

The *CASA* software package

Dirk Petry (ESO/ARC), February 2012



Outline

- What is *CASA*?
 - main features
- Who develops *CASA*?
 - development team
- What are the main requirements and how does *CASA* meet them?
 - design and implementation
- *CASA* status and release plans
- How does *CASA* look and feel?
 - installation and the typical analysis session

CASA main features

- **CASA = Common Astronomy Software Applications**
- **Development started in the 90s as the next generation of AIPS**
- **Refocussed in 2003 to be *the ALMA/EVLA analysis package***
- **Has the intention to be a *general software package to reduce both interferometer and single-dish data***
- **Internally consists of two parts:**

User interface, higher-level analysis routines, viewers
= *casa non-core*



General physical and astronomical utilities, infrastructure
= *casacore*

- **Implements the “Measurement Equation” (Hamaker, Bregman & Sault 1996)**
- **Internal data format is the “Measurement Set” (Kemball & Wieringa 2000)**
- **more than 1.5 Million lines of code (mostly C++)**
- **In public release under GNU Public License since December 2009**

CASA – development team



CASA Developers Meeting, NRAO, Socorro, May 2010

CASA – development team



Since mid 2008, two CASA developers at ESO, since Sept. 2009 three



CASA – development team



**Originally only developed at NRAO (Socorro, NM) ,
now approx. 19 FTE developers are at work at**

US (NRAO and others): 12
Japan (NAOJ): 3
Europe (ESO and others): 4

+ 1 CASA manager (NRAO Socorro) = Jeff Kern

+ 1 CASA Project Scientist (NRAO Socorro) = Jürgen Ott

+ 1 ALMA CASA subsystem scientist (NRAO CV) = Crystal Brogan

+ 1 EVLA CASA subsystem scientist (NRAO Socorro) = Steve Myers

+ 2 ARC CASA leads (ESO: D. Petry, NAOJ: A. Higuchi)

+ a few 5% FTEs at ASTRON, ATNF, and other places

Also involved:

ALMA Computing Managers = B. Glendenning (NRAO), E. Schmid, P. Ballester (ESO)



CASA design and implementation



Overall architecture:

- 1) A data structure
- 2) A set of data import/export facilities
- 3) A set of tools for data access, display, and editing
- 4) A set of tools for science analysis
- 5) A set of high-level analysis procedures (“tasks”)
- 6) A programmable command line interface with scripting
- 7) Documentation



CASA design and implementation



Overall architecture:

1) A data structure

Tables: Images, Caltables, and the Measurement Set (MS)

2) A set of data import/export facilities

the so-called *fillers*: (ASDM, UVFITS, FITS-IDI, VLA archive) → MS, FITS → Image

3) A set of tools for data access, display, and editing

tools to load/write data into/from casacore data types,

Qt-based table browser, viewer, and (beta) x/y plotter, *matplotlib*-based x/y plotter

4) A set of tools for science analysis

built around the *Measurement Equation* (developed in 1996),

a toolkit for radio astronomical calibration, imaging, and simulation

5) A set of high-level analysis procedures (“tasks”)

user-friendly implementations of the solutions for all common analysis problems

6) A programmable command line interface with scripting

Python (augmented by *IPython*) gives a MATLAB-like interactive language

7) Documentation

an extensive cookbook (500 pages) + documentation through help commands

(help, ?, pdoc) + online help pages, See <http://casa.nrao.edu/>

Overall architecture:

1) A data structure

a) → *Tables: Images, Caltables, and the Measurement Set (MS)*

2) A set of data import/export facilities

the so-called *fillers*: (ASDM, UVFITS, FITS-IDI, VLA archive) → MS, FITS → Image

3) A set of tools for data access, display, and editing

tools to load/write data into/from casacore data types,

Qt-based table browser, viewer, and (beta) x/y plotter, *matplotlib*-based x/y plotter

4) A set of tools for science analysis

b) → built around the *Measurement Equation* (developed in 1996),

a toolkit for radio astronomical calibration, imaging, and simulation

5) A set of high-level analysis procedures (“tasks”)

user-friendly implementations of the solutions for all common analysis problems

6) A programmable command line interface with scripting

c) → *Python* (augmented by *IPython*) gives a MATLAB-like interactive language

7) Documentation

an extensive cookbook (500 pages) + documentation through help commands

(help, ?, pdoc) + online help pages, See <http://casa.nrao.edu/>



CASA design and implementation



CASA special features:

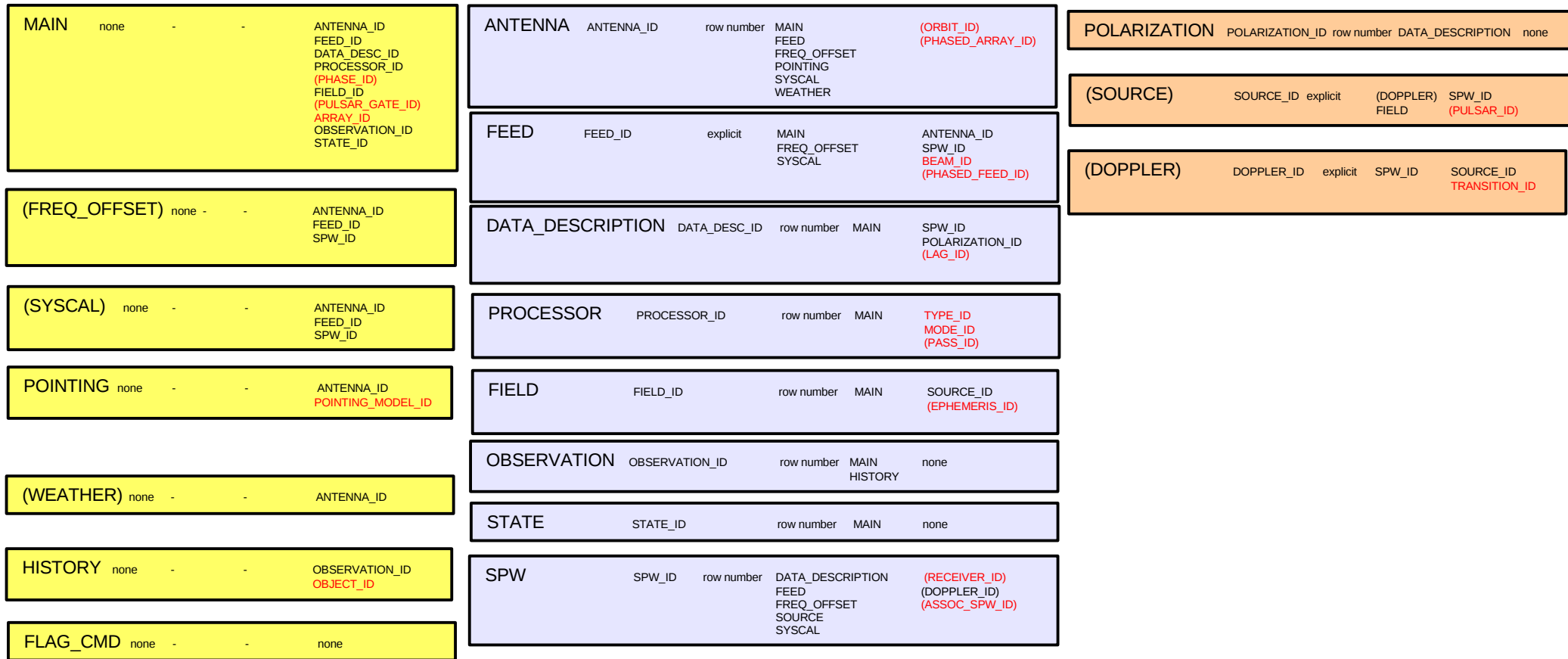
a) the *Measurement Set* (MS)

- developed by Cornwell, Kemball, & Wieringa between 1996 and 2000
- designed to store both interferometry (multi-dish) and single-dish data
- supports (in principle) any setup of radio telescopes
- supports description and processing of the data via the Measurement Equation
- fundamental storage mechanism: *CASA Tables* (inspired by *MIRIAD*)
- *MS = table for radio telescope data (visibilities) + auxiliary sub-tables*



CASA design and implementation

The Measurement Set



Legend:

[Table Name]	[Key defined in this table]	[key definition method]	[referenced by]	[referenced keys] (optional) reference to table outside the MS definition
--------------	-----------------------------	-------------------------	-----------------	--

- Level 1: Tables not referenced by other tables
- Level 2: Tables referenced by level 1
- Level 3: Tables referenced by level 2

CASA special features:

- b) A toolkit for radio astronomical calibration, imaging, and simulation built around the *Measurement Equation* (Hamaker, Bregman, & Sault 1996 + Sault, Hamaker, & Bregman 1996)

$$\vec{V}_{ij} = \vec{M}_{ij} \vec{B}_{ij} \vec{G}_{ij} \vec{D}_{ij} \int \vec{E}_{ij} \vec{P}_{ij} \vec{T}_{ij} \vec{F}_{ij} S \vec{I}_v(l, m) e^{-i2\pi(u_{ij}l + v_{ij}m)} dl dm + \vec{A}_{ij}$$

where

the vectors are: V = observed visibility = $f(u, v)$, I = Image to be derived,

A = additive baseline-based error component

the matrices are: M = multiplicative, baseline-based error component

B = bandpass response

G = generalised electronic gain

D = polarisation leakage

E = antenna voltage pattern, i.e. primary beam effects

P = parallactic angle dependence

T = tropospheric effects

F = ionospheric Faraday rotation

S = mapping of I to the polarization basis of the observation

other variables and indices are:

l, m = image plane coordinates, i, j = telescope ID pairs = baseline, u, v = Fourier plane coordinates

CASA special features:

- b) A toolkit for radio astronomical calibration, imaging, and simulation built around the *Measurement Equation* (Hamaker, Bregman, & Sault 1996 + Sault, Hamaker, & Bregman 1996)
(continued)

Assuming, e.g., independence of the matrices from (l,m) , the ME can be solved for individual calibration components.

$$\vec{V}_{ij}^{obs} = \vec{B}_{ij} \vec{G}_{ij} \vec{D}_{ij} \vec{P}_{ij} \vec{T}_{ij} \vec{F}_{ij} \vec{V}_{ij}^{ideal}$$

ideal visibility known from calibrator source

⇒ have set of linear equations.

The actual calculation of the component is then a χ^2 minimization.

(For wide-field imaging the above assumption doesn't hold and the solution is more complex but still possible.)

CASA contains a set of **solvers** for the different calibration components.



CASA design and implementation



CASA special features:

b) [A toolkit for radio astronomical calibration, imaging, and simulation](#) (continued)

Imaging in CASA: Combinations of Major and Minor Cycle Algorithms

Imaging (Major Cycle):

- 1) Standard (no dir.-dep. effects, uv-grid sampling uses convolutional regridding)
- 2) with dir.-dep. effects:
 - a) W-term (image domain faceting, uv domain faceting, W projection)
 - b) PB correction (image domain, A projection)
 - c) Pointing Offset correction by phase gradient
 - d) Mosaicing (linear (separate) deconvolution, joined deconv. of combined dirty images, mosaicing by regridding all uv data onto one grid)

Deconvolution (Minor Cycle):

- 1) CLEAN (delta function model)
- 2) MS-CLEAN (blob model)
- 3) MSMFS CLEAN (model of blobs with polynomial spectrum)
- 4) MEM (maximum entropy method using prior image and delta function model)

see nice overview compiled by Urvashi Rau: <https://safe.nrao.edu/wiki/bin/view/Software/AlgorithmList>

CASA special features:

b) [A toolkit for radio astronomical calibration, imaging, and simulation](#) (continued)

A sophisticated radio-astronomical data simulator: *simdata*

- **Create Measurement Sets of simulated data**

(and for convenience: analyse the simulated MS to create simulated image)

- **Input:**

a) **FITS image**

b) **“antenna list” file describing your interferometer (incl. site name)**

sites: `browsetable(os.getenv("CASAPATH").split(' ')[0]+"/data/geodetic/Observatories")`

arrays: `ls os.getenv("CASAPATH").split(' ')[0]+"/data/alma/simmos/"`

c) **observation setup parameters**

(central direction, time, mosaicing, spectral, integration time, etc.)

d) **corrupting effect parameters**

(thermal noise from atmosphere and receiver)

- *uses realistic site-dependent troposphere model*

- *knows about ALMA and EVLA receiver parameters*

- *phase noise and gain drift can be applied to the MS later via CASA tools*

e) **for convenience: clean task parameters for output image creation**



CASA design and implementation



CASA special features:

c) A programmable command line interface with scripting

Framework Architecture of 20 tools bound to Python (augmented by IPython)

```
at : ATM - atmosphere simulation library
cb : Calibration utilities
cp : Calibration solution plotting utilities
cs : Coordinate system utilities
fg : Flagging/Flag management utilities
ia : Image analysis utilities
im : Imaging utilities
me : Measures utilities
ms : MeasurementSet (MS) utilities
mp : MS plotting (data (amp/phase) versus other quantities)
pm : PlotMS utilities
rg : Region manipulation utilities
tb : Table utilities (selection, extraction, etc)
tp : Table plotting utilities
qa : Quanta utilities
sl : Spectral line import and search
sm : Simulation utilities
vp : Voltage pattern/primary beam utilities
pl : pylab (matplotlib) functionality
sd : (after running asap_init()) Single dish utilities
```



CASA design and implementation



CASA special features:

c) **A programmable command line interface with scripting**

(continued)

Python (augmented by IPython)

Gives features such as

- tab completion
- autoparenthesis
- command line numbering
- access to OS, e.g.
 - Lines starting with '!' go to the OS.
 - `a = !ls *.py` to capture the output of 'ls *.py'.
 - `!cmd $myvar` expands Python var `myvar` for the shell.
- history
- `execfile()`
- comfortable help



CASA design and implementation



CASA special features:

- c) A programmable command line interface with scripting
(continued)

In addition to toolkit: high-level tasks for the standard user

tasks (implemented in Python) → **tools** (implemented in C++)

e.g. the task *importfits* is based on the tool *ia* (image analysis):

```
#Python script
casalog.origin('importfits')
ia.fromfits(imagename, fitsimage, whichrep, whichhdu, zeroblanks, overwrite)
ia.close()
```

CASA 3.3 comes with 109 implemented tasks.



CASA status

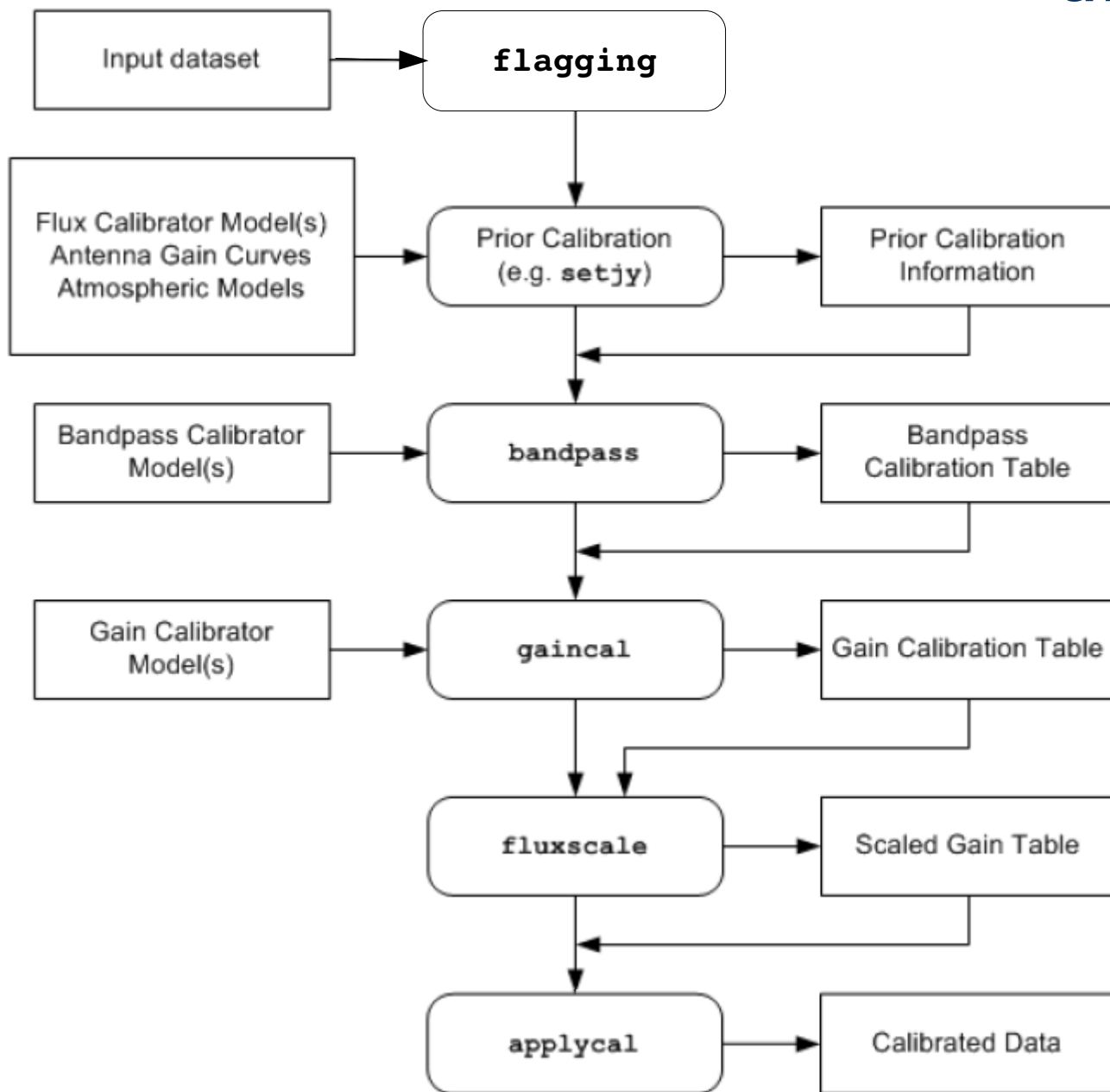


- Since Dec 2009 in public release under GPL = anybody can download, no warranty (see <http://casa.nrao.edu>), limited support (help desk, needs registration)
- Tutorials for the user community regularly given
- The first public release was CASA 3.0.0 (Dec 2009), release 3.3.0 out Nov 2011
- Development platforms: Linux (RHEL) + Mac OS X
- Supported platforms (binary distribution): RHEL, Fedora, openSuSE, Ubuntu, Mac OS X (10.6 and 10.7, no longer 10.5)
- Code kept in *svn* repository at NRAO, Socorro
- Have approx. 4300 modules, > 1.5E6 lines of code, > 1E6 lines of comments
- The core functionality (*casacore*, also available at <http://code.google.com/p/casacore/>) is also used by other projects
- *Hot topics*:
 - Support for High Performance Computing and Parallelisation
 - Advanced Imaging: wide fields, continuum imaging over wide spectral ranges

How does CASA look and feel?

A typical analysis session

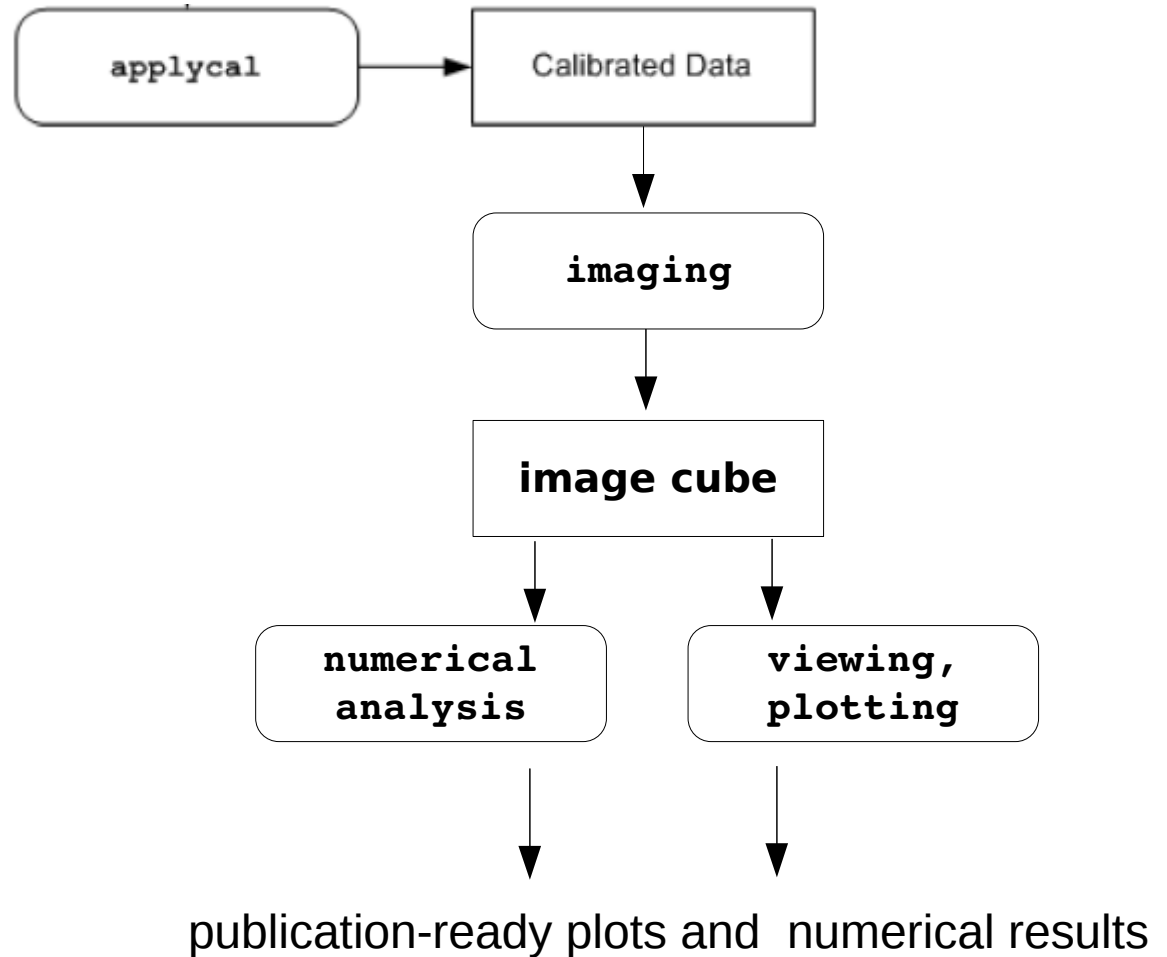
Part 1: flagging and calibration



How does CASA look and feel?

A typical analysis session

Part 2: imaging and
image analysis





How does CASA look and feel?



Installation - CASA comes as a tgz-file for Linux or a dmg-file for Mac OS-X

See “Obtaining CASA” link on <http://casa.nrao.edu/>

Download latest version at

https://svn.cv.nrao.edu/casa/linux_distro

or

https://svn.cv.nrao.edu/casa/osx_distro

Linux:

Unpack tgz file in a location of your choice.
cd into the created casapy directory.

```
export PATH=$PWD:$PATH
```

Mac OS-X:

Open the CASA disk image file (if your browser does not do so automatically).

Read the README for special instructions!

Drag the CASA application to the Applications folder of your hard disk.

Eject the CASA disk image.

Double-click the CASA application to run it for the first time.

Distribution contains all necessary libraries and Python. No external packages needed.

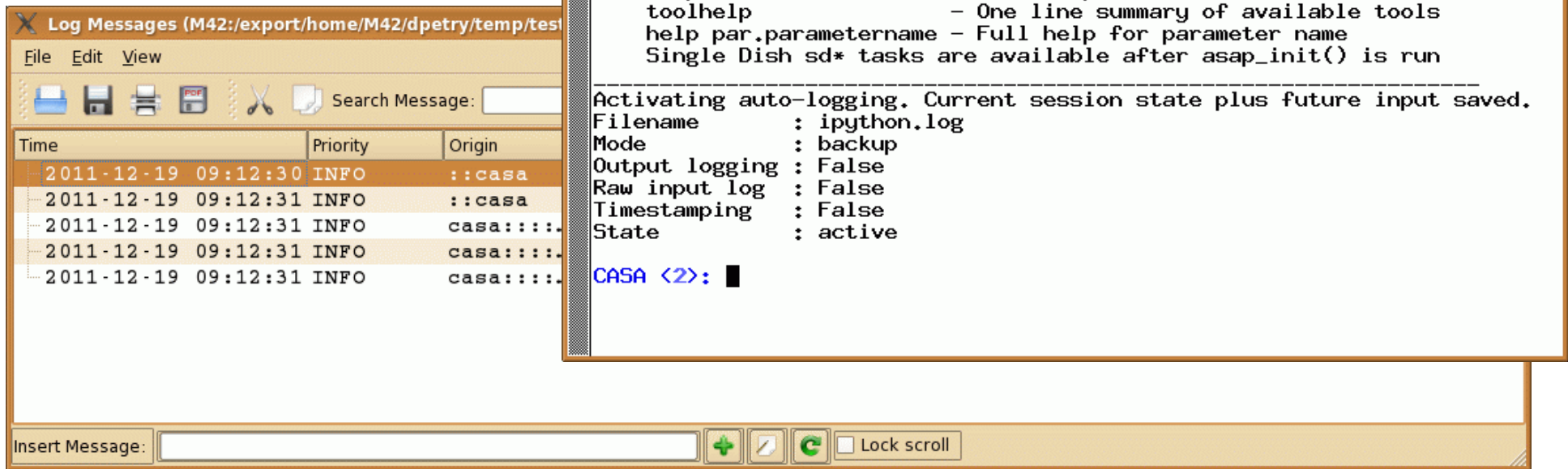
Exception: Ubuntu users should install Xvfb!

How does CASA look and feel?

Pictures from a typical analysis session

- 1) Startup:
open terminal and start *casapy*

Basic help tools are listed and the logger window is opened.



The image shows two overlapping windows from a Linux desktop environment. The top window is a terminal titled 'bash' showing the output of the 'casapy' command. The bottom window is a 'Log Messages' window showing a list of log entries.

Terminal Window Output:

```

bash-3.2$ casapy
CASA Version 3.3.0 (r16856)
Compiled on: Thu 2011/11/03 18:34:23 UTC

-----
For help use the following commands:
tasklist           - Task list organized by category
taskhelp           - One line summary of available tasks
help taskname      - Full help for task
toolhelp           - One line summary of available tools
help par.parametername - Full help for parameter name
Single Dish sd* tasks are available after asap_init() is run

-----
Activating auto-logging. Current session state plus future input saved.
Filename          : ipython.log
Mode               : backup
Output logging    : False
Raw input log     : False
Timestamping      : False
State              : active

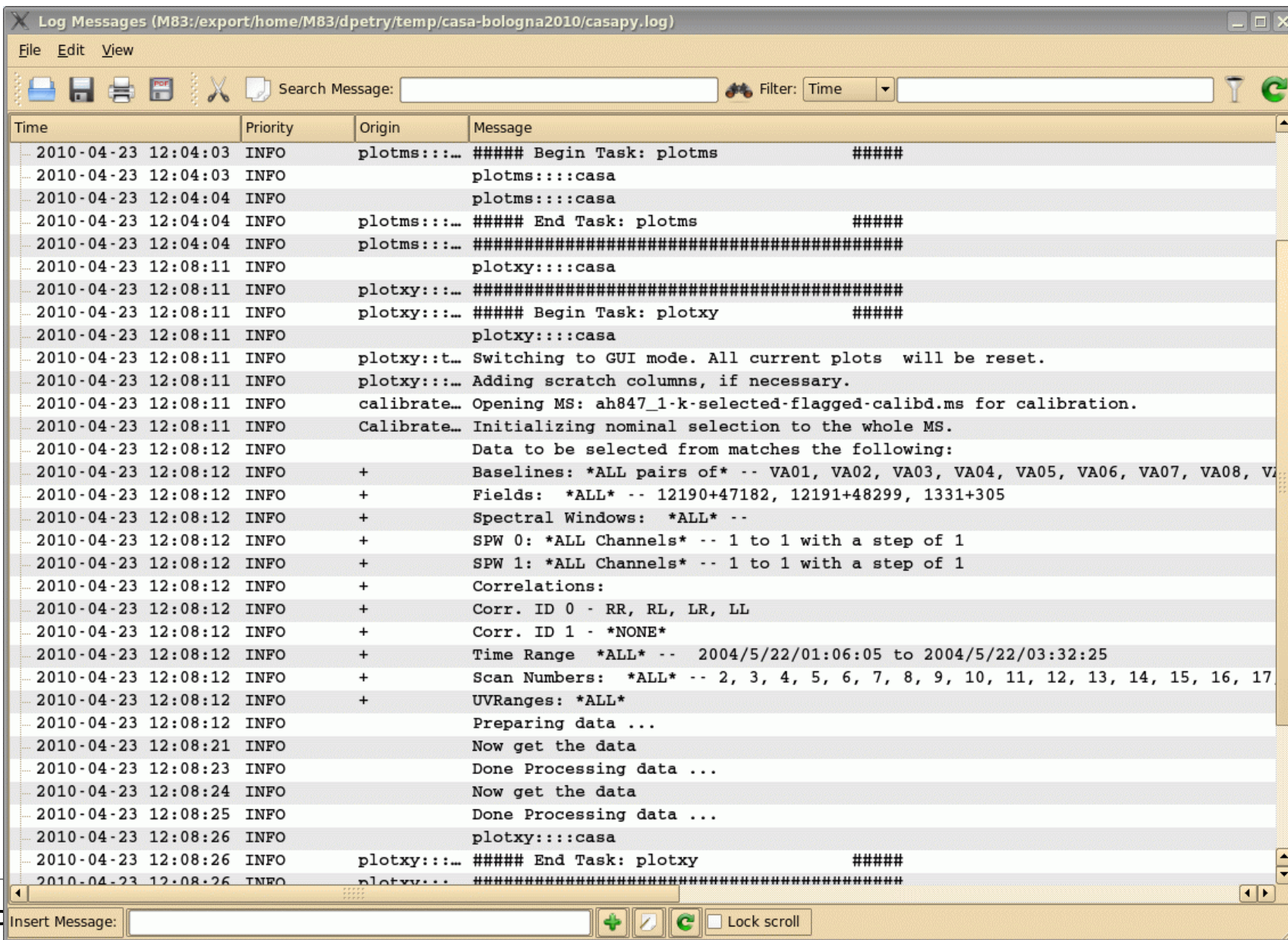
CASA <2>: █
  
```

Log Messages Window:

Time	Priority	Origin
2011-12-19 09:12:30	INFO	::casa
2011-12-19 09:12:31	INFO	::casa
2011-12-19 09:12:31	INFO	casa:::..
2011-12-19 09:12:31	INFO	casa:::..
2011-12-19 09:12:31	INFO	casa:::..

The CASA user interface

The logger provides functionality for monitoring and debugging command execution.



Log Messages (M83:/export/home/M83/dpetry/temp/casa-bologna2010/casapy.log)

Time	Priority	Origin	Message
2010-04-23 12:04:03	INFO	plotms::...	##### Begin Task: plotms #####
2010-04-23 12:04:03	INFO		plotms:::casa
2010-04-23 12:04:04	INFO		plotms:::casa
2010-04-23 12:04:04	INFO	plotms::...	##### End Task: plotms #####
2010-04-23 12:04:04	INFO	plotms::...	#####
2010-04-23 12:08:11	INFO		plotxy:::casa
2010-04-23 12:08:11	INFO	plotxy::...	#####
2010-04-23 12:08:11	INFO	plotxy::...	##### Begin Task: plotxy #####
2010-04-23 12:08:11	INFO	plotxy::...	plotxy:::casa
2010-04-23 12:08:11	INFO	plotxy::t...	Switching to GUI mode. All current plots will be reset.
2010-04-23 12:08:11	INFO	plotxy::...	Adding scratch columns, if necessary.
2010-04-23 12:08:11	INFO	calibrate...	Opening MS: ah847_1-k-selected-flagged-calibd.ms for calibration.
2010-04-23 12:08:11	INFO	Calibrate...	Initializing nominal selection to the whole MS.
2010-04-23 12:08:12	INFO		Data to be selected from matches the following:
2010-04-23 12:08:12	INFO	+	Baselines: *ALL pairs of* -- VA01, VA02, VA03, VA04, VA05, VA06, VA07, VA08, VA09, VA10, VA11, VA12, VA13, VA14, VA15, VA16, VA17, VA18, VA19, VA20, VA21, VA22, VA23, VA24, VA25, VA26, VA27, VA28, VA29, VA30, VA31, VA32, VA33, VA34, VA35, VA36, VA37, VA38, VA39, VA40, VA41, VA42, VA43, VA44, VA45, VA46, VA47, VA48, VA49, VA50, VA51, VA52, VA53, VA54, VA55, VA56, VA57, VA58, VA59, VA60, VA61, VA62, VA63, VA64, VA65, VA66, VA67, VA68, VA69, VA70, VA71, VA72, VA73, VA74, VA75, VA76, VA77, VA78, VA79, VA80, VA81, VA82, VA83, VA84, VA85, VA86, VA87, VA88, VA89, VA90, VA91, VA92, VA93, VA94, VA95, VA96, VA97, VA98, VA99, VA100
2010-04-23 12:08:12	INFO	+	Fields: *ALL* -- 12190+47182, 12191+48299, 1331+305
2010-04-23 12:08:12	INFO	+	Spectral Windows: *ALL* --
2010-04-23 12:08:12	INFO	+	SPW 0: *ALL Channels* -- 1 to 1 with a step of 1
2010-04-23 12:08:12	INFO	+	SPW 1: *ALL Channels* -- 1 to 1 with a step of 1
2010-04-23 12:08:12	INFO	+	Correlations:
2010-04-23 12:08:12	INFO	+	Corr. ID 0 - RR, RL, LR, LL
2010-04-23 12:08:12	INFO	+	Corr. ID 1 - *NONE*
2010-04-23 12:08:12	INFO	+	Time Range *ALL* -- 2004/5/22/01:06:05 to 2004/5/22/03:32:25
2010-04-23 12:08:12	INFO	+	Scan Numbers: *ALL* -- 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100
2010-04-23 12:08:12	INFO	+	UVRanges: *ALL*
2010-04-23 12:08:12	INFO		Preparing data ...
2010-04-23 12:08:21	INFO		Now get the data
2010-04-23 12:08:23	INFO		Done Processing data ...
2010-04-23 12:08:24	INFO		Now get the data
2010-04-23 12:08:25	INFO		Done Processing data ...
2010-04-23 12:08:26	INFO		plotxy:::casa
2010-04-23 12:08:26	INFO	plotxy::...	##### End Task: plotxy #####
2010-04-23 12:08:26	INFO	plotxy::...	#####

Insert Message: Lock scroll

Pictures from a typical analysis session

2) enter commands in a
MATLAB-like environment

recall previous settings

list present settings
for current task
(includes parameter
verification)

```

dpetry@pc014720:~/temp/radio-analysis/cqtau+mwc480 - Shell - Konsole
Session Edit View Bookmarks Settings Help

CASA <15>: fluxscale(vis='AT352_A071103-K', caltable='AT352_A071103-K-gain', fluxtable
='0', transfer='1')

CASA <16>: applycal(vis='AT352_A071103-K', gaintable='AT352_A071103-K-gain', field='2'

CASA <17>: tget clean
-----> tget(clean)
Restored parameters from file clean.last

CASA <18>: inp
-----> inp()
# clean :: Deconvolve an image with selected algorithm
vis                = 'AT352_A071103-K' # name of input visibility file
imagenam          = 'cqtau-3-target'   # Pre-name of output images
field             = '2'                # Field Name
spw               = ''                # Spectral windows:channels: '' is all
selectdata       = False              # Other data selection parameters
mode              = 'mfs'             # Type of selection (mfs, channel, velocity,
niter             = 500                # Maximum number of iterations
gain              = 0.1                # Loop gain for cleaning
threshold         = '0.0mJy'         # Flux level to stop cleaning. Must include
psfmode          = 'clark'            # method of PSF calculation to use during min
imagermode       = ''                 # Use csclean or mosaic. If '', use psfmode
multiscale        = []                # set deconvolution scales (pixels), default:
interactive       = True               # use interactive clean (with GUI viewer)
npercycle         = 100                # Number of iterations before interactive pro

mask              = []                # cleanbox(es), mask image(s), and/or region(
imsize            = [512, 512]        # x and y image size in pixels, symmetric for
cell              = ['0.03arcsec', '0.03arcsec'] # x and y cell size. default unit
phasecenter       = ''                # Image phase center: position or field index
restfreq          = ''                # rest frequency to assign to image (see help
  
```


Pictures from a typical analysis session

3) where needed, tools have GUIs:

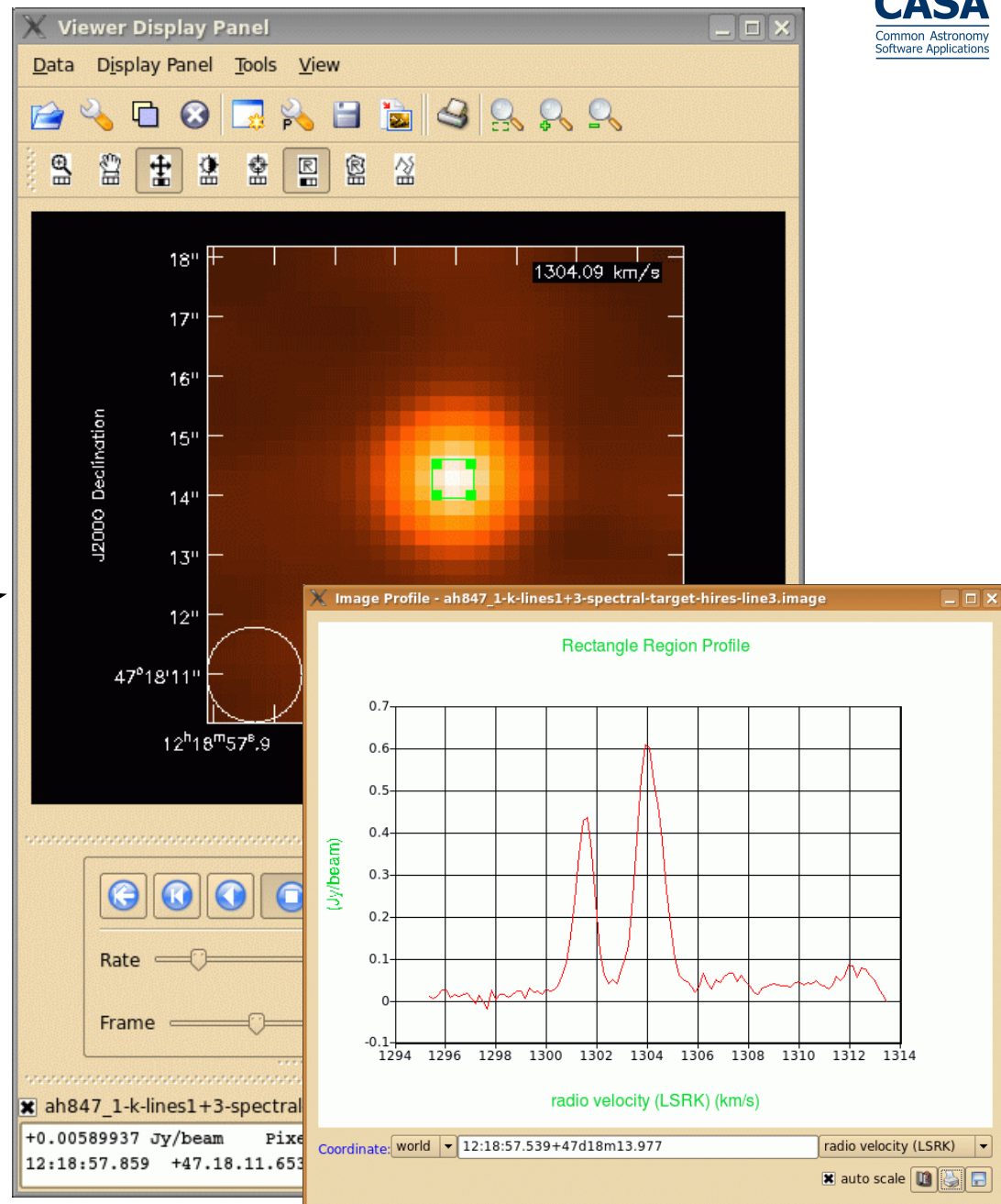
plotxy, plotcal, browsetable,
viewer, clean, plotms

(started in separate threads)

The **viewer** is a powerful multi-function tool for data selection and visualization.

Uses Qt widget set
(but 80% independent)

Rendering based on pgplot



A typical analysis session

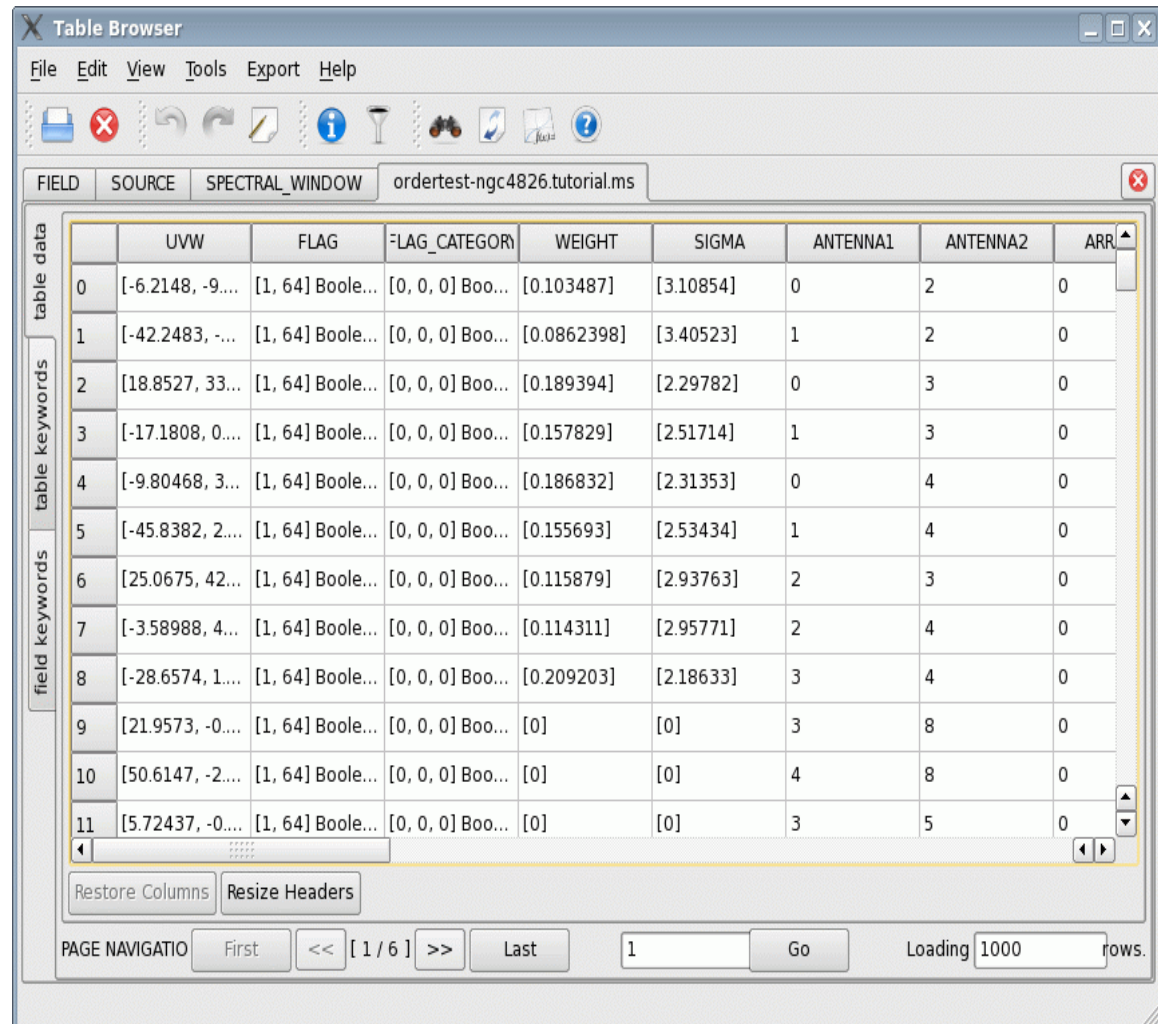
3) where needed, tools have GUIs:

plotxy, plotcal, browsetable,
viewer, clean, plotms

(started in separate threads)

browsetable permits you to
explore any CASA table, e.g.
Measurement Sets

Also Qt-based.



FIELD	SOURCE	SPECTRAL_WINDOW	UVW	FLAG	FLAG_CATEGORY	WEIGHT	SIGMA	ANTENNA1	ANTENNA2	ARRIVAL_TIME
0			[-6.2148, -9...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.103487]	[3.10854]	0	2	0
1			[-42.2483, -...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.0862398]	[3.40523]	1	2	0
2			[18.8527, 33...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.189394]	[2.29782]	0	3	0
3			[-17.1808, 0...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.157829]	[2.51714]	1	3	0
4			[-9.80468, 3...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.186832]	[2.31353]	0	4	0
5			[-45.8382, 2...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.155693]	[2.53434]	1	4	0
6			[25.0675, 42...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.115879]	[2.93763]	2	3	0
7			[-3.58988, 4...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.114311]	[2.95771]	2	4	0
8			[-28.6574, 1...	[1, 64] Boole...	[0, 0, 0] Boo...	[0.209203]	[2.18633]	3	4	0
9			[21.9573, -0...	[1, 64] Boole...	[0, 0, 0] Boo...	[0]	[0]	3	8	0
10			[50.6147, -2...	[1, 64] Boole...	[0, 0, 0] Boo...	[0]	[0]	4	8	0
11			[5.72437, -0...	[1, 64] Boole...	[0, 0, 0] Boo...	[0]	[0]	3	5	0

A typical analysis session

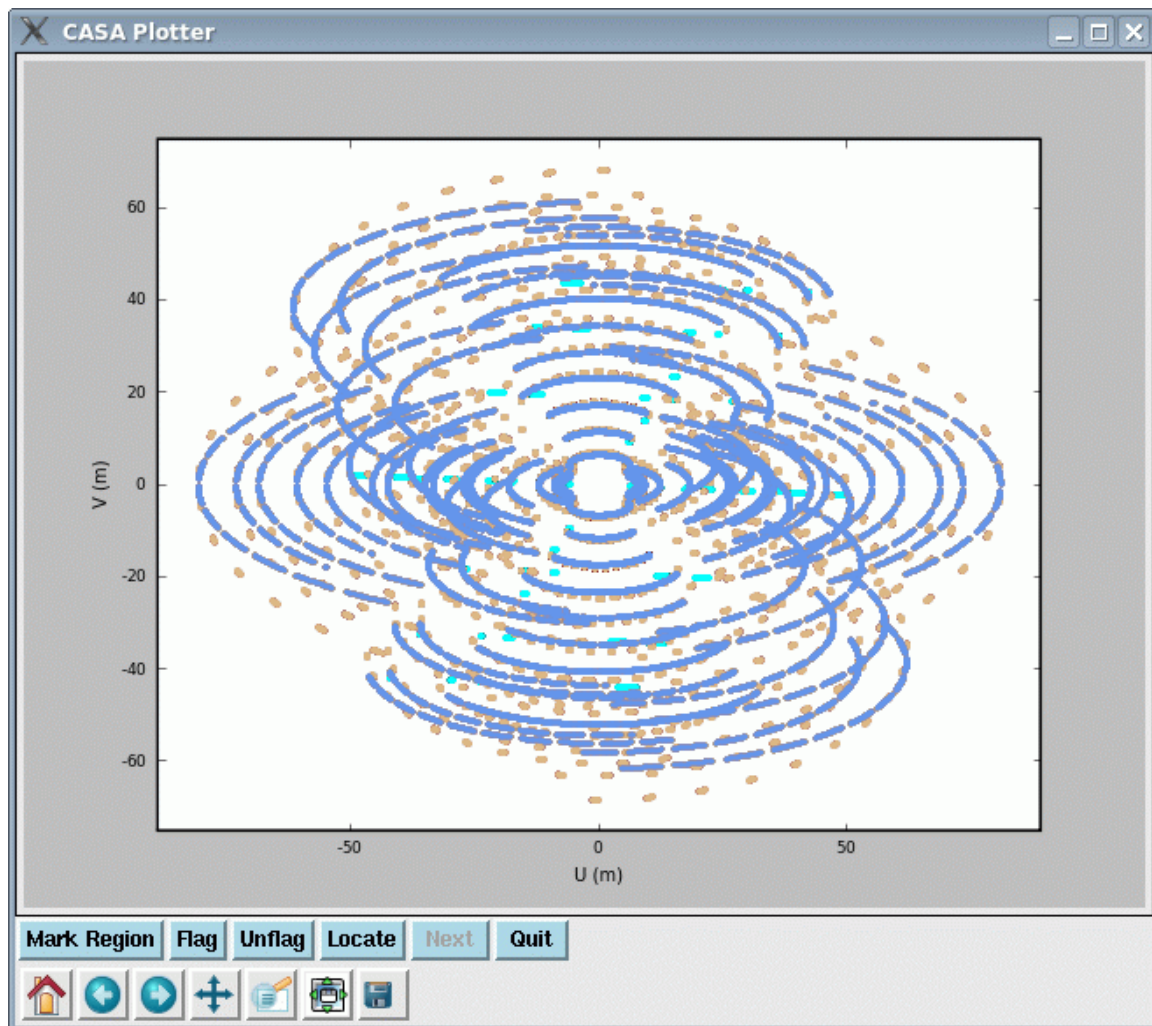
3) where needed, tools have GUIs:

plotxy, plotcal, browsetable,
viewer, clean, plotms

(started in separate threads)

plotxy is a specialised tool
for diagnostic plots and
data selection

To be phased out.



A typical analysis session

3) where needed, tools have GUIs:

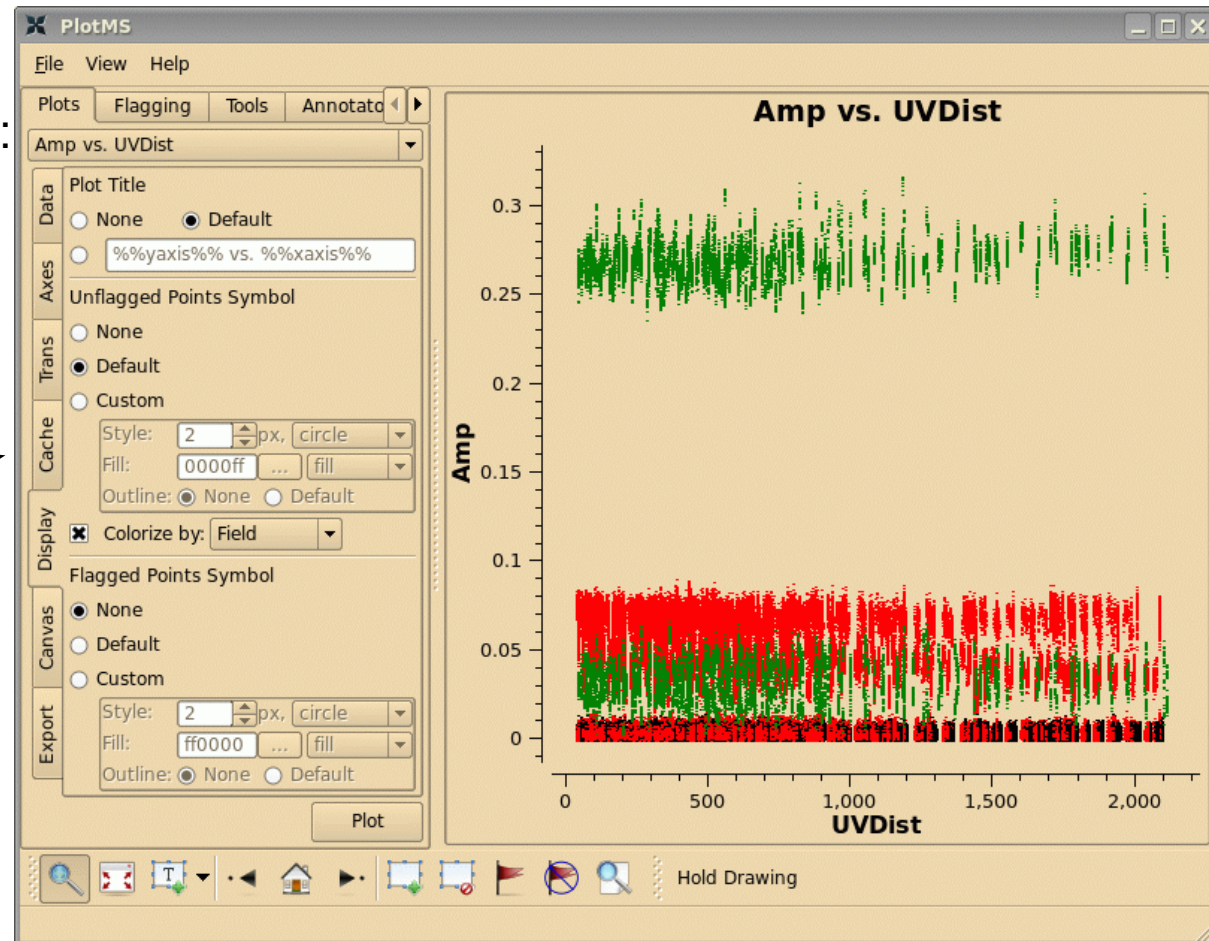
plotxy, plotcal, browsetable,
viewer, clean, plotms

(started in separate threads)

plotms is going to replace
plotxy. Release 3.3 contains
first useful version.

plotms is Qt-based and much
faster than plotxy.

Uses generic plotting class
which in turn uses **Qwt**.



A typical analysis session

4) A sophisticated radio-astronomical data simulator: *simdata*

```

CASA <48>: inp
-----> inp()
# simdata :: mosaic simulation task:
project      = 'Itziar_full'      # root for output file
modifymodel  = True              # modify model image
skymodel     = 'M1-mp1em9.a0p001.60d50r.c17o3.fits' # mo
inbright     = 'unchanged'      # set peak surface brig
indirection  = 'J2000 00h00m00.00 -30d00m00.0' # "J2000
incell       = ''               # cell/pixel size e.g.
incenter     = '200GHz'         # frequency of center c
inwidth      = '0.2GHz'        # channel width e.g. "1

setpointings = True             #
integration  = '10s'           # integration (sampling
direction    = ['J2000 00h00m00.00 -30d00m00.0'] # "J200
mapsize      = ['1arcmin', '1arcmin'] # angular size of
maptype      = 'hexagonal'     # hexagonal, square, et
pointingspacing = '6arcsec'    # spacing in between po

predict      = True            # calculate visibilities
complist     = ''              # optional componentlis
antennalist  = '/home/dpetry/temp/casa/data/alma/simmos/a
# interferometric MS
refdate      = '2011/10/21/03:25:00' # time/date of obse
totaltime    = '12000s'       # total time of observa
caldirection = ''              # pt source calibrator
calflux      = '1Jy'          #
sdantlist    = ''             # single dish antenna p
sdant        = 0              # single dish antenna i

thermalnoise = ''             # add thermal noise: [t
leakage      = 0.0            # cross polarization
image        = True           # (re)image $project.ms
vis          = '$project.ms'  # Measurement Set(s) to
modelimage   = ''            # prior image to use in
imsize       = 0              # output image size in
cell         = '0.3arcsec'    # cell size with units
niter        = 10000          # maximum number of ite
threshold    = '0.01 mJy'     # flux level (+units) t
weighting    = 'briggs'       # weighting to apply to
outertaper   = []            # uv-taper on outer bas
stokes       = 'I'           # Stokes params to imag

analyze      = False          # (only first 6 selecte
graphics     = 'screen'      # display graphics at e
verbose      = True           #
overwrite    = True           # overwrite files start
async       = False          # If true the taskname

```

CASA <49>: █



Summary



- The standard science data analysis package for ALMA and EVLA is **CASA**
- Data from other observatories can also be processed, e.g. BIMA, ATCA, CARMA, SMA
- CASA is mostly C++ code (libraries for general use available as **casacore**)
- approx. 22 people are working on CASA in North America, Europe, and Japan
- CASA is a **comprehensive toolbox** with
 - MATLAB-like, scriptable user interface using **Python/iPython**
 - **procedures for calibration, imaging, spectral and spatial analysis, simulation and more**
 - GUI tools for data selection, browsing, plotting, and image processing
- The command-line interface has two levels:
 - **tasks** for the common analysis problems
 - **tools** for everything else including *your own tasks*
- the heart of the science analysis code is the **Measurement Equation**
- the internal data format are **CASA Tables**
- the **Measurement Set** is the CASA data format for visibility data
- CASA is publicly available under GPL for **Linux and Mac OS X**, installation is simple, see <http://casa.nrao.edu/>
- The latest release is version 3.3 (November 2011)