

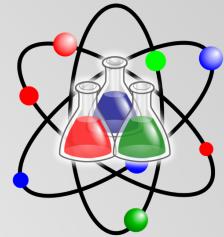
# ObsPy

A Python Framework for Seismology

Developing a scientific software library  
(... and spreading the word)

Tobias Megies, Lion Krischer (...)  
June 2014

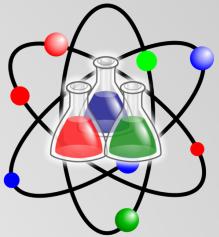
# Software in Science



- Going from research code to distributable and reusable code easily is 5-10 times as expensive
- Gain in hard research currency (e.g. publications) is questionable

## WHY DO IT?

- Science needs it and the need will only increase
- Helps the whole community - no need to do it 100 times (quickly? badly?) if done once but properly
- In the long run, if embraced by all, greatly **reduces the time to research for all**
- Helps with reproducibility - a still unsolved issue



# Software in Science - Issues

- Software development skills

*Not a thoroughly taught skill but many of us spend a lot of time doing it.*

- Sustainability

*How to keep it going after the project finishes?*

- Community building

*Good software without users has little value. How to spread the word?*

- Limited resources in money and time

*Most scientific software is a by-product of actual research; very little funding for software developments.*

- Recognition and rewards

*Not the same value as publications and hard to build an academic career from it.*

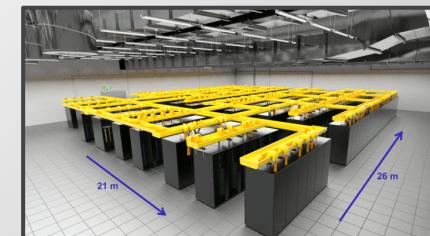
# Outline

1. Introduction to Python and ObsPy
  - a. Why Python?
  - b. Functionality of ObsPy
  - c. Basic Usage Examples
2. Some Technical Details
  - a. Testing
  - b. Code Management and Communication
3. “spreading the word”

# Why Python?

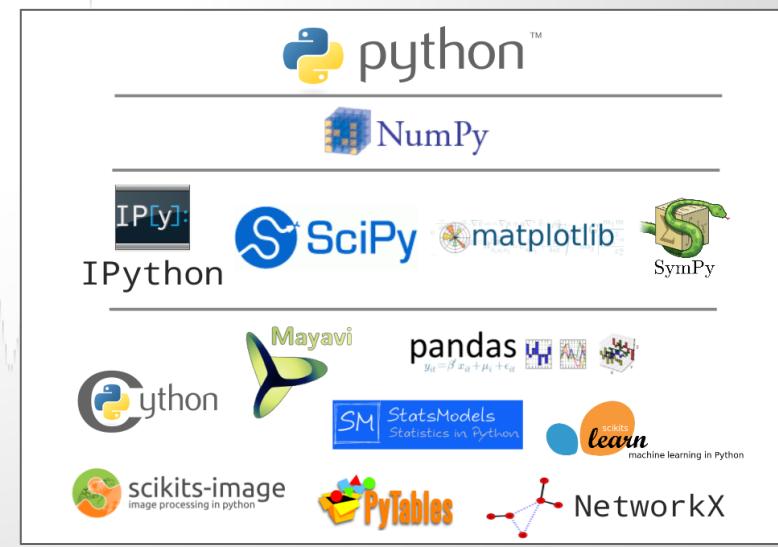
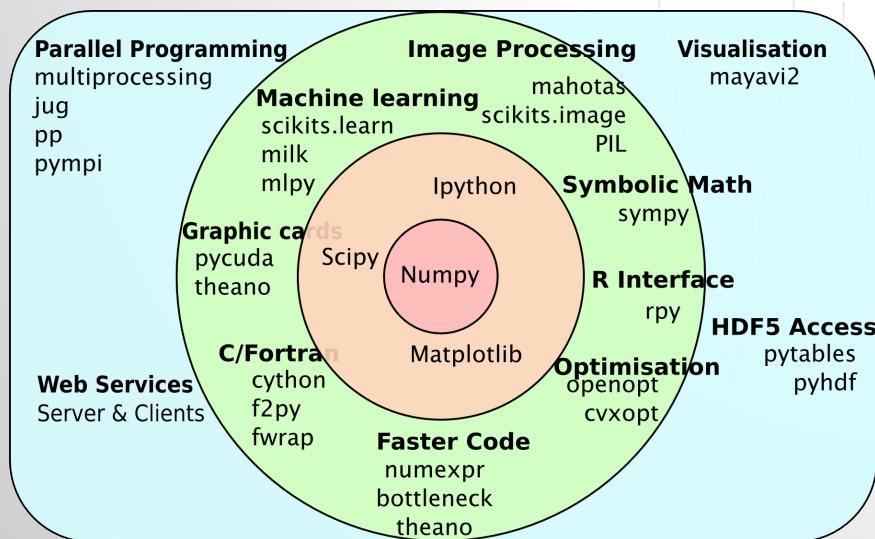


- Widely used in all areas, picking up lots of momentum in many sciences
- Simple, concise, and easy-to-read syntax
- **Free and Open Source**, large scientific community
  - ⇒ potentially high impact / user base
- general purpose programming language
- Cross-platform: from RaspberryPi to large supercomputers



# Why Python?

- No need to compile, interactive shell available
- Easy to interact with existing C and Fortran code
- Vast scientific ecosystem; taking advantage of developments in other sciences



# What is ObsPy?



Python library to work with seismological data

- waveform data
- station metadata
- event metadata

Facilitates development

- from short code snippets
- to complex processing workflows

Develop once, use everywhere

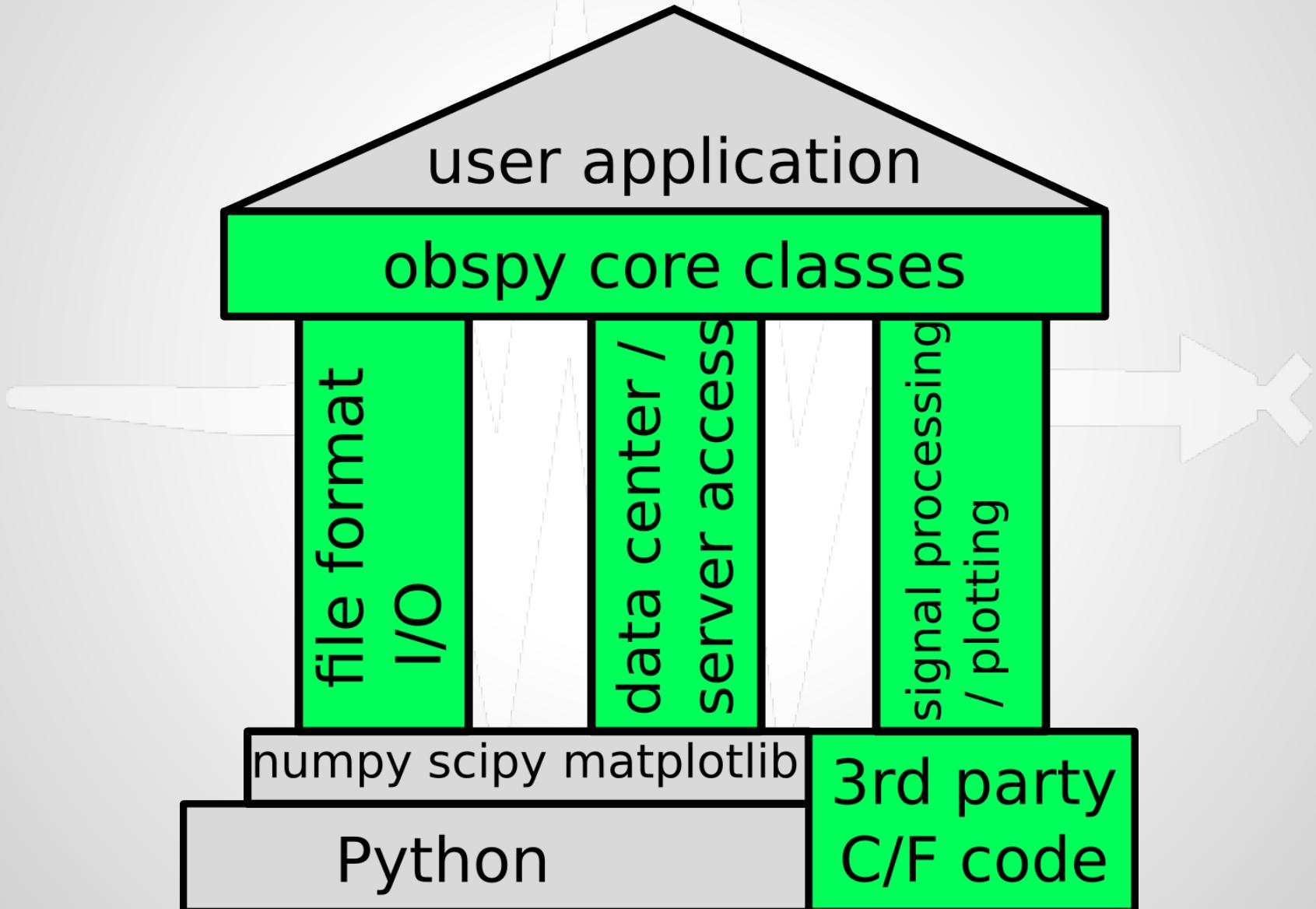
=> a bridge for seismologists into the scientific Python ecosystem

# Processing needs

What I need is..

- .. I/O of local data
- .. fetch data/metadata from data centers
- .. convenient handling of the parsed data/metadata
- .. basic signal processing / data analysis / math
- .. visualization capabilities

# Functionality?



# File Formats

# Situation before ObsPy

GSE2GSE	<b>convseis</b>	<b>manual (postscript)</b>	Onicescu & Rizescu	GSE1.0 <-> GSE2.0 DOS/Windows
GSE2MSEED	<b>codeco_3.3c</b>	<b>Documentation</b>	<b>Urs Kradolfer</b>	GSE1.0/2.x UNIX HP/SUN, Linux
GSE2MSEED	<b>gse2mseed</b>		IRIS, Chad Trabant	GSE 2.x/IMS 1.0, INT or CM6 Solaris, Linux, Mac OSX and Windows
GSE2SAC	<b>codeco_3.3c gsesac</b>	<b>Documentation README</b>	<b>Urs Kradolfer/Hugues Dufumier</b>	GSE1.0/2.x, SAC(A/B)
GSE2SEED	<b>gse2seed (version2.31)</b>	<b>README</b>	<b>ORFEUS, Sleeman</b>	Handling metadata
GSE2SUDS	<b>convseis</b>	<b>manual (postscript)</b>	Onicescu & Ritzescu	GSE1.0 -> PC-SUDS (.DMX)
MARS2MSEED	<b>mars2mseed</b>	<b>Documentation</b>	<b>Chad Trabant</b>	
MSEED2AH	<b>ms2ah</b>	<b>QUG UCB README</b>	IRIS/PASSCAL	-
MSEED2ASCII	<b>mseed2ascii</b>	<b>Documentation</b>	<b>Chad Trabant</b>	
MSEED2CSS	<b>codeco_3.3c</b>	<b>Documentation</b>	<b>Urs Kradolfer</b>	CSS2.8/3.0
MSEED2GSE	<b>codeco_3.3c</b>	<b>Documentation</b>	<b>Urs Kradolfer</b>	GSE1.0/2.x
MSEED2SAC	<b>mseed2sac</b>	<b>Documentation</b>	<b>Chad Trabant</b>	
MSEED2SAC	<b>codeco_3.3c ms2sac</b>	<b>Documentation QUG UCB README</b>	<b>Urs Kradolfer Quanterra Users Group</b>	-



WTF!?

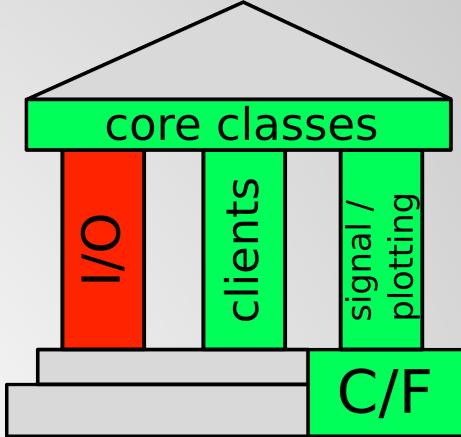
What am I supposed to do with it???

<http://www.demotivation.us/search/all/wtf-1266995.html>

# Functionality?

read/write support for lots of formats:

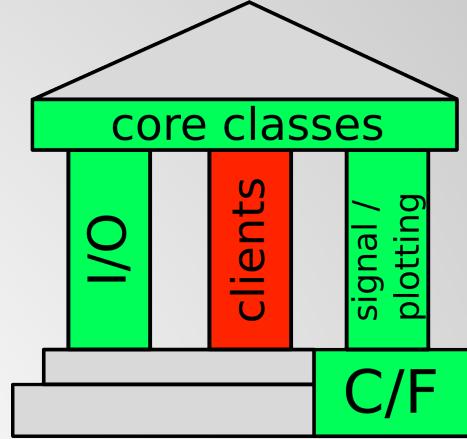
- waveforms (MiniSEED, GSE2, SAC, ...)
  - many different ways to store binary/encoded timeseries
- station metadata (SEED, [StationXML](#))
  - complex, esp. instrument response
- event metadata ([QuakeML](#), NDK, PDE)
  - complex associations, owed to how data is assembled in realtime systems



# Functionality?

data center access (archived data):

- [FDSN web service](#) client
  - [IRIS](#), [Orfeus](#), USGS, RESIF, NCEDC, ...
- ArcLink client: [EIDA](#) / Orfeus
- [SeisHub](#): FFB, local database systems
- ...



server access (near-realtime / ringbuffer data):

- SeedLink, Earthworm
- ⇒ different types of servers,  
but usage of clients very similar

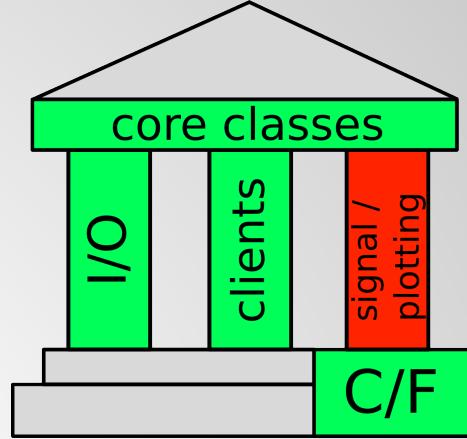
# Functionality?

## basic signal processing:

- trim, merge, rotate, ...
- filter, resample, instrument correction
- array analysis, cross correlations
- (coincidence) triggering (incl. master event detection)
- probabilistic power spectral densities

## basic plotting:

- waveform preview plots
- stations/events location plots
- channel instrument response plots (bode plots)



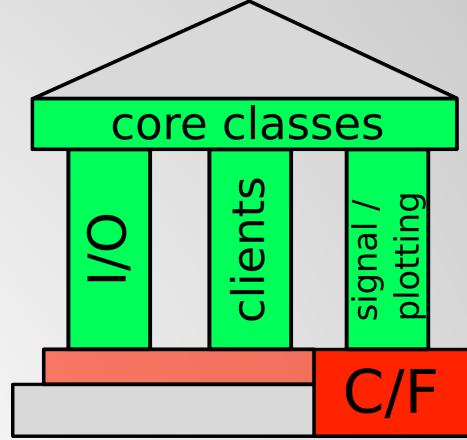
# Functionality?

3rd party code:

- don't reinvent the wheel
- reuse well established and maintained code

we use..

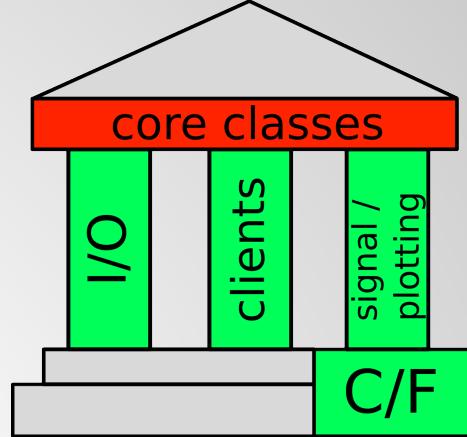
- [numpy](#): fast array operations
- [scipy](#): signal processing routines
- libmseed: MiniSEED I/O ([IRIS](#))
- evalresp: instrument correction ([ISTI/IRIS](#))
- iaspei-tau: theoretical traveltimes ([Snoke et al.](#))
- GSE\_UTI: GSE2 I/O (Stange et al.)



# Functionality?

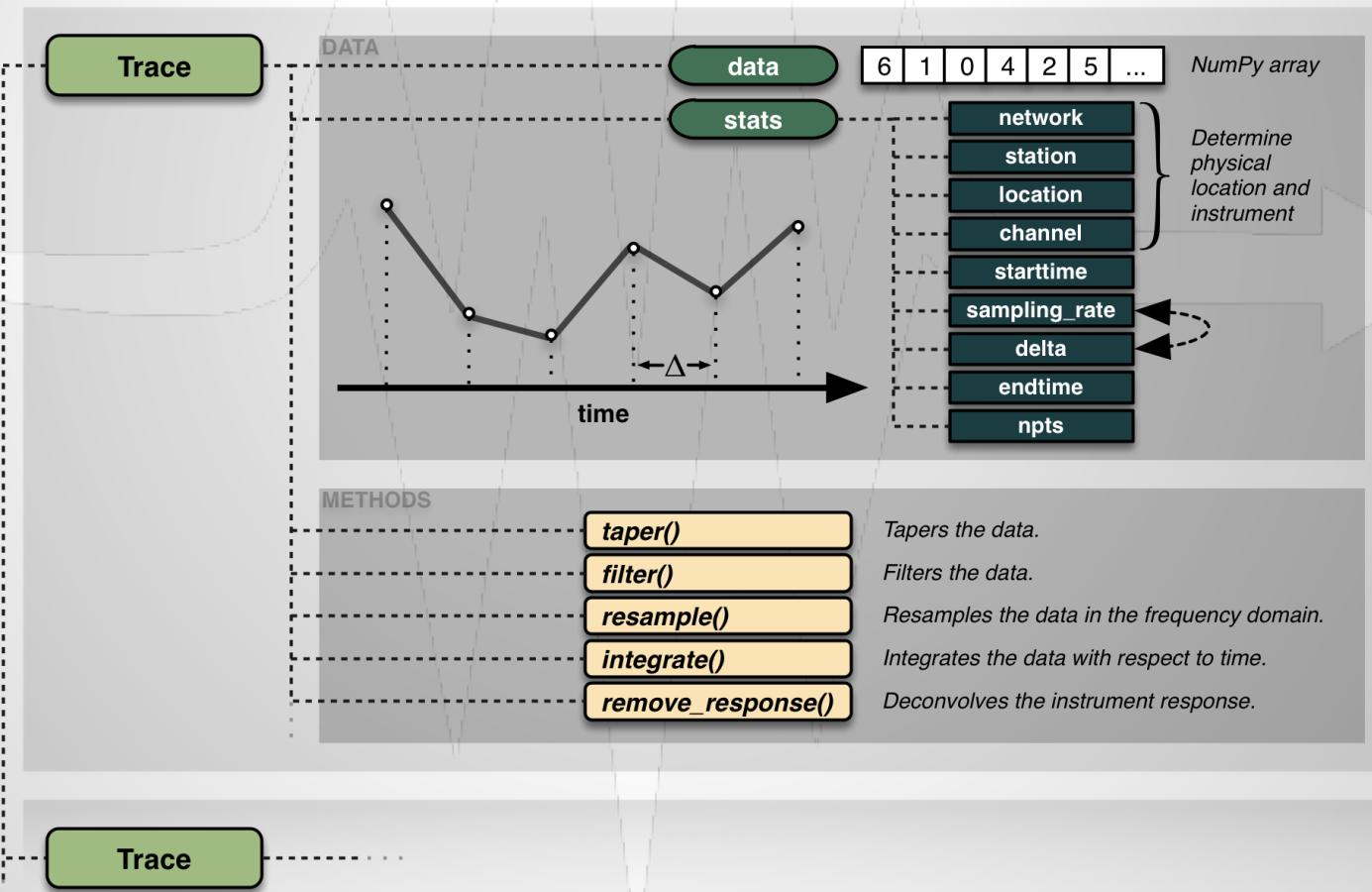
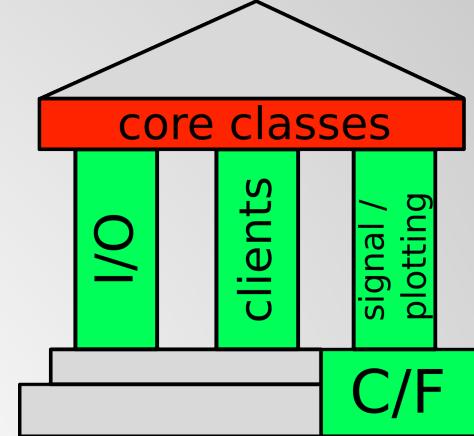
core object classes:

- waveforms
  - ⇒ Trace / Stream
- station metadata
  - ⇒ Inventory / Network / Station / ...
- event metadata
  - ⇒ Catalog / Event / ...



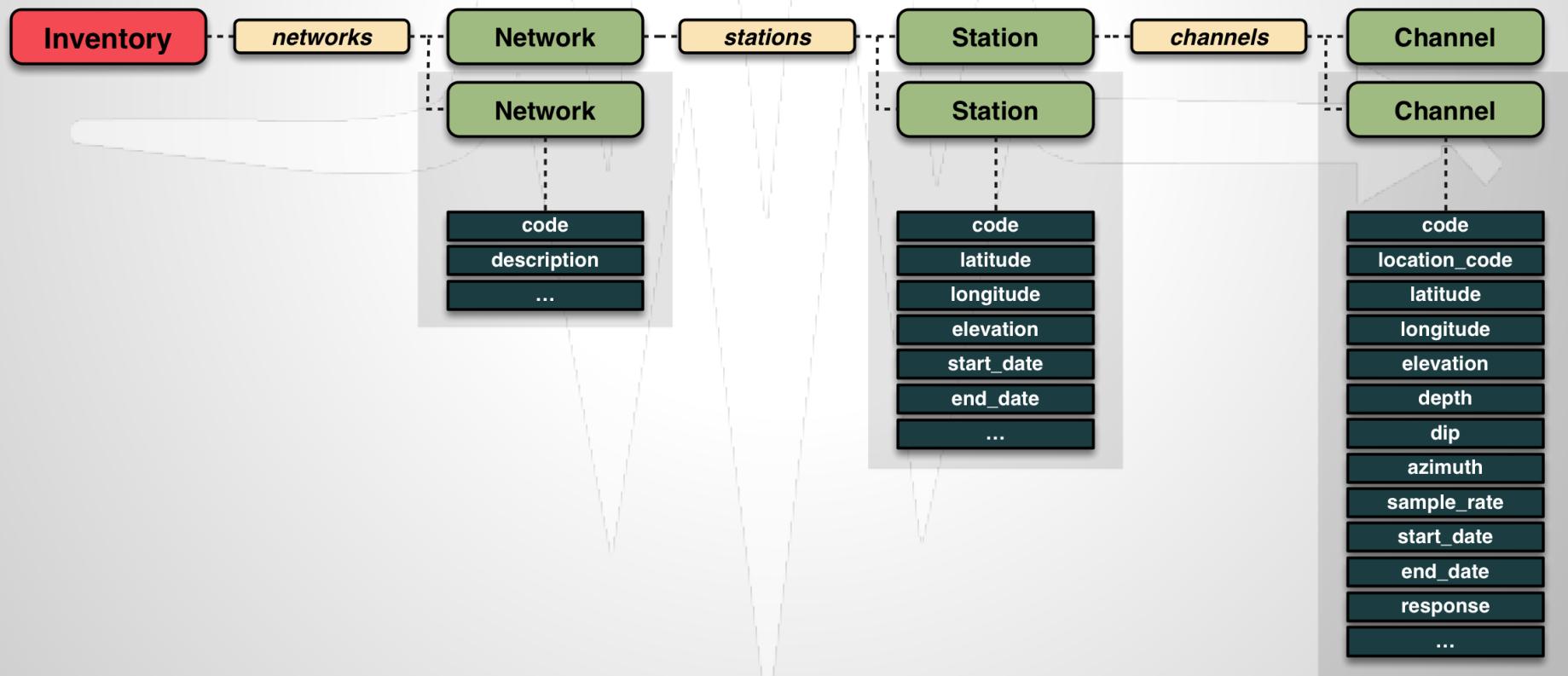
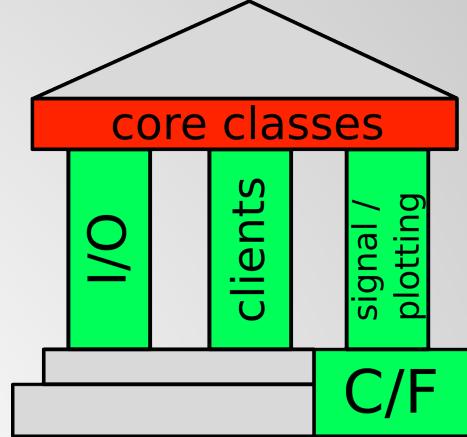
# Functionality?

core object classes: waveforms



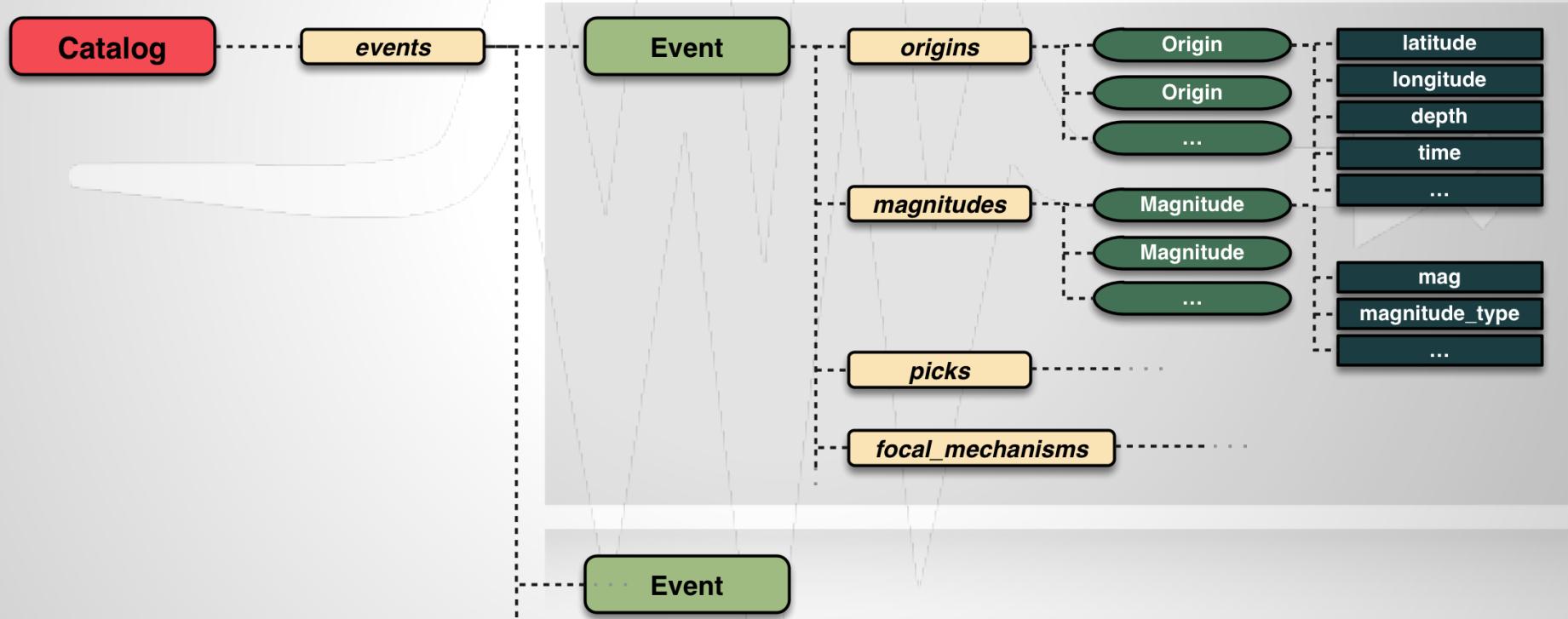
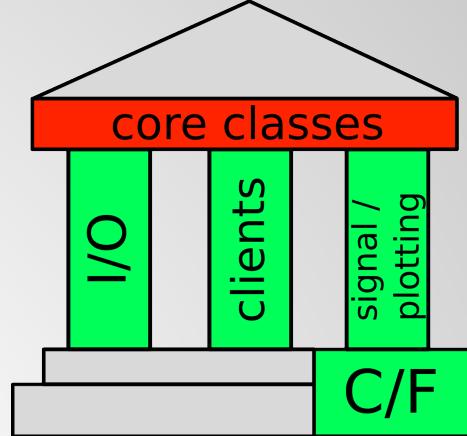
# Functionality?

core object classes: station metadata



# Functionality?

core object classes: event metadata



waveforms /  
station + event metadata

IRIS, EIDA, EMSC, ...  
..., seedlink, earthworm

read local files

access data centers / servers

**input**

obspy  
signal  
processing  
routines

obspy core classes  
Trace / Stream  
Inventory / Station  
Catalog / Event

fast array  
operations

numpy

signal  
processing

scipy

reuse  
legacy code

ctypes

**process**

**output**

write local files

publication quality figures

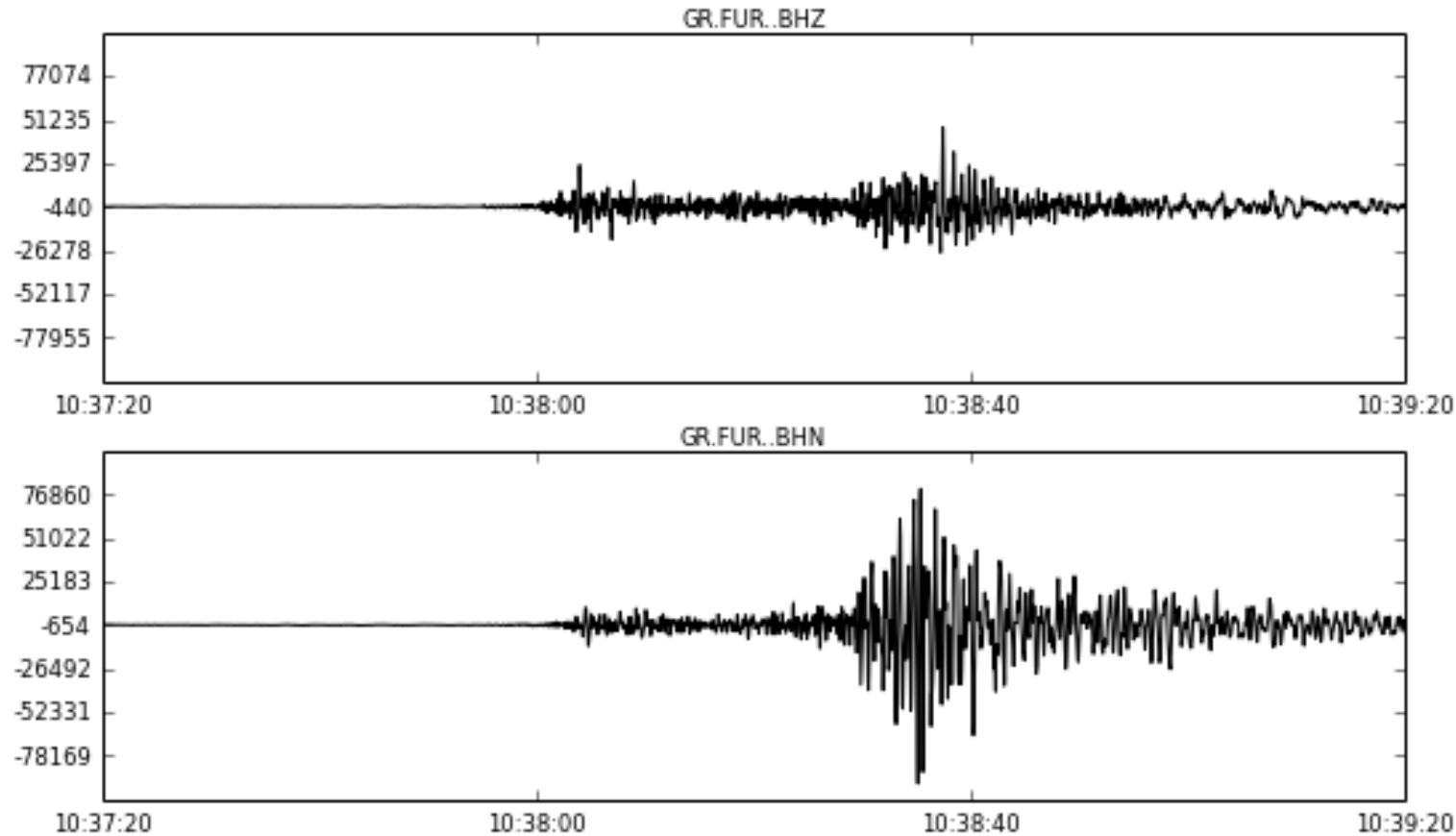
derived waveforms /  
station / event metadata

matplotlib / basemap

# Functionality?

```
In [2]: from obspy import read  
stream = read("waveform.mseed")  
stream.plot()
```

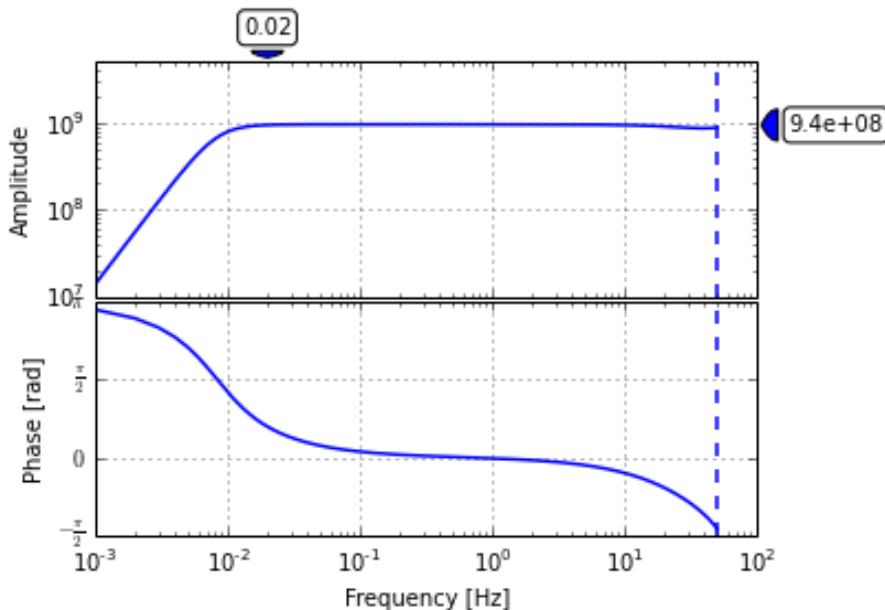
2014-05-31T10:37:20Z - 2014-05-31T10:39:20Z



# Functionality?

```
In [3]: from obspy import read_inventory
inventory = read_inventory("station.xml")
channel = inventory[0][0][0]
print channel
channel.plot(0.001)
```

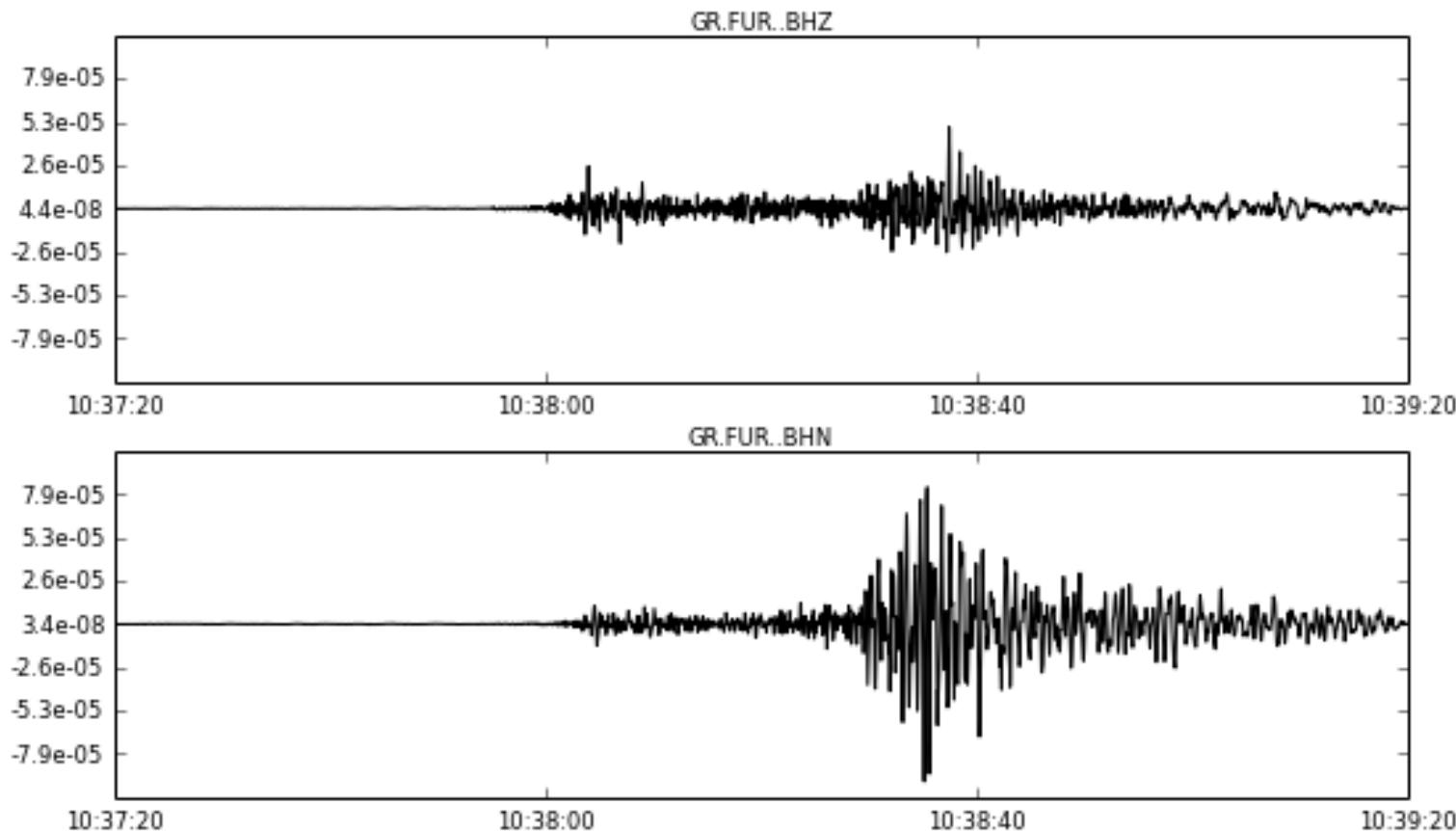
```
Channel 'HHZ', Location ''
    Timerange: 2006-12-16T00:00:00.000000Z - --
    Latitude: 48.16, Longitude: 11.28, Elevation: 565.0 m, Local Depth: 0.0 m
    Azimuth: 0.00 degrees from north, clockwise
    Dip: -90.00 degrees down from horizontal
    Channel types: TRIGGERED, GEOPHYSICAL
    Sampling Rate: 100.00 Hz
    Sensor: Streckeisen STS-2/N seismometer
    Response information available
```



# Functionality?

```
In [4]: stream.attach_response(inventory)
stream.remove_response(output="VEL")
stream.plot()
```

2014-05-31T10:37:20Z - 2014-05-31T10:39:20Z



# Technical Aspects

- Testing Framework
  - Testing code
  - Test reporting / Continuous Integration
- Version Control + Code Hosting
  - git: distributed version control
  - github: central platform for hosting, “social coding”

# Technical Aspects: Testing? Why?

- correctness of code
- stability of code
- when you fix a bug you want to make sure it  
*stays fixed*
- everybody can work on the code even if he/she  
does not know all details of the implementation
- documentation

# Technical Aspects: Doctests

```
class UTCDateTime(object):

    def __add__(self, value):
        """
        Adds seconds and microseconds to current UTCDateTime object.

        :type value: int, float
        :param value: Seconds to add
        :rtype: :class:`~obspy.core.utcdatetime.UTCDateTime`
        :return: New UTCDateTime object.

        .. rubric:: Example

        >>> dt = UTCDateTime(1970, 1, 1, 0, 0)
        >>> dt + 1.123456
        UTCDateTime(1970, 1, 1, 0, 0, 1, 123456)
        """
        if isinstance(value, datetime.timedelta):
            value = (value.microseconds + (value.seconds +
                86400) * 1000000) / 1000000.0
        return UTCDateTime(self.timestamp + value)

    if __name__ == '__main__':
        import doctest
        doctest.testmod(exclude_empty=True)
```

SOURCE CODE



DOCUMENTATION

obspy.core.utcdatetime.UTCDateTime.\_\_add\_\_

`UTCDateTime.__add__(value)`

[source]

Adds seconds and microseconds to current UTCDateTime object.

Parameters: `value (int, float)` Seconds to add

Return type: `UTCDateTime`

Returns: New UTCDateTime object.

Example

```
>>> UTCDateTime(1970, 1, 1, 0, 0) + 1.123456
UTCDateTime(1970, 1, 1, 0, 0, 1, 123456)
```

# Technical Aspects: Unit tests

```
from obspy.core import UTCDateTime
import unittest

class UTCDateTimeTestCase(unittest.TestCase):
    """
    Test suite for obspy.core.utcdatetime.UTCDateTime.
    """

    def test_weekday(self):
        """
        Tests weekday method.
        """
        dt = UTCDateTime(2008, 10, 1, 12, 30, 35, 45020)
        self.assertEquals(dt.weekday, 2)

    def test_invalidDates(self):
        """
        Tests invalid dates.
        """
        self.assertRaises(ValueError, UTCDateTime, 2010, 9, 31)
        self.assertRaises(ValueError, UTCDateTime, '2010-09-31')

    def suite():
        return unittest.makeSuite(UTCDateTimeTestCase, 'test')

if __name__ == '__main__':
    unittest.main(defaultTest='suite')
```

# Technical Aspects: Test Reporting

```
megies@wintermute: ~/git/obspy
megies@wintermute: ~/git/obspy                               megies@wintermute: ~/git/obspy/obspy/mseed
wintermute:~/git/obspy[releases]$ obspy-runitests -r
.
.
.
s

Ran 983 tests in 103.554s
OK

Test report has been sent to tests.obspy.org. Thank you!
wintermute:~/git/obspy[releases]$
```

# Technical Aspects: Test Reporting

```
megies@wintermute: ~/git/obspy/obspy/mseed
megies@wintermute: ~/git/obspy                         megies@wintermute: ~/git/obspy/obspy/mseed
14
15 def getStartAndEndTime(file_or_file_object):
16     """
17         Returns the start- and endtime of a Mini-SEED file or file-like object.
18
19         :type file_or_file_object: basestring or open file-like object.
20         :param file_or_file_object: Mini-SEED file name or open file-like object
21             containing a Mini-SEED record.
22         :return: tuple (start time of first record, end time of last record)
23
24         This method will return the start time of the first record and the end time
25         of the last record. Keep in mind that it will not return the correct result
~/git/obspy/obspy/mseed/util.py [TYPE=PYTHON]          15,5           1%
70     >>> f.close()
71     """
72     # Get the starttime of the first record.
73     info = getRecordInformation(file_or_file_object)
74     starttime = info['starttime']
75     # Get the endtime of the last record.
76     info = getRecordInformation(
77         file_or_file_object,
78         (info['number_of_records'] - 1) * info['record_length'])
79     endtime = info['endtime']
80     return starttime, endtime
81
~/git/obspy/obspy/mseed/util.py [TYPE=PYTHON]          78,39          10%
```

# Technical Aspects: Test Reporting

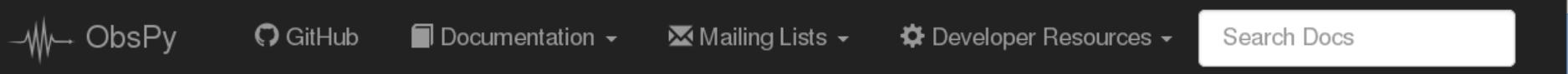
```
megies@wintermute: ~/git/obspy
megies@wintermute: ~/git/obspy                         megies@wintermute: ~/git/obspy/obspy/mseed
-----
File "/home/megies/git/obspy/obspy/mseed/util.py", line 67, in obspy.mseed.util.getStartAndEndTime
Failed example:
    getStartAndEndTime(file_object) # doctest: +NORMALIZE_WHITESPACE
Exception raised:
Traceback (most recent call last):
  File "/home/megies/python276/lib/python2.7/doctest.py", line 1289, in __run
    compileflags, 1) in test.globs
  File "<doctest obspy.mseed.util.getStartAndEndTime[12]>", line 1, in <module>
    getStartAndEndTime(file_object) # doctest: +NORMALIZE_WHITESPACE
  File "/home/megies/git/obspy/obspy/mseed/util.py", line 73, in getStartAndEndTime
    info = getRecordInformation(file_or_file_object)
  File "/home/megies/git/obspy/obspy/mseed/util.py", line 253, in getRecordInformation
    endian=endian)
  File "/home/megies/git/obspy/obspy/mseed/util.py", line 288, in _getRecordInformation
    elif file_object.read(8)[6] not in ['D', 'R', 'Q', 'M']:
IndexError: string index out of range

-----
Ran 983 tests in 100.804s

FAILED (failures=1, errors=1)

Test report has been sent to tests.obspy.org. Thank you!
wintermute:~/git/obspy〔releases〕$
```

# Technical Aspects: Test Reporting



## Test Reports Overview of the latest 20 test reports

Test Reports Overview of the latest 20 test reports							
Report			Errors / ObsPy		System		
Report	Failures	Version	Tests	Modules	Node	Python Version	System
#15439	1	0.9.2-805-gdbc825	1053	21	travis-ci	3.4.0	Linux (64bit)
#15438	3	0.9.2-746-ga194a	1047	21	docker-debian_7_wheezy	2.7.3	Linux (64bit)
#15437	3	0.9.2-746-ga194a	1047	21	docker-fedora_20	2.7.5	Linux (64bit)
#15436	-	0.9.2-805-gdbc825	1053	21	travis-ci	3.3.5	Linux (64bit)
#15435	26	0.9.2-787-gf176	1186	28	sphinx	2.7.3	Linux (32bit)

records per page: 20 ▾

show all  errors only

System

- all
- Darwin
- Linux
- Windows

Architecture

- all
- 32bit
- 64bit

Python

- all
- 2.6.0
- 2.6.1
- 2.6.5
- 2.6.6
- 2.6.7

# Technical Aspects: Testing

## Continuous Integration

- automate the build (incl. dependencies)
- automate running all tests
- trigger building/testing automatically  
on changes in the repository
- online overview of build/test results
- keep building/testing fast
- each pull request is automatically tested

# Technical Aspects: Testing

[Travis](#): free CI service for open source projects

Travis Home Blog Status Help Travis CI for Private Repositories Lion Krischer

Search all repositories

My Repositories Recent +

**obspy/obspy** 1870 Duration: 23 sec Finished: -

**megies/obspy** 236 Duration: 27 sec Finished: 5 days ago

**barsch/seishub.core** 42 Duration: 2 min 56 sec Finished: 6 days ago

**krischer/LASIF** 115 Duration: 5 min 9 sec Finished: 27 days ago

**krischer/wfs\_input\_generator** 27 Duration: 1 min 33 sec Finished: about a month ago

**barsch/obspy** 23

**obspy/obspy**

ObsPy: A Python Toolbox for seismology/seismological observatories.

Current Build History Pull Requests Branch Summary Build #1871

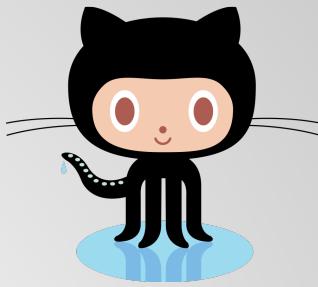
**dtype-compatibility** - Use native\_str() for dtype setting. **#1871 failed** ran for 53 min 6 sec 10 minutes ago Commit c370a8c Compare 62f831f..c370a8c

Lion Krischer authored and committed

Build Matrix

Job	Duration	Finished	Python
<a href="#">1871.1</a>	9 min 16 sec	17 minutes ago	2.6
<a href="#">1871.2</a>	10 min 25 sec	17 minutes ago	2.7
<a href="#">1871.3</a>	18 min 1 sec	10 minutes ago	3.3
<a href="#">1871.4</a>	15 min 24 sec	11 minutes ago	3.4

# Technical Aspects: GitHub



GitHub: Free hosting of code repositories

- De facto standard for code hosting
- Worldwide community of developers
- Issue / bug tracker
  - (*add comments, commits, integration with Travis CI*)
- **Social coding and communication**
- Forking, pull requests
- Avoids overhead of self-hosting
- Visibility
- New: DOIs for Code => citable

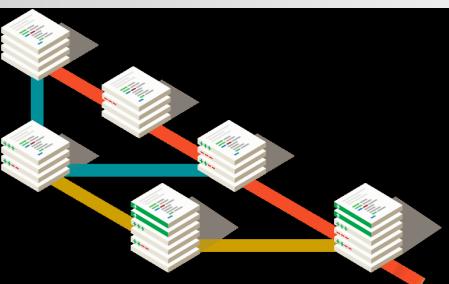
# Technical Aspects: git



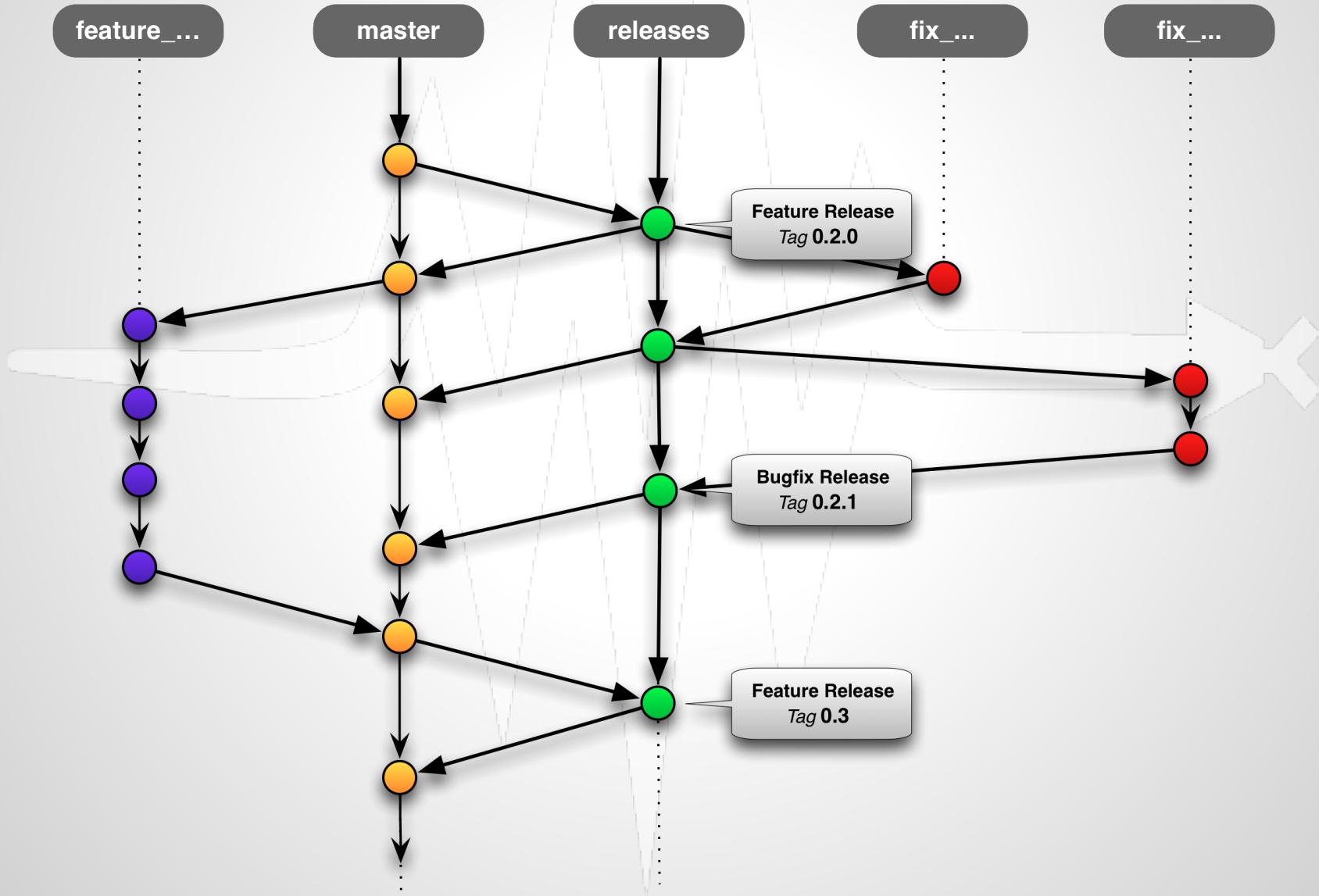
git: **distributed** version control software

- Initialized 2005 by Linus Torvalds for managing the Linux kernel
- Every local copy of the repository is self-contained with full history
- Possible to work offline
- Strong support for branching / merging

*“Subversion used to say ‘CVS done right’ [...] There is no way to do CVS right.”*  
Linus Torvalds



# Technical Aspects: ObsPy Dev Model



# Spreading the word: why?

start of the project ..

- .. group of a few dedicated (under)grad students

but ..

- .. people go different ways

- .. and unmaintained projects die fast

ultimate goal: self-sustaining project

- .. get more contributors/developers/maintainers

- .. raise impact

- .. reach critical mass of users / institutions  
relying (depending?) on the project

# Spreading the word: homework..

make it ..

.. useful

.. easy to use

- good [documentation](#)
- [tutorial](#), [gallery](#), [workshop documents](#) online

.. easy to install (on any platform)

- [available at pypi](#)
- [binaries, installers, packages](#)

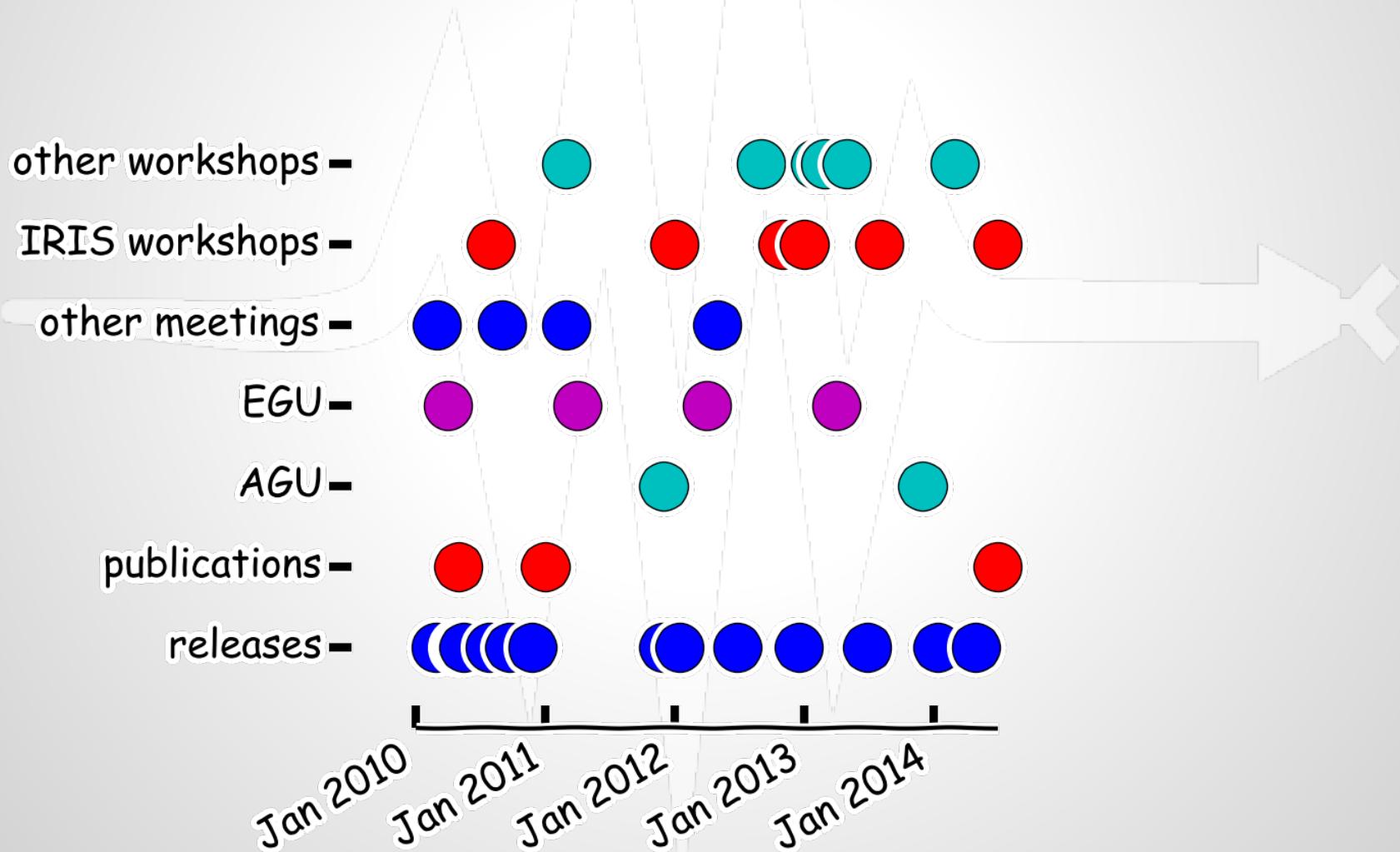
.. reliable (tests!)

.. interactive and responsive

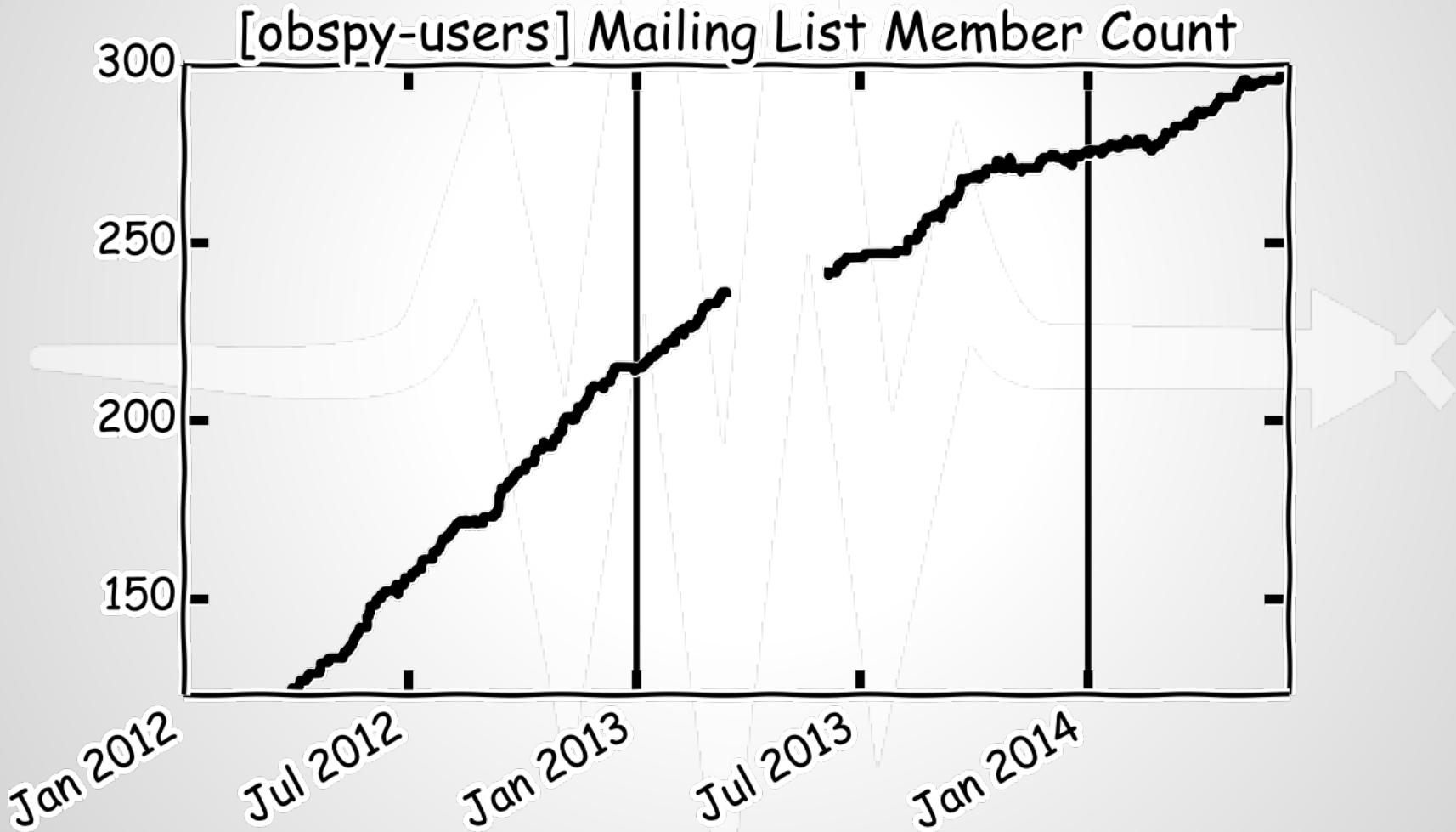
- [github](#), [wiki](#), [mailing list](#)

# Spreading the word: how?

..wherever we go, we tell people about it

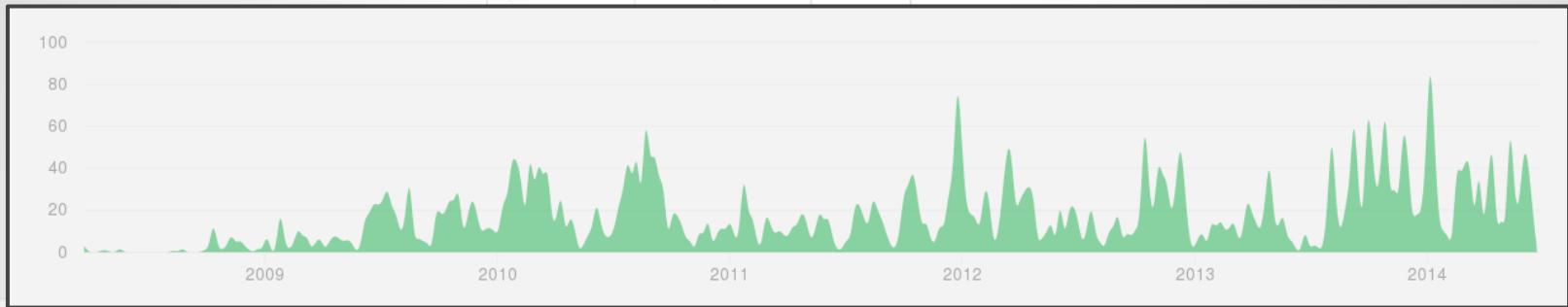


# Spreading the word: success so far?



# Spreading the word: success so far?

some stats..



- 30 people have actively committed changesets on github
- ~ 10 people sent contributions as Python files via email
- 59 citations of our publications

# Spreading the word: success so far?

All requests 12    Open Closed    Sort: Newest ▾    New pull request

Yours 6

Find a user...

ThomasLecocq 2

markcwill 1

QuLogic 1

krischer 1

jwassermann 1

**ObsPy test suite on Docker images** #828

This PR adds a couple of scripts and resources to test ObsPy on various Linux based operating sys...  
by krischer 1 days ago    krischer:docker\_tests    9 comments

**Using argparse?** #826

Now I realize that argparse was probably left out on purpose for 2.6 support. However, future kin...  
by QuLogic 2 days ago    QuLogic:argparse    11 comments

**Problems with unicode dtypes with old numpy versions** #823

NumPy 1.4 (and potentially 1.5 ?) cannot deal with dtypes specified as unicode strings, .e.g. all...  
by megies 4 days ago    dtype-compatibility    13 comments

**PPSD: enable getting/plotting mode and mean psd values** #804

Should have some minimal tests before merging..  
by megies on May 23    megies:ppsd\_stats

**Kinemetrics EVT data format support.** #781

Added support for the format + tests + reference PDF.  
by ThomasLecocq on Apr 28    ThomasLecocq:feature\_evt    20 comments

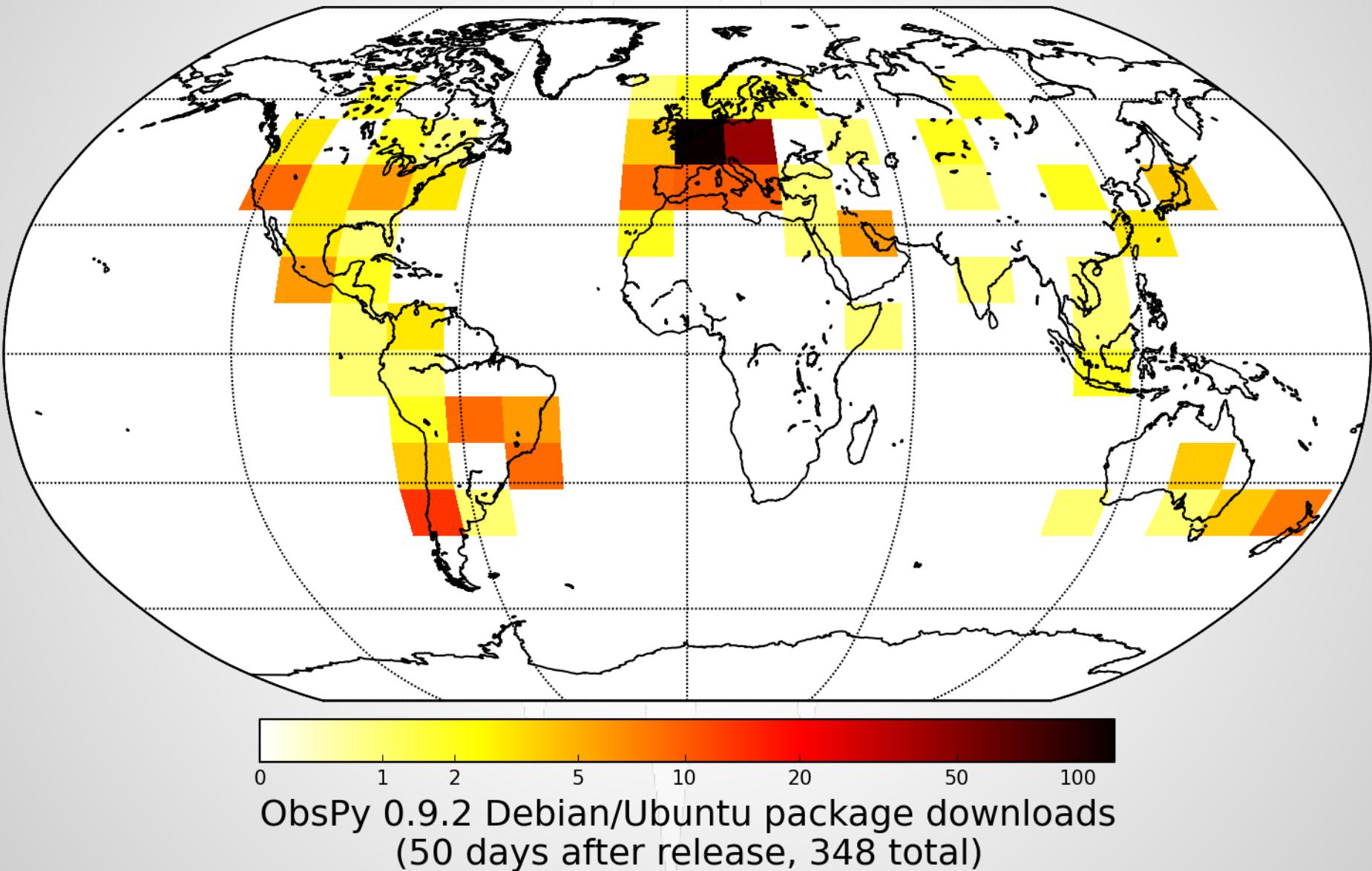
- lot of active discussions about improvements

# Spreading the word: success so far?

growing number of contributions from “outside”

- clients (3/9): seedlink, earthworm, neic
- file format plugins: css, datamark, pde, y
- packaging efforts:
  - Fedora / RHEL / CentOS
  - ArchLinux
  - FreeBSD
  - NetBSD
  - MacPorts

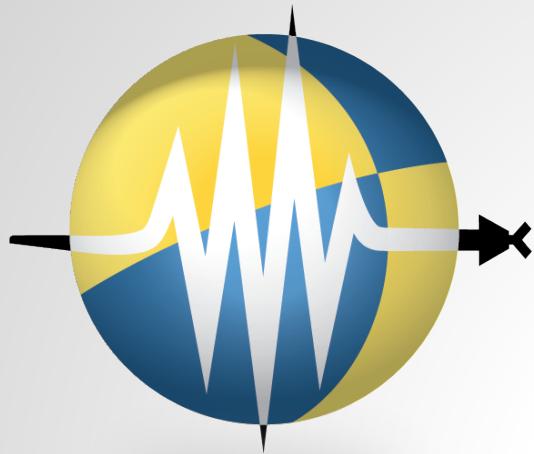
# Spreading the word: success so far?



# Summary

## ObsPy

- ..addresses most needs for successful and fast development of custom processing
- ..being reproducible
- ..and easing up exchange of processing workflows
- ..it is widely known, used and acknowledged
- ..and has grown from a 2-4 man show to being developed by people from all around the world



# ObsPy

A Python Framework for Seismology

**Thanks for your attention!**

[www.obspy.org](http://www.obspy.org)