Analysing ALMA data - the *CASA* software package

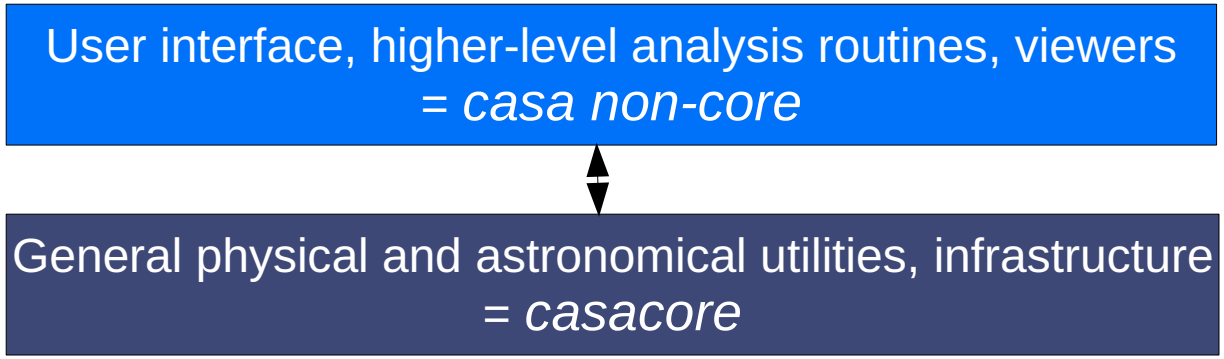
Dirk Petry (ESO), June 2010

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
→ How does CASA look and feel?	- the typical analysis session
→ CASA status and release plans	

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:**



- **Implements the “Measurement Equation” (Hamaker, Bregman & Sault 1996)**
- **Internal data format is the “Measurement Set” (Kemball & Wieringa 2000)**
- **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. 17 FTE developers are at work at

US (NRAO and others): 10.5
Japan (NAOJ): 3.0
Europe (ESO and others): 3.5

+ 1 CASA manager (NRAO Socorro) = Nick Elias

+ 1 Project Scientist (NRAO Socorro) = Jürgen Ott

+ a few 5% FTEs at ASTRON, ATNF, and other places
Also involved:
ALMA Computing Managers = B. Glendenning (NRAO), G. Raffi, 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 → MS, FITS → Image, UVFITS → MS, VLA → MS, etc.

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 set of C++ classes for radio astronomical calibration and imaging

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

   special procedures for each required task such as CLEAN etc.

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 for users and developers

CASA design and implementation

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 → MS, FITS → Image, UVFITS → MS, VLA → MS, etc.

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 set of C++ classes for radio astronomical calibration and imaging

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

special procedures for each required task such as CLEAN etc.

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 for users and developers

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

MAIN	none	-	-	ANTENNA_ID FEED_ID DATA_DESC_ID PROCESSOR_ID (PHASE_ID) FIELD_ID (PULSAR_GATE_ID) ARRAY_ID OBSERVATION_ID STATE_ID
(FREQ_OFFSET)	none	-	-	ANTENNA_ID FEED_ID SPW_ID
(SYSCAL)	none	-	-	ANTENNA_ID FEED_ID SPW_ID
POINTING	none	-	-	ANTENNA_ID POINTING_MODEL_ID
(WEATHER)	none	-	-	ANTENNA_ID
HISTORY	none	-	-	OBSERVATION_ID OBJECT_ID
FLAG_CMD	none	-	-	none

ANTENNA	ANTENNA_ID	row number	MAIN FEED FREQ_OFFSET POINTING SYSCAL WEATHER		(ORBIT_ID) (PHASED_ARRAY_ID)
FEED	FEED_ID	explicit	MAIN FREQ_OFFSET SYSCAL		ANTENNA_ID SPW_ID BEAM_ID (PHASED_FEED_ID)
DATA_DESCRIPTION	DATA_DESC_ID	row number	MAIN		SPW_ID POLARIZATION_ID (LAG_ID)
PROCESSOR	PROCESSOR_ID	row number	MAIN		TYPE_ID MODE_ID (PASS_ID)
FIELD	FIELD_ID	row number	MAIN		SOURCE_ID (EPHEMERIS_ID)
OBSERVATION	OBSERVATION_ID	row number	MAIN HISTORY		none
STATE	STATE_ID	row number	MAIN		none
SPW	SPW_ID	row number	DATA_DESCRIPTION FEED FREQ_OFFSET SOURCE SYSCAL		(RECEIVER_ID) (DOPPLER_ID) (ASSOC_SPW_ID)

POLARIZATION	POLARIZATION_ID	row number	DATA_DESCRIPTION		none
(SOURCE)	SOURCE_ID	explicit	(DOPPLER) FIELD		SPW_ID (PULSAR_ID)
(DOPPLER)	DOPPLER_ID	explicit	SPW_ID		SOURCE_ID TRANSITION_ID

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

V1, D.Petry, 13.2.09

CASA design and implementation

CASA special features:

b) the *Measurement Equation* (Hamaker, Bregman, & Sault 1996 + Sault, Hamaker, & Bregman 1996) implemented as a set of C++ classes for radio astronomical calibration and imaging

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

where

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

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

P = paralactic angle

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 design and implementation

CASA special features:

b) the *Measurement Equation* (Hamaker, Bregman & Sault 1996)
implemented as a set of C++ classes for radio astronomical calibration and imaging
(continued)

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

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

ideal visibility known from calibrator source

⇒ have set of linear equations.

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

The calibrater (cb) tool contains a set of ***solvers*** for the different calibration components.

CASA design and implementation

CASA special features:

c) A programmable command line interface with scripting

Framework Architecture of 17 tools can be bound to any scripting language, presently selected is ***Python (augmented by IPython)***

at – atmosphere library
ms – Measurement Set utilities
mp – Measurement Set Plotting, e.g. data (amp/phase) versus other quantities
cb – Calibration utilities
cp – Calibration solution plotting utilities
im – Imaging utilities
ia – Image analysis utilities
fg – flagging utilities
tb – Table utilities (selection, extraction, etc.)
me – Measures utilities
tp – table plot
vp – voltage patterns
qa – Quanta utilities
cs – Coordinate system utilities
pl – matplotlib functionality
sd - ASAP = ATNF Spectral Analysis Package (single-dish analysis imported from ATNF)
sm - simulation

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

toolkit (implemented in C++) ⟶ tasks (implemented in Python)

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 3.0.1 comes with 91 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 3.0.2 published this month

- Development platforms: Linux (RHEL) + Mac OS X
- Supported platforms (binary distribution): RHEL, Fedora, openSuSE, Ubuntu, Max OS X

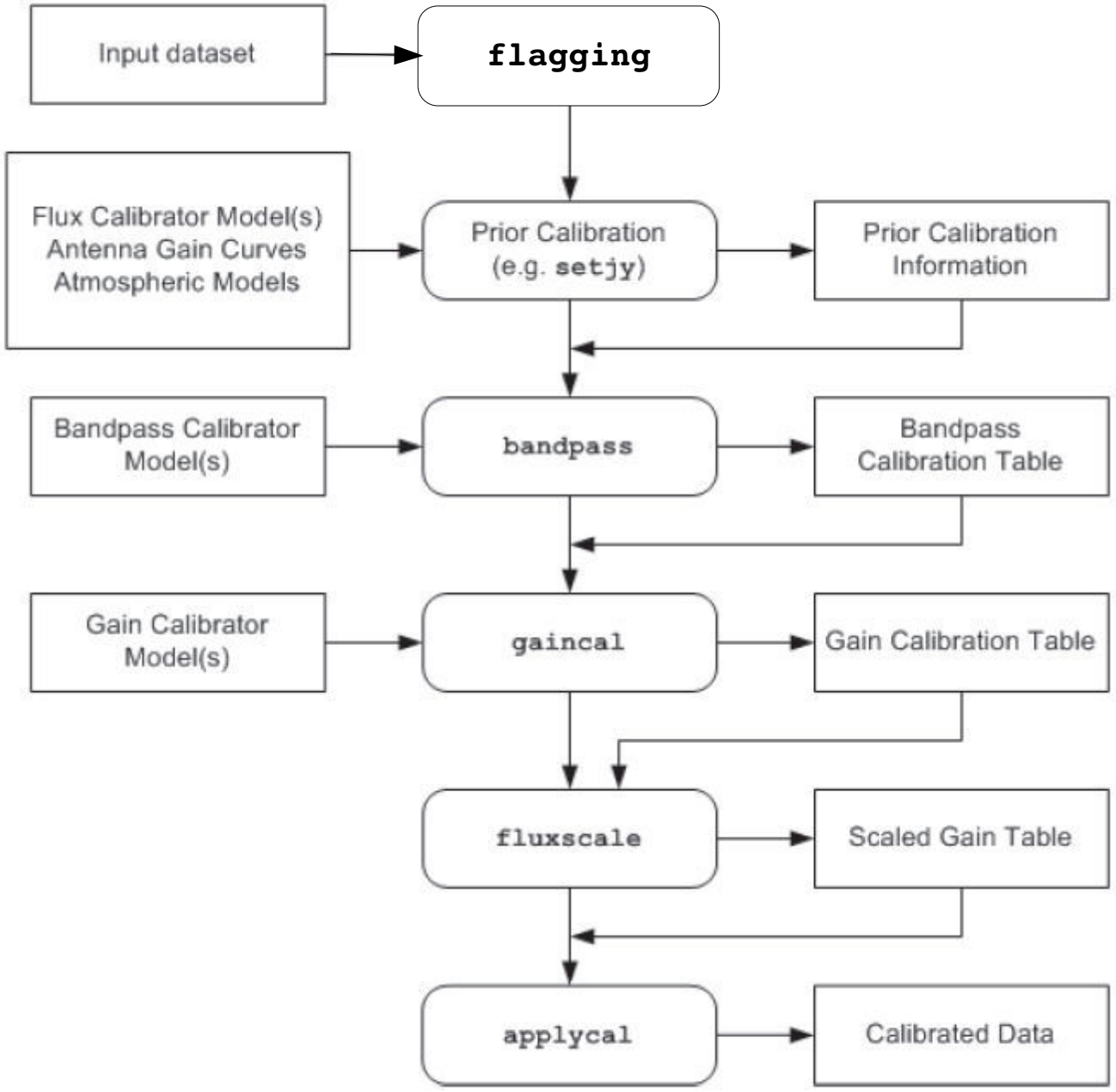
- Code kept in *svn* repository at NRAO, Socorro
- Presently have approx. 4300 modules, 1.5E6 lines of code, 1E6 lines of comments
- The core functionality (*casacore,* also available at http://code.google.com/p/casacore/)
  is also used by other projects

- *Hot topics*:
  - Support for High Performace Computing and Parallelisation
  - Advanced Imaging: wide fields, continuum imaging over wide spectral ranges
  - Interoperability: using CASA for other observatories and VLBI

How does *CASA* look and feel?

A typical analysis session

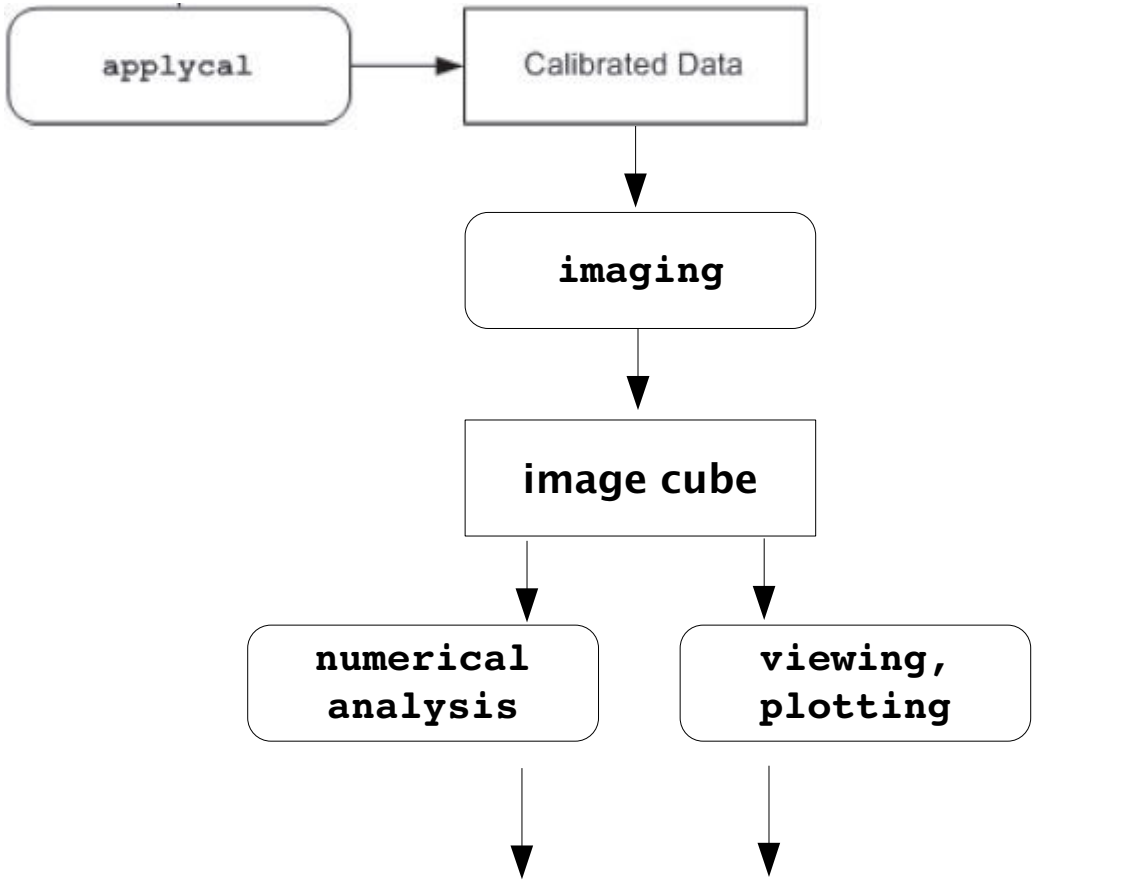
Part 1: flagging and calibration



How does *CASA* look and feel?

A typical analysis session

Part 2: imaging and
image analysis

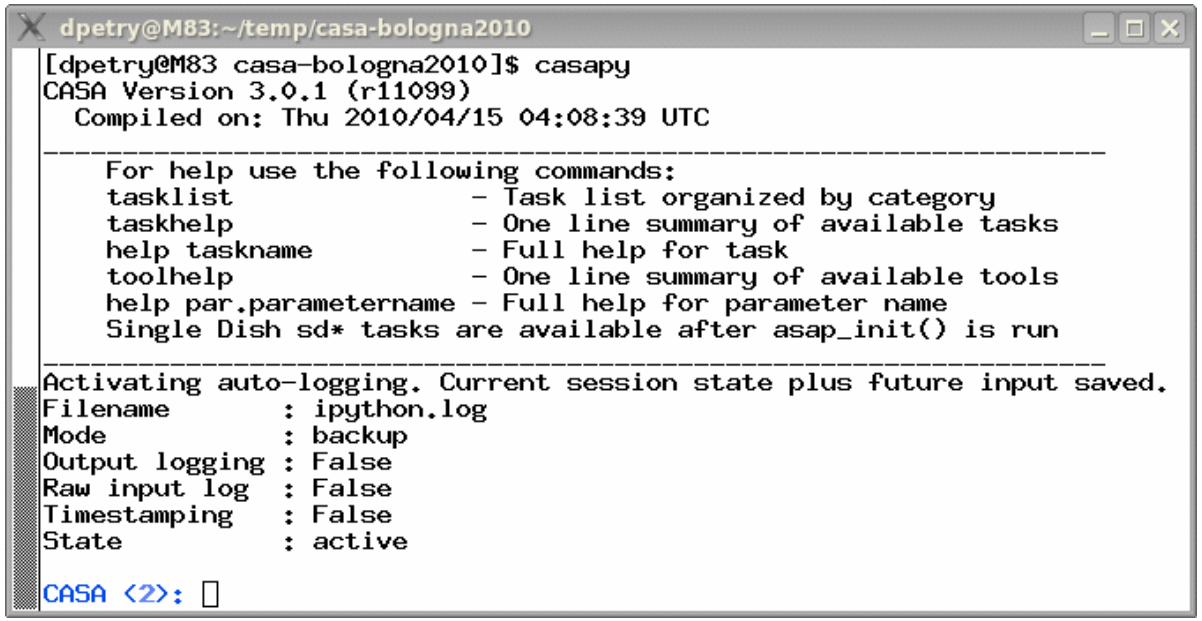


How does *CASA* look and feel?

Pictures from a typical analysis session

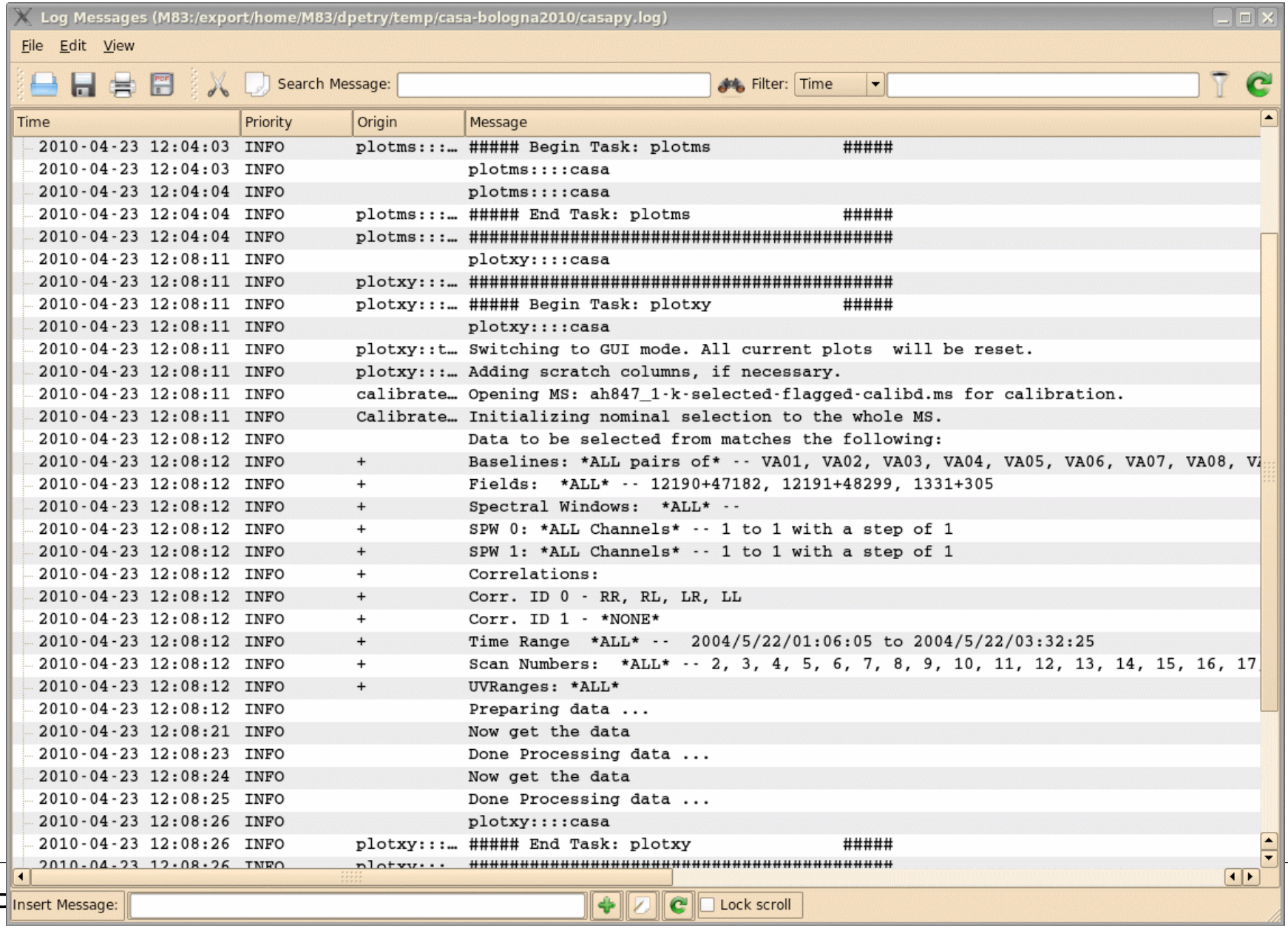
1) Startup:
open terminal and start *casapy*

Available tasks and tools are listed and the logger window is opened.



The *CASA* user interface

The logger provides functionality for monitoring and debugging command execution.



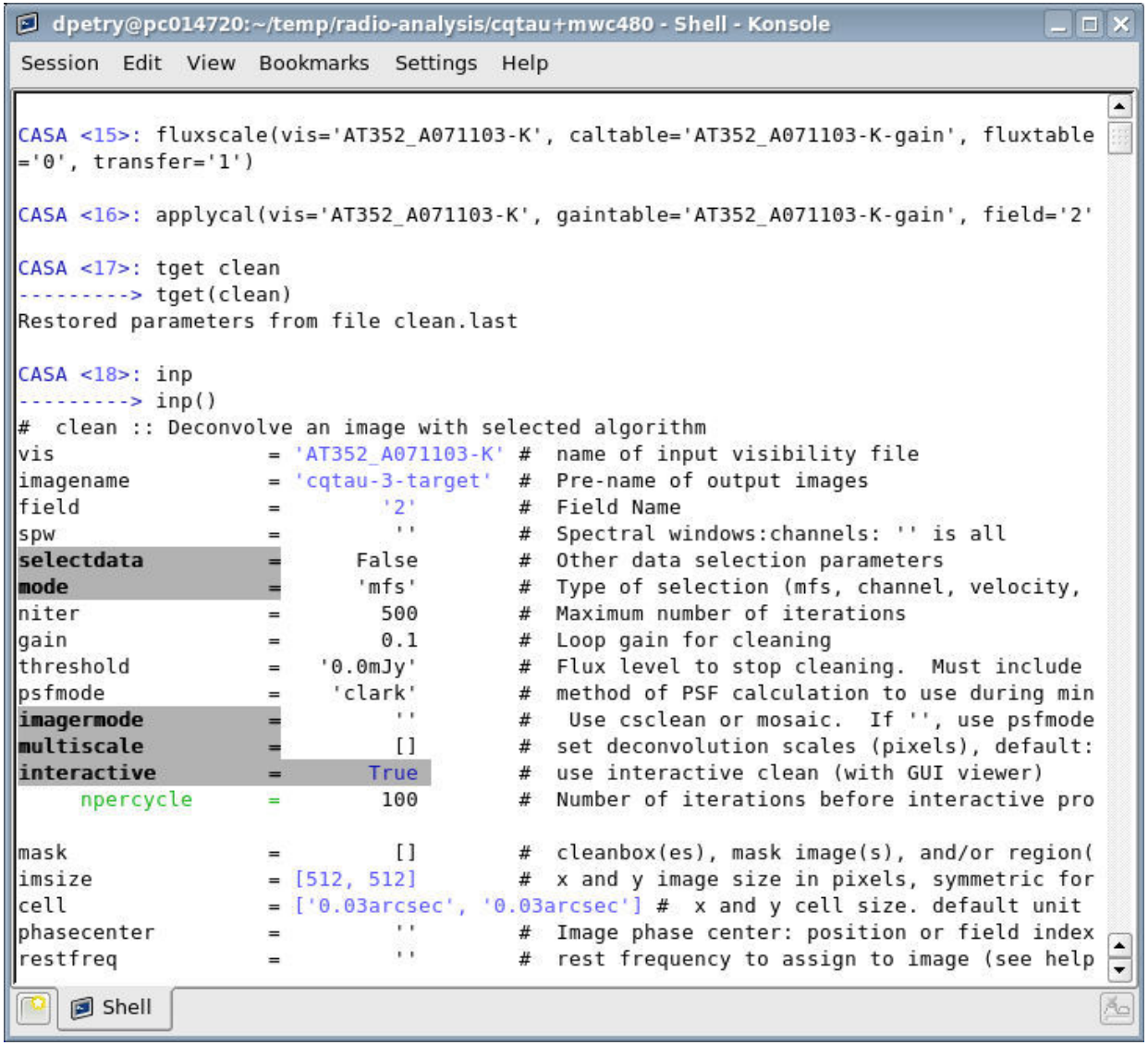
The *CASA* user interface

Pictures from a typical analysis session

2) enter commands in a MATLAB-like environment

`recall previous settings`

`list present settings for given task (includes parameter verification)`



The *CASA* user interface

Pictures from a typical analysis session

3) where needed, tools have GUIs:

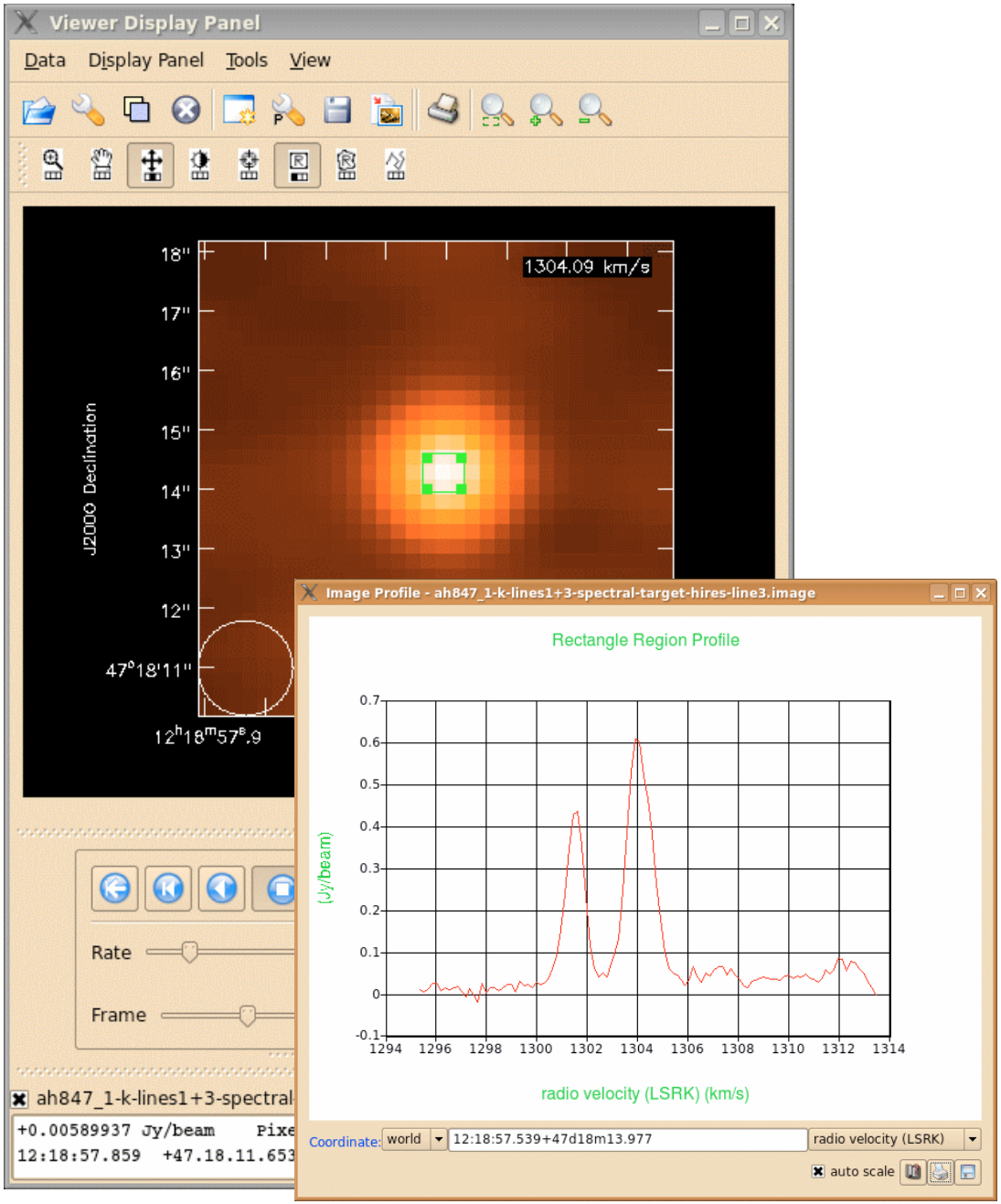
plotxy, plotcal, browsetable, viewer, clean

(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



The *CASA* user interface

A typical analysis session

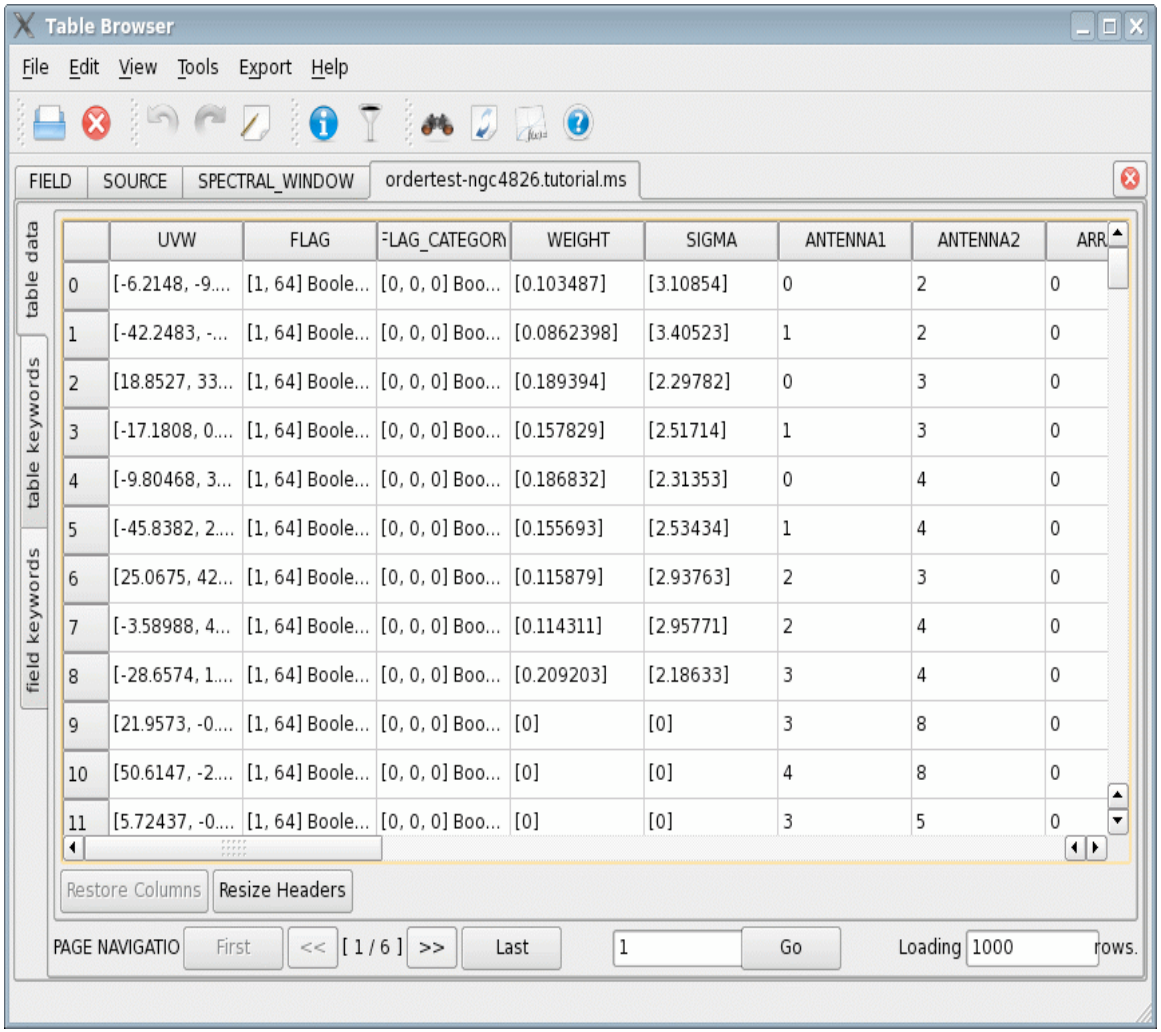
3) where needed, tools have GUIs:

plotxy, plotcal, browsetable, viewer, clean

(started in separate threads)

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

Also Qt-based.



The *CASA* user interface

A typical analysis session

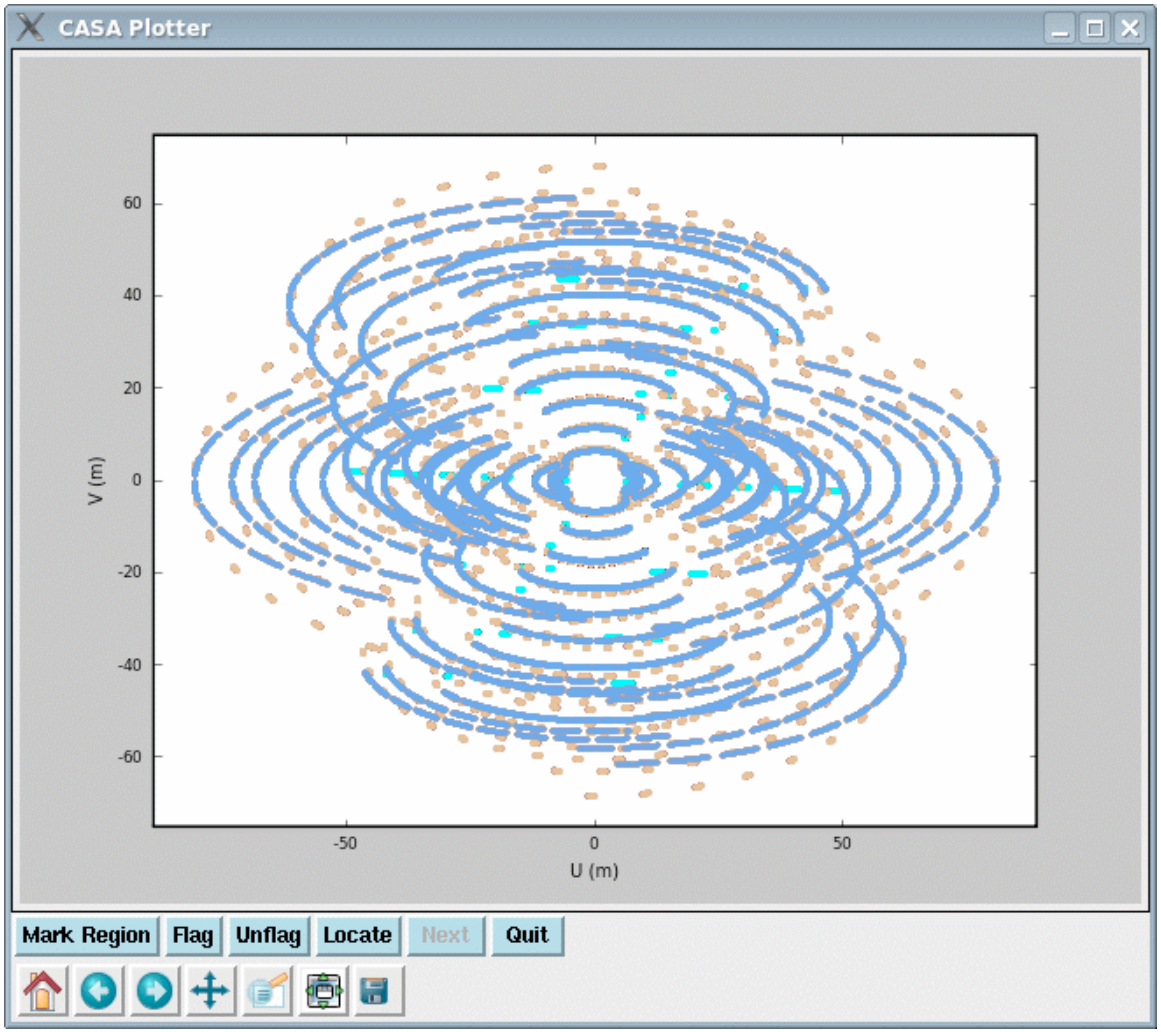
3) where needed, tools have GUIs:

plotxy, plotcal, browsetable, viewer, clean

(started in separate threads)

plotxy is a specialized tool for diagnostic plots and data selection

To be phased out.



The *CASA* user interface

A typical analysis session

3) where needed, tools have GUIs:

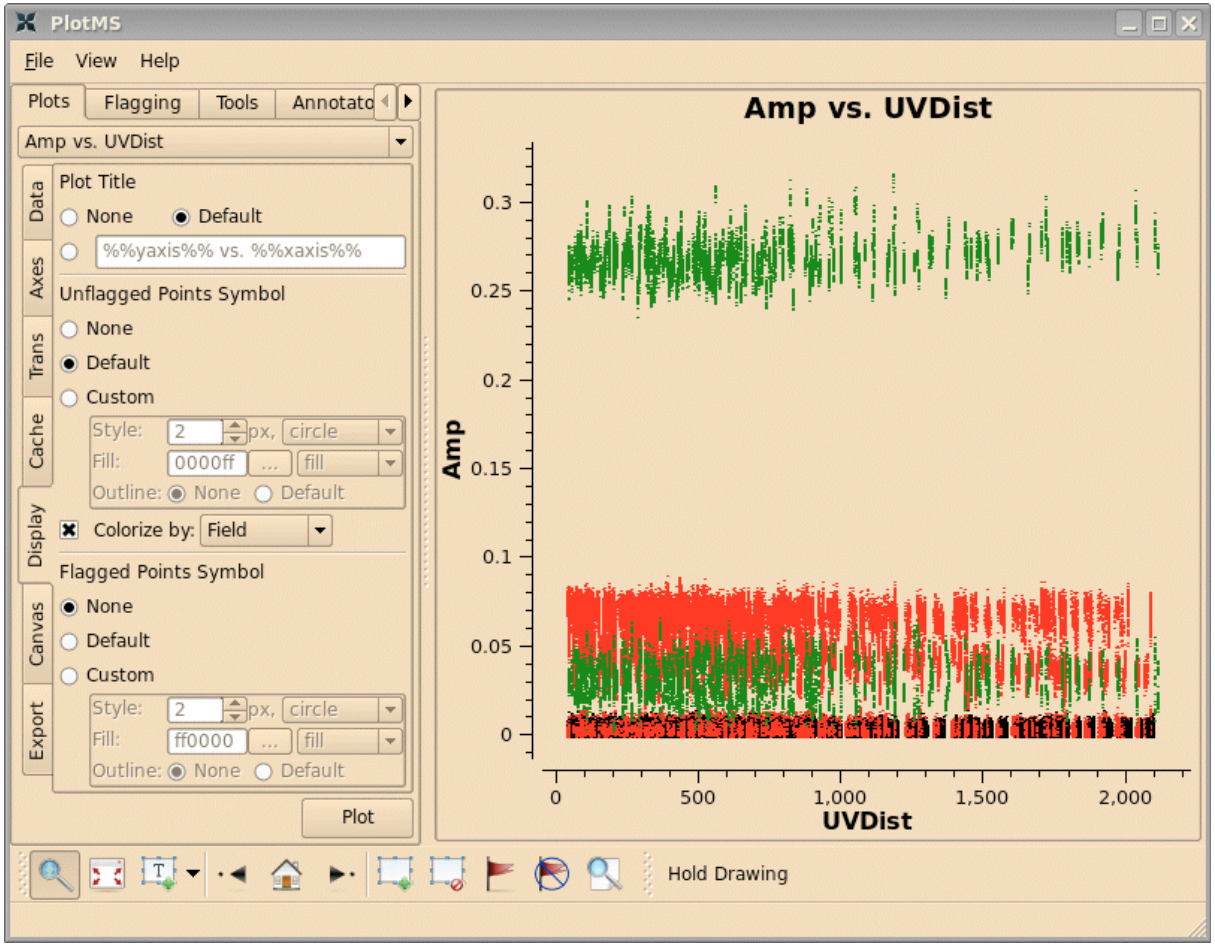
plotxy, plotcal, browsetable, viewer, clean

(started in separate threads)

plotms is going to replace plotxy. Release 3.0.2 contains beta version.

plotms is Qt-based and much faster than plotxy.

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



Summary

- **The standard science data analysis package for ALMA and EVLA is *CASA***
- **Data from other observatories can also be processed, e.g. VLA, BIMA, ATCA, ...**
- ***CASA* derives from AIPS++ (partially survives in *casacore*)**
- **approx. 20 people are working on *CASA***
  in North America, Europe, and Japan
- ***CASA* is a toolbox with**
  - **MATLAB-like user interface**
  - **GUI tools for data selection, browsing, and image processing**
- **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**
  (it is technically a Table with several well-defined sub-tables)
- ***CASA* is publicly available under GPL for Linux and Mac OS X**

- **The first public release of *CASA* (version 3.0.0) became available in December 2009**
- **The latest release is version 3.0.2**

Additional remarks

Dirk Petry (ESO), June 2010

CASA Measurement Sets, ASDMs, and uvfits

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

- Presently supported input formats:

ALMA: ALMA Science Data Model (ASDM)

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

VLA: VLA archive format

FITS IDI: planned for later this year

and the transport format uvfits

CASA Measurement Sets, ASDMs, and uvfits

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

CASA Measurement Sets, ASDMs, and uvfits

The ASDM

- 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 stored in the MAIN table

- no compression

- on disk, table descriptions in XML files, table data in binary MIME format files

- import into CASA using the task `importasdm` (for v1) or `importoldasdm` (for v0.9)

- in release 3.0.2 there is a first version of `exportasdm` (MS to ASDM)

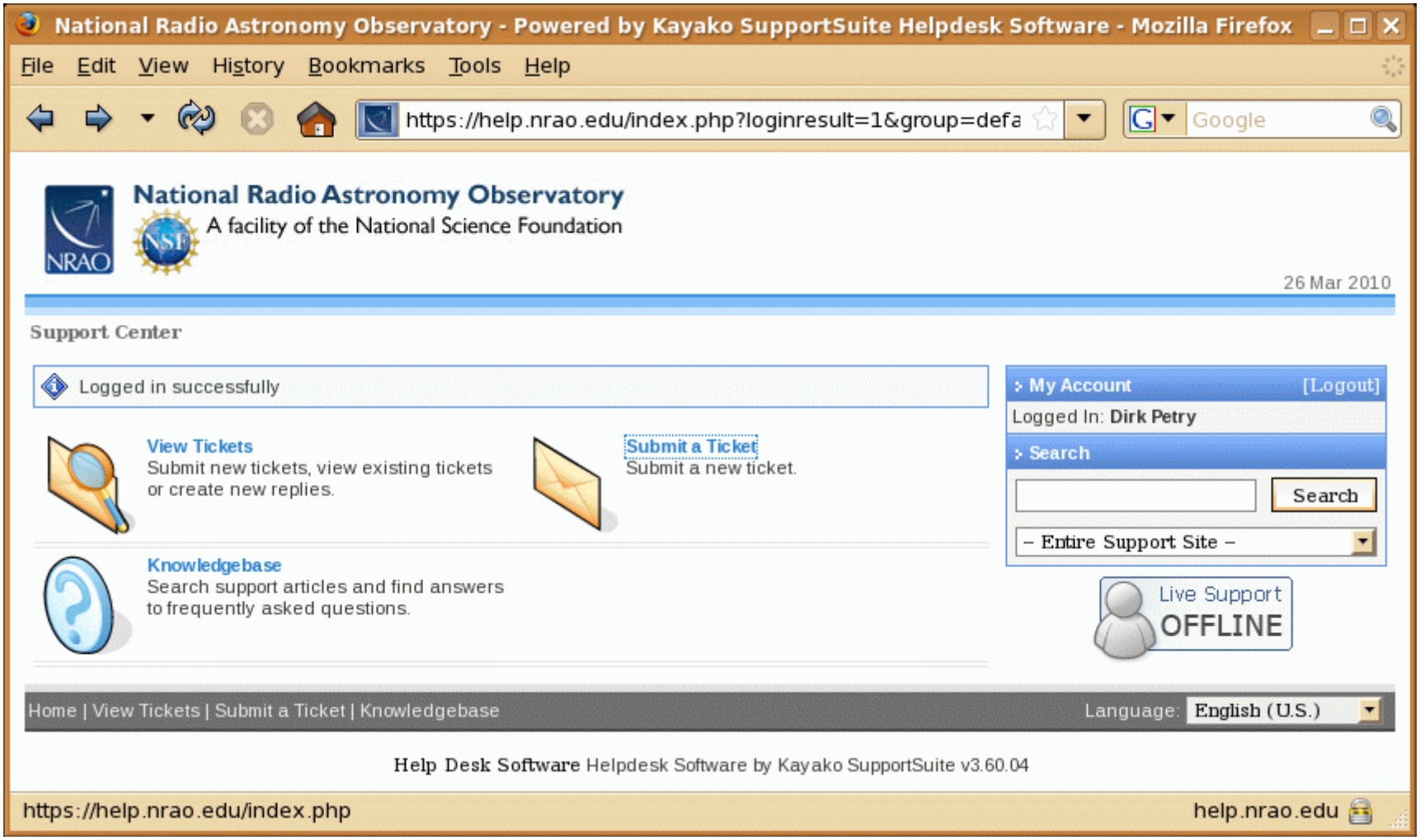
CASA - further information

Further information on CASA

- CASA home page	**http://casa.nrao.edu**
- CASA release notes	**http://casa.nrao.edu/release_notes.shtml**
- CASA cookbook	**http://casa.nrao.edu/Doc/Cookbook/casa_cookbook.pdf**
- CASA guides	**http://casaguides.nrao.edu**

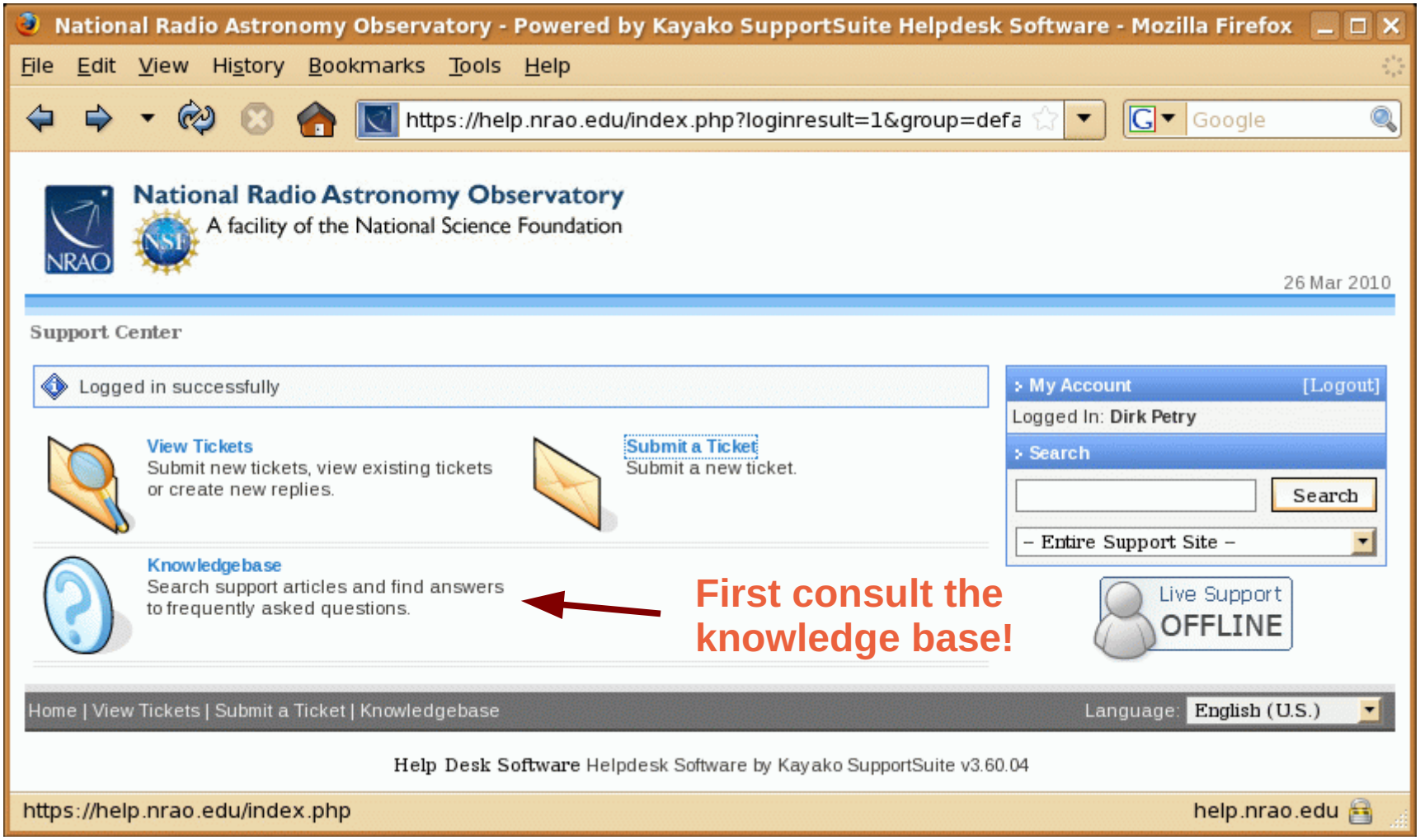
In case of problems ...

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



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



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*

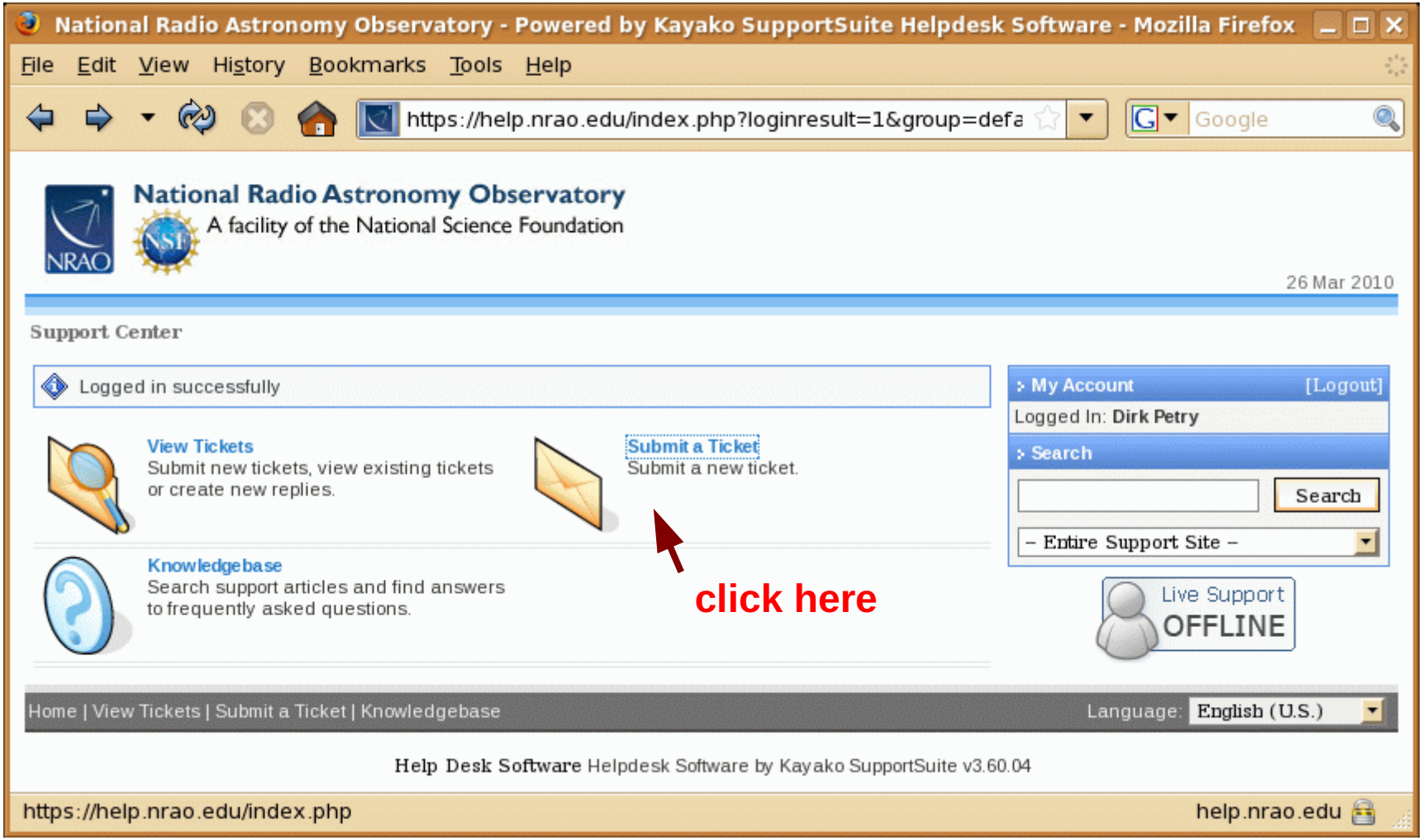
- **Try to reproduce your problem, ideally by writing a Python *script* which will demonstrate the problem.**
- **Put your *test data (if needed)* on some web or ftp server where it can remain for at least several months.**
- **File a helpdesk ticket including the script, a short description of the problem, 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.**

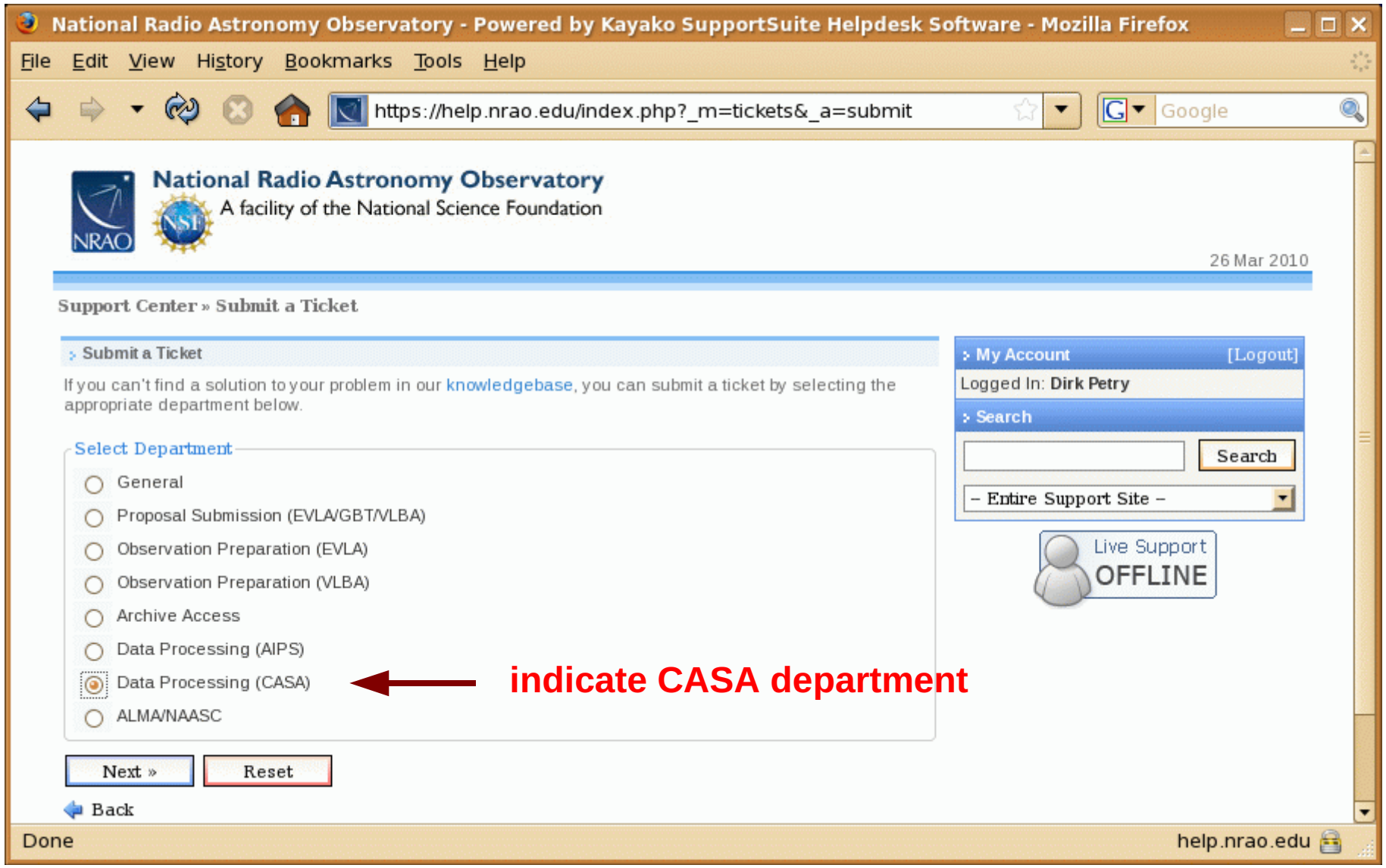
In case of problems ...

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



In case of problems ...

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



In case of problems

How to file a helpdesk ticket:

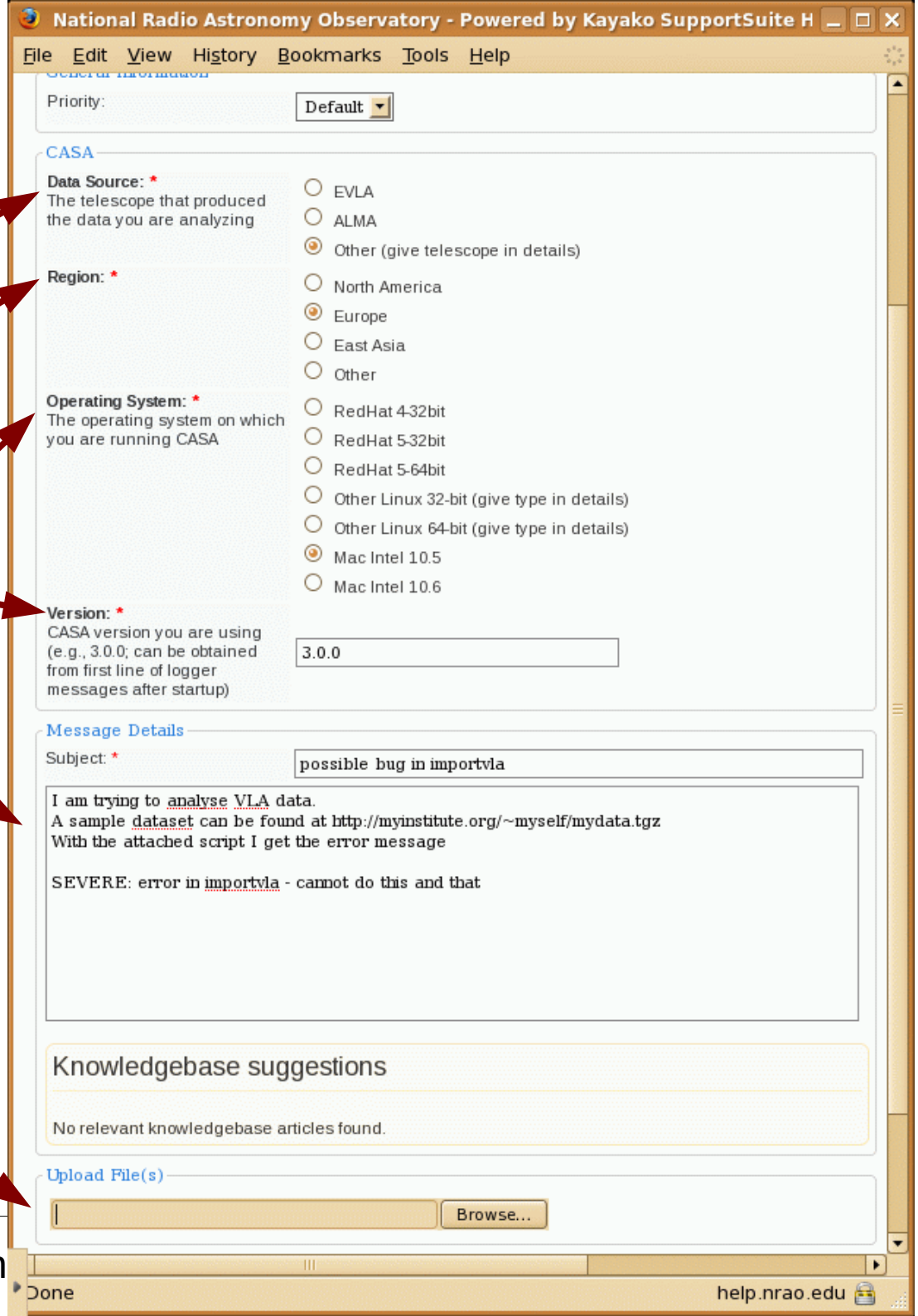
Where does your data come from?
(identify necessary expertise)

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 ...