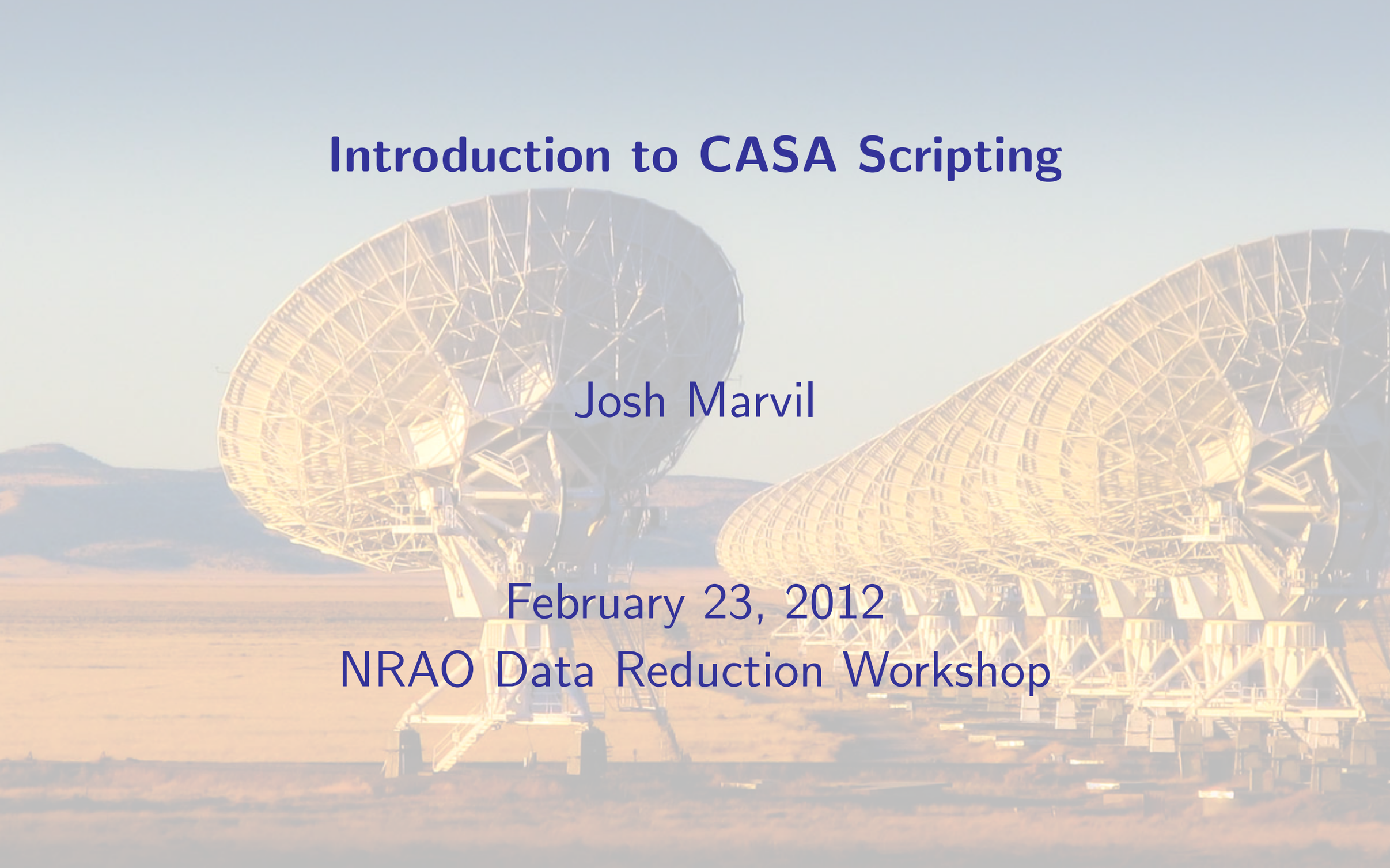


Introduction to CASA Scripting

Josh Marvil

February 23, 2012

NRAO Data Reduction Workshop

Outline of this Talk

A CASA Script
Python Basics
The CASA Toolkit
Writing CASA Tasks
Examples, Tips, Tricks

A CASA Script

- Example Script: G55.7 Tutorial

```
setjy(vis='G55.7+3.4_10s.ms', field='0542*',
          spw='2~3,5~6', modimage='3C147_L.im')

gaincal(vis='G55.6+3.4_10s.ms', caltable='G55.6+3.4_10s.G0',
     spw='2~3,5~6',  solint='int', calmode='p', field='0542*')
```

- Name your script almost anything you want, *e.g.* myScript.py

A CASA Script

- Run your script in CASA:

```
CASA>> execfile('myScript.py')
```

- Run your script from the terminal:

```
bash$ casapy -c myScript.py
```

- Run your script remotely:

```
bash$ nohup casapy -c myScript.py &
```

A CASA Script

- Another Example: listobs.last

```
taskname              = "listobs"
vis                   =  "G55_split1.ms"
verbose               =  True
listfile              =  "G55_split1.ms.listobs.txt"

#listobs(vis="G55_split1.ms",verbose=True,
        listfile="G55_split1.ms.listobs.txt")
```

Python Basics

Strings, Lists
for, if, else
Dictionary
Numpy Array

Python Basics

- Data types

```
x=3        x=3.0        x=3e0        x='3'        x=[3]
```

- Index selection

```
x = 'abc'                          x = [ 'a', 'b', 'c' ]
                                         0    1    2    3
a = x[ 0 ]
b = x[ 1:2 ]
c = x[ -1: ]
```

Python Basics

- Adding strings

```
ab = 'a' + 'b'
abc = ab + 'c'
```

- Adding lists

```
x = [ 1, 2 ] + [ 3, 4 ]
x = x + [ 5 ]
```

Python Basics

- Example Script: G55.7 Tutorial

```
setjy(vis='G55.7+3.4_10s.ms', field='0542*',
          spw='2~3,5~6', modimage='3C147_L.im')

gaincal(vis='G55.6+3.4_10s.ms', caltable='G55.6+3.4_10s.G0',
     spw='2~3,5~6',  solint='int', calmode='p', field='0542*')
```

Python Basics

- Python strings in scripts

```
vis = 'G55.7+3.4_10s.ms'
field = '0542*'
spw = '2~3,5~6'
modimage = '3C147_L.im'

setjy(vis=vis, field=field, spw=spw,  modimage=modimage)

gaincal(vis=vis, caltable=vis[ :-2 ]+'G0', field=field,
        spw=spw,  solint='int', calmode='p')
```

Python Basics

- Conditional statements and logical operators

```
x = 3.0

if (x == '3'): print 'this will not happen'
elif (x > 5.0): print 'this will not happen either'
else: print 'this will happen'
```

Python Basics

- Objects with length can be iterated

```
x = 'abc'

for item in x:  print item
```

- *E.g.* clean multiple fields

```
allFields = [ '0', '3', '7' ]

for field in allFields:
    clean( field = field ...
```

Python Basics

- The Python dictionary

```
x = { 'firstKey' : 3.0, 'secondKey' : 'a' }
x[ 'thirdKey' ] = [ 1, 2 ]

a = x[ 'secondKey' ]
x_keys = x.keys()
```

Python Basics

- The Python dictionary

```
raster = { 'file' : 'DSS_poss1_red.image',
           'colormap' : 'Greyscale 1'  }

contour = { 'file' : 'EVLA_Cband.image',
            'levels' : [ 1, 2, 3, 5 ],
            'base' : 0, 'unit' : 0.45 }

imview(raster=raster, contour=contour, out='filename.ps')
```

Python Basics

- The Numpy array

```
x = pl.array( [1,2,3,4] )
y = 2*x**3 - 4

y2 = y[ y > 2 ]
x2 = x[ y > 2 ]

x = pl.linspace( 1, 10, 100 )
x_log = pl.logspace( 0, 1, 100 )
noise = pl.randn( 100 )
```

The CASA Toolkit

Quanta

Measures

Tables

Image Analysis

Pylab

The CASA Toolkit

- The quanta tool (qa)

```
x = qa.quantity( 12.4, 'deg' )
x_rad = qa.convert( x, 'rad' )
x_dms = qa.angle( x )
x_hms = qa.time( x )
```

The CASA Toolkit

- The measures tool (me)

```
me.doframe( me.observatory( 'VLA' ) )
me.doframe( me.epoch( 'utc', thisTime ) )
mySun = me.measure( me.direction( 'SUN' ), 'AZELGEO' )
```

The CASA Toolkit

- Opening tables with browsetable

```
CASA>> browsetable('myTable')

bash$ casabrowser myTable
```

- Opening tables with the tb tool:

```
tb.open( 'myTable' )
myData = tb.getcol( 'DATA' )
tb.close()
```

The CASA Toolkit

- Using the table query language (TaQL)

```
tb.open( 'myTable'  )
stb = tb.query( 'ANTENNA1 == 3 && FIELD_ID == 1' )
myGains, myTimes = stb.getcol( 'GAIN' ), stb.getcol( 'TIME' )
tb.close()
```

The CASA Toolkit

- Using the imval task

```
myData = imval( imagename = 'myImage' , region = 'myRegion' )
```

- Opening images with the ia tool:

```
ia.open( 'myImage' )
myData = ia.getregion( region = 'myRegion' )
ia.close()
```

The CASA Toolkit

- The pl tool (matplotlib)

```
pl.plot( x )
pl.plot( x, y, 'k--' )
pl.errorbar( x, y, xerr=xerr, yerr=yerr )

pl.semilogy( x, y, 'bo', ms=4 )
pl.hist( x, bins=pl.linspace(0,20,21) )
pl.imshow( X, cmap = 'gray')
```

The CASA Toolkit

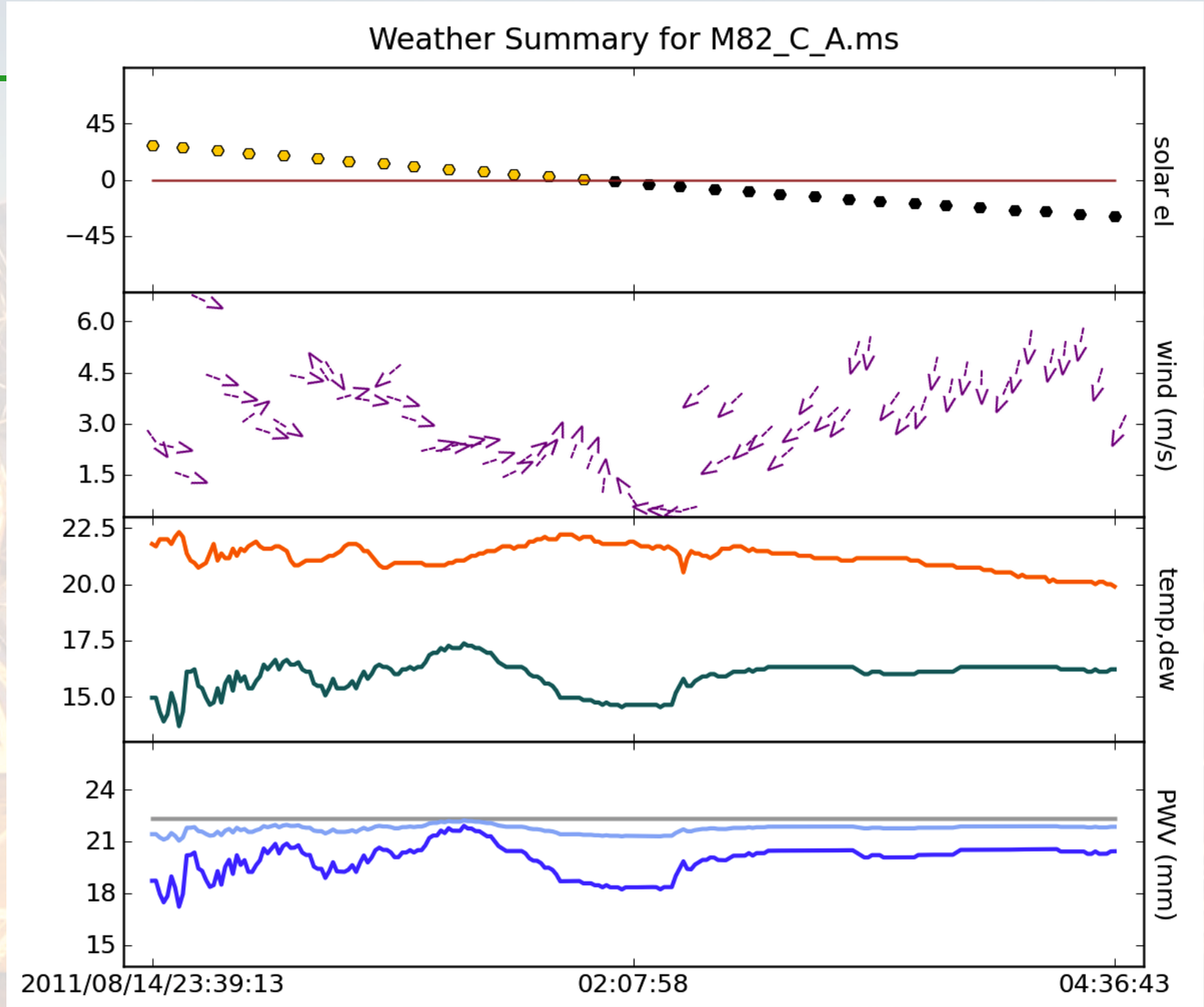
- More with matplotlib

```
pl.plot( x1, y1, 'bo', label= 'firstLabel' )
pl.plot( x2, y2, 'gd', label= 'secondLabel' )
pl.legend()

pl.title( 'Sample Title' )
pl.xlabel( 'Sample Label' )
pl.text( 3, 4.5, 'Sample Text' )
```

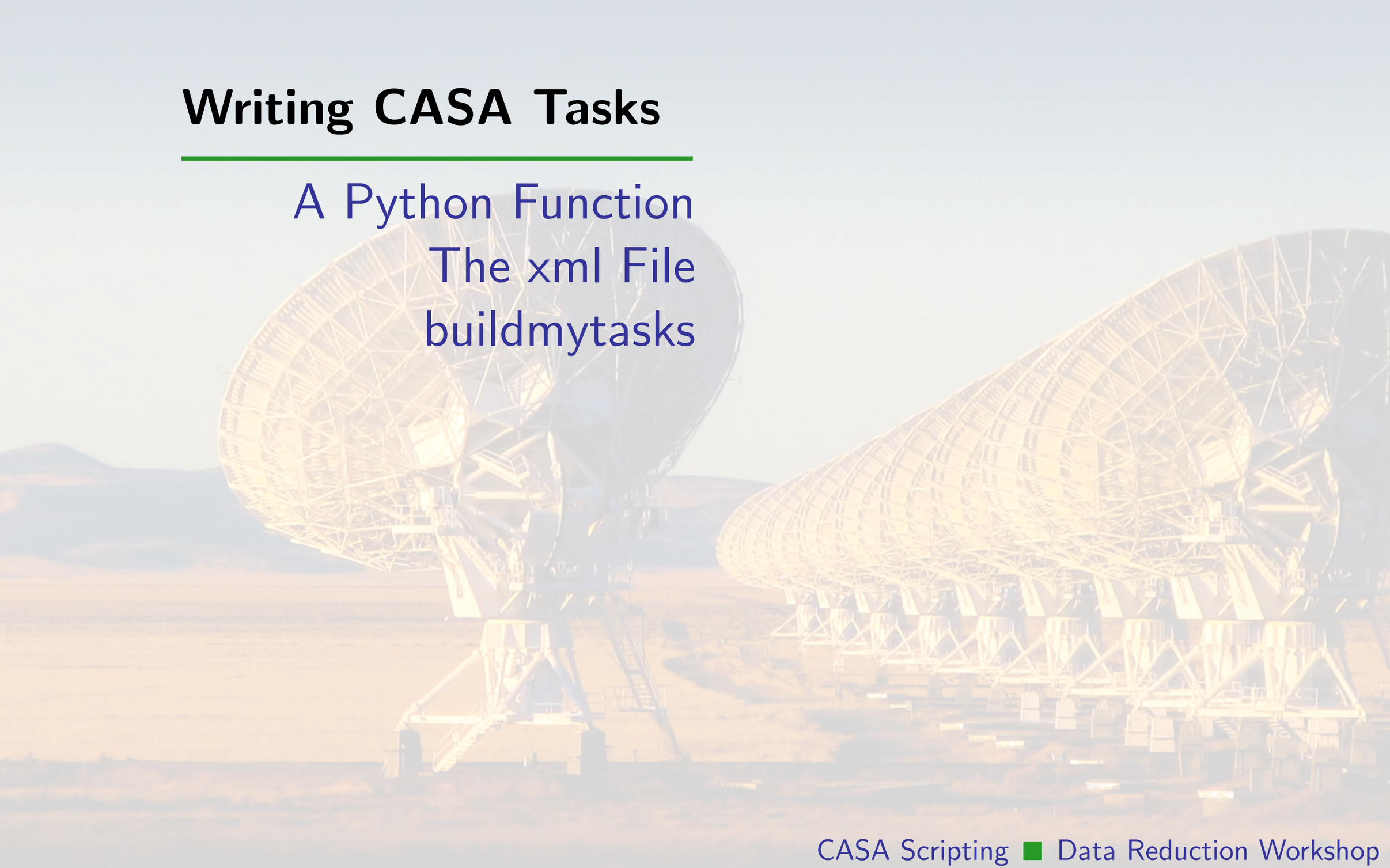
plotweather

- the tb tool
- the qa tool
- the me tool
- the pl tool



Writing CASA Tasks

A Python Function
The xml File
buildmytasks



Writing CASA Tasks

- Turn your script into a Python function

```
from casa import table as tb
...
import pylab as pl

def plotWX(vis='', seasonal_weight=0.5, doPlot=True):

    tb.open( vis + '/WEATHER' )
    myTimes = tb.getcol( 'TIME' )
    ....
```

Writing CASA Tasks

- Write the xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" ?>
<casaxml xmlns="http://casa.nrao.edu/schema/psetTypes.html"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://casa.nrao.edu/schema/casa.xsd
file:///opt/casa/code/xmlcasa/xml/casa.xsd">

<task type="function" name="plotWX">
...
</task>
</casaxml>
```

Writing CASA Tasks

- Inside the `task` tag

```
<shortdescription> short description </shortdescription>
<input>
<param type="string" name="vis" kind="ms" mustexist="true">
<description>MS name</description>
<value></value>
</param>
...
</input>
<example>example text</example>
```

Writing CASA Tasks

- Name your function file task_taskname.py
- Name your xml file taskname.xml
- Building and importing your task

```
CASA>> !buildmytasks taskname
CASA>> execfile( 'mytasks.py' )
CASA>> inp taskname
```

Tips for getting started

- Take advantage of the CASA command line (ipython)
  - TAB completion
  - Object Inspection
    - print x, type(x), len(x)
    - pl.shape(x) or x.shape

Tips for getting started

- Use a Python-friendly text editor
  - syntax highlighting, block quote and indent
  - kwrite, gedit, emacs

Tips for getting started

- Look at examples –
  - `http://casaguides.nrao.edu`
  - `http://casa.nrao.edu/casa_cookbook.pdf`
  - `http://casa.nrao.edu/docs/casaref/CasaRef.html`
- This talk can be found here:
  - `https://science.nrao.edu/facilities/evla/early-science/DRW-spring2012/CASA-Scripting.pdf`

Python Basics

- Searching strings

```
x = 'abc'
myIndex = x.find( 'b' )
```

- Searching lists

```
x = [ 'a', 'b', 'c' ]
myIndex = x.index( 'b' )
```

Python Basics

- Handling errors in your script:

```
x = [ 2, 3 ]
searchThis = 1
stopOnError = True

try:
    myIndex = x.index( searchThis )

except:
    print 'index not found: ', searchThis
    if stopOnError: raise
```

Python Basics

- Open and parse a text file:

```
for line in open('myText.txt', 'r'):
  line1 = line.split(' ')
```

- Append a value to a text file:

```
x = 3.0
out1 = open('myText.txt', 'a')
out1.write( str(x) + '\n' )
out1.close()
```

Python Basics

- Running commands from the system shell

```
os.system( 'xv myPlotFile.png &' )

os.system( 'pdflatex myTexFile.tex' )

os.system( 'mutt -s '+thisSubject+' -a '+thisAttachment+ \
           '  '+thisAddress+' < '+thisBody )
```

Retrieving Data from CASA Objects

- Opening the ms main table

```
ms.open( 'myMS.ms')
ms.iterinit( interval=1000 )
ms.iterorigin()
moretodo=True

while moretodo:
  origDat = ms.getdata( items= 'data' );
  moretodo = ms.iternext();

ms.close();
```

Fitting Data using Built-in Methods

- scipy.linalg.lstsq

```
from scipy.linalg import lstsq

A = pl.vstack( [x, np.ones(len(x))] ).T
myCoef, res, rank, s = lstsq( A, y )
myFit = myCoef[1] * x + myCoef[0]
```

Fitting Data using Built-in Methods

- scipy.linalg.lstsq

```
from scipy.linalg import lstsq

A = pl.vstack( [pl.sqrt(x)] ).T
myCoef, res, rank, s = lstsq( A, y )
myFit = myCoef * pl.sqrt(x)
```

Fitting Data using Built-in Methods

- pl.polyfit (from numpy.lib.polynomial)

```
myCoef = pl.polyfit( x, y, deg=1 )
myFit = myCoef[1] * x + myCoef[0]
```

- scipy.stats.linregress

```
from scipy.stats import linregress

m, b, r, p, m_err = linregress( x, y )
myFit = m * x + b
```

Fitting Data using Built-in Methods

- scipy.optimize.curve_fit

```
from scipy.optimize import curve_fit

fp = lambda x, p1, p2, p3: p1/(x**p2)*pl.sin(p3*x)

p, p_cov = curve_fit( fp, x, y, p0=p0, sigma=y_err )
myFit = fp( x, p[0], p[1], p[2] )
```

Fitting Data using Custom Methods

- Weighted, non-linear least squares from [R]

```
import rpy2.robjects as robjects

robjects.globalenv["x"] =  robjects.FloatVector(x)
robjects.globalenv["y"] =  robjects.FloatVector(y)

myNLS = robjects.r.nls('y~A0*sin(x)+ A1*exp(-2*x)')
myCoef = robjects.r.summary(myNLS).rx('coefficients')[0]
myFit = myCoef[0] * sin(x) + A1 * exp(-2*x)
```

Plotting Data using Built-in Methods

- 3d plots with matplotlib

```
import mpl_toolkits.mplot3d.axes3d as p3

ax = p3.Axes3D( pl.figure() )

ax.plot_wireframe( X, Y, data )
ax.plot_surface( X, Y, data, color='red', alpha=.3 )
ax.plot3D( X.ravel(), Y.ravel(), data.ravel(), 'gd' )
```