A closer look at CASA
Dirk Petry (ESO), December 2010

Outline

→ application overview
   - the CASA system of applications

→ tasks and tools
   - the two-level user interface revisited

→ global variables
   - Python global variables and the task parameters

→ non-interactive casapy
   - casapy command line options

→ Measurement Set, ASDM, uvfits, ...
   - the visibility data formats

→ calibration tables
   - CASA tables for calibration data

→ CASA images and FITS
   - image storage in tables and FITS files

→ if you encounter problems
   - how and where to file a helpdesk ticket
CASA application overview

In release 3.1.0: 8 independent applications exposed to the user:

- **casapy** ........................................ the CASA “shell”
- **casapyinfo** ...................................... returns info about how CASA was built
- **casabrowser** .................................... == browsetable() task within casapy
- **casalogger** ...................................... the logger started by default with casapy
- **casaplotms** ..................................... will be == plotms() task within casapy
- **casaviewer** ..................................... == viewer() task within casapy
- **asdm2MS** ......................................... converts ASDM to MS, ==importasdm() in casapy

- **buildmytasks** ................................. integrates user-provided tasks into casapy
  (see appendix G of the cookbook)
CASA tasks and tools

Two level user interface: Top level == **Tasks** *(104 in release 3.1, see taskhelp)*

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Tasks</th>
<th>Tasks</th>
<th>Tasks</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>accum</td>
<td>fixvis</td>
<td>importfits</td>
<td>pclean</td>
<td>sdscale</td>
</tr>
<tr>
<td>applycal</td>
<td>flagautocorr</td>
<td>importfitsidi</td>
<td>peel</td>
<td>sdsMOOTH</td>
</tr>
<tr>
<td>autoclean</td>
<td>flagcmd</td>
<td>importgmrt</td>
<td>plotants</td>
<td>sdstat</td>
</tr>
<tr>
<td>bandpass</td>
<td>flagdata</td>
<td>importoldasdm</td>
<td>plotcal</td>
<td>sdtpimaging</td>
</tr>
<tr>
<td>blcal</td>
<td>flagdata2</td>
<td>importuvfits</td>
<td>plotms</td>
<td>setjy</td>
</tr>
<tr>
<td>boxit</td>
<td>flagmanager</td>
<td>importvla</td>
<td>plotxy</td>
<td>simdata</td>
</tr>
<tr>
<td>browsetable</td>
<td>fluxscale</td>
<td>imregrid</td>
<td>polcal</td>
<td>slsearch</td>
</tr>
<tr>
<td>calstat</td>
<td>fringecal</td>
<td>imsmooth</td>
<td>rmtables</td>
<td>smoothcal</td>
</tr>
<tr>
<td>clean</td>
<td>ft</td>
<td>imstat</td>
<td>sdaverage</td>
<td>specfit</td>
</tr>
<tr>
<td>clearcal</td>
<td>gaincal</td>
<td>imtrans</td>
<td>sdbaseline</td>
<td>splottotable</td>
</tr>
<tr>
<td>clearplot</td>
<td>gencal</td>
<td>imval</td>
<td>sdcal</td>
<td>split</td>
</tr>
<tr>
<td>clearstat</td>
<td>hanningsmooth</td>
<td>imview</td>
<td>sdcoadd</td>
<td>ssolux</td>
</tr>
<tr>
<td>concat</td>
<td>imcollapse</td>
<td>listcal</td>
<td>sdfit</td>
<td>uvcontsub</td>
</tr>
<tr>
<td>csvclean</td>
<td>imcontsub</td>
<td>listhistory</td>
<td>sdflag</td>
<td>uvcontsub2</td>
</tr>
<tr>
<td>cvel</td>
<td>imfit</td>
<td>listobs</td>
<td>sdflagmanager</td>
<td>uvmodelfit</td>
</tr>
<tr>
<td>deconvolve</td>
<td>imhead</td>
<td>listvis</td>
<td>sdimaging</td>
<td>uvsub</td>
</tr>
<tr>
<td>exportasdm</td>
<td>immath</td>
<td>mosaic</td>
<td>sdimprocess</td>
<td>viewer</td>
</tr>
<tr>
<td>exportfits</td>
<td>immoments</td>
<td>msmoments</td>
<td>sdlist</td>
<td>vishead</td>
</tr>
<tr>
<td>exportuvfits</td>
<td>importasdm</td>
<td>msvview</td>
<td>sdmath</td>
<td>visstat</td>
</tr>
<tr>
<td>feather</td>
<td>importevla</td>
<td>newflagdata</td>
<td>sdplot</td>
<td>widefield</td>
</tr>
<tr>
<td>find</td>
<td>importevla2</td>
<td>oldsimdata</td>
<td>sdsave</td>
<td></td>
</tr>
</tbody>
</table>
CASA tasks and tools

Two level user interface: Top level == Tasks (104 in release 3.1, see taskhelp)

```plaintext
accum    fixvis    importfits    pclean    sdscale
applycal flagautocorr importfitsidi peel sdsmooth
autoclean flagflagdr importart plotants sdstat
boxit    flagmanager importvla    plotcal    sdtpimaging
browsetable fluxscale imregrid polcal smoothcal
bandpass flagdata2 importgmrt    plotants    specfit
blcal    flagcmd    importgmrt    plotants    splattable
browsetable fluxscale imregrid polcal smoothcal
```

CASA includes entire ATNF Spectral Analysis Package (ASAP) for single dish data analysis (see chapter 8 of the cookbook)

Initialise in CASA using

`asap_init`

Then have sd tool and sdtasks

```plaintext
deconvolve    imhead    listvis    sdaverage
exportasdm    immath    mosaic    sdbaseline
exportfits    immoments    mmsmoments    sdcal
exportuvfits importasdm    msvview    sdcont
feather    importevla newflagdata sdplot    wipimaging
find    importevla2 oldsimdata sdsave
```
CASA tasks and tools

Two level user interface: Top level == **Tasks**

documented in

a) *built-in documentation*

```
help <taskname>
pdoc <taskname>
<taskname> ?
```

b) *task reference web page*

`http://casa.nrao.edu/docs/taskref/TaskRef.html`

c) *cookbook*


*Note: there is also the CASAGuides wiki*

`http://casaguides.nrao.edu/`
CASA tasks and tools

Two level user interface: Top level == *Tasks*

- provide all basic analysis functionality for inexperienced users
  (without Python knowledge)

- provide the common analysis functionality for experienced users

- user interface optimised for interactive work with additional helper commands

<table>
<thead>
<tr>
<th>command</th>
<th>example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>default(clean)</td>
<td>- reset all input parameters</td>
</tr>
<tr>
<td>inp</td>
<td>inp</td>
<td>- show parameters of current task</td>
</tr>
<tr>
<td>go</td>
<td>go</td>
<td>- start current task</td>
</tr>
<tr>
<td>saveinputs</td>
<td>saveinputs(clean, parfile)</td>
<td>- store parameters in file</td>
</tr>
<tr>
<td>tget</td>
<td>tget(clean, parfile)</td>
<td>- restore parameters from file</td>
</tr>
</tbody>
</table>
CASA tasks and tools

Two level user interface: bottom level == **Tools** *(18 of them for release 3.1.0)*

- cb (calibrator)
- cp (cal plot)
- fg (flagger)
- ia (image analysis)
- im (imager)
- me (measures)
- mp (MS plot)
- ms (MS)
- qa (quanta)
- sm (simulation)
- tb (table)
- tp (table plot)
- vp (voltage patterns)
- cs (coord. sys.)
- at (atmosphere)
- sl (spectral lines)
- pl (pylab functions)
- sd (ASAP functions – run asap_init() to import into CASA)
CASA tasks and tools

Two level user interface: bottom level == Tools
documented in

a) built-in documentation

```
help <toolname>
help <toolname>.<methodname>
```

b) toolkit manual web page

`http://casa.nrao.edu/docs/CasaRef/CasaRef.html`
CASA tasks and tools

Two level user interface: bottom level == Tools

- contain all the special CASA functionality as Python objects
- not optimised for interactive use, behave just like Python objects

⇒ user calls methods of the tools:

<toolname>.<methodname>(<parameters>)

e.g., ms.open('mydata.ms')  - open an MS read-only with the MS tool

- anything possible with tasks is also possible using tools alone
- tasks are Python scripts using the tools + xml interface definition
CASA tasks and tools

Example: the task `flagautocorr(vis)` - flag the rows with autocorrelation data in an MS

```python
import os
from taskinit import *

def flagautocorr(vis=None):
    casalog.origin('flagautocorr')
    try:
        fg.clearflagselection(0)
        if ((type(vis)==str) & (os.path.exists(vis))):
            fg.open(vis)
        else:
            raise Exception, 'Visibility data set not found'
        fg.setdata()
        fg.setmanualflags(autocorrelation=True)
        fg.run()
        fg.done()
        ms.open(vis,nomodify=False)
        ms.writehistory(message='flagautocorr',origin='flagautocorr')
        ms.close()
    except Exception, instance:
        fg.done()
        print '*** Error ***',instance
```
CASA tasks and tools

Example: the task `flagautocorr(vis)` - flag the rows with autocorrelation data in an MS

```python
import os
from taskinit import *

def flagautocorr(vis=None):
    casalog.origin('flagautocorr')
    try:
        fg.clearflagselection(0)
        if ((type(vis)==str) & (os.path.exists(vis))):
            fg.open(vis)
        else:
            raise Exception, 'Visibility data set not found'
        fg.setdata()
        fg.setmanualflags(autocorrelation=True)
        fg.run()
        fg.done()
        ms.open(vis,nomodify=False)
        ms.writehistory(message='flagautocorr',origin='flagautocorr')
        ms.close()
    except Exception, instance:
        fg.done()
        print '*** Error ***',instance
```
CASA tasks and tools

Example: the task `flagautocorr(vis)` - *flag the rows with autocorrelation data in an MS*

```python
import os
from taskinit import *
def flagautocorr(vis=None):
    casalog.origin('flagautocorr')
    try:
        fg.clearflagselection(0)
        if ((type(vis)==str) & (os.path.exists(vis))):
            fg.open(vis)
        else:
            raise Exception, 'Visibility data set not found'
        fg.setdata()
        fg.setmanualflags(autocorrelation=True)
        fg.run()
        fg.done()
        ms.open(vis,nomodify=False)
        ms.writehistory(message='flagautocorr',origin='flagautocorr')
        ms.close()
    except Exception, instance:
        fg.done()
        print '*** Error ***',instance
```
CASA tasks and tools

Example: the task `flagautocorr(vis)` - flag the rows with autocorrelation data in an MS

```python
import os
from taskinit import *

def flagautocorr(vis=None):
    casalog.origin('flagautocorr')
    try:
        fg.clearflagselection(0)
        if ((type(vis)==str) & (os.path.exists(vis))):
            fg.open(vis)
        else:
            raise Exception, 'Visibility data set not found'

        fg.setdata()
        fg.setmanualflags(autocorrelation=True)
        fg.run()
        fg.done()
    except Exception, instance:
        fg.done()
        print '*** Error ***',instance

    ms.open(vis,nomodify=False)
    ms.writehistory(message='flagautocorr',origin='flagautocorr')
    ms.close()
```

MS tool
CASA global variables and task parameters

- casapy == Python shell with CASA extensions

- in casapy variables defined on the command line are global, i.e.
  scripts started with
    execfile('scriptfilename')
  have access to the variable values

Example:

script “myscript.py”:

    # test global variables
    print “value of a is “, a

command line:

    CASA <2>: a = 10
    CASA <3>: execfile('myscript.py')

output:

    value of a is  10
CASA global variables and task parameters

- `taskname`  
  = name of the current task which will be executed by `go`  

- every task parameter name behaves like a global variable  
  ⇒ e.g., if you specify the input parameter `field` in a command line

  ```
  field = 'NGC4826'
  ```

  then `field` will keep this value until you change it, for all tasks!

- the parameters of all tasks are coherently named so they can be shared:  
  `vis` - the input MS  
  `outputvis` - the output MS  
  `field` - the selection condition on the field table in an MS  
  `spw` - the selection condition on the spectral window table in an MS  
  ...

- get help on a parameter by typing `help par.<parametername>`
casapy command line options

Useful casapy command line options:

--logfile filename .......... use this filename instead of “casapy.log”
--nologger ............... don't start a logger
--log2term ............... print the log messages in the terminal
--nogui ............... don't permit any GUIs (implies --nologger)

-c filename ............... execute the Python script filename, then exit

Example: run a pipeline script non-interactively

casapy --nologger -c mypipeline.py
CASA internal and external visibility data formats

- Internal CASA visibility data format is the Measurement Set (MS)

- Presently supported input formats:

  ALMA:  ALMA Science Data Model (ASDM) - importasdm

  EVLA:  Science Data Model (SDM, essentially the same as the ASDM) - importevla

  VLA:  VLA archive format - importvla

  EVN, eMERLIN et al.: FITS-IDI - importfitsidi

  and the general transport format  uvfits - importuvfits
CASA internal and external visibility data formats

The **MS**

- relational database system with fixed structure made from **CASA Tables**

- consists of a main *table* with 15 required *sub-tables* + several optional ones

- uses OS directory structure (need to copy with `cp -R`, remove with `rm -r`)

- visibilities stored in the MAIN table

- no compression

- manipulate an MS with the `ms` and the `tb` tool or with `browsetable()`

- during processing, CASA may add “scratch columns” to the MS main table
The **ASDM (ALMA) and SDM (EVLA)**

- relational database system with fixed structure

- consists of set of up to 56 tables (also observatory setup information!)

- uses OS directory structure (need to copy with `cp -R`, remove with `rm -r`)

- visibilities accessed via the MAIN table

- on disk, bulk data stored in Binary Large Objects (BLOBs) using MIME format

- remaining data stored in XML files

- import into CASA using the task `importasdm` or `importevla`

- new in release 3.1: `exportasdm` (MS to ASDM conversion, e.g. for simulations)
CASA calibration tables

Calibration tables for visibility data

- CASA tables with defined columns and subtables

- contain calibration solutions and/or parametrisations

- serve communication between calibration tasks and storage of final result
CASA images and fits

Two formats for images in CASA:

a) CASA images
   - based on CASA Tables
   - proper name in casacore: PagedImage
   - approach: make the image accessible as a multi-dimensional lattice
   - arbitrary size on disk, paged into memory

b) FITS
   - translation to/from CASA images by importfits and exportfits tasks
   - follows the IAU FITS standard v3.0 (2008)
   - special additions for compatibility with AIPS for spectral image cube axes
History

- All CASA data formats contain history or log sub-tables

- Access via browsetable() or special tasks/tool methods:

  MS:  listhistory() or ms.listhistory()
  Image: ia.history()
In case of problems ...

What to do if you encounter a problem with CASA:
If the cookbook and the release notes don't help, go to http://help.nrao.edu/
Don't have an account? Register at http://my.nrao.edu

A) You think you have found a bug in CASA
- Try to reproduce your problem, ideally by writing a Python script
- Put your test data (if needed) and the script on some web or ftp server where it can remain for at least several months.
- File a helpdesk ticket including a short description of the problem and links to the script and the data.
- Please mention CASA version and your operating system (32 bit or 64 bit?)

B) You don't know how to perform a certain analysis task in CASA
- Read the documentation described earlier in this talk.
- If you can't make progress, then, as in (A) try to prepare a script for your analysis up to the point where you don't know how to go further.
- File a helpdesk ticket including the script and a description of what you would like to achieve.
In case of problems ...

What to do if you encounter a problem with CASA:
If the cookbook and the release notes don't help, go to http://help.nrao.edu/
Don't have an account? Register at http://my.nrao.edu

A) You think you have found a bug in CASA
- Try to reproduce your problem, ideally by writing a Python script which will demonstrate the problem.
- Put your test data (if needed) and the script on some web or ftp server where it can remain for at least several months.
- File a helpdesk ticket including a short description of the problem and links to the script and the data.
- Please mention CASA version and your operating system (32 bit or 64 bit?)

B) You don't know how to perform a certain analysis task in CASA
- Read the documentation described earlier in this talk.
- If you can't make progress, then, as in (A) try to prepare a script for your analysis up to the point where you don't know how to go further.
- File a helpdesk ticket including the script and a description of what you would like to achieve.

First consult the knowledge base!
In case of problems ...

What to do if you encounter a problem with CASA and you can't find the solution in the documentation or the knowledge base:

A) You think you might have found a bug in CASA
   - Try to reproduce your problem, ideally by writing a Python script which will demonstrate the problem.
   - Put your test data (if needed) on some web or ftp server where it can remain for at least several months.
   - File a helpdesk ticket including the script, a short description of the problem, and the URL of the data.
   - Need to mention CASA version and your operating system (32 bit or 64 bit?)

B) You don't know how to perform a certain analysis task in CASA
   - If you can't make progress, then, as in (A) try to prepare a script for your analysis up to the point where you don't know how to go further.
   - File a helpdesk ticket including the script and a description of what you would like to achieve.
In case of problems ...

How to file a helpdesk ticket at help.nrao.edu:

[Image of helpdesk interface]

Click here
In case of problems ...

How to file a helpdesk ticket at help.nrao.edu:

indicate CASA department
In case of problems

How to file a helpdesk ticket:

Where does your data come from? (identify necessary expertise)

Where do you come from? (who is responsible?)

What OS and CASA version are you using? (for reproducing your problem)

Give at least a description of what you are trying to do and the URL of your test data if needed to reproduce your problem. Also quote error messages.

Upload a Python script which demonstrates your problem.
In case of problems ...

Your ticket has been submitted to our department successfully. One of our support agents will get back to you with more information shortly.

**Ticket Information**
- Ticket ID: CBM-918027
- Department: Data Processing (CASA)
- Full Name: Dirk Petry
- E-mail: dpetry@eso.org
- Priority: Default

**CASA**
- Data Source: ALMA
- Region: Europe
- Operating System: RedHat 5-32bit
- Version: 3.0.0

(confirmation email with ID in subject will arrive from do-not-reply@nrao.edu)