CASA Introduction

Ronny Zhao-Geisler 趙家驊

National Taiwan Normal University,
Department of Earth Sciences, Taiwan, ROC

AIW, January 2013

background: 33 antennas of ALMA (ESO/NAOJ/NRAO)

CASA: Getting Started

CASA (**C**ommon **A**stronomy **S**oftware **A**pplications) is a comprehensive software package to calibrate, image, and analyze radioastronomical data from interferometers (such as ALMA and EVLA) as well as single dish telescopes.

Current release:

CASA 4.0 (http://casa.nrao.edu/index.shtml)

Help on CASA:

CASA Cookbook (~ 600 pages),
CASA Guides (http://casaguides.nrao.edu/index.php),
Tasks (http://casa.nrao.edu/docs/TaskRef/TaskRef.html),
Help Desk (my.nrao.edu) and **"help"** within CASA

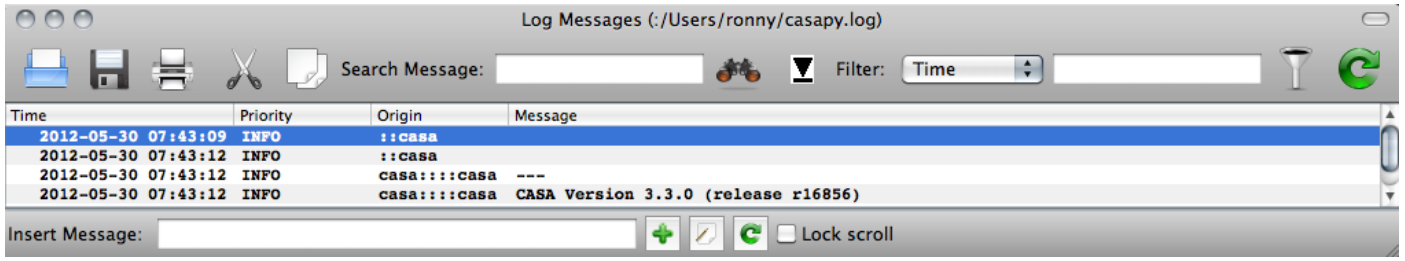
CASA: Startup

to start CASA type: **'casapy'**

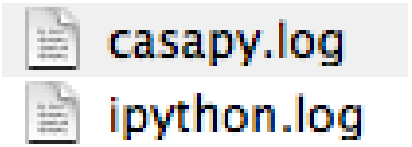
=> iphyton interface

```
CASA <2>:
```

=> casalogger GUI



=> session logging



(CASA messages)
(ipython command history)

to end CASA type: **'quit'** or **'exit'**

CASA: Tasks (I)


```
> tasklist      CASA <2>: tasklist
```

Imaging	Analysis	Import/export	Information	Editing	Manipulation
clean	imcollapse	exportfits	imhead	fixplanets	concat
deconvolve	imcontsub	exportuvfits	imstat	fixvis	conjugatevis
feather	imfit	importaipscaltable	imval	flagautocorr	cvel
ft	imhead	importasdm	listcal	flagcmd	fixvis
imcontsub	immath	importfits	listhistory	flagdata	hanningsmooth
(boxit)	immoments	importfitsidi	listobs	flagmanager	imhead
(csvclean)	impbcor	importuvfits	listvis	msview	msmoments
{mosaic}	imregrid	importvla	plotms	plotms	plotms
{widefield}	imsmooth	(exportasdm)	plotuv	plotxy	plotxy
	imstat	(importevla)	plotxy	(flagdata2)	split
	imtrans	(importgmrt)	vishead	(testautoflag)	testconcat
	imval	{importoldasdm}	visstat		uvcontsub
	listvis		(listsdm)		vishead
	slsearch				{uvcontsub2}
	splattotable				
	(specfit)				

() … experimental tasks, {} … deprecated tasks

CASA: Tasks (II)

Visualization	Simulation	Single dish	Utility	Calibration	Modeling
clearplot	sim_analyze	asap_init	browsetable	accum	setjy
imview	sim_observe	sdaverage	clearplot	applycal	uvcontsub
msview	simdata	sdbaseline	clearstat	bandpass	uvmodelfit
plotants		sdcal	concat	blcal	uvsub
plotcal		sdcoadd	conjugatevis	calstat	{uvcontsub2}
plotms		sdfit	find	clearcal	
plotuv		sdflag	help par.parameter	fixplanets	
plotxy		sdflagmanager	help taskname	fluxscale	
viewer		sdimaging	imview	ft	
		sdimprocess	msview	gaincal	
		sdlist	plotms	gencal	
		sdmath	rmtables	listcal	
		sdplot	startup	plotants	
		sdsave	taskhelp	plotcal	
		sdscale	tasklist	polcal	
		sdsmooth	testconcat	setjy	
		sdstat	toolhelp	smoothcal	
		sdtpimaging		uvmodelfit	
				uvsub	

Tools are also available (functions, underneath tasks).

CASA: Task Description


```
> taskhelp      CASA <2>: taskhelp

Available tasks:

accum              : Accumulate incremental calibration solutions into a calibration table
applycal           : Apply calibrations solutions(s) to data
autoclean          : CLEAN an image with automatically-chosen clean regions.
bandpass           : Calculates a bandpass calibration solution
blcal              : Calculate a baseline-based calibration solution (gain or bandpass)
boxit              : Box regions in image above given threshold value.
browsetable        : Browse a table (MS, calibration table, image)
calstat            : Displays statistical information on a calibration table
clean              : Invert and deconvolve images with selected algorithm
clearcal           : Re-initializes the calibration for a visibility data set
clearplot          : Clear the matplotlib plotter and all layers
clearstat          : Clear all autolock locks
concat             : Concatenate several visibility data sets.
conjugatevis       : Change the sign of the phases in all visibility columns.
csvclean           : This task does an invert of the visibilities and deconvolve in the image plane.
cvel               : regrid an MS to a new spectral window / channel structure or frame
deconvolve         : Image based deconvolver
exportasdm         : Convert a CASA visibility file (MS) into an ALMA Science Data Model
exportfits         : Convert a CASA image to a FITS file
exportuvfits       : Convert a CASA visibility data set to a UVFITS file;
feather            : Combine two images using their Fourier transforms
find               : Find string in tasks, task names, parameter names;
fixplanets         : Changes FIELD and SOURCE table entries based on user given direction or POINTING table,
fixvis             : Recalculates (u, v, w) and/or changes Phase Center
flagautocorr       : Flag autocorrelations
flagcmd            : Flagging task based on flagging commands
flagdata           :  All purpose flagging task based on selections
```

CASA: Task Use

use standard tasking interface:

- **`default, input, go`**
- uses global variables for task parameters

call as functions with arguments:

- **`taskname`**`(arg1=val1,arg2=val2,...)`
- unspecified parameters will be defaulted (globals are not used)

Standard Task Call (I)


```
CASA <2>: default('plotms')

CASA <3>: inp('plotms')
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 =          ''            #  input visibility dataset (blank for none)
xaxis               =          ''            #  plot x-axis (blank for default/current)
yaxis               =          ''            #  plot y-axis (blank for default/current)
selectdata          =        True            #  data selection parameters
     field          =          ''            #  field names or field index numbers (blank for all)
     spw            =          ''            #  spectral windows:channels (blank for all)
     timerange      =          ''            #  time range (blank for all)
     uvrange        =          ''            #  uv range (blank for all)
     antenna        =          ''            #  antenna/baselines (blank for all)
     scan           =          ''            #  scan numbers (blank for all)
     correlation    =          ''            #  correlations (blank for all)
     array          =          ''            #  (sub)array numbers (blank for all)
     observation    =          ''            #  Select by observation ID(s)
     msselect       =          ''            #  MS selection (blank for all)

averagedata         =        True            #  data averaging parameters
     avgchannel     =          ''            #  average over channel?  (blank = False, otherwise value in channels)
     avgtime        =          ''            #  average over time? (blank = False, other value in seconds)
     avgscan        =       False            #  only valid if time averaging is turned on.  average over scans?
     avgfield       =       False            #  only valid if time averaging is turned on.  average over fields?
     avgbaseline    =       False            #  average over all baselines?  (mutually exclusive with avgantenna)
     avgantenna     =       False            #  average by per-antenna?  (mutually exclusive with avgbaseline)
     avgspw         =       False            #  average over all spectral windows?
     scalar         =       False            #  Do scalar averaging?

transform           =       False            #  transform data in various ways?
extendflag          =       False            #  have flagging extend to other data points?
iteraxis            =          ''            #  the axis over which to iterate
...
```

set 'plotms' parameters to default & see input

expend-able para-meters

defaults are in **black**

Standard Task Call (II)


```
CASA <4>: vis = 'OrionKL.ms'

CASA <5>: xaxis = 'channels'

CASA <6>: inp('plotms')
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 = 'OrionKL.ms'       #  input visibility dataset (blank for none)
xaxis               = 'channels'         #  plot x-axis (blank for default/current)
yaxis               =          ''        #  plot y-axis (blank for default/current)
selectdata          =        True        #  data selection parameters
...
```

change global variables

errors are marked in **red**

```
CASA <7>: xaxis = 'channel'

CASA <8>: yaxis = 'amplitude'

CASA <9>: inp('plotms')
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 = 'OrionKL.ms'       #  input visibility dataset (blank for none)
xaxis               =  'channel'         #  plot x-axis (blank for default/current)
yaxis               = 'amplitude'        #  plot y-axis (blank for default/current)
     ydatacolumn    =          ''        #  data column to use for y-axis (blank for default/current)
...
```

correct error & check input again
(non-default values are in **blue**)

now also expended

```
CASA <10>: go('plotms')
Executing:  plotms()
```

execute task

Functional Call & Help

call task as function

```
CASA <11>: plotms(vis='OrionKL.ms',xaxis='channel',yaxis='amplitude')
```

some tasks return Python dictionaries, e.g. `myval=imval()`

---

save and restore global variables:

help on task 'saveinputs'

```
CASA <12>: help saveinputs
---------> help(saveinputs)
Help on function saveinputs in module __main__:

saveinputs(taskname=None, outfile='', myparams=None, ipython_globals=None)
    Save current input values to file on disk for a specified task:

    taskname -- Name of task
        default: <unset>; example: taskname='bandpass'
        <Options: type tasklist() for the complete list>
    outfile -- Output file for the task inputs
        default: taskname.saved; example: outfile=taskname.orion
```

parameters of current task 'plotms'
are saved / restored
as / from the file 'plotms.last'

```
CASA <13>: saveinputs
---------> saveinputs()

CASA <14>: tget
---------> tget()
Restored parameters from file plotms.last
```

CASA Help

```
CASA <15>: help clean
---------> help(clean)
Help on clean task:

Invert and deconvolve images with selected algorithm
       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
       imagename -- Pre-name of output images:
               default: none; example: imagename='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
        ...
```

another example:
help on task 'clean'

exit help by
typing 'q'

CASA Loops & Scripts

Arrays and loops (indentation matters!):

```
CASA <16>: basename=['observationA','observationB','observationC']

CASA <17>: for name in basename:
      ....:     listobs(vis=name+'.ms')
      ....:
      ....:
```

for <variable> in <array>

Scripting:

```
CASA <18>: execfile('OrionKL_calib.py')
```

name of iphyton script

Data Selection Syntax (I)


```
CASA <19>: inp('plotms')
#  plotms :: A plotter/interactive flagger for visibility data.
vis                 =         ''            #  input visibility dataset (blank for none)
xaxis               =         ''            #  plot x-axis (blank for default/current)
yaxis               =         ''            #  plot y-axis (blank for default/current)
selectdata          =       True            #  data selection parameters
     field          =         ''            #  field names or field index numbers (blank for all)
     spw            =         ''            #  spectral windows:channels (blank for all)
     timerange      =         ''            #  time range (blank for all)
     uvrange        =         ''            #  uv range (blank for all)
     antenna        =         ''            #  antenna/baselines (blank for all)
     ...
```

data selection input

---

vis … string with name of measurement set:
- example: vis = 'OrionKL.ms'

---

caltable … string with name of calibration table:
- example: caltable = 'OrionKL.tsys' / '.cal'

Data Selection Syntax (II)

field … string with source name or field ID:
- wildcard '*' can be used
- first checks for name, then ID
- example: field = '1331+305' / 'Orion*' / '0,3~5'

---

spw … string with spectral window ID plus channels:
- use ':' as separator of spw from optional channelization,
- use '^' as separator of channels from step/width
- example: spw = '1~8' / '1:0~127' / '<2' / '0~3:5~54^5' / '1412~1415MHz'

---

xaxis … string with x-axis data:
- example: xaxis = 'time' / 'chan' / 'freq' / 'antenna' / 'amp' / 'phase' / 'real' / 'imag' / 'snr' / 'u' / 'v'

---

yaxis … string with y-axis data:
- example: yaxis = 'amp' / 'phase' / 'real' / 'imag' / 'antenna'

Data Selection Syntax (III)

antenna … string with antenna name or ID:
- first check for name, then ID
- use '&' to select only cross-correlation of baseline(s)
- example: antenna = 'PM03,DV07' / '0,3~15' / '3&*' / '!5' (all but not 5)

---

timerange … string with date/time range:
- specify 'T0~T1' , missing parts of T1 are default to T0, can give 'T0+dT', example: timerange = '2012/04/13/09:15:00 ~ 09:25:00'

---

uvrange … string with uv-range:
- example: uvrange = '100~200km' / '>30klambda'

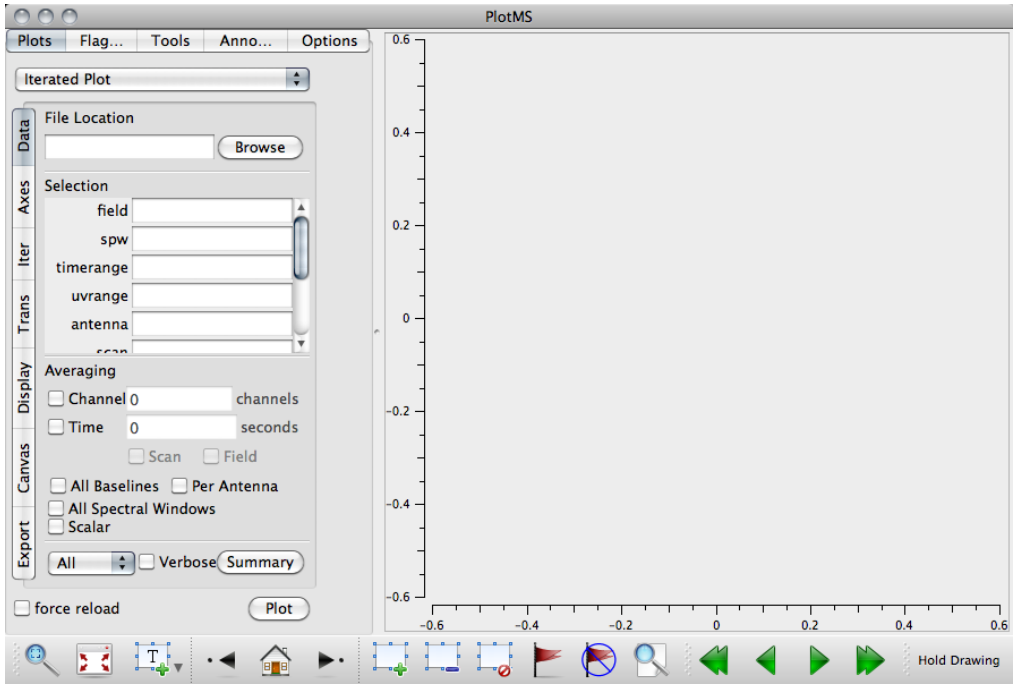
---

correlation … string with correlation product to be used:
- example: correlation = 'XX' / 'YY'

Plotting Tools

PlotMS:

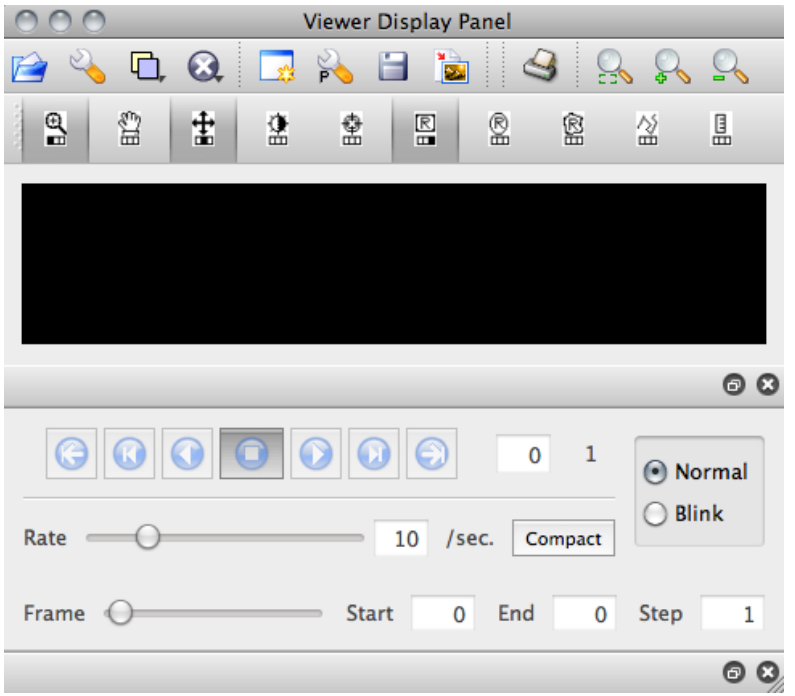
```
CASA <20>: plotms
--------> plotms()
```




```
plotms(vis='OrionKL.ms',spw='0',
  xaxis='frequency',yaxis='amp',
  field='1',antenna='1~8',...)
```

Viewer:

```
CASA <21>: viewer
--------> viewer()
```




```
viewer('OrionKL.image','raster')
```

Or as stand alone application:

```
> casaviewer OrionKL.fits &
```

Others:

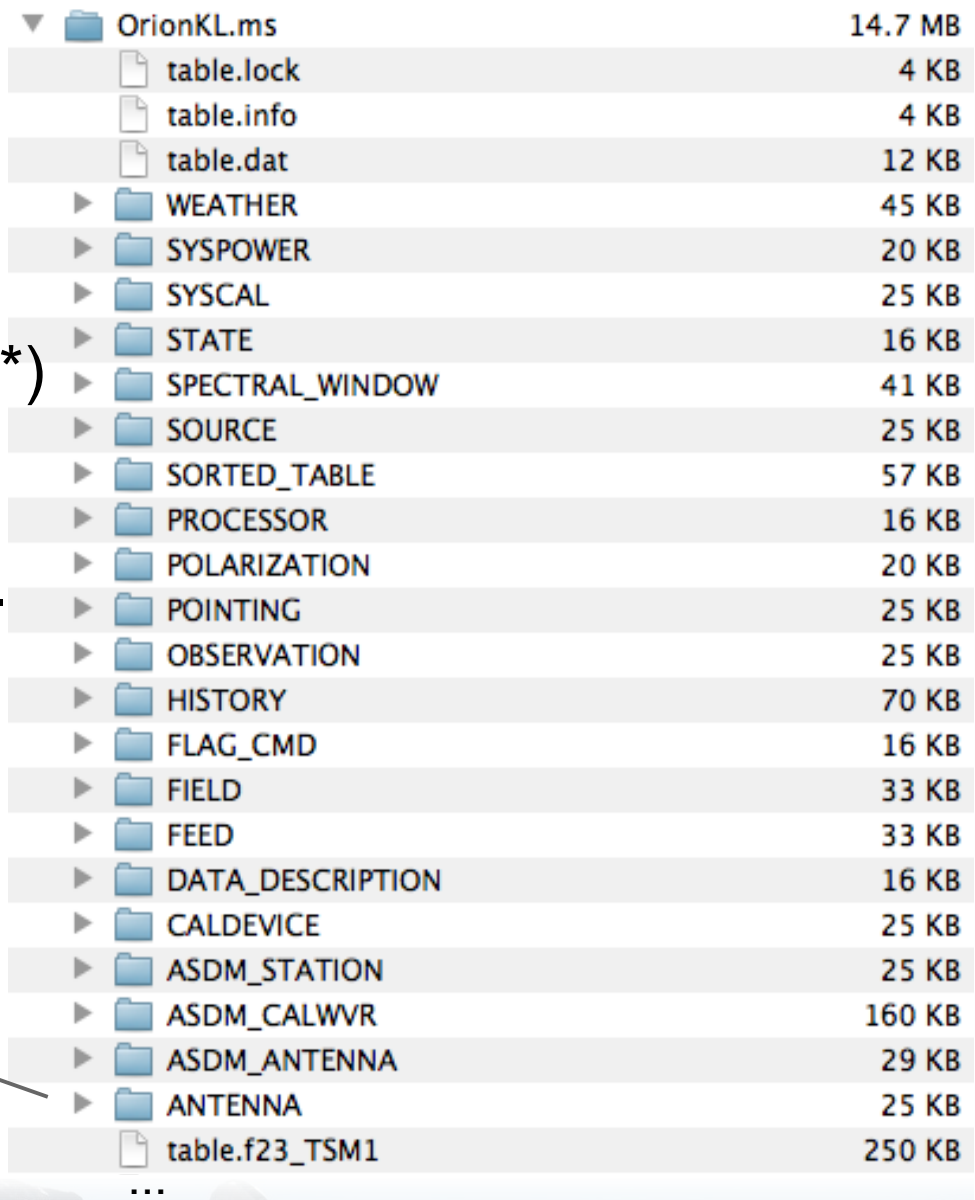
```
plotants(vis='OrionKL.ms',...)
plotcal(caltable='OrionKL.tsys',...)
```

Measurement Set

- data are organized in a **Measurement Set (MS)**

- a MS is a directory on disk:
  => it contains the MAIN tables (table.*)
  => it also contains sub-tables (as sub-directories): e.g. FIELD, SOURCE, ANTENNA, etc.

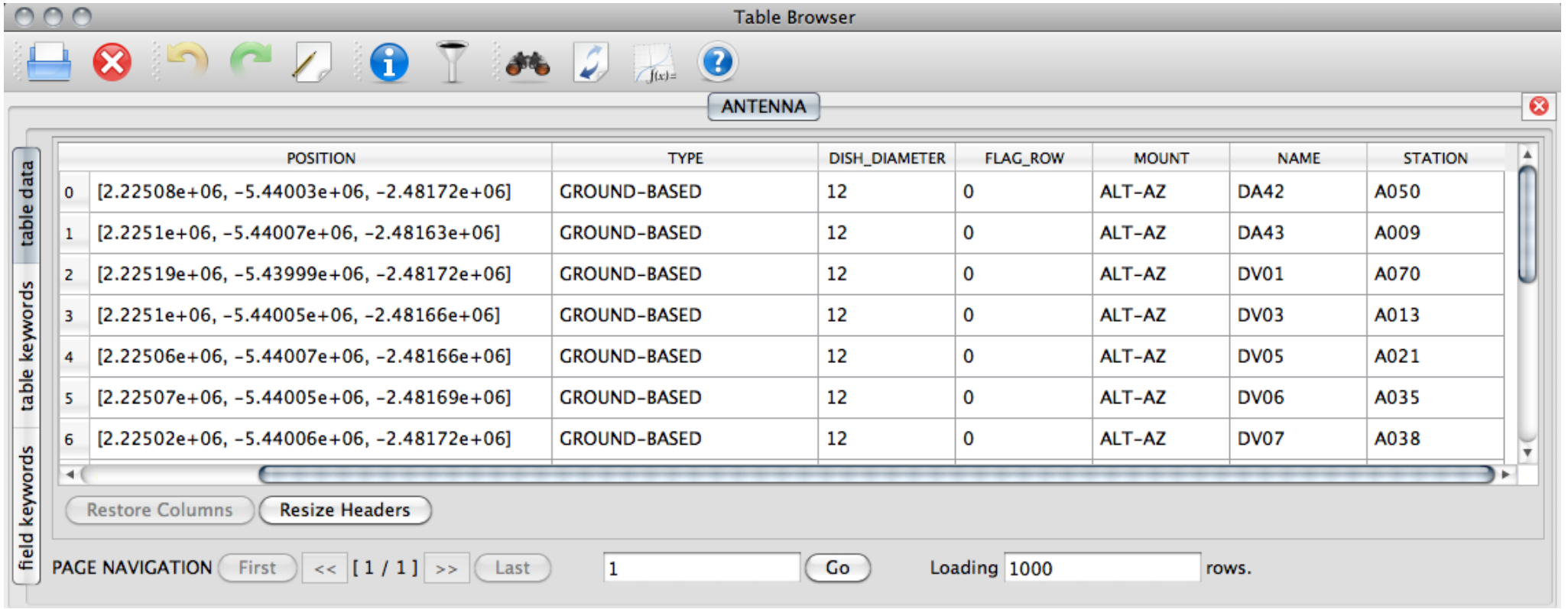
- to remove a MS:
  `rmtables('filename')`



Measurement Set Tables

use **browstable** to inspect tables:

```
CASA <22>: browsetable
---------> browsetable()
```



	POSITION	TYPE	DISH_DIAMETER	FLAG_ROW	MOUNT	NAME	STATION
0	[2.22508e+06, -5.44003e+06, -2.48172e+06]	GROUND-BASED	12	0	ALT-AZ	DA42	A050
1	[2.2251e+06, -5.44007e+06, -2.48163e+06]	GROUND-BASED	12	0	ALT-AZ	DA43	A009
2	[2.22519e+06, -5.43999e+06, -2.48172e+06]	GROUND-BASED	12	0	ALT-AZ	DV01	A070
3	[2.2251e+06, -5.44005e+06, -2.48166e+06]	GROUND-BASED	12	0	ALT-AZ	DV03	A013
4	[2.22506e+06, -5.44007e+06, -2.48166e+06]	GROUND-BASED	12	0	ALT-AZ	DV05	A021
5	[2.22507e+06, -5.44005e+06, -2.48169e+06]	GROUND-BASED	12	0	ALT-AZ	DV06	A035
6	[2.22502e+06, -5.44006e+06, -2.48172e+06]	GROUND-BASED	12	0	ALT-AZ	DV07	A038

```
browsetable(tablename='OrionKL.ms/ANTENNA')
```