MANCHESTER 1824

The University of Manchester

Introduction to CASA

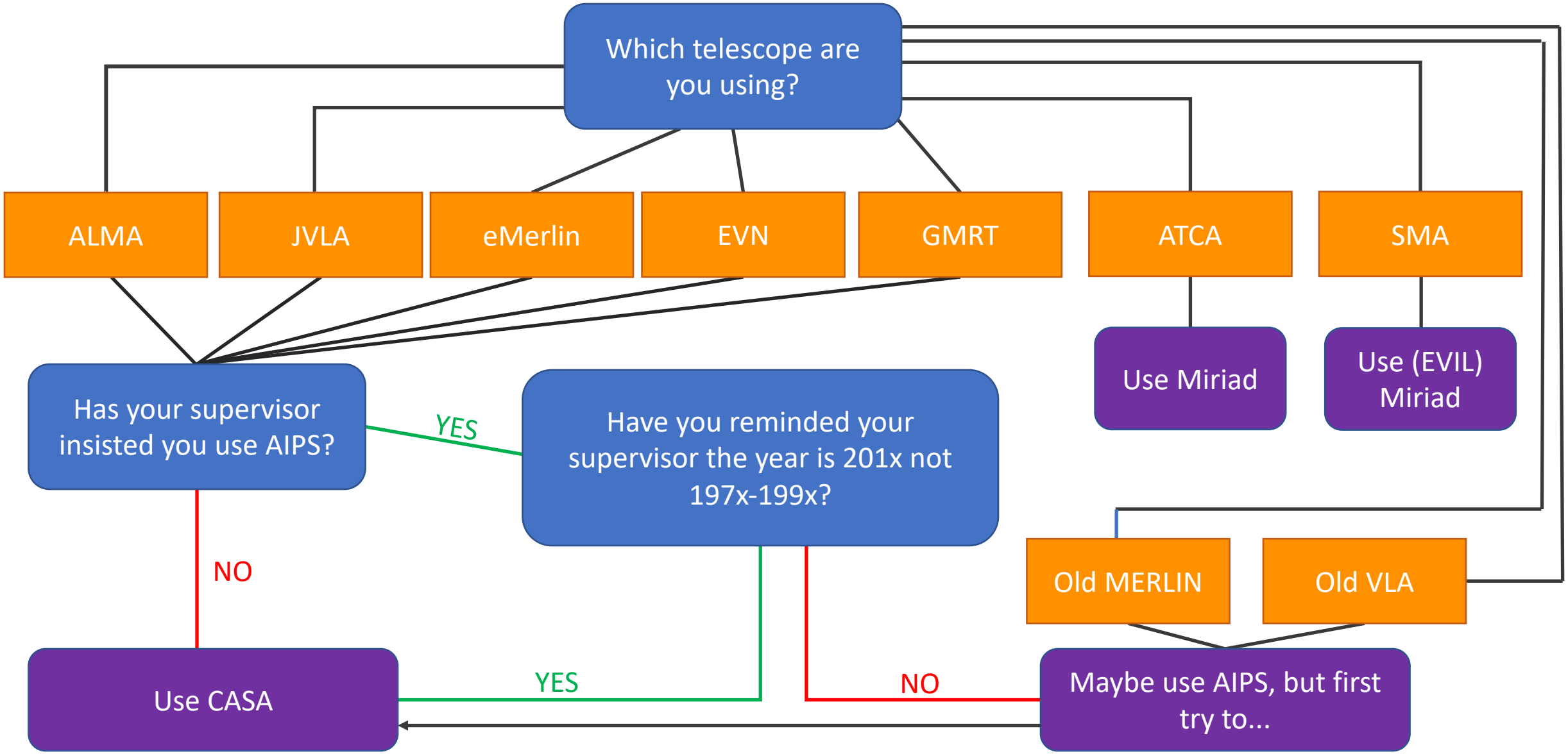
Adam Avison

ALMA

EUROPEAN ARC

ALMA Regional Centre || UK

Which software to use for data reduction?



What is CASA?

- Common Astronomy Software Applications

- A software package made up of C++ tools under an iPython interface

- Aims to support the next generation of radio telescopes (ALMA, JVLA, ngVLA and SKA)

- Basically it does everything you need to take raw visibilities from a telescope and turn them into science ready data.

Getting started

- The lastest version of CASA is 5.4.1 (and hopefully you have it all installed already)

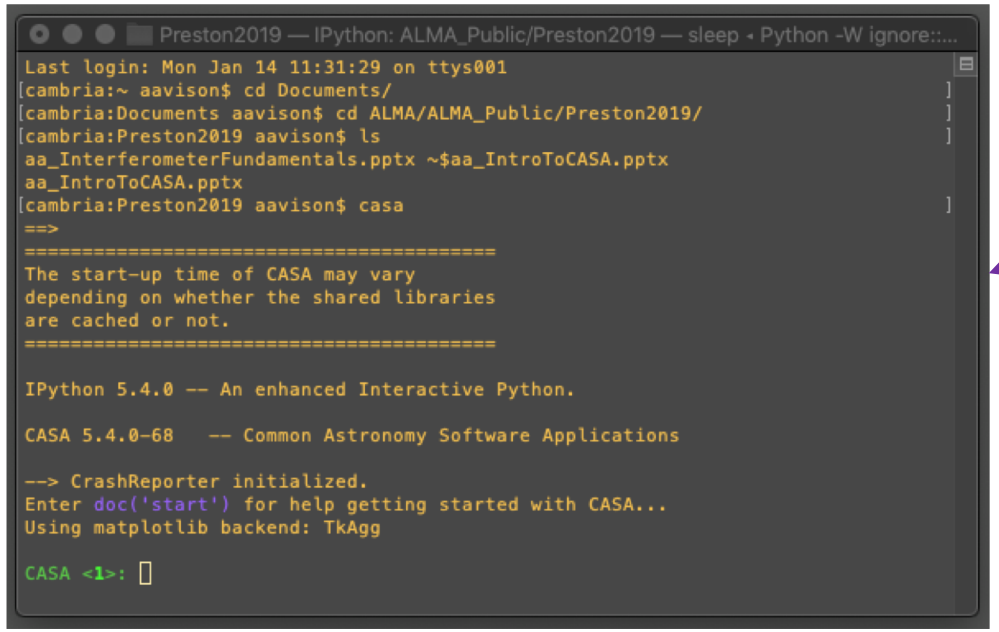
- If not you can get a copy from:

  https://casa.nrao.edu/casa_obtaining.shtml

- Versions from Linux and macOS.

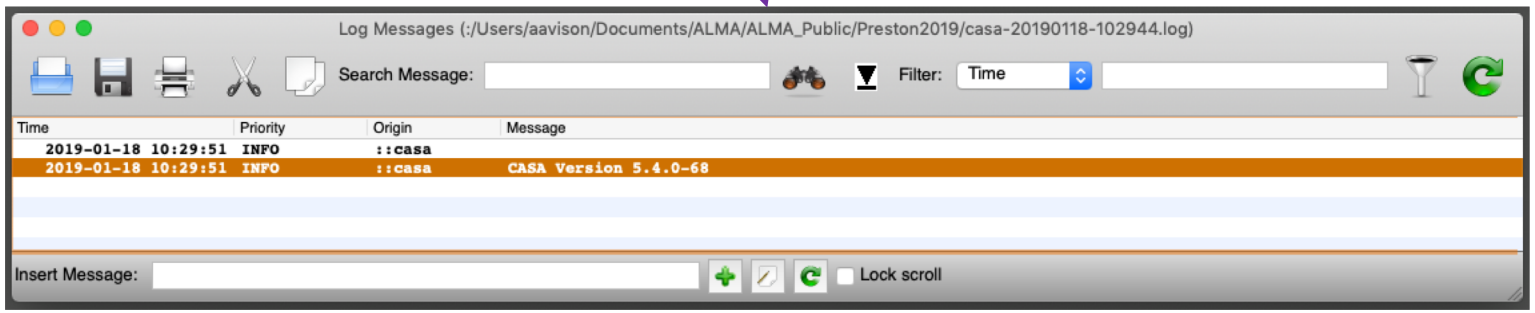
Working with CASA

- Once installed CASA can be started by typing 'casa' in a terminal.
- This will startup the iPython interface in the terminal and launch the Logger GUI



The iPython interface is where the work gets done

Logger will show you *(lots)* of useful *(and occasionally useless)* message from the the tasks being run.

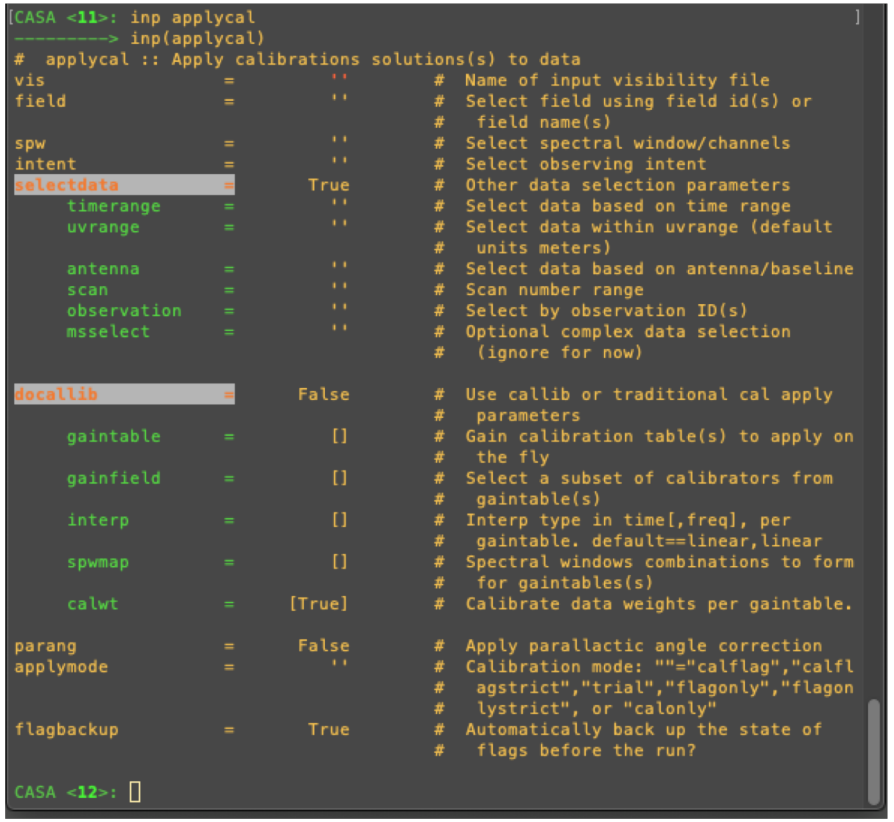


CASA tasks

- Tasks in CASA are the commands which are used to preform a specific function.
- Each contain a set of user definable parameters.
- To see what parameters a task has we can use the `inp` command.

Example: applycal

- applycal is the task used to apply calibration tables to the data. (You'll see this in use in the hands on later).



- Typing just inp will give you the inputs for the last CASA task you used.

Getting more information

- For most parameters within a task you can get more information on what it wants by typing help(par.<param_name>)

```
[CASA <17>: help par.field                                                    ]
----------> help(par.field)
Help on function field in module parameter_dictionary:

field()
    field -- Select field using field id(s) or field name(s).
             [run listobs to obtain the list ids or names]
    default: 0  (for sdimaging)
             '' = all fields (for the other ASAP tasks)

    If field string is a non-negative integer, it is assumed a field index
    otherwise, it is assumed a field name
             field='0~2'; field ids 0,1,2
             field='0,4,5~7'; field ids 0,4,5,6,7
             field='3C286,3C295'; field named 3C286 adn 3C295
             field = '3,4C*'; field id 3, all names starting with 4C
    This selection is in addition to scanlist, iflist, and pollist.

    See help par.selectdata for additional syntax.
    See specific task for any additional details.
[(END)_                                                                       ]
```

Getting even more information

- CASA comes with a HTML version of the CASA documentation which it will boot in a browser if you type `doc()`
- For the table of contents you can type `doc('toc')`
- For some specific tasks you can type e.g. `doc('applycal')`
- Alternately a Google search for 'CASA NRAO <task> name' should bring up the relevant page. *(Beware Google seems to have cached the 2010 docs so make sure you get the more recent versions)*

Navigating the docs

<escape to a browser>

Working with CASA

- To execute a task:

1. Default the task parameters with `default(taskname)`
2. Fill in all the parameters you need
3. Do an `inp` to check you've filled everything in the right format
4. Type the task name to execute it

```
[CASA <38>: default(applycal)                                                ]
```

```
[CASA <39>: vis='myvis.ms'                                                   ]

[CASA <40>: field='G123.45'                                                  ]

[CASA <41>: spw=0                                                            ]
```

```
[CASA <42>: inp                                                              ]
---------> inp()
#  applycal :: Apply calibrations solutions(s) to data
vis                 = 'myvis.ms'         #  Name of input visibility file
field               =  'G123.45'         #  Select field using field id(s) or
                                         #   field name(s)
spw                 =          0         #  Select spectral window/channels
```

```
[CASA <43>: spw='0'                                                          ]
```

```
[CASA <45>: inp                                                              ]
---------> inp()
#  applycal :: Apply calibrations solutions(s) to data
vis                 = 'myvis.ms'         #  Name of input visibility file
field               =  'G123.45'         #  Select field using field id(s) or
                                         #   field name(s)
spw                 =        '0'         #  Select spectral window/channels
```

```
[CASA <46>: applycal                                                         ]
---------> applycal()
```

Scripts

- As you can imagine typing each parameter for each task every time you want to execute it can be time consuming.
- Especially if you have a whole sample of sources to process in a similar way.
- Thankfully you can write Python scripts with CASA commands and execute them in CASA (this is how the hands on will run later today)
- To execute a script in CASA use the command `execfile('scriptname.py')`

Example script

```
import numpy as np #standard Python module importing
import os

default(applycal) #default task parameters
vis='myvis.ms' #input task parameters
field='G123.45'
spw='0'
gaintable=['sometable.amp']
gainfield=[2]
applycal() #execute task
```

Simple example, apply cal for one spectral window

```
import numpy as np #standard Python module importing
import os

for j in range(4):
        default(applycal) #default task parameters
        vis='myvis.ms' #input task parameters
        field='G123.45'
        spw=str(j)
        gaintable=['sometable.amp']
        gainfield=[2]
        applycal() #execute task
```

Slightly more advanced example, looping through each spectral window

CASA Tasks and Tools

TASKS:
Front end, user friend command line functions for data reduction, manipulation in CASA.

Built upon the TOOL kit functions available in CASA.

Typically have a bit more functionality than functions available in GUIs e.g. viewer

vs

TOOLS:
'Under the hood' basic functions upon which tasks are built.

Preform simple tasks but can be useful in image manipulation and some simulation tasks.