A Very Brief Introduction to Python

Emily Wolin

Goal: Overview of Python basics, with a few hands-on examples

What is Python?

- Interpreted
- High-level
- Free and open-source
- Object-oriented: each object has various *attributes* and *methods*

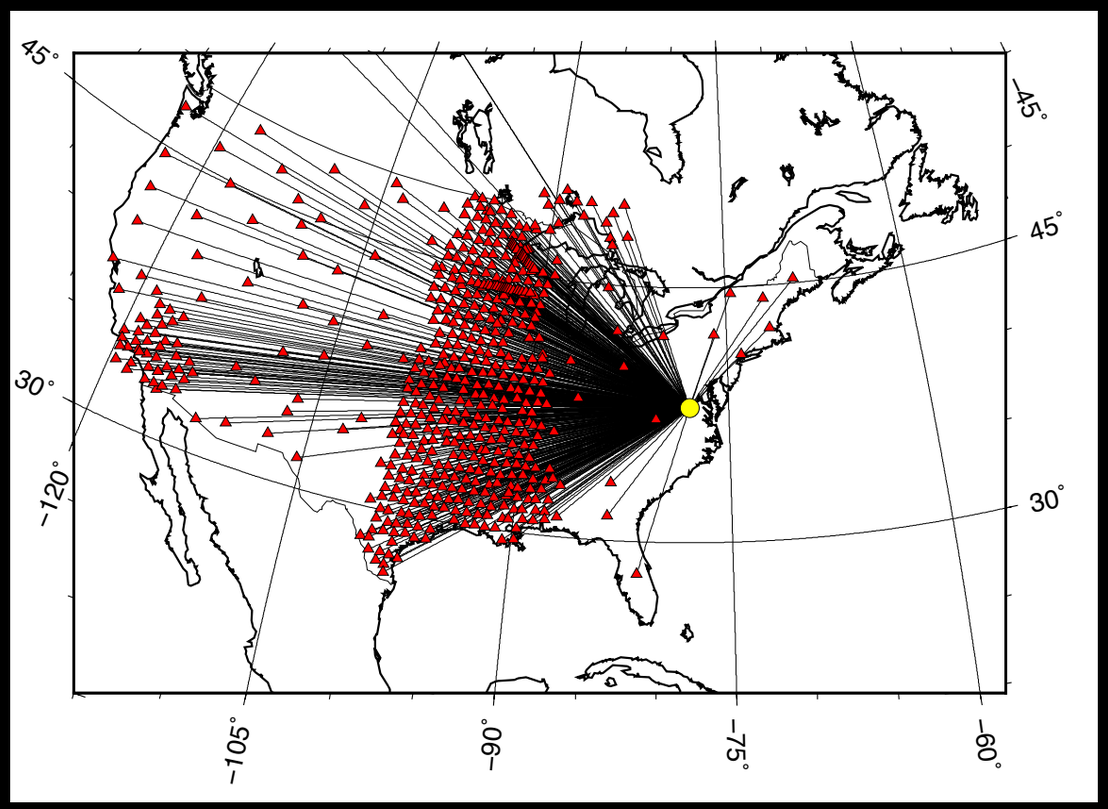
Why do I use Python?

"How good are current tomographic models of North America?"

Why do I use Python?

~~"How good are current tomographic models of North America?"~~

"We can use current tomographic models to predict *S* and Rayleigh wavetrains from earthquakes within North America. Do these synthetic seismograms agree with observations at SPREE and the TA?"



Why do I use Python?

Things I need to know to compare two waveforms:

- Time series of ground motion (of course)
- Station, network, component, location
- Lat, lon, depth of station
- Start and end time of traces
- Sampling rate
- Event hypocenter and origin time
- Phase picks

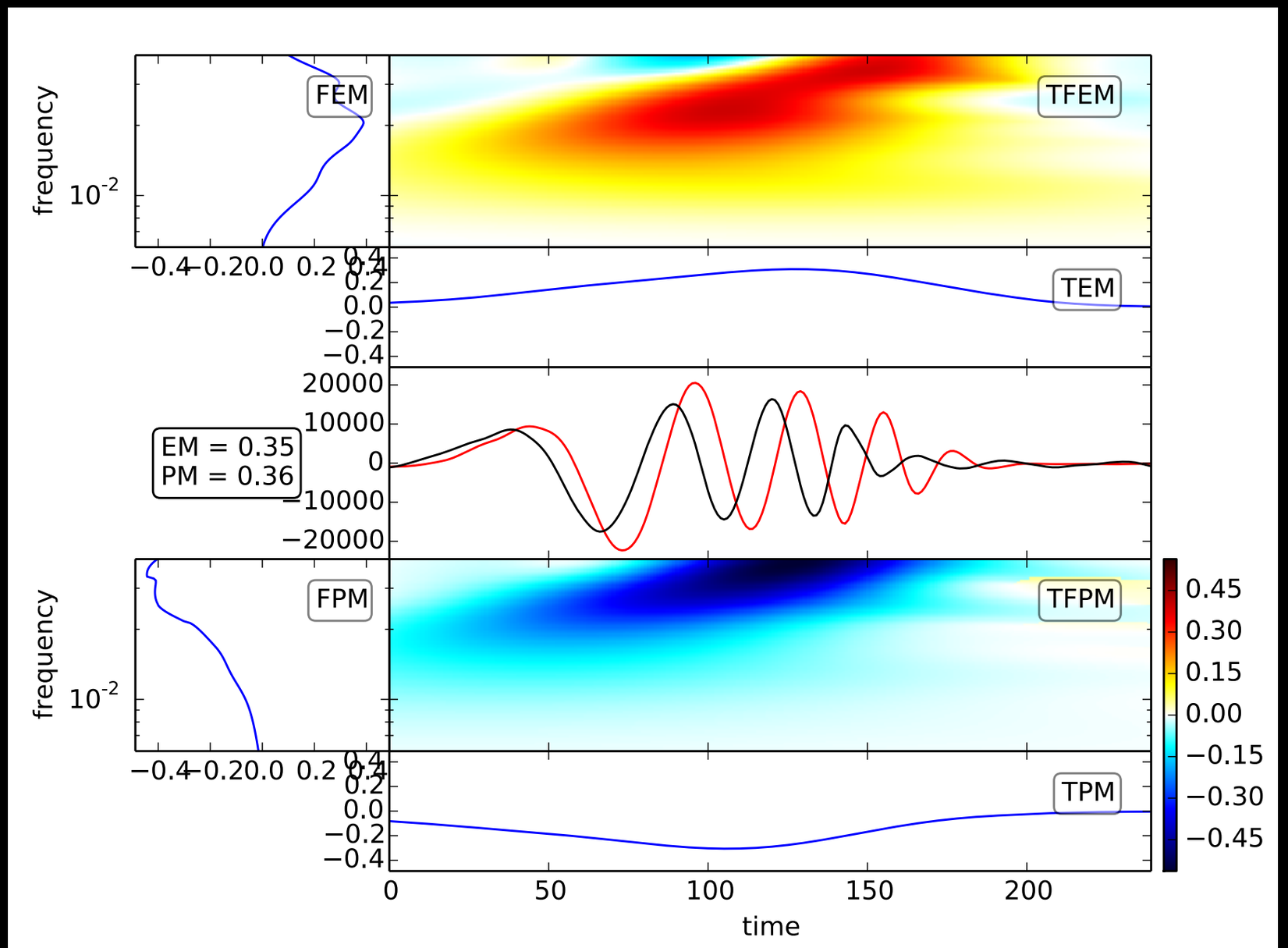
SAC deals with headers nicely, and provides a lot of nice functions.

But I *hate* writing SAC macros.

And I need to do things that SAC can't do.

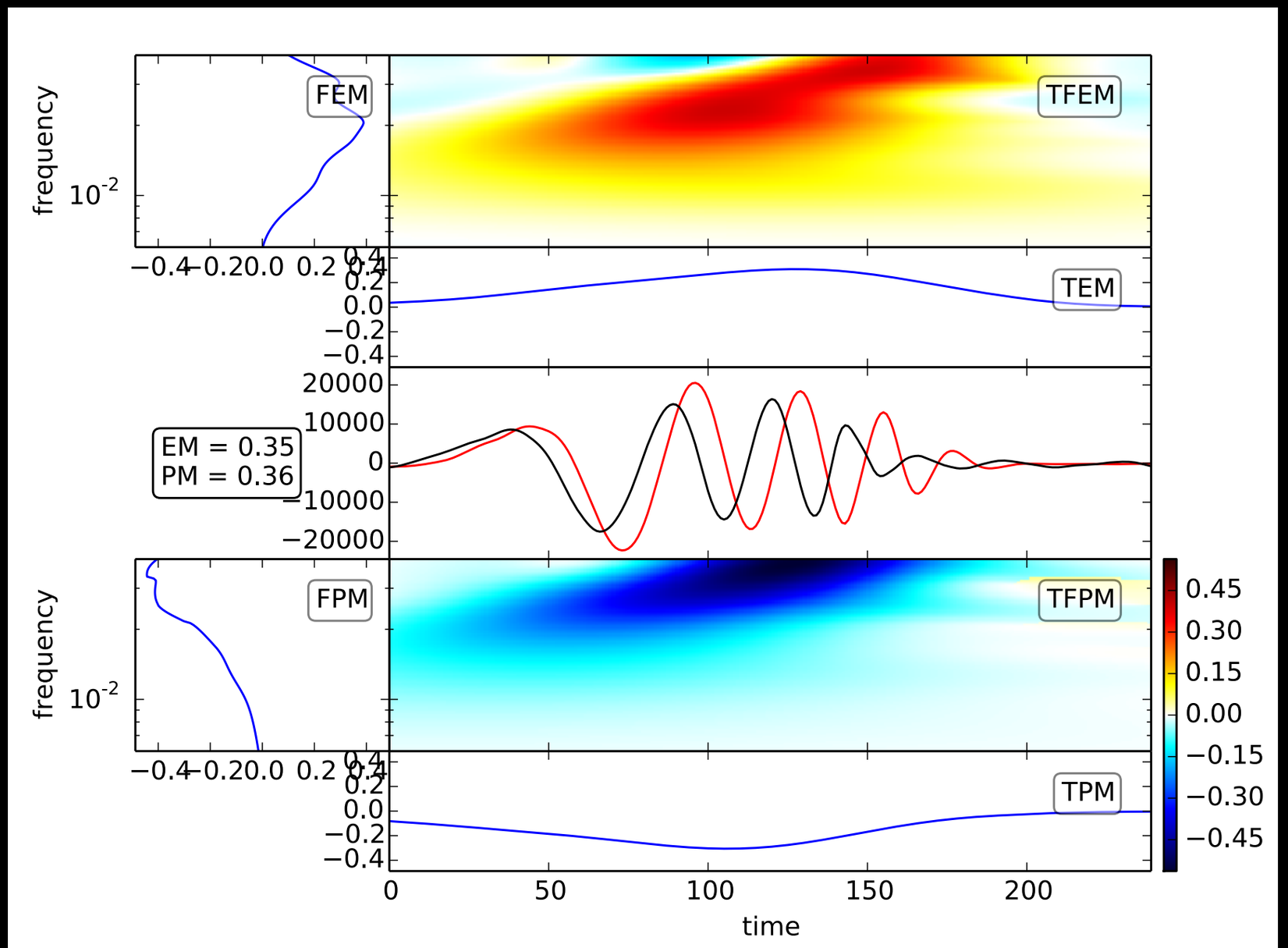
Why do I use Python?

Goal: Measure misfit between observed and synthetic seismograms



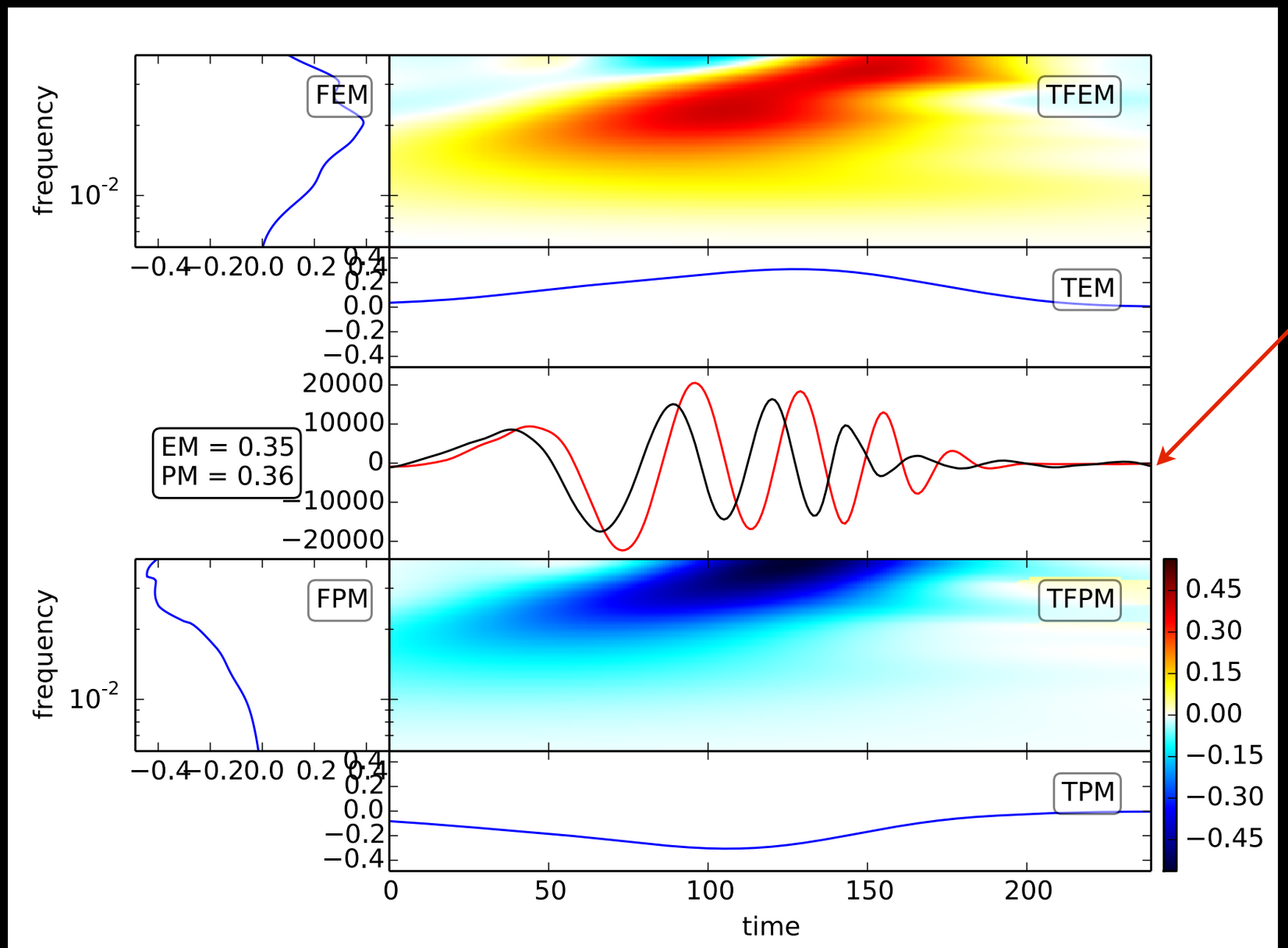
Why do I use Python?

Method: Calculate time-frequency misfit



Why do I use Python?

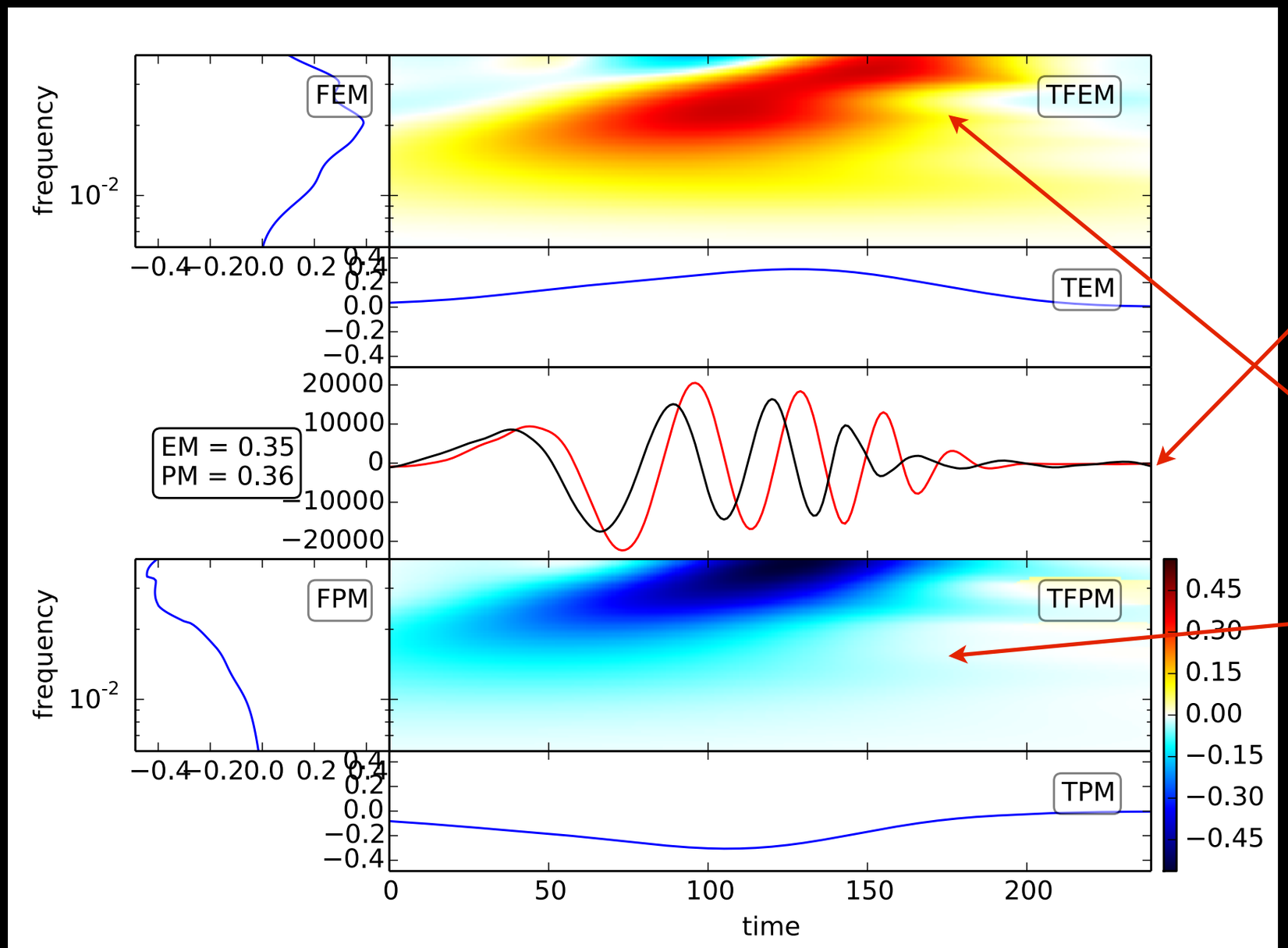
Method: Calculate time-frequency misfit



Input: two data arrays

Why do I use Python?

Method: Calculate time-frequency misfit

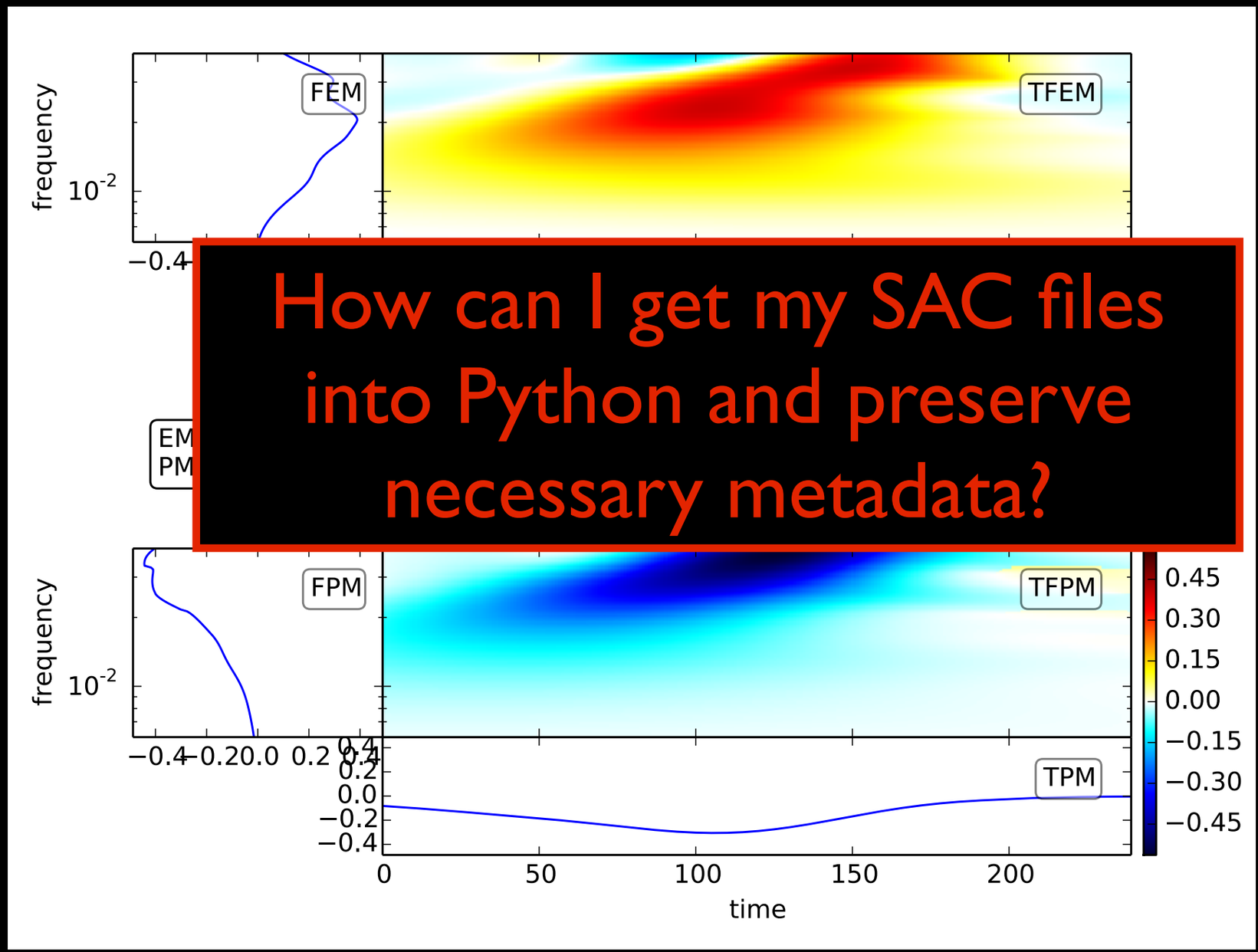


Input: two data arrays

Output: amplitude and phase differences

Why do I use Python?

Method: Calculate time-frequency misfit



How can I get my SAC files into Python and preserve necessary metadata?

Why do I use Python?

In Python, I could build an *object* containing the time series and the headers.

```
mytrace.header.stnm
mytrace.header.stla
mytrace.data
```

} *attributes*

I could also define functions that act on or modify the object's attributes.

```
mytrace.trim(starttime=t1, endtime=t2)
mytrace.filter("highpass", freq=0.02)
mytrace.remove_response(output="DISP")
```

} *methods*

Note: these are all pseudocode

Why do I use Python?

Good news:

Seismology-friendly data structures already exist in a package called ObsPy.

Even better news:

I can use powerful numerical and scientific Python libraries to process and visualize my data.

Tip: Use a text editor with syntax highlighting.

In Vim (my favorite), set up ~/.vimrc:

```
syntax on
set number
set noeb vb t_vb=
```

(turn on syntax highlighting and line numbers; turn off annoying beeps)

Getting started: IPython

In a terminal window, type `ipython --pylab`

Getting started: IPython

In a terminal window, type `ipython --pylab`

```
print 'Hello, world!'
```

Getting started: IPython

```
a='1'       str
a=1         int
a=1.1       float
a=1+2j      complex
```

Getting started: IPython

```
a=[42,17,6]
```
← a *list*

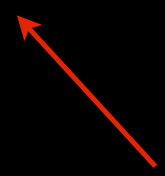
```
a[1]
```

```
a.<tab>
```

in IPython, use <tab> and ? to explore attributes/methods of an object

Getting started: IPython

```
s='hello there    ' ← a string
s[1]
s.<tab>
```



in IPython, use <tab> and ? to explore attributes/methods of an object

Getting started: IPython

```
if answer != 42:
    print ‘that is not correct’
for i in range(5):
    print ‘Hello, world!’
```

Spaces! (4)

Using modules

Python doesn't load modules unless you ask for them

```
import os
help(os)
os.environ
myname=os.environ['USER']
```

← a *dictionary*

Write your own module

In a file called mymodule.py:

```
import os
def sayhello():
    myname=os.environ['USER']
    print 'Hello, {0}!'.format(myname)

def addthese(a,b):
    c=a+b
    print '{0}+{1}={2}'.format(a,b,c)
    return c
```

← a *function*

← a *function* with *arguments*

Use your new module

In the IPython shell:

```
import mymodule
mymodule.sayhello()
a=mymodule.addthese(3.1,2.7)
```

Or, if your fingers are getting tired:

```
import mymodule as mm
mm.sayhello()
a=mm.addthese(3.1,2.7)
```

Using classes

We can define a class (an object) that has its own attributes and methods.

```
from birdclasses import Swallow
bird=Swallow(species='African', status='unladen')
bird.loadstatus
bird.velocity
```

http://style.org/unladenswallow/

Using classes

We can define a class (an object) that has its own attributes and methods.

```
from birdclasses import Swallow
bird=Swallow(species='African', status='unladen')
bird.loadstatus
```

```
bird.velocity
```

Try giving the bird a coconut:

```
bird.giveCoconut()
```

What is the airspeed velocity of an unladen swallow?

http://style.org/unladenswallow/

Useful Modules

NumPy:

"the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities"

http://www.numpy.org/

Useful Modules

SciPy:

"a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics and much more"

http://www.scipy.org/about.html

Useful Modules

matplotlib:

"matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms"

⟶ pylab gives MATLAB-like syntax

http://matplotlib.org/

Useful Modules

ObsPy:

“an open-source project dedicated to provide a Python framework for processing seismological data. It provides parsers for common file formats and seismological signal processing routines which allow the manipulation of seismological time series”

http://docs.obspy.org/

Python Basics

Now you know a little about:

- Data types
- Object orientation: attributes, methods
- Flow control statements (if/while/for): mandatory indentation
- Importing modules and writing simple functions

And now for something a little different...

Scripting with Python

Sample code (mylinefit.py):

- Generate a linearly-spaced array of x values
- Calculate a function y(x) at each point
- Add random noise to y(x)
- Fit a line through the noisy data
- Plot the noisy data, original function, and best-fit line

Scripting with Python

Run script from IPython: `run mylinefit.py`

Scripting with Python

Run script from IPython: `run mylinefit.py`

Run in Terminal:

```
chmod +x mylinefit.py
./mylinefit.py
```

(The first line of mylinefit.py should be

```
#!/usr/bin/env python
```

so the shell knows this is a Python program.)

Scripting with Python

Fit line in 2 ways:

- SciPy: stats.linregress module
- "By hand" with NumPy matrices and some inverse theory

$$y_n = ax_n + b$$

$$\mathbf{Gm} = \mathbf{d}$$

$$\mathbf{m} = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{d}$$

$$\mathbf{G} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}$$

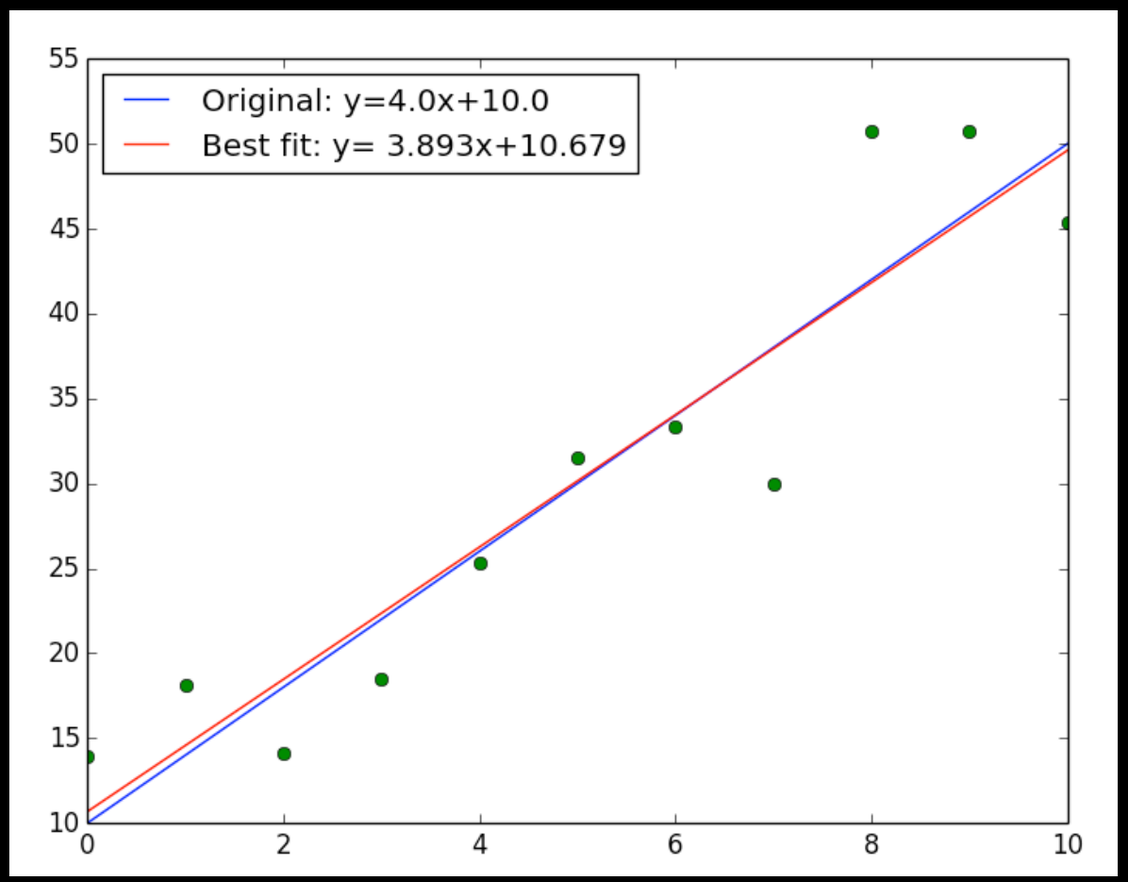
$$\mathbf{m} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Scripting with Python

Fit line in 2 ways:

- SciPy: stats.linregress module
- "By hand" with NumPy matrices and some inverse theory



$$y_n = ax_n + b$$

$$\mathbf{Gm} = \mathbf{d}$$

$$\mathbf{m} = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{d}$$

$$\mathbf{G} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}$$

$$\mathbf{m} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Read and plot a waveform

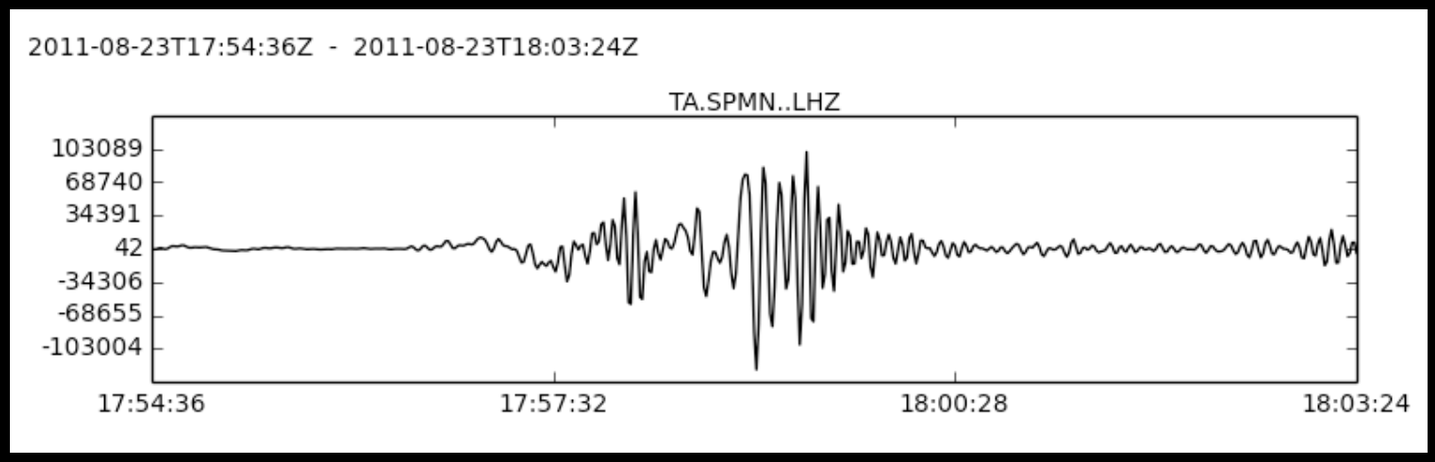
```
from obspy.core import read
st = read('TA.SPMN..LHZ.disp')
st.plot()
```

Also try:

```
print st
len(st)
tr = st[0]
print tr
print tr.stats
print tr.stats[‘station’]
tr.data
```

Read and plot a waveform

```
from obspy.core import read
st = read('TA.SPMN..LHZ.disp')
st.plot()
```



Also try:

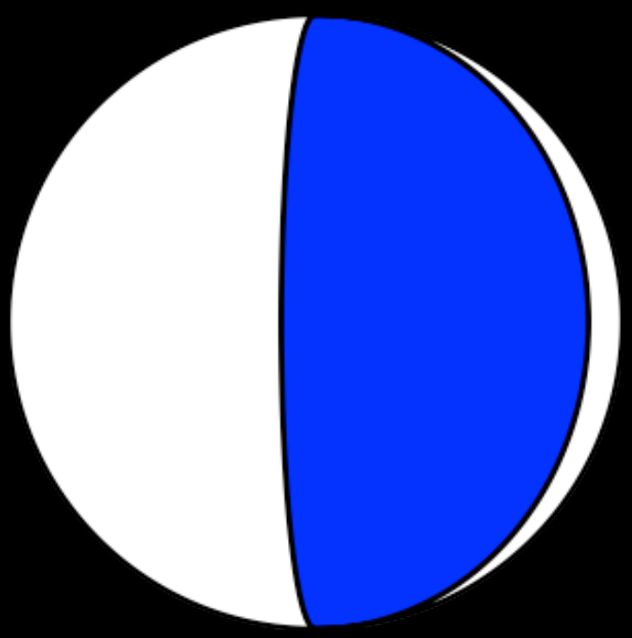
```
print st
len(st)
tr = st[0]
print tr
print tr.stats
print tr.stats['station']
tr.data
```

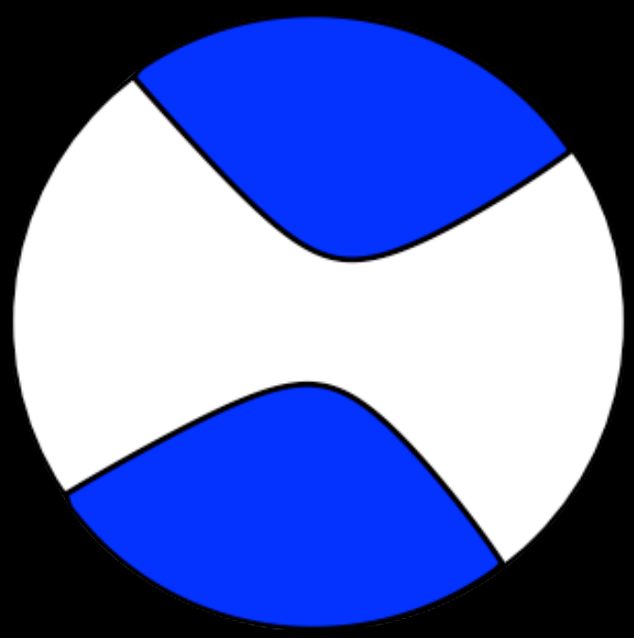
Plot a focal mechanism

```
from obspy.imaging.beachball import Beachball
mt = [ 180, 80, 90 ]
Beachball(mt, size=500)

mt2 = [-0.463, 4.61, -4.15, -0.0633, -0.171, -1.49]
Beachball(mt2, size=500)
```

Plot a focal mechanism

```
from obspy.imaging.beachball import Beachball
mt = [ 180, 80, 90 ]
Beachball(mt, size=500)

mt2 = [-0.463, 4.61, -4.15, -0.0633, -0.171, -1.49]
Beachball(mt2, size=500)
```





Scripting with Python

Plot any two-column ASCII file from Terminal:

Scripting with Python

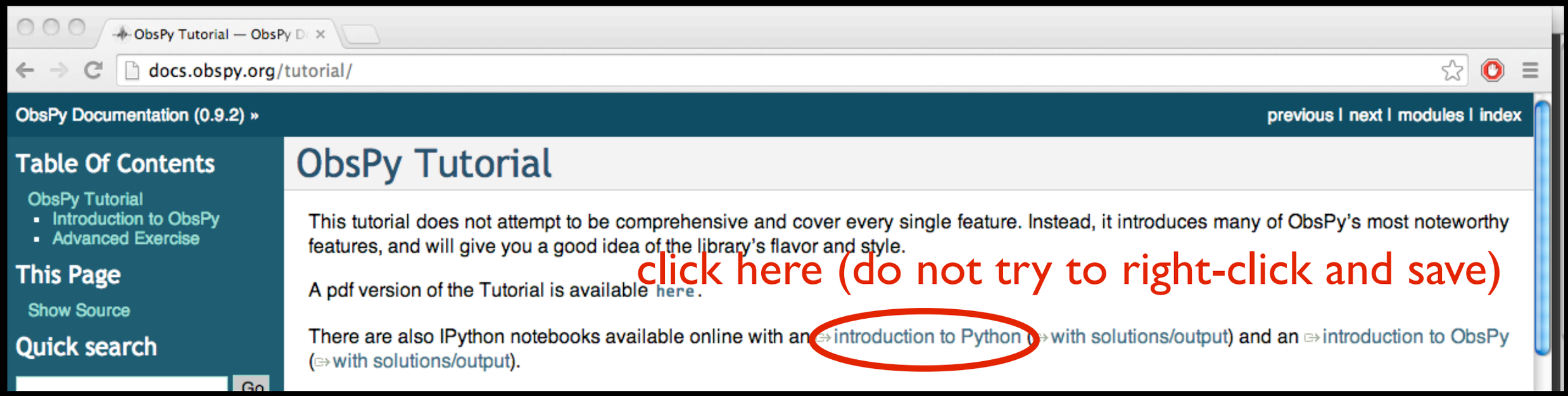
Plot any two-column ASCII file from Terminal:

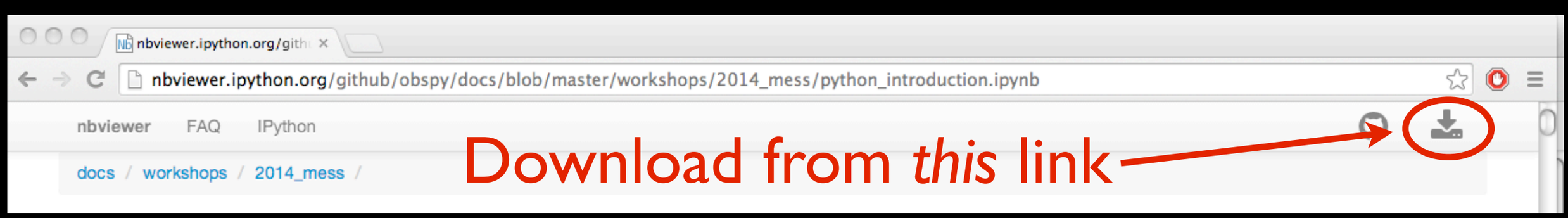
```
chmod +x plotanything.py

./plotanything.py vs.ak135

./plotanything.py vs.*
```

(Again, note the #!/usr/bin/env python in the first line of plotanything.py)

If we have time: IPython notebooks

http://docs.obspy.org/tutorial





Then run the notebook (will open in a browser):

```
ipython notebook python_introduction.ipynb --pylab inline
```

Want to learn more?

- *A Byte of Python*

  http://swaroopch.com/notes/python/

- *ObsPy tutorials:*

  http://docs.obspy.org/tutorial/

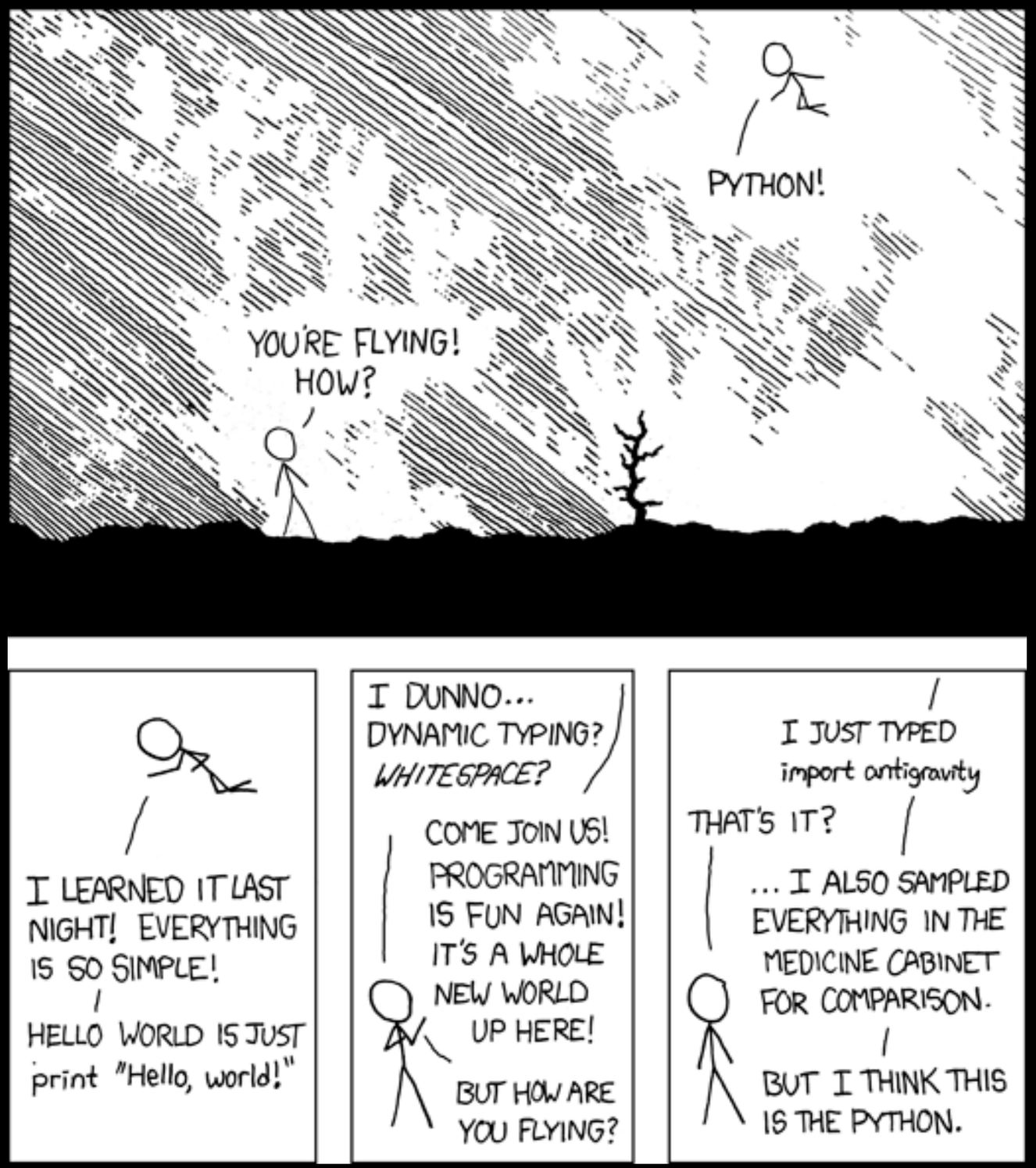
- *Python Scripting for Computational Science*

  http://folk.uio.no/hpl/scripting/index.html

- *Python Scientific Lecture Notes*

  http://scipy-lectures.github.io/index.html

Thank you!



http://xkcd.com/353/

Read and plot a waveform

```
from obspy.core import read
st = read('http://examples.obspy.org/RJOB_061005_072159.ehz.new')
print st
len(st)
tr = st[0]
print tr
print tr.stats
print tr.stats[‘station’]
tr.data
tr.plot()
```

Importing: which style is best?

OK for IPython/quick testing

```
import mymodule
from mymodule import *
```

Better for scripting!

```
import mymodule as mm
from mymodule import sayhello, addthese
```

Beware of name clashes
(and don't trust your memory)

What's an object?

How could you describe an earthquake?

- Hypocenter (lat, lon, depth)
- Origin time
- Rise time/other source-time function
- Moment tensor
- Who contributed the solution (may be several solutions--ISC, PDE, Global CMT, etc.)?
- Event ID

It'd be nice if we had a way of tying all of these different pieces of information together.

Instead of using a bunch of different variables...

M6.0 - Northern Mid-Atlantic Ridge

2014-07-27 01:28:38 UTC

PAGER - GREEN ShakeMap - I

Google Earth KML

Return to the EQ List/Map/Search

Scientific

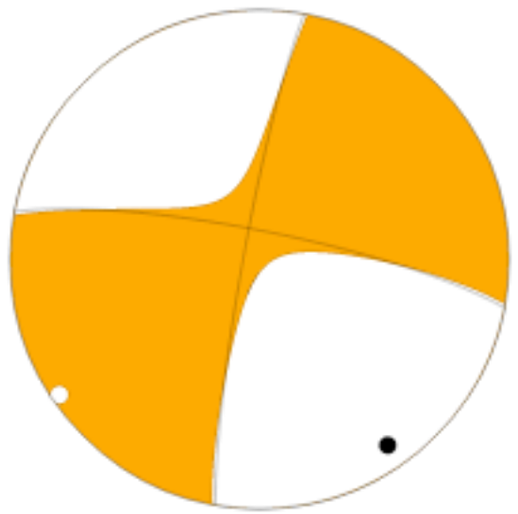
Location and Magnitude contributed by: USGS National Earthquake Information Center

Preferred Location Parameters

Parameter	Value	Uncertainty
Magnitude	6.0 mwb	± 0.03
Location	23.761°N, 45.646°W	Not Specified
Depth	10.0 km	± 1.7 km
Number of Stations Used	Not Specified	
Number of Phases Used	122	
Minimum Distance	1880.0 km (16.89°)	
Travel Time Residual	1.18 sec	
Azimuthal Gap	60°	
Review Status	MANUAL	
Event ID	usb000rxni	

Moment Tensor

US Mww



Instead of using a bunch of different variables...

M6.0 - Northern Mid-Atlantic Ridge

2014-07-27 01:28:38 UTC

PAGER - GREEN ShakeMap - I

Google Earth KML

Return to the EQ List/Map/Search

Scientific

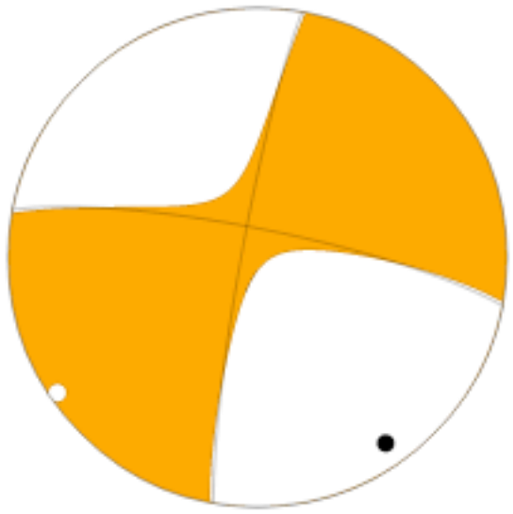
Location and Magnitude contributed by: USGS National Earthquake Information Center

Preferred Location Parameters

evla=23.761

Parameter	Value	
Magnitude	6.0 mwb	± 0.03
Location	23.761°N, 45.646°W	Not Specified
Depth	10.0 km	± 1.7 km
Number of Stations Used	Not Specified	
Number of Phases Used	122	
Minimum Distance	1880.0 km (16.89°)	
Travel Time Residual	1.18 sec	
Azimuthal Gap	60°	
Review Status	MANUAL	
Event ID	usb000rxni	

Moment Tensor



Instead of using a bunch of different variables...

M6.0 - Northern Mid-Atlantic Ridge

2014-07-27 01:28:38 UTC

PAGER - GREEN ShakeMap - I

Google Earth KML

Return to the EQ List/Map/Search

Scientific

Location and Magnitude contributed by: USGS National Earthquake Information Center

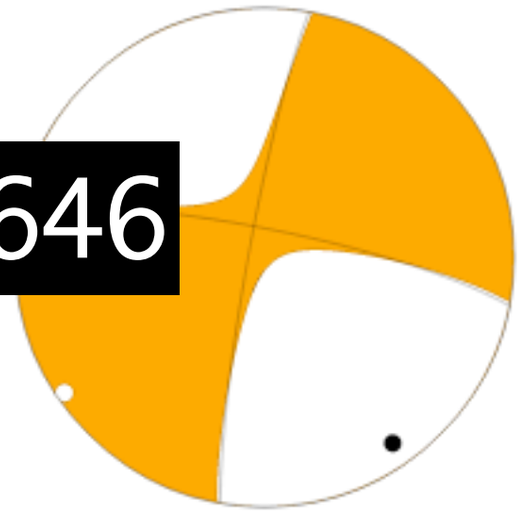
Preferred Location Parameters

Moment Tensor

evla=23.761

evlo=45.646

Parameter	Value	
Magnitude	6.0 mwb	± 0.03
Location	23.761°N, 45.646°W	
Depth	10.0 km	
Number of Stations Used	Not Specified	
Number of Phases Used	122	
Minimum Distance	1880.0 km (16.89°)	
Travel Time Residual	1.18 sec	
Azimuthal Gap	60°	
Review Status	MANUAL	
Event ID	usb000rxni	



Instead of using a bunch of different variables...

M6.0 - Northern Mid-Atlantic Ridge

2014-07-27 01:28:38 UTC

PAGER - GREEN ShakeMap - I

Google Earth KML

Return to the EQ List/Map/Search

Scientific

Location and Magnitude contributed by: USGS National Earthquake Information Center

Preferred Location Parameters

Moment Tensor

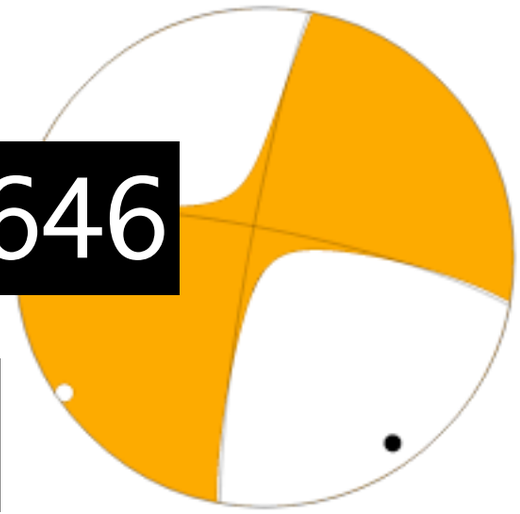
Parameter	Value
Magnitude	6.0 mwb
Location	23.761°N, 45.646°W
Depth	10.0 km
Number of Stations Used	Not Specified
Number of Phases Used	122
Minimum Distance	1880.0 km (16.89°)
Travel Time Residual	1.18 sec
Azimuthal Gap	60°
Review Status	MANUAL
Event ID	usb000rxni

± 0.03

evla=23.761

evlo=45.646

evdp=10.0



Instead of using a bunch of different variables...

M6.0 - Northern Mid-Atlantic Ridge

2014-07-27 01:28:38 UTC

PAGER - GREEN ShakeMap - I

Google Earth KML

Return to the EQ List/Map/Search

origin_time=...?

Scientific

Location and Magnitude contributed by: USGS National Earthquake Information Center

Preferred Location Parameters

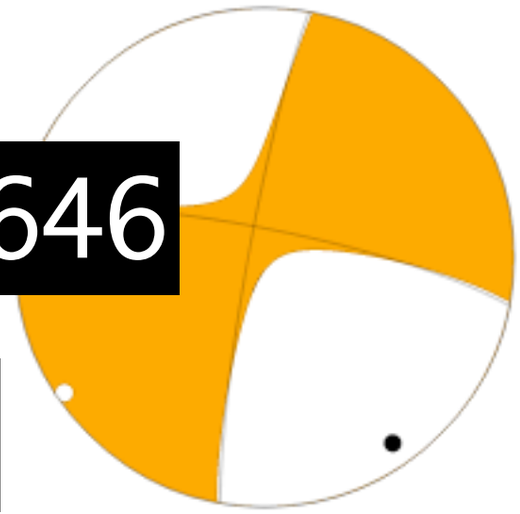
Moment Tensor

evla=23.761

evlo=45.646

evdp=10.0

Parameter	Value
Magnitude	6.0 mwb
Location	23.761°N, 45.646°W
Depth	10.0 km
Number of Stations Used	Not Specified
Number of Phases Used	122
Minimum Distance	1880.0 km (16.89°)
Travel Time Residual	1.18 sec
Azimuthal Gap	60°
Review Status	MANUAL
Event ID	usb000rxni



an Event *object* would help us organize all of this together: