or file name
imsize = [256, 256] # Image size in pixels (nx,ny)
cell = '' # Cell size in arcseconds (x,y)
stokes = 'I' # Stokes parameter to image (I,IV,IQUV)
fieldid = 0 # Field index identifier
field = '' # Field name
spwid = 0 # Spectral window identifier
weighting = 'natural' # Weighting to apply to visibilities
# (natural, uniform, briggs, radial, superuniform)
uvfilter = False # Apply additional filtering/uv tapering of the visibilities
timerange = '1960/01/01/00:00:00~2020/12/31/23:59:59' # range of time to select from
ata
restfreq = '' # restfrequency to use in image

CASA <9>: █
```

Figure 1 CASA format of 'inp' command for clean task at startup. Note the **mode**, **alg**, **weighting** and **uvfilter** parameters are bold indicating that they may expand based on user choices. The use of grey highlighting also distinguishes these parameters.

```

CASA <5>: run clean.last
CASA <6>: inp
-----> inp()
vis                    = 'ngc5921_src.split.ms' # Name of input visibility file
imagename              = 'ngc5921_task'         # Pre-name of output images
mode                   = 'channel'             # Type of selection (mfs, channel, velocity, frequency)
  nchan                 = 46                   # Number of channels to select
  start                 = 0                    # Start channel
  step                  = 1                    # Increment between channels/velocity
  width                 = 1                    # Channel width (value > 1 indicates channel averaging)
alg                     = 'hogbom'             # Algorithm to use (hogbom, clark, csclean, multiscale)
niter                  = 6000                  # Number of iterations
gain                   = 0.1                   # Loop gain for cleaning
threshold              = 8.0                  # Flux level to stop cleaning (mJy)
mask                   = ''                   # Name of mask image used in cleaning
cleanbox               = []                   # clean box regions or file name
imsize                 = [256, 256]           # Image size in pixels (nx,ny)
cell                   = [15.0, 15.0]         # Cell size in arcseconds (x,y)
stokes                 = 'I'                  # Stokes parameter to image (I,IV,IQU,IQUV)
fieldid                = 0                    # Field index identifier
field                  = ''                   # Field name
spwid                  = 0                    # Spectral window identifier
weighting              = 'briggs'             # Weighting to apply to visibilities
  rmode                 = 'norm'              # Robustness mode (for Briggs weighting)
  robust                = 0.5                 # Briggs robustness parameter
  noise                 = '0.03y'            # noise parameter for briggs weighting when rmode='abs'
  npixels               = 0                   # number of pixels to determine uv-cell size 0=> field of view
uvfilter               = False                # Apply additional filtering/uv tapering of the visibilities
timerange              = '1960/01/01/00:00:00~2020/12/31/23:59:59' # range of time to select from data
ata                    = ''                   # restfrequency to use in image
restfreq               = ''
CASA <7>:

```

Figure 2 CASA format of 'inp' command for clean task after task execution. The blue-colored values indicate parameters set to values different from the default. The combination of indenting and green color indicates sub parameters.

```

CASA <7>: vis=2
CASA <8>: inp
-----> inp()
vis                    = 2                    # Name of input visibility file
imagename              = ''                   # Pre-name of output images
mode                   = 'happy'             # Type of selection (mfs, channel, velocity, frequency)
alg                     = 'clark'             # Algorithm to use (hogbom, clark, csclean, multiscale)
niter                  = 500                  # Number of iterations
gain                   = 0.1                   # Loop gain for cleaning
threshold              = 0.0                  # Flux level to stop cleaning (mJy)
mask                   = []                   # Name of mask image used in cleaning
cleanbox               = []                   # clean box regions or file name
imsize                 = [256, 256]           # Image size in pixels (nx,ny)
cell                   = ['1.0arcsec', '1.0arcsec'] # Cell size in arcseconds (x,y)
stokes                 = 'I'                  # Stokes parameter to image (I,IV,IQU,IQUV)
fieldid                = 0                    # Field index identifier
field                  = ''                   # Field name
spwid                  = 0                    # Spectral window identifier
weighting              = 'natural'           # Weighting to apply to visibilities
  (natural, uniform, briggs, radial, superuniform)
uvfilter               = False                # Apply additional filtering/uv tapering of the visibilities
timerange              = '1960/01/01/00:00:00~2020/12/31/23:59:59' # range of time to select from data
restfreq               = ''                   # restfrequency to use in image
CASA <9>:

```

Figure 3 CASA format of 'inp' command illustrating 'bad' values for the vis and mode parameters. The vis parameter has the wrong type; the mode parameter is not an acceptable option - both are drawn in red to draw attention to the problem.

Data selection enhancement

As was noted, the data selection rules varied across task and lacked flexibility. These have been made uniform across the tool level and are in the process of being standardized across the tasks (scheduled for the June 15 patch). The plotxy task currently features this and illustrates how it will be implemented for the other tasks:

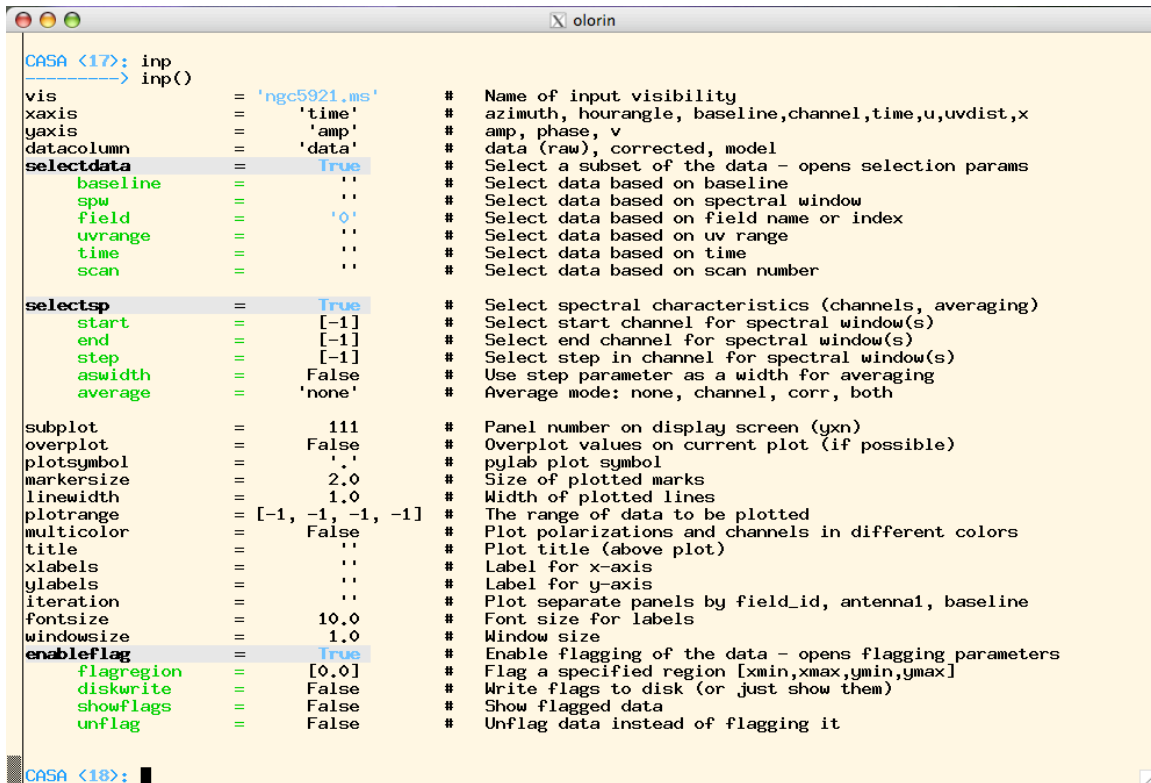
The standard selection parameters available and some examples are given below:

--- Select Data ---

```
selectdata -- Select a subset of the visibility file data to plot
  default: False; example: selectdata=True
baseline -- Select data based on baseline
  default: " (all); example: baseline='5&6' baseline 5-6
  baseline='5&6;7&8' #baseline 5-6 and 7-8
  baseline='5' # all baselines with antenna 5
  baseline='5,6' # all baselines with antennas 5 and 6
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
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 based on uvrange(s)
  default: " (all); example:
  uvrange='0~1000' # uvranges from 0-1000 meters
  uvrange='>4kl' #uvranges greater than 4 kilo lambda
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 - Not Yet Implemented
  default: " (all); example: scan='>3'
```

XY display enhancement/interface change

- Interface change: the **flagxy** task has been removed and its functionality has been added to the **plotxy** task.
- Interface change: the revised data selection facilities have been included in the **plotxy** task.
- Display and flagging on spectral/polarization averages are now available. See the example below:



```
CASA <17>: inp
inp()
vis = 'ngc5921.ms' # Name of input visibility
xaxis = 'time' # azimuth, hourangle, baseline,channel,time,u,uvdist,x
yaxis = 'amp' # amp, phase, v
datacolumn = 'data' # data (raw), corrected, model
selectdata = True # Select a subset of the data - opens selection params
baseline = '' # Select data based on baseline
spw = '' # Select data based on spectral window
field = '0' # Select data based on field name or index
uvrange = '' # Select data based on uv range
time = '' # Select data based on time
scan = '' # Select data based on scan number

selectsp = True # Select spectral characteristics (channels, averaging)
start = [-1] # Select start channel for spectral window(s)
end = [-1] # Select end channel for spectral window(s)
step = [-1] # Select step in channel for spectral window(s)
aswidth = False # Use step parameter as a width for averaging
average = 'none' # Average mode: none, channel, corr, both

subplot = 111 # Panel number on display screen (yxn)
overplot = False # Overplot values on current plot (if possible)
plotsymbol = '.' # pylab plot symbol
markersize = 2.0 # Size of plotted marks
linewidth = 1.0 # Width of plotted lines
plotrange = [-1, -1, -1, -1] # The range of data to be plotted
multicolor = False # Plot polarizations and channels in different colors
title = '' # Plot title (above plot)
xlabel = '' # Label for x-axis
ylabel = '' # Label for y-axis
iteration = '' # Plot separate panels by field_id, antenna1, baseline
fontsize = 10.0 # Font size for labels
windowsize = 1.0 # Window size
enableflag = True # Enable flagging of the data - opens flagging parameters
flagregion = [0.0] # Flag a specified region [xmin,xmax,ymin,ymax]
diskwrite = False # Write flags to disk (or just show them)
showflags = False # Show flagged data
unflag = False # Unflag data instead of flagging it

CASA <18>: █
```

Figure 4 Full set of parameters for plotxy task. Setting ‘enableflag=True’ will trigger the interactive mode allowing the user to draw regions on the plot and then choose to either flag, unflag or identify (locate) the data.

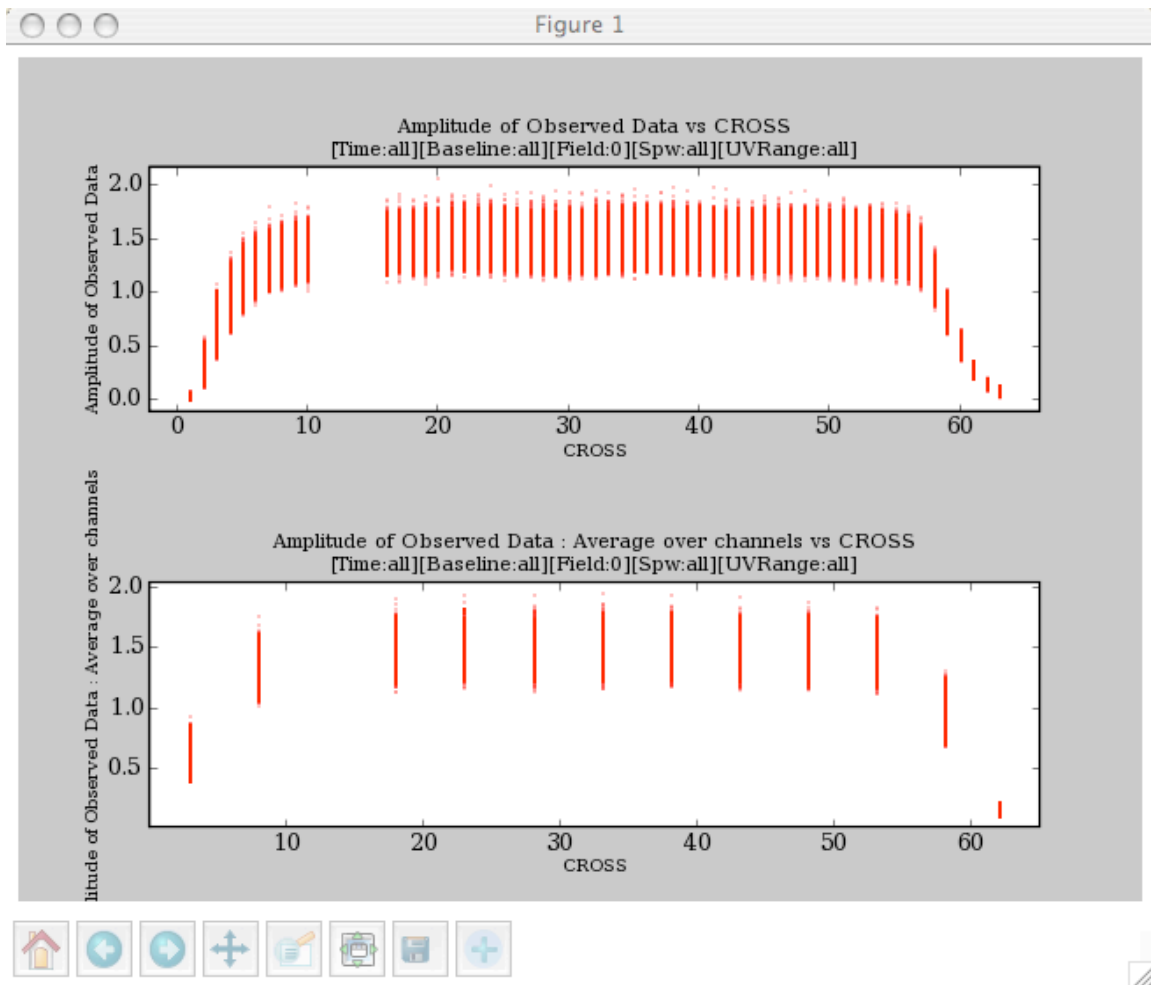


Figure 5 Results of flagging the third 5-channel average in the lower plot; note that the five channels which composed the average are then also flagged.

The script to make this plot is:

```
plotxy(vis="ngc5921.ms", xaxis="channel", yaxis="amp", datacolumn="data",
selectdata=True, field="0", subplot=211)
plotxy(vis="ngc5921.ms", xaxis="channel", yaxis="amp", datacolumn="data",
selectdata=True, field="0", selectsp=True, start=[-1], end=[-1], step=[5], aswidth=True, average="chan",
subplot=212, enableflag=True, diskwrite=True)
#select the region around the third, 5-channel average:
#click the Right '+' box to mark the region and then click it again to end region selection
#type 'f' in the command console to 'flag' the data
```

Flag versions have been enabled

- There is a new task, *flagmanager*:

```
olorin
CASA <1>: inp flagmanager
-----> inp(flagmanager)
vis          =          ..      # Name of input visibility file (MS)
mode        = 'list'         # Flag management operation (list,save,restore,delete)

CASA <2>: mode='save'

CASA <3>: inp
-----> inp()
vis          =          ..      # Name of input visibility file (MS)
mode        = 'save'         # Flag management operation (list,save,restore,delete)
versionname =              ''   # Name of flag version (no spaces)
comment     =              ''   # Short description of flag version
merge       = 'replace'      # Merge option (replace, and, or)

CASA <4>: mode='restore'

CASA <5>: inp
-----> inp()
vis          =          ..      # Name of input visibility file (MS)
mode        = 'restore'      # Flag management operation (list,save,restore,delete)
versionname =              ''   # Name of flag version (no spaces)
merge       = 'replace'      # Merge option (replace, and, or)

CASA <6>: mode='delete'

CASA <7>: inp
-----> inp()
vis          =          ..      # Name of input visibility file (MS)
mode        = 'delete'      # Flag management operation (list,save,restore,delete)
verssionname =              ''  #

CASA <8>: █
```

Figure 6 Illustration of the flagmanager task in each of its modes.

Example:

- 1) plot data of interest, e.g., uvdist plot
 - a. `plotxy('ngc5921.ms','uvdist',selectdata=True,field='0')`

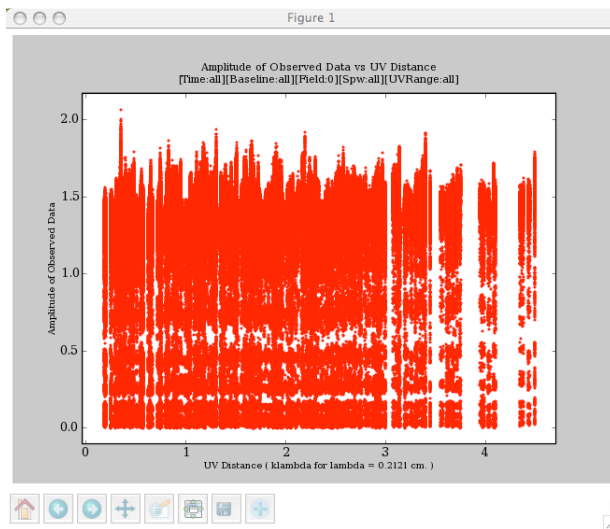


Figure 7 Starting data display (no flagging).

- 2) save starting flags
 - a. `flagmanager('ngc5921.ms',mode='save',versionname='original',comment='starting flags')`

- 3) list flag versions
 - a. `flagmanager('ngc5921.ms',mode='list')`
 - b. #look in logger to see the entry:
 - i. main: working copy in main table
 - ii. original: starting flags
- 4) flag some data
 - a. `plotxy('ngc5921.ms','uvdist',selectdata=True,field='0',enableflag=True)`
 - b. #click on the right-most matplotlib button (the cross), mark region(s) and then click on the button again. You will be prompted to type: 'f' for flag, 'u' for unflag, or 'l' for locate; type 'f' and <return>

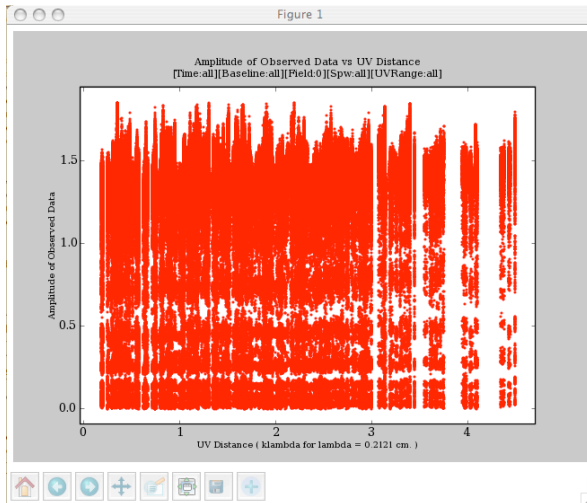


Figure 8 Flag version 'cliphigh'; data was clipped above 1.85.

- 5) save this version of flags
 - a. `flagmanager('ngc5921.ms',mode='save',versionname='cliphigh',comment='clip above 1.85')`
- 6) restore previous flag version
 - a. `flagmanager('ngc5921.ms',mode='restore',versionname='original')`
 - b. #default is to replace the flags with the version specified
- 7) flag different data
 - a. `plotxy('ngc5921.ms','uvdist',selectdata=True,field='0',enableflag=True)`
 - b. #click on the right-most matplotlib button (the cross), mark region(s) and then click on the button again. You will be prompted to type: 'f' for flag, 'u' for unflag, or 'l' for locate; type 'f' and <return>

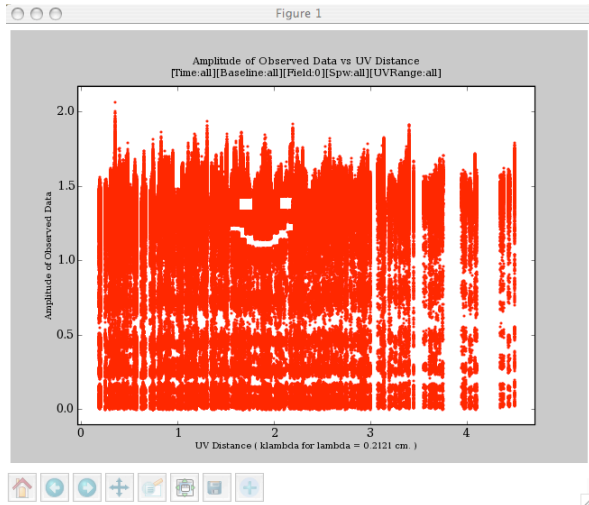


Figure 9 Flag version 'smiley'; intended to be easily recognizable!

- 8) save this version of flags
 - a. `flagmanager('ngc5921.ms',mode='save',versionname='smiley', comment='easily recognizable flag region')`
- 9) restore previous flag version and merge it into the existing flags
 - a. `flagmanager('ngc5921.ms',mode='restore',versionname='cliphigh',merge='or')`

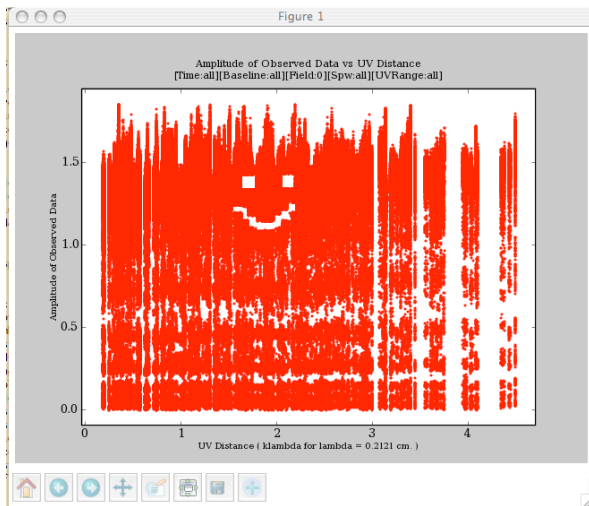
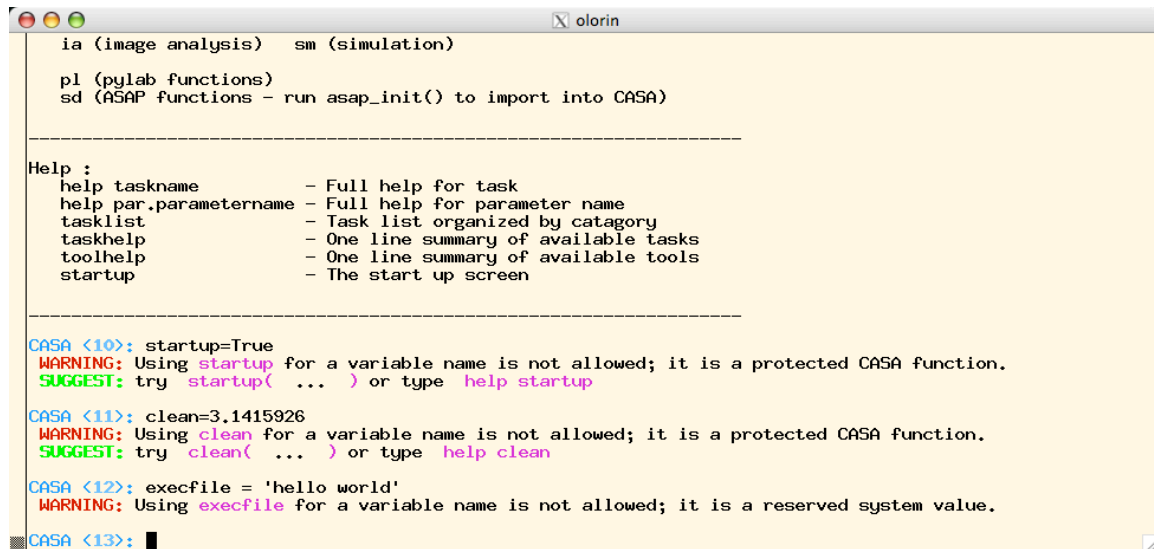


Figure 10 Combined flags using 'smiley' and 'cliphigh' flag versions.

Reassignment of CASA tools/tasks is disabled

- Python natively does not allow variable names to be protected. A means of protecting CASA and system variables has been enabled:



```
ola (image analysis)  sm (simulation)

pl (pylab functions)
sd (ASAP functions - run asap_init() to import into CASA)

-----

Help :
help taskname          - Full help for task
help par.parametername - Full help for parameter name
tasklist               - Task list organized by category
taskhelp               - One line summary of available tasks
toolhelp               - One line summary of available tools
startup                - The start up screen

-----

CASA <10>: startup=True
WARNING: Using startup for a variable name is not allowed; it is a protected CASA function.
SUGGEST: try startup( ... ) or type help startup

CASA <11>: clean=3.1415926
WARNING: Using clean for a variable name is not allowed; it is a protected CASA function.
SUGGEST: try clean( ... ) or type help clean

CASA <12>: execfile = 'hello world'
WARNING: Using execfile for a variable name is not allowed; it is a reserved system value.

CASA <13>: █
```

Figure 11 Illustration of the protection of CASA and system functionality.

ALMA TST5 Stability issues

The following issues resulting in a crash of CASA have been resolved:

- ✓ Imager (infinite loop when dirty beam is poorly defined)
- ✓ Imager (corrupted HISTORY table)
- ✓ Crash of flagxy
- ✓ Crash of ploxy when MS doesn't exist
- ✓ "Crash" of flagxy when there is no data to plot
- ✓ Crash of the split task

deferred: crash of flagdata (new flag tool is under development).

postponed: tablebrowser issues

matplotlib profiling

- Case:
 - 700 MB dataset
 - DATA
 - = 2 polarizations x 7 channels x 1876352 rows
 - = 26268928 points
 - = 210 MB (stored as Double or Complex)
 - = 100 MB (stored as Float)
 - Note: 100 MB: only Y-axis data (Float) + 15 MB for X-axis data (Double)
- Read time from disk
 - Expectation:
 - = 20 MB/s (standard)
 - = ~10 seconds read time
 - Result:
 - = 29 seconds reported
 - Note: includes 26 million of the following operation: $\text{SQRT}(\text{REAL}^2 + \text{IMAG}^2)$
 - Current effort: exploring a Python compile option difference between Mac and Linux. Exploring time versus memory optimization efforts (e.g., if we only need to plot, then a column access (rather than row-based) can be used to speed up the read.
- Memory allocated
 - Expectation:
 - = 420 MB (Double for both X and Y) + flags
 - Result:
 - = 273 MB (before matplotlib does its allocation)
 - = 676 MB after plotting
 - matplotlib backend keeps a copy of both X and Y in Double precision (to facilitate panning and zooming without having to re-read from disk)
 - Current effort: exploring alternate backends; working to write a backend that does not require this.
- Plotting time
 - Expectation:
 - = 52 seconds (matplotlib does ~500K pts/sec natively)
 - Result:
 - = 60 seconds
- SUMMARY:

- The matplotlib backend used by CASA makes an additional copy of the data to be plotted to enable faster zoom/pan operations (without having to read from the disk again). In addition, we found that the plotting code within CASA was making an additional, superfluous copy of the data; this was removed.
- The disk-read times for Linux systems were found to be worse than for outside of CASA (not seen on Mac OSs); we are working to resolve this.
- The CASA plotting speed is now roughly the same as for native matplotlib.

Key interface changes

The following lists the items that likely have to change in scripts:

- **importvla** now labels the antenna names as:
 - VAx_{xx}: for VLA antennas; xx indicates the two-character name of the antenna, e.g., VA01, VA18
 - EA_{xx}: for EVLA antennas; xx is the same as above
 - **Result:** *refant* parameters will need to change in scripts if they were specified.
- **restore** task has been removed; use the **default** task instead.
- **flagxy** task has been removed; its functionality has been combined into the **plotxy** task.
- **plotxy** task has new parameters for data selection.