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)*

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

CASA tasks and tools

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

accum	fixvis	importfits	pclean	sdscale
applycal	flagautocorr	importfitsidi	peel	sdsmooth
[illegible]	[illegible]	[illegible]rt	plotants	sdstat
[illegible]	[illegible]	[illegible]dasdm	plotcal	sdtpimaging
[illegible]	[illegible]	[illegible]fits	plotms	setjy
[illegible]	[illegible]	[illegible]a	plotxy	simdata
[illegible]	[illegible]	[illegible]	polcal	slsearch
[illegible]	[illegible]	[illegible]	rmtables	smoothcal
[illegible]	[illegible]	[illegible]	sdaverage	specfit
[illegible]	[illegible]	[illegible]	sdbaseline	splattotable
[illegible]	[illegible]	[illegible]	sdcal	split
[illegible]	[illegible]	[illegible]	sdcoadd	ssoflux
[illegible]	[illegible]	[illegible]	sdfit	uvcontsub
[illegible]	[illegible]	[illegible]tory	sdflag	uvcontsub2
[illegible]	[illegible]	[illegible]	sdflagmanager	uvmodelfit
deconvolve	imhead	listvis	sdimaging	uvsub
exportasdm	immath	mosaic	sdimprocess	viewer
exportfits	immoments	msmoments	sdlist	vishead
exportuvfits	importasdm	msview	sdmath	visstat
feather	importevla	newflagdata	sdplot	widefield
find	importevla2	oldsimdata	sdsave	

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

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

> ***http://casa.nrao.edu/Doc/Cookbook/casa_cookbook.pdf***

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

command	*example*	
default	`default(clean)`	**- reset all input parameters**
inp	`inp`	**- show parameters of current task**
go	`go`	**- start current task**
saveinputs	`saveinputs(clean, parfile)`	**- store parameters in file**
tget	`tget(clean, parfile)`	**- restore parameters from file**

CASA tasks and tools

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

```
cb (calibrater)          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*

```
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*

one input parameter

```
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*

```
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
```

autoflag tool

CASA tasks and tools

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

```
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
```

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

CASA internal and external visibility data formats

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:

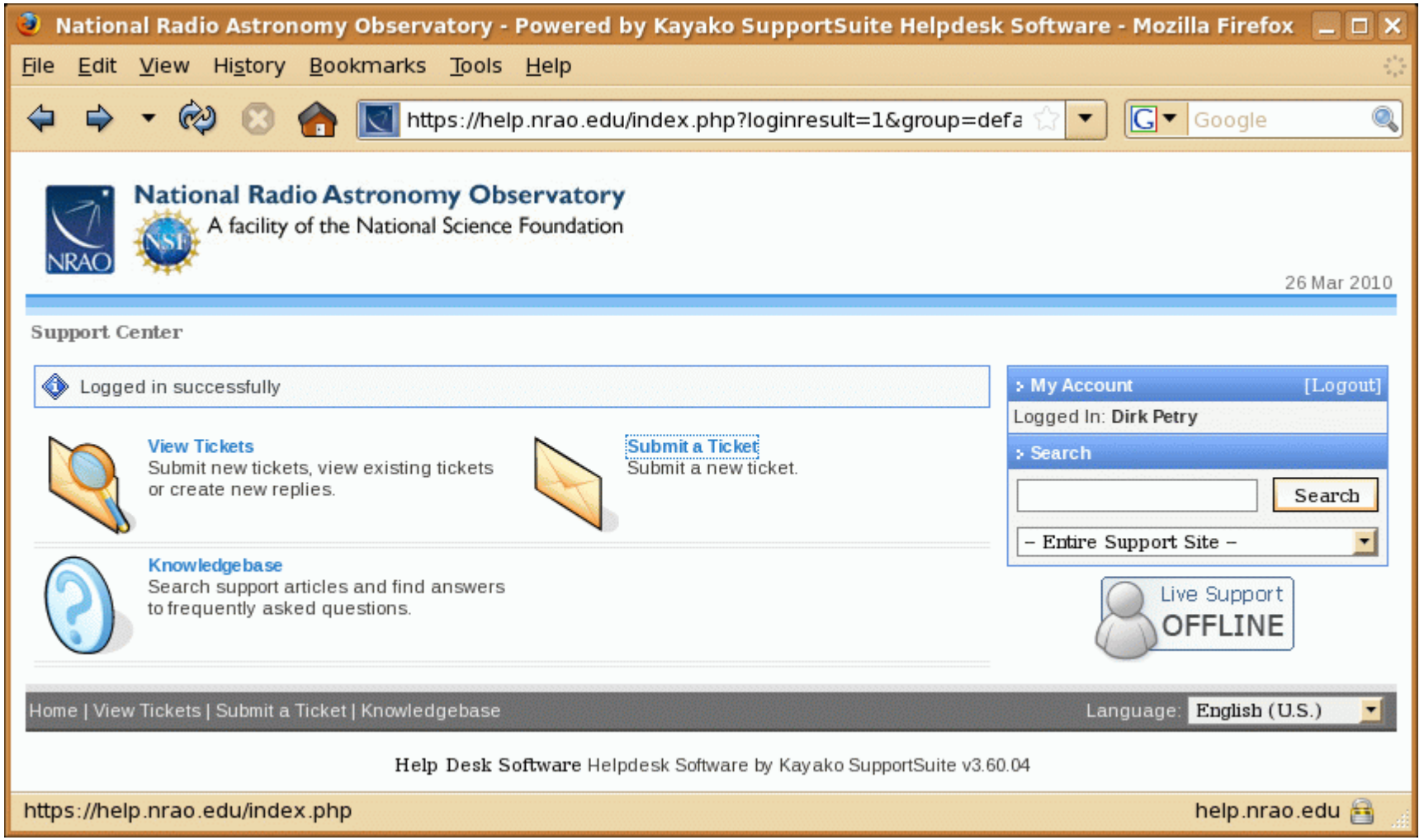
MS: listhistory() or ms.listhistory()
Image: ia.history()

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

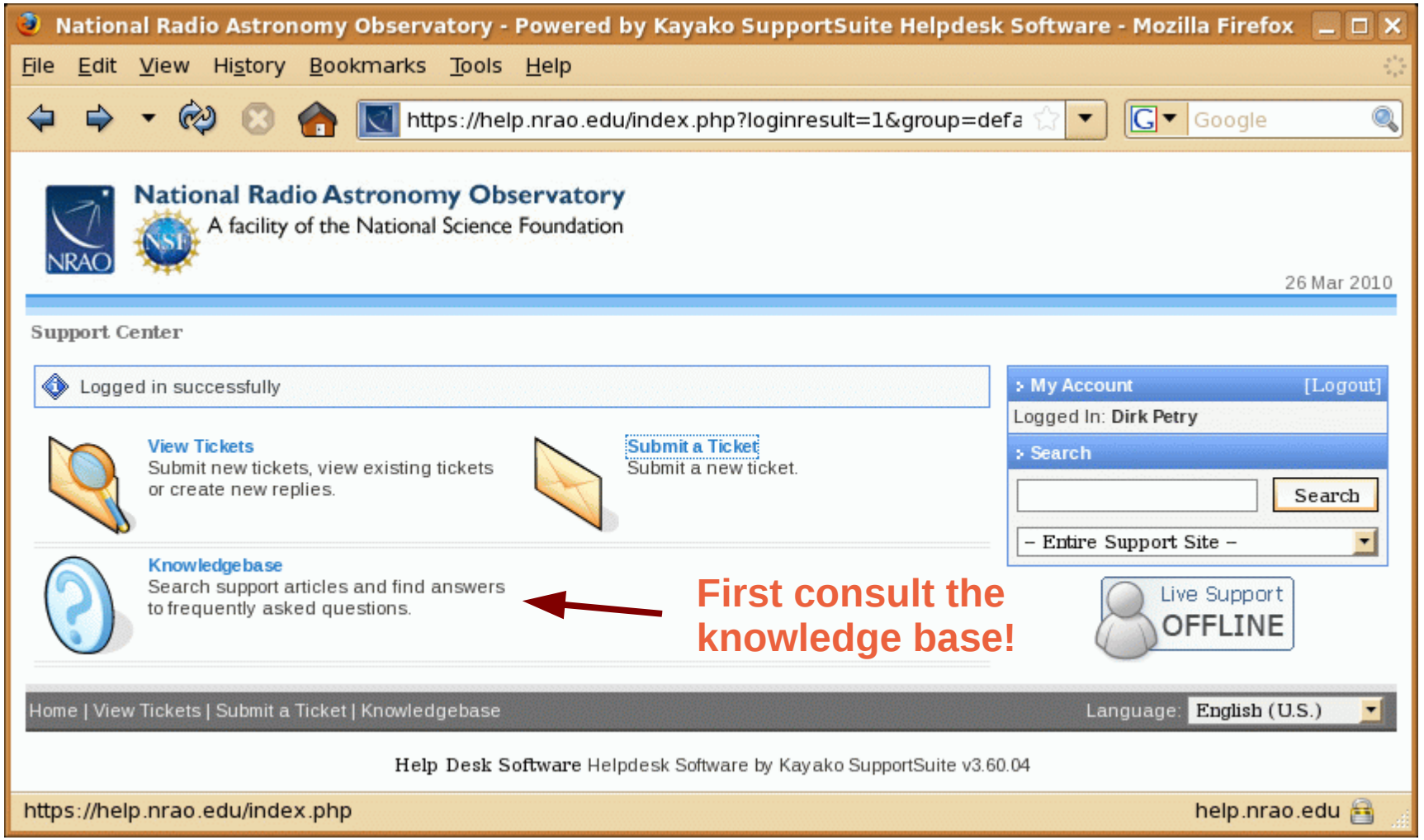


In case of problems ...

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*

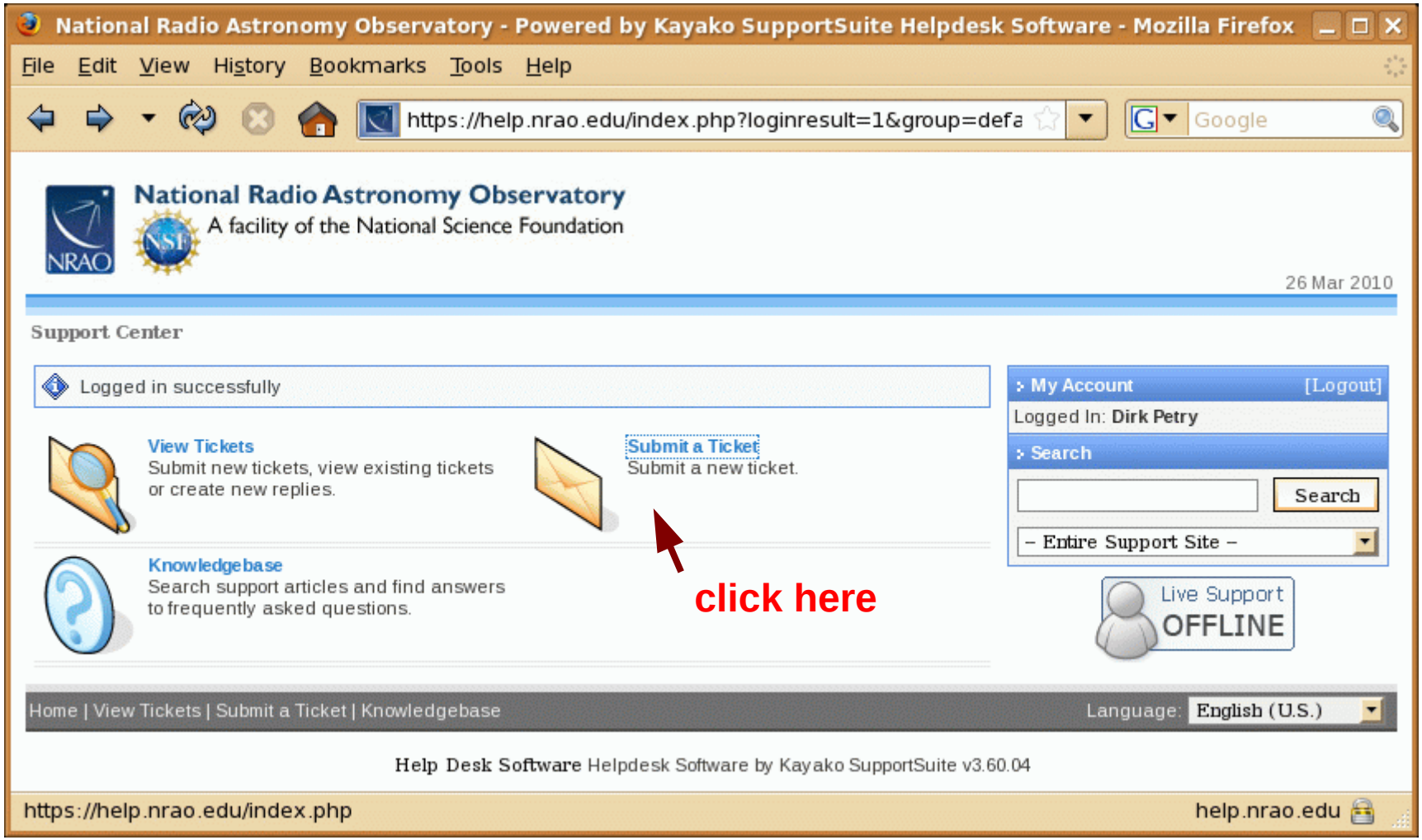
- **Try to reproduce your problem, ideally by writing a Python *script* which will demonstrate the problem.**
- **Put your *test data (if needed)* on some web or ftp server where it can remain for at least several months.**
- **File a helpdesk ticket including the script, a short description of the problem,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

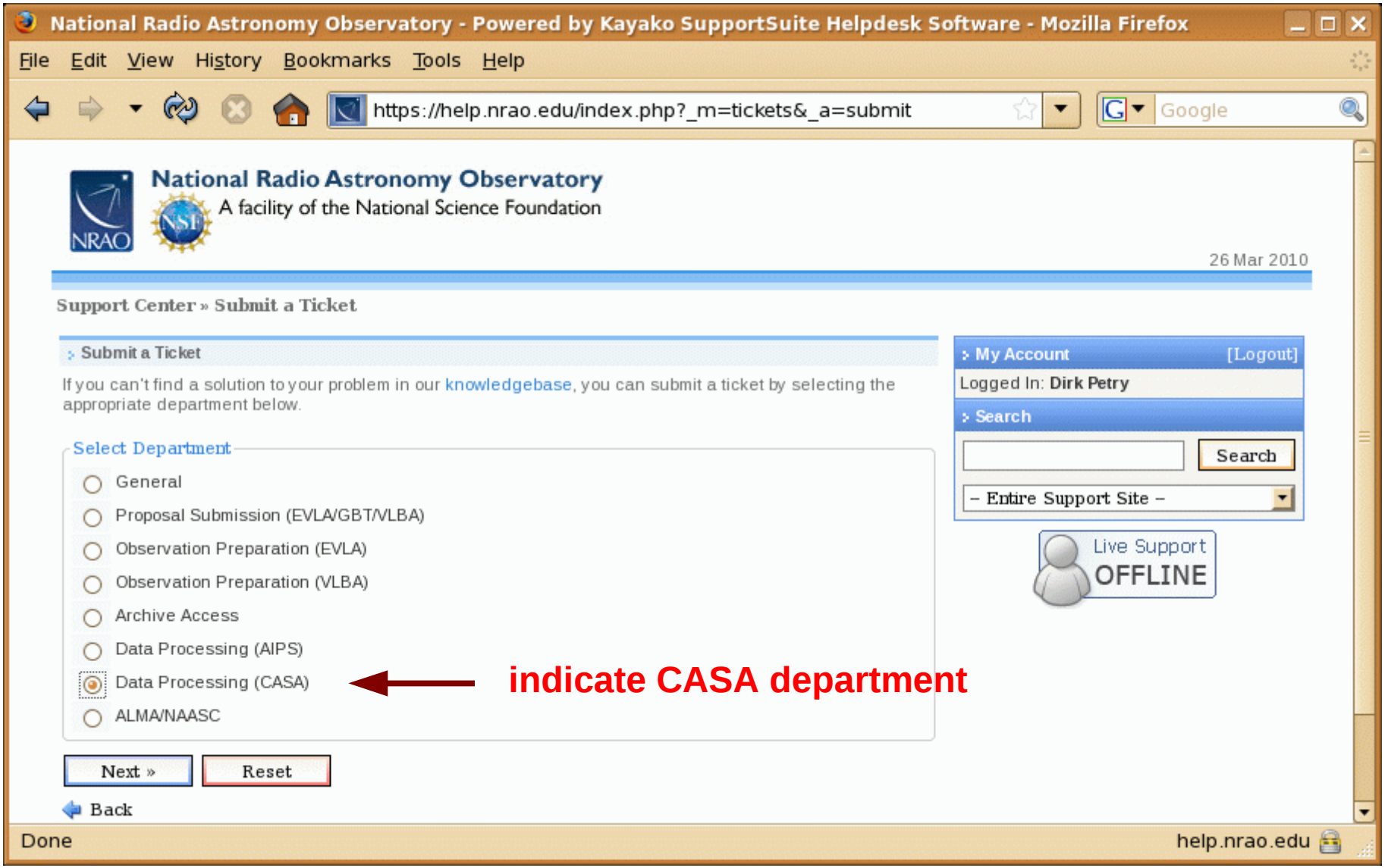
In case of problems ...

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:

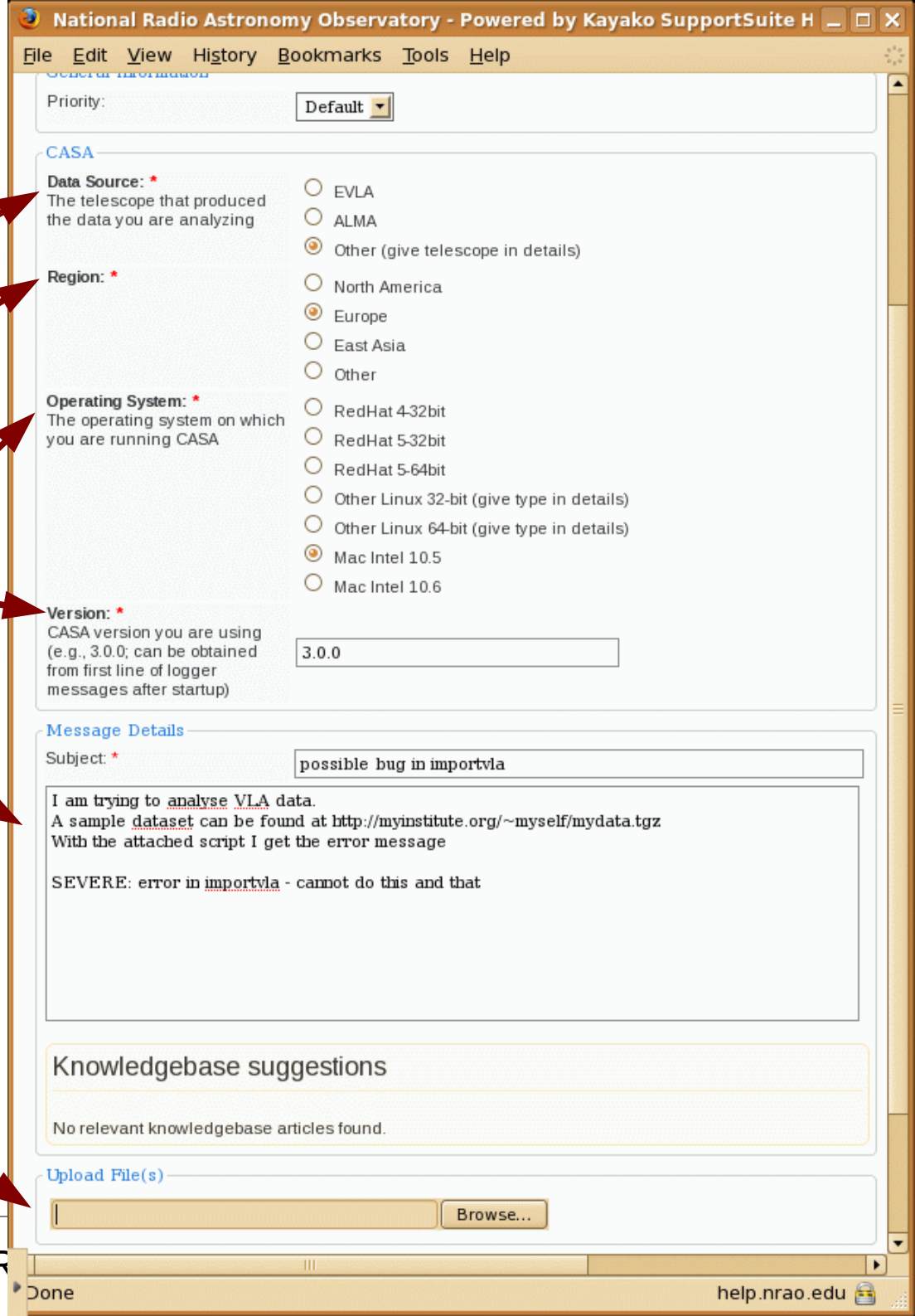
Where does your data come from?
(identify necessary expertise)

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 ...