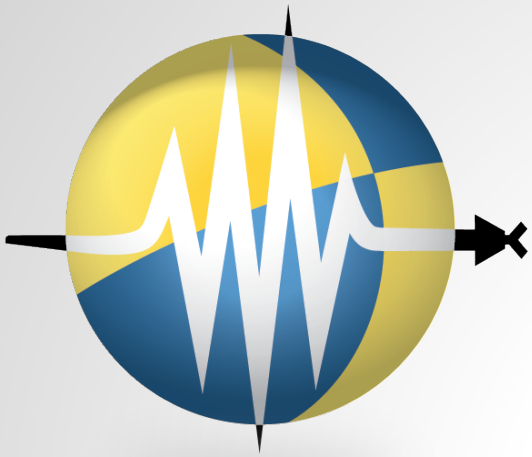


ObsPy

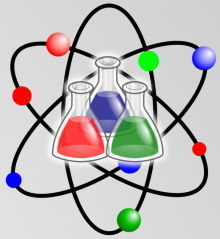
A Python Framework for Seismology

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

Tobias Megies, Lion Krischer (...)

June 2014

Software in Science

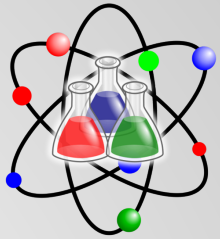


- Going from research code to distributable and reusable code easily is 5-10 times as expensive
- Gain in hard research currency (e.g. publications) is questionable

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.

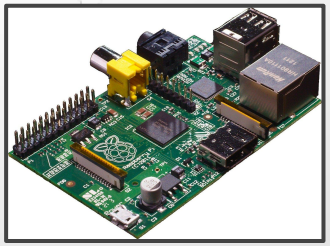
Outline

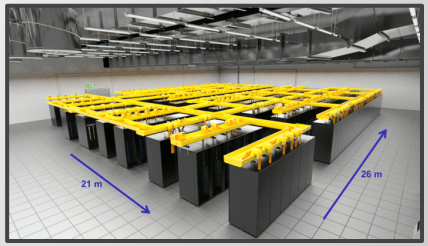
1. Introduction to Python and ObsPy
   a. Why Python?
   b. Functionality of ObsPy
   c. Basic Usage Examples
2. Some Technical Details
   a. Testing
   b. Code Management and Communication
3. “spreading the word”

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

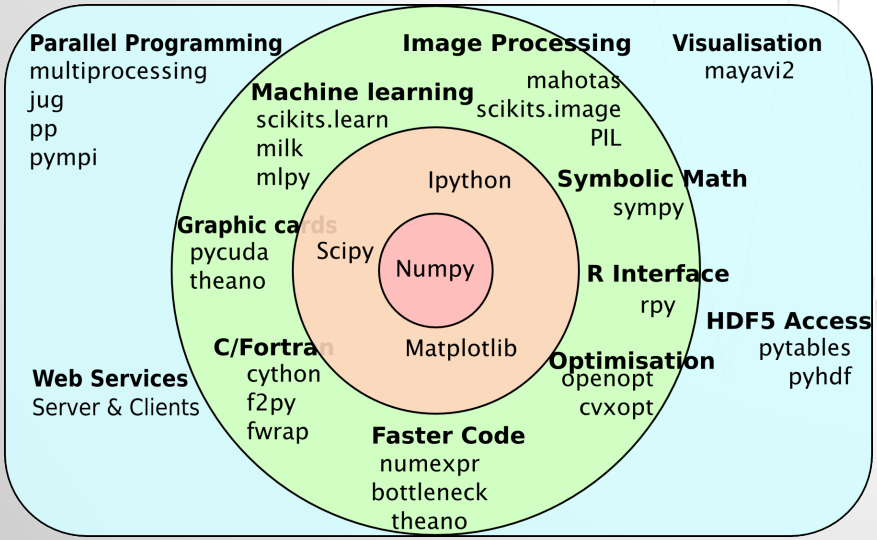
  ⇒ potentially high impact / user base
- general purpose programming language
- Cross-platform: from RaspberryPi to large supercomputers

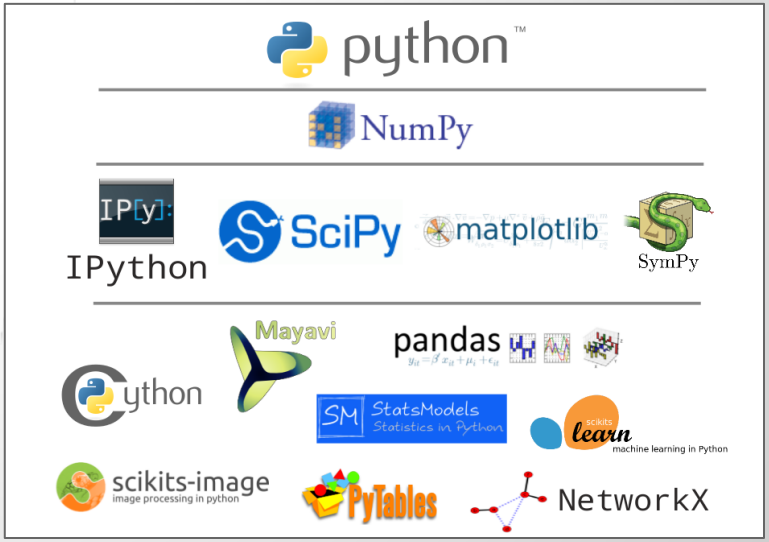




Why Python?

- 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





http://luispedro.org/files/talks/2013/EuBIAS/mahotas.html#/

Picture by Fernando Pérez

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

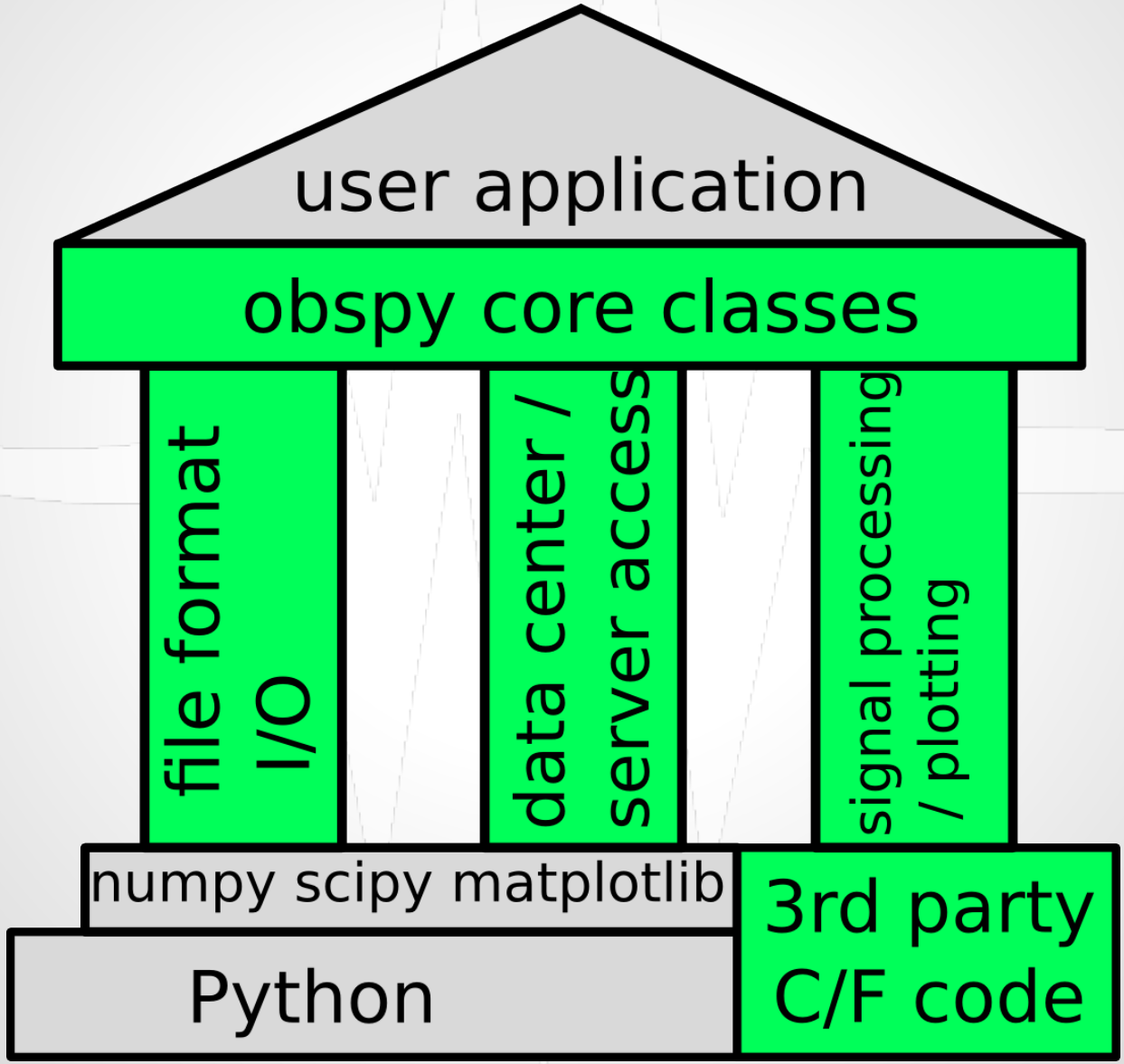
=> a bridge for seismologists into the scientific Python ecosystem

Processing needs

What I need is..

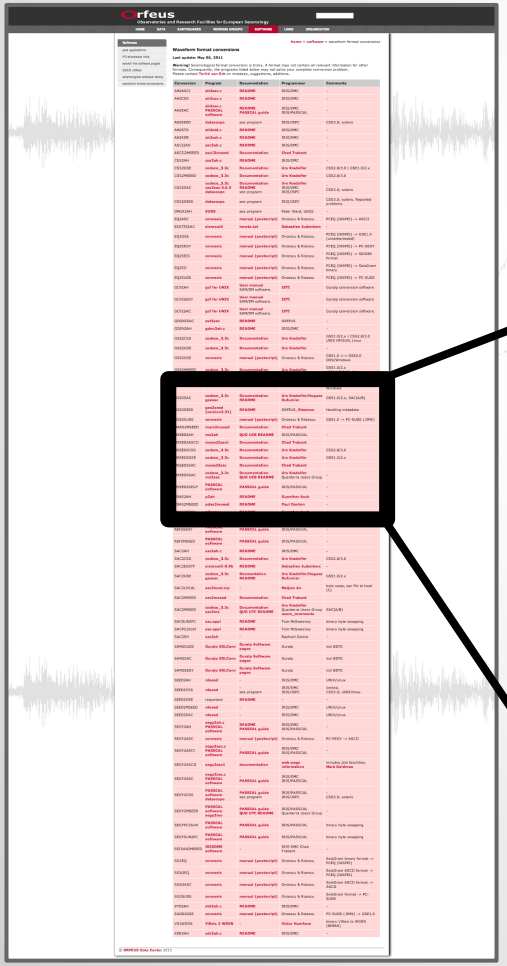
.. I/O of local data
.. fetch data/metadata from data centers
.. convenient handling of the parsed data/metadata
.. basic signal processing / data analysis / math
.. visualization capabilities

Functionality?



File Formats

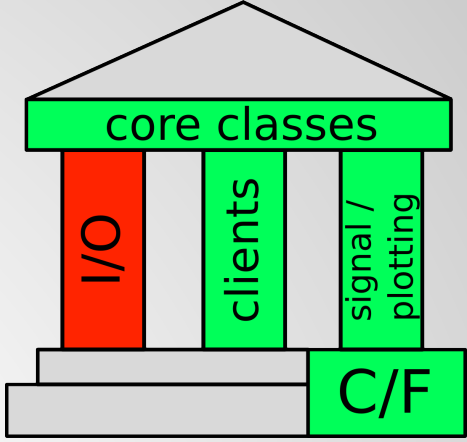
Situation before ObsPy



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

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

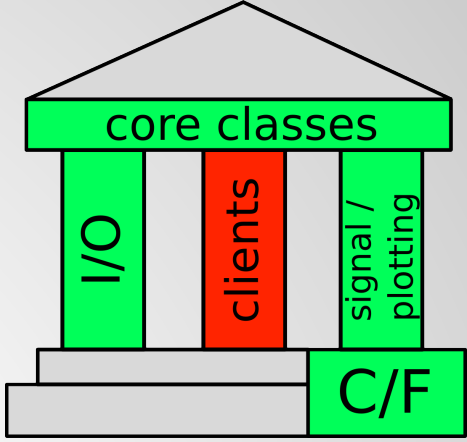
Functionality?



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):

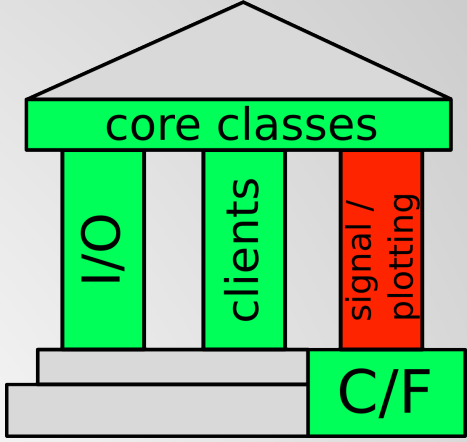
- FDSN web service client
  - IRIS, Orfeus, USGS, RESIF, NCEDC, ...
- ArcLink client: EIDA / Orfeus
- SeisHub: FFB, local database systems
- ...

server access (near-realtime / ringbuffer data):

  - SeedLink, Earthworm

⇒ different types of servers,
but usage of clients very similar

Functionality?



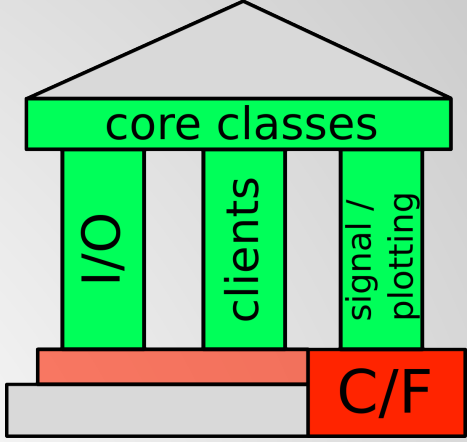
basic signal processing:

- 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?



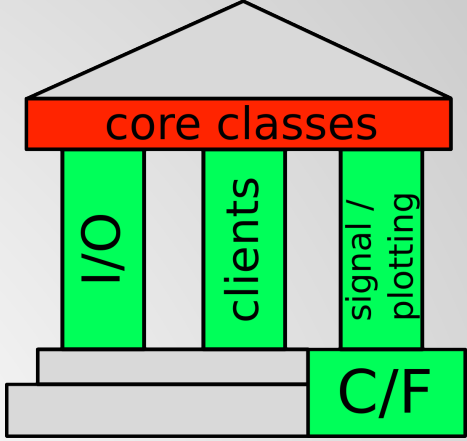
3rd party code:

- 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 etal.)
- GSE_UTI: GSE2 I/O (Stange etal.)

Functionality?

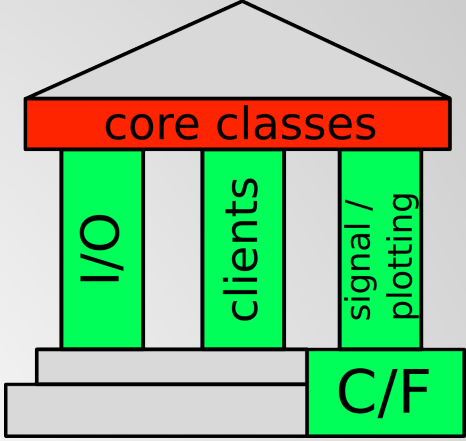


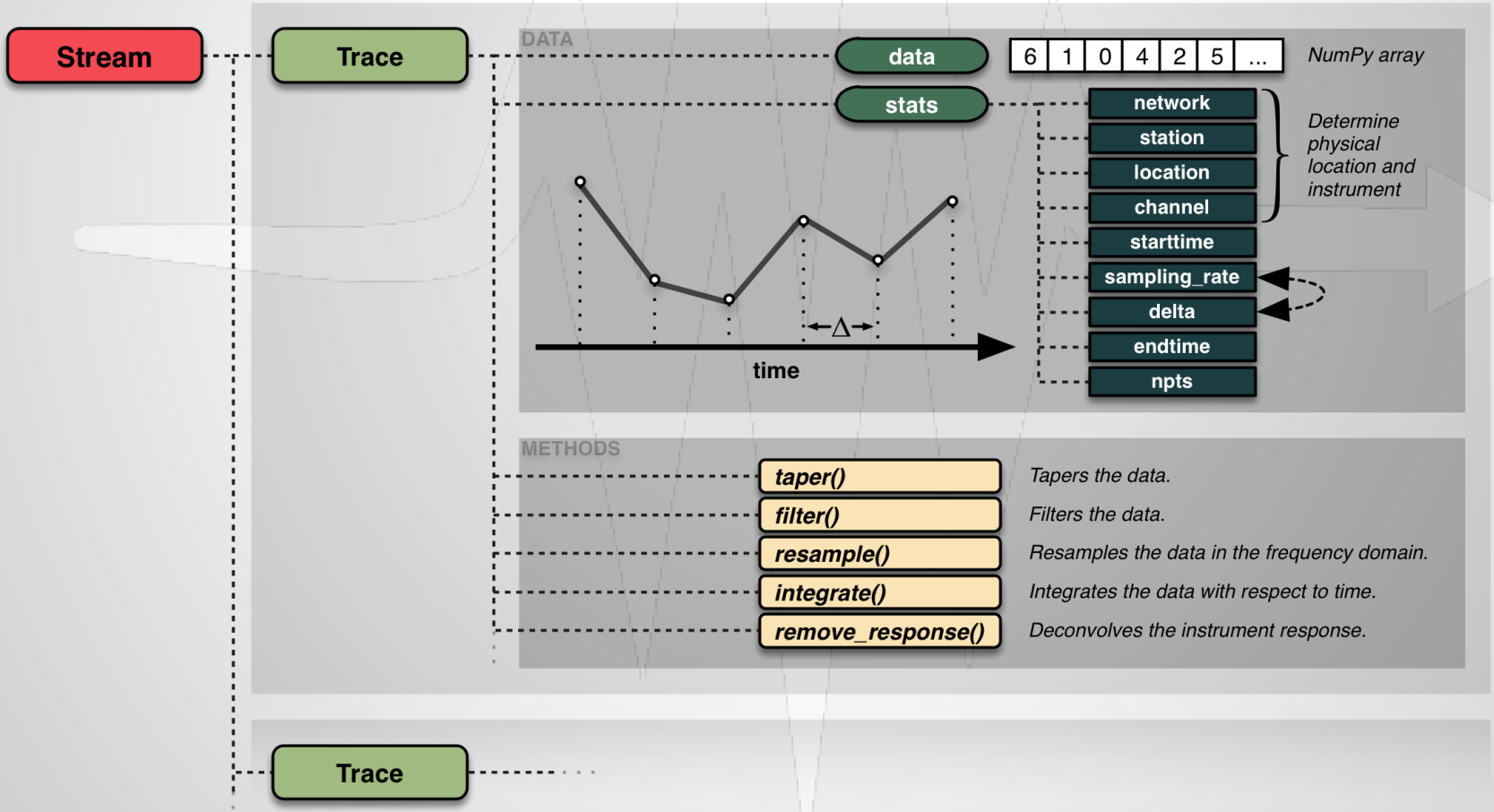
core object classes:

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

Functionality?

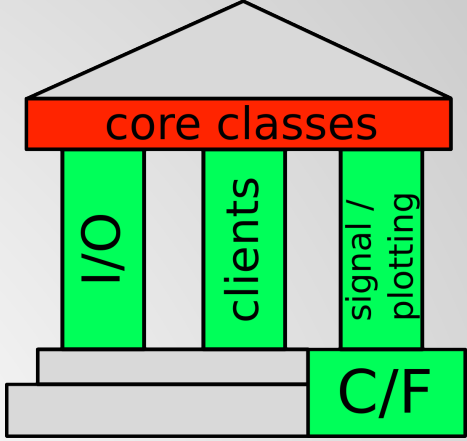
core object classes: waveforms

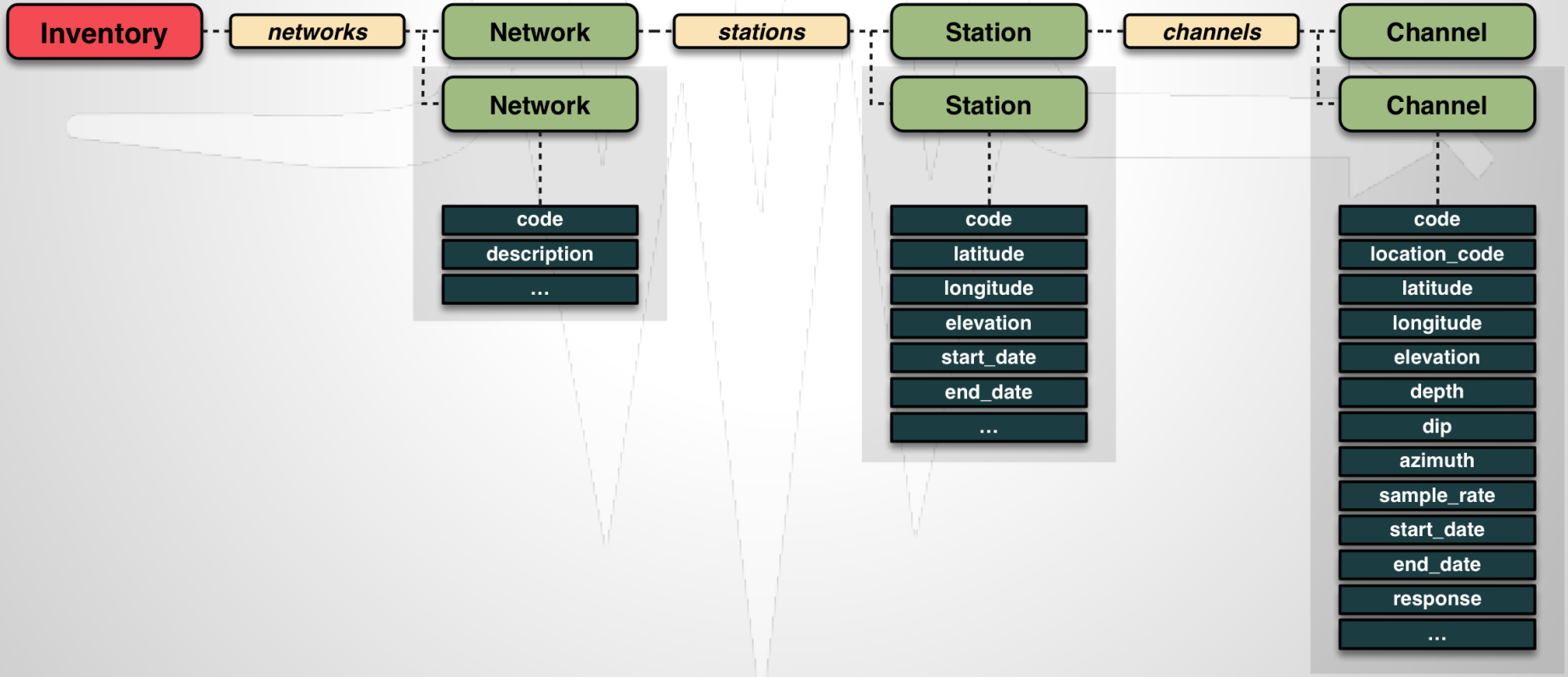




Functionality?

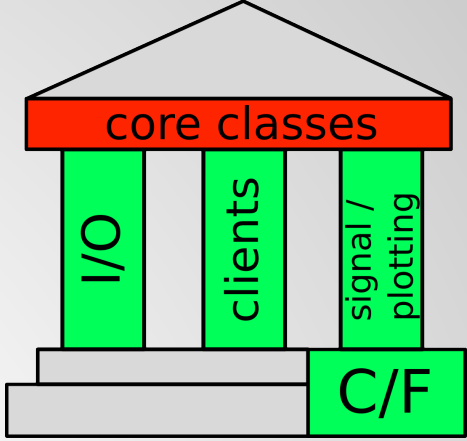
core object classes: station metadata

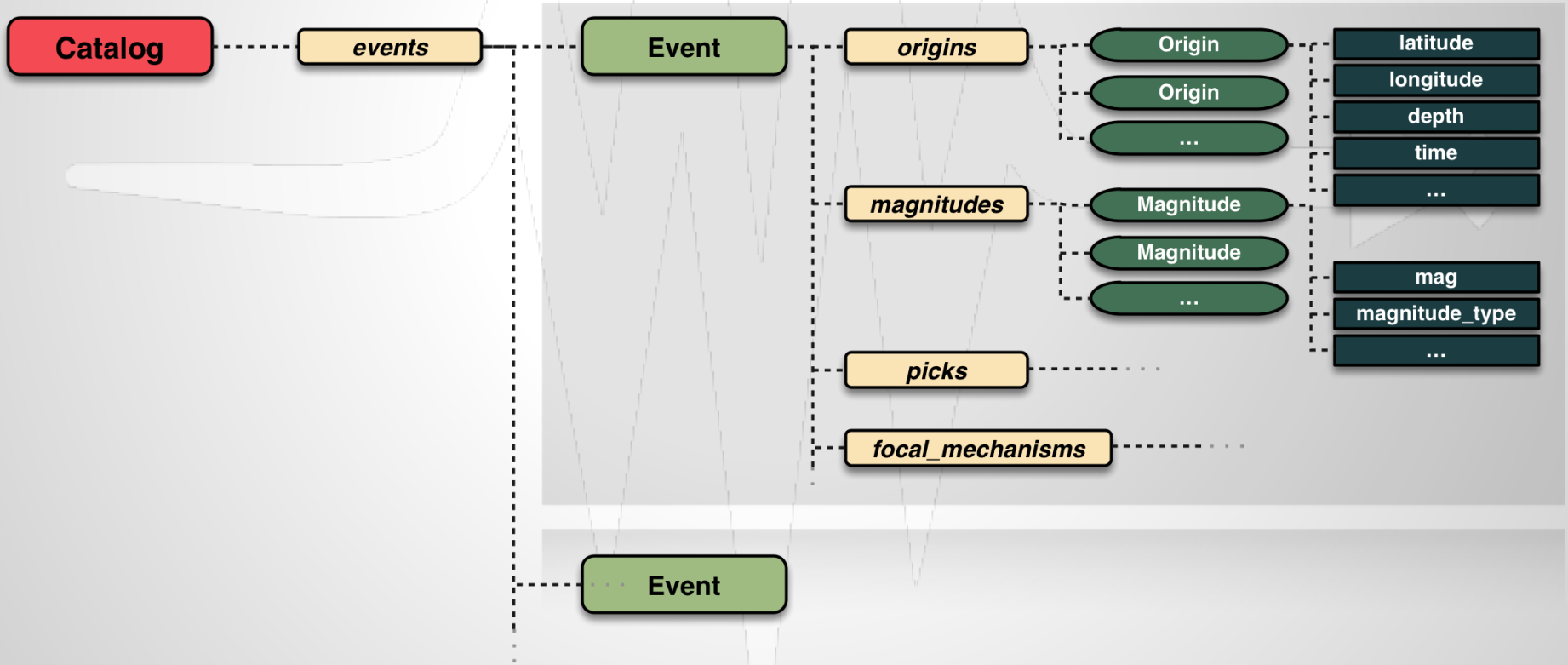


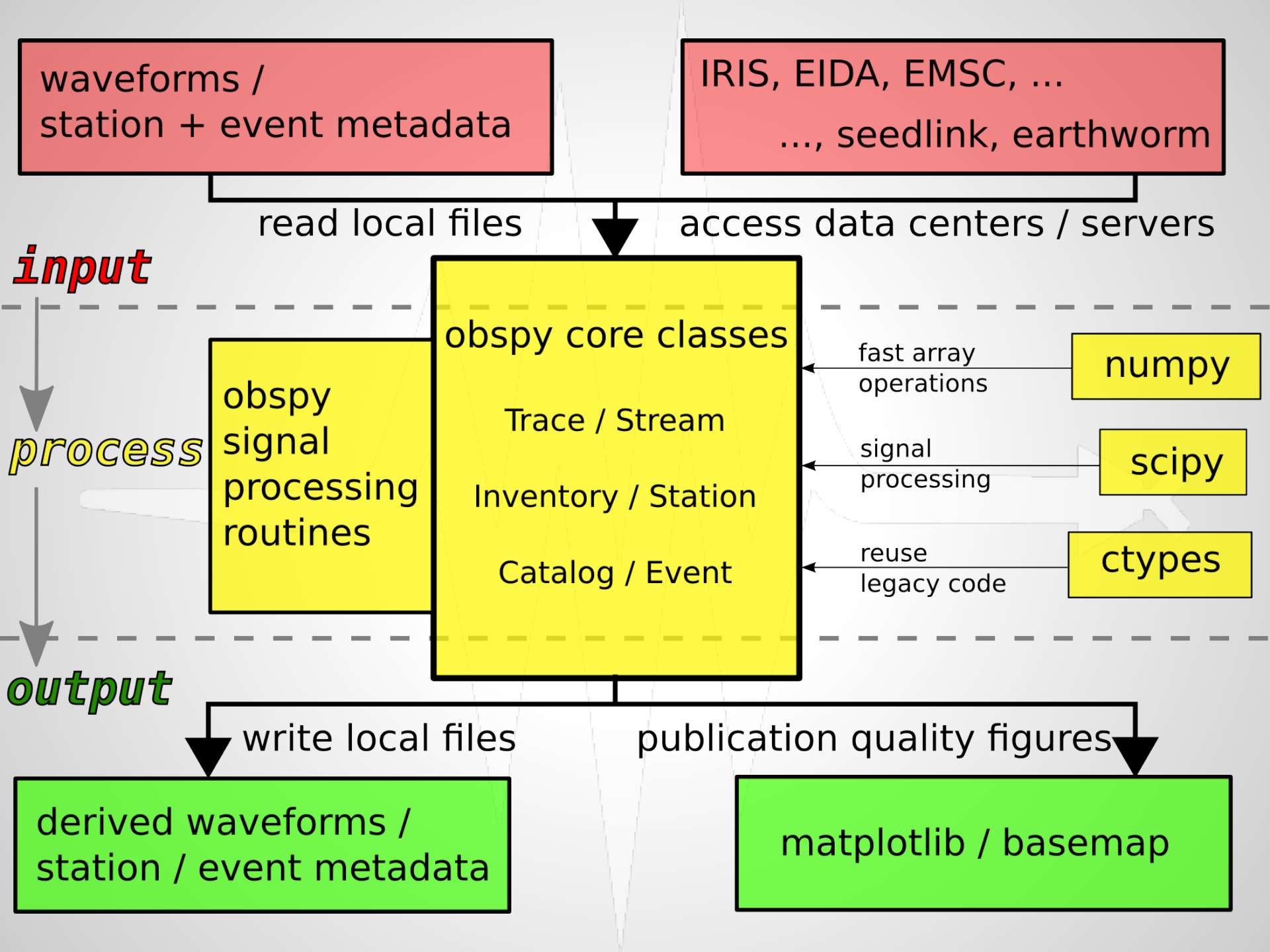


Functionality?

core object classes: event metadata

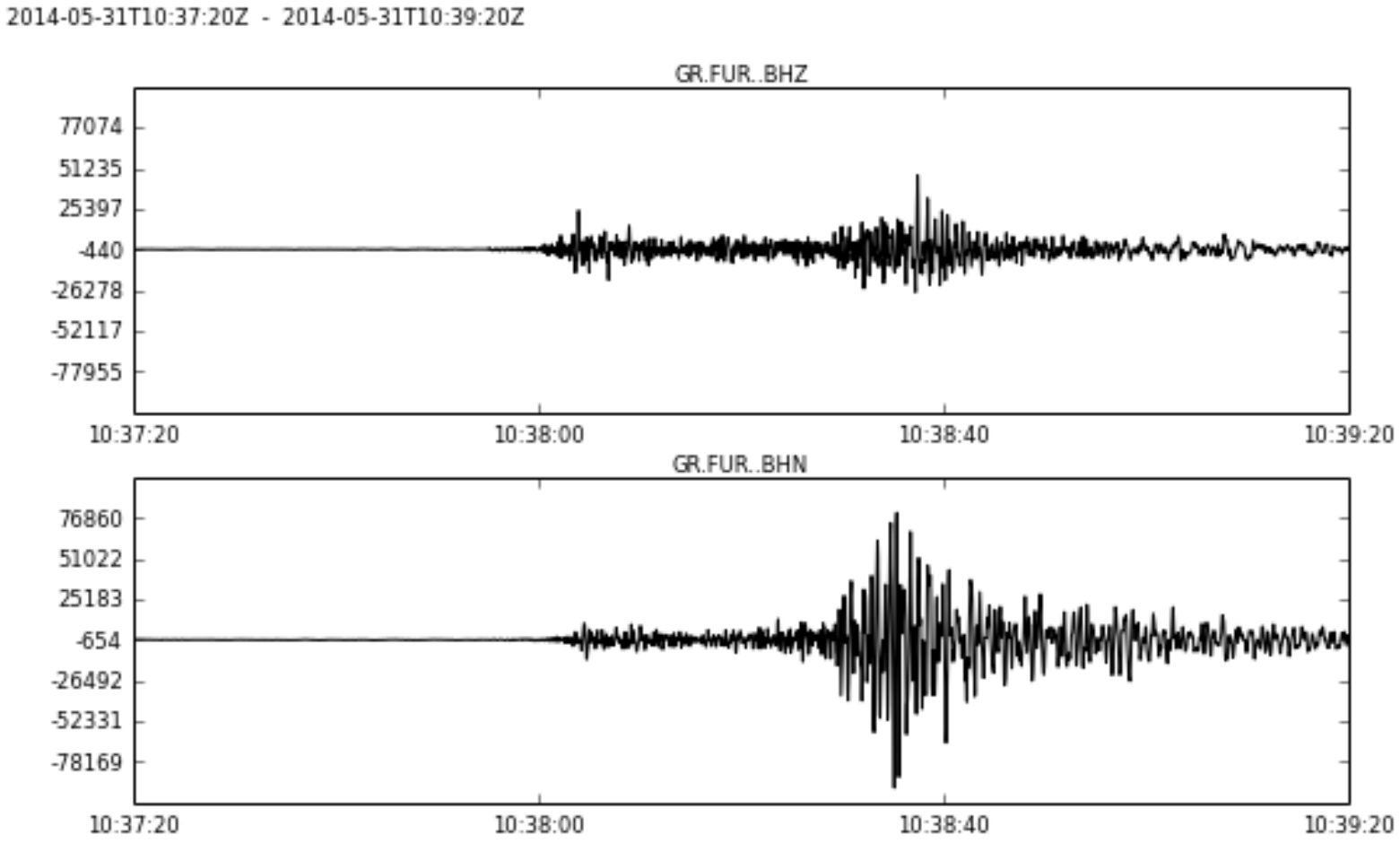





waveforms /
station + event metadata
IRIS, EIDA, EMSC, ...
..., seedlink, earthworm
read local files
access data centers / servers
input
process
output
obspy core classes
Trace / Stream
Inventory / Station
Catalog / Event
obspy
signal
processing
routines
fast array
operations
numpy
signal
processing
scipy
reuse
legacy code
ctypes
write local files
publication quality figures
derived waveforms /
station / event metadata
matplotlib / basemap

Functionality?

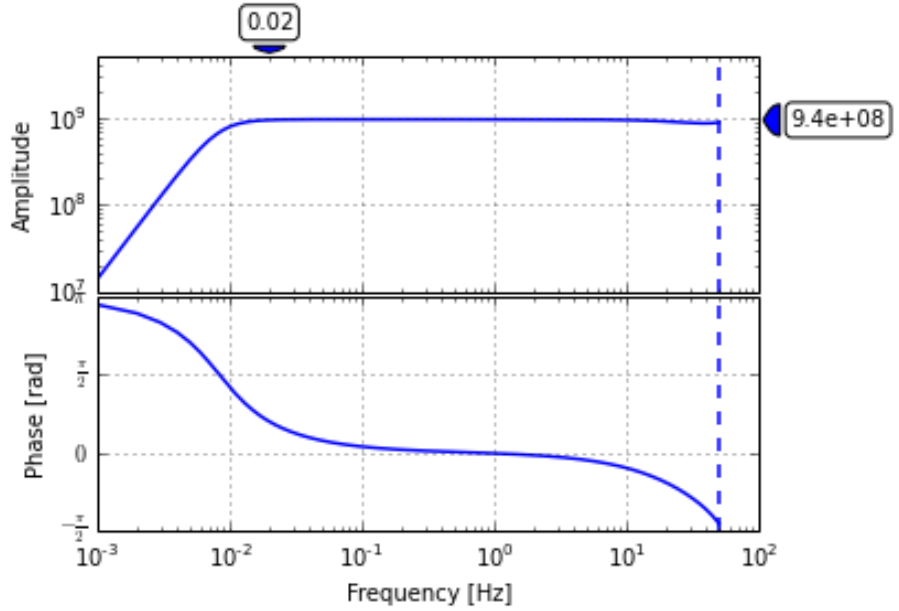
```
In [2]: from obspy import read
        stream = read("waveform.mseed")
        stream.plot()
```



Functionality?

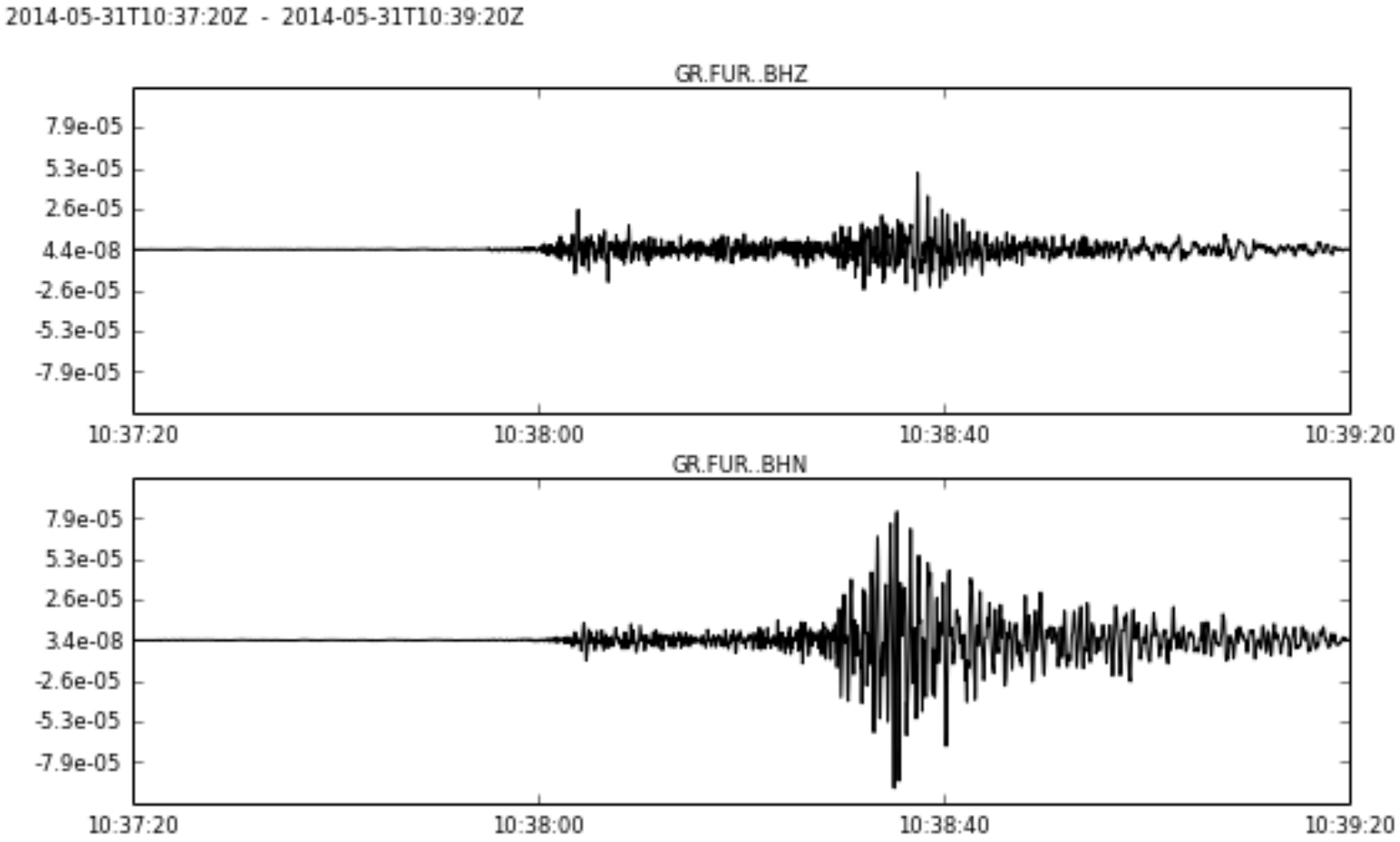
In [3]:
```
from obspy import read_inventory
inventory = read_inventory("station.xml")
channel = inventory[0][0][0]
print channel
channel.plot(0.001)
```

```
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()
```



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

```python
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
                86400) * 1000000) / 1000000.0
        return UTCDateTime(self.timestamp + valu

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.

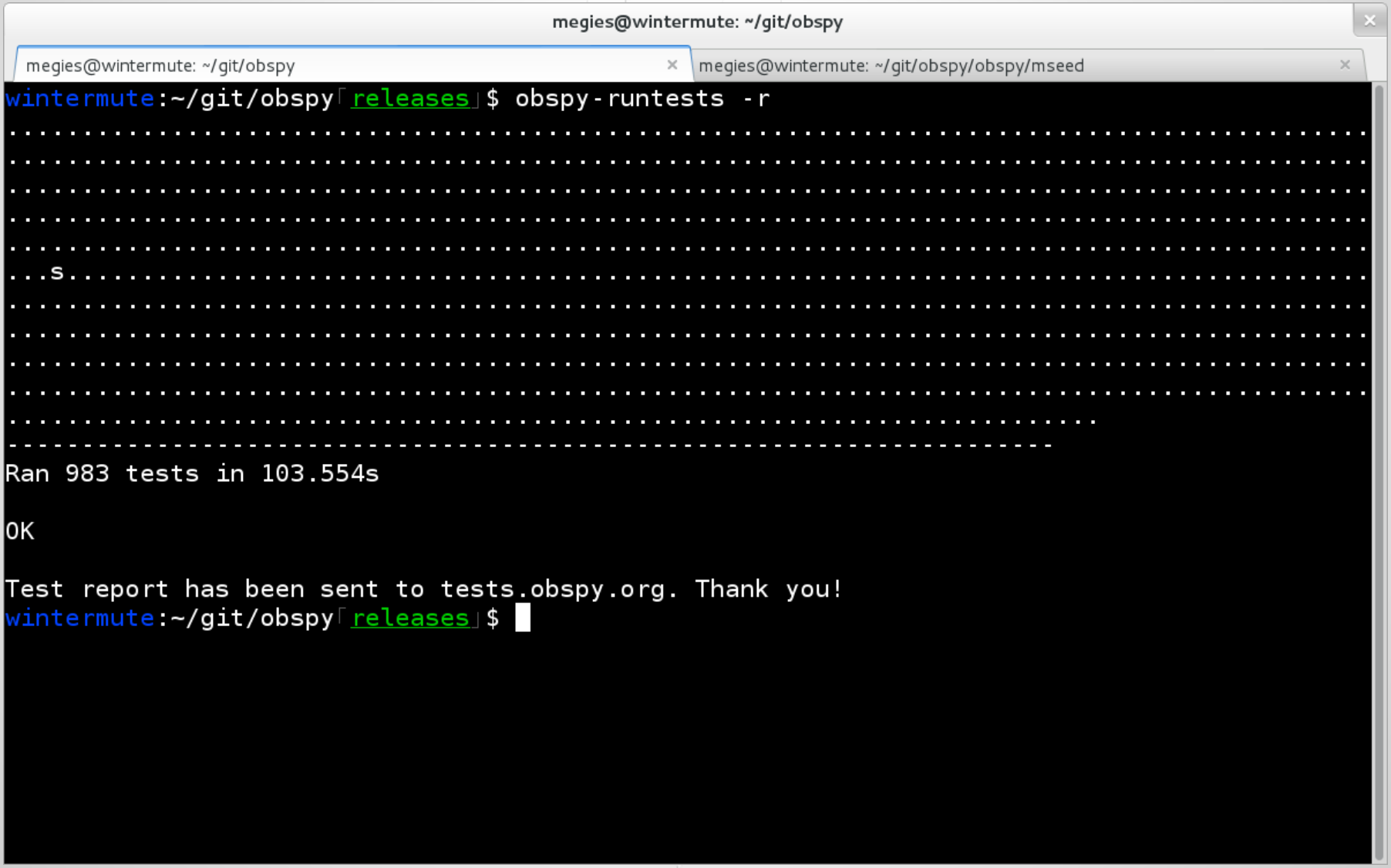
Example

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

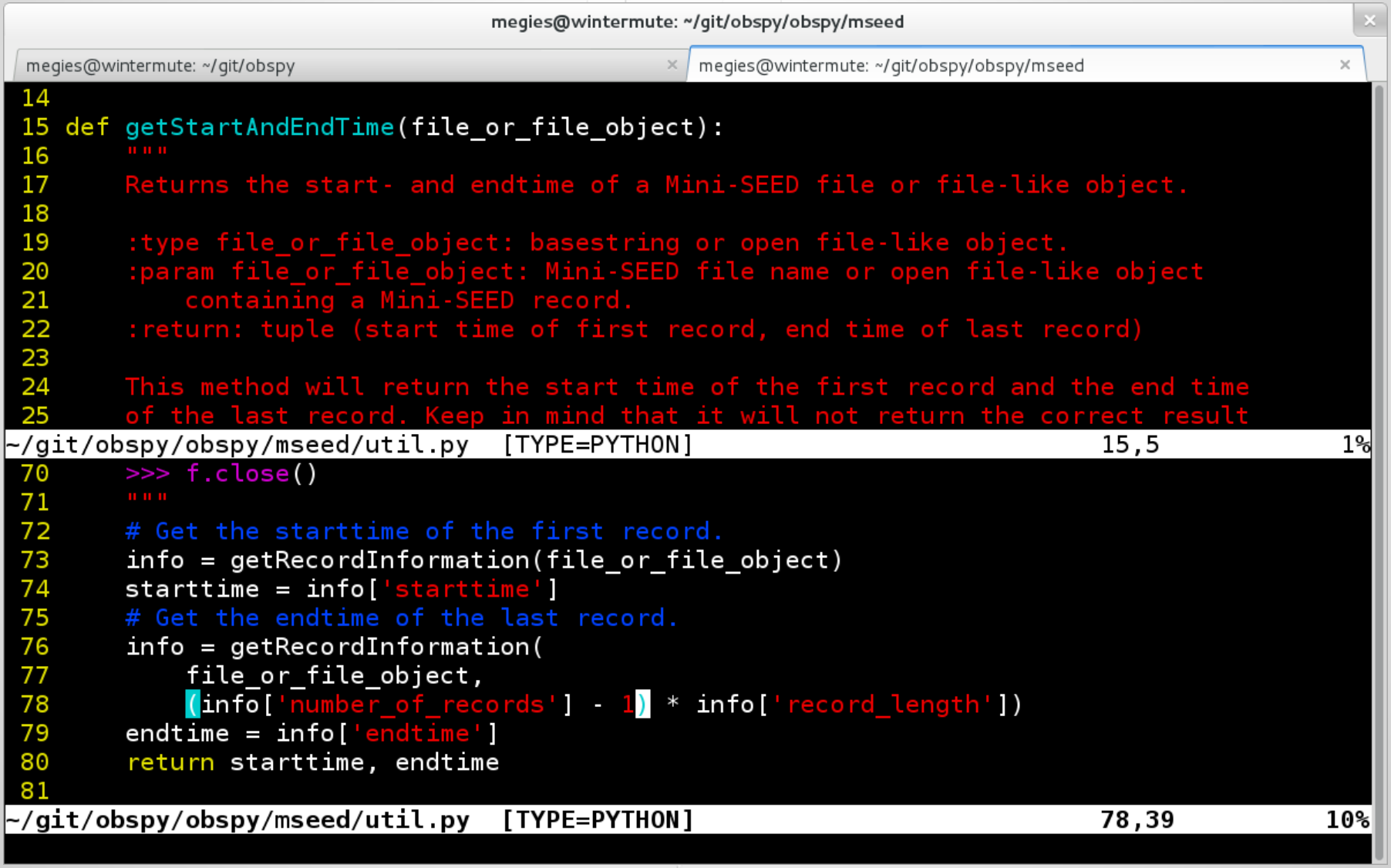
Technical Aspects: Unit tests

```python
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')
```

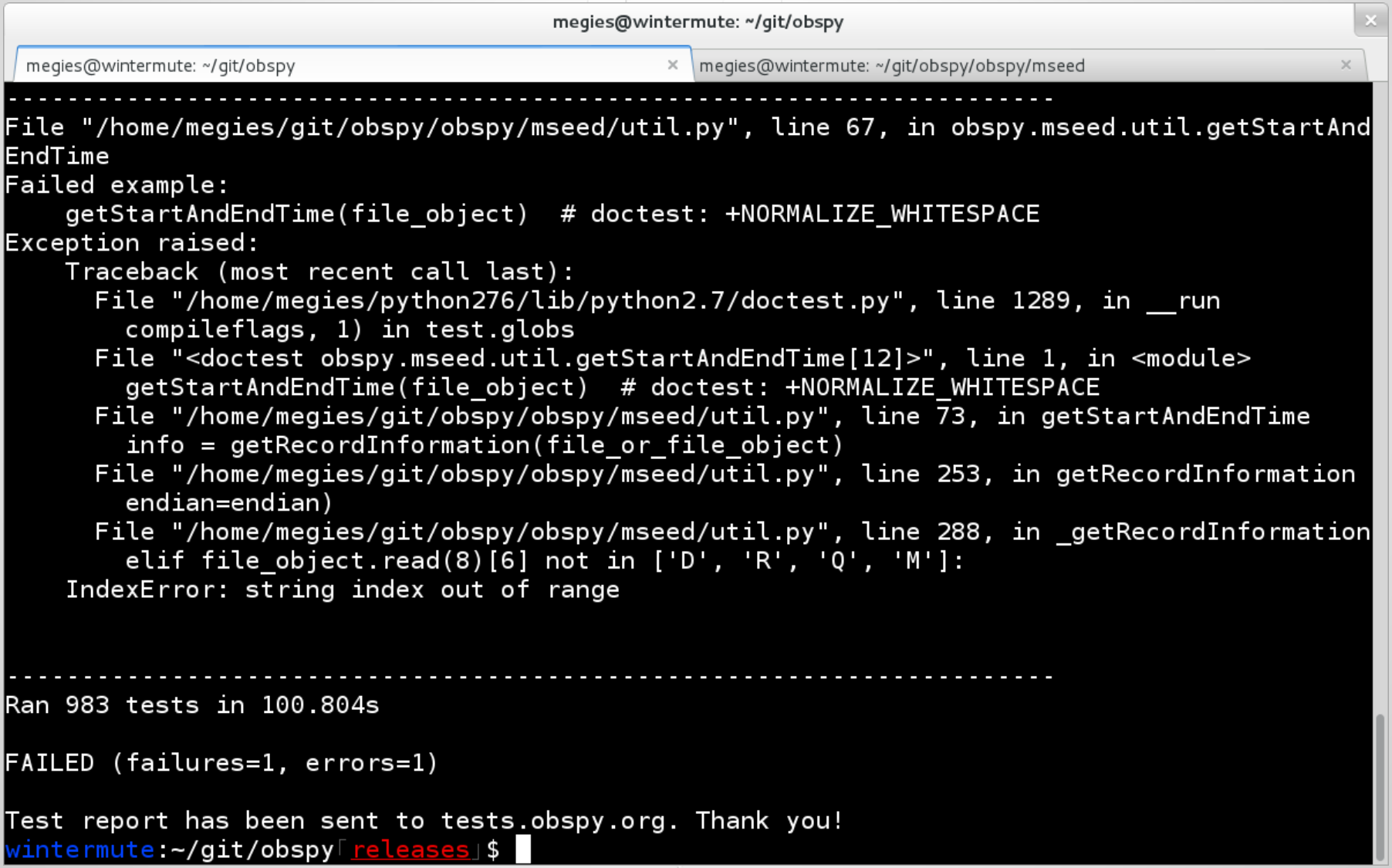
Technical Aspects: Test Reporting



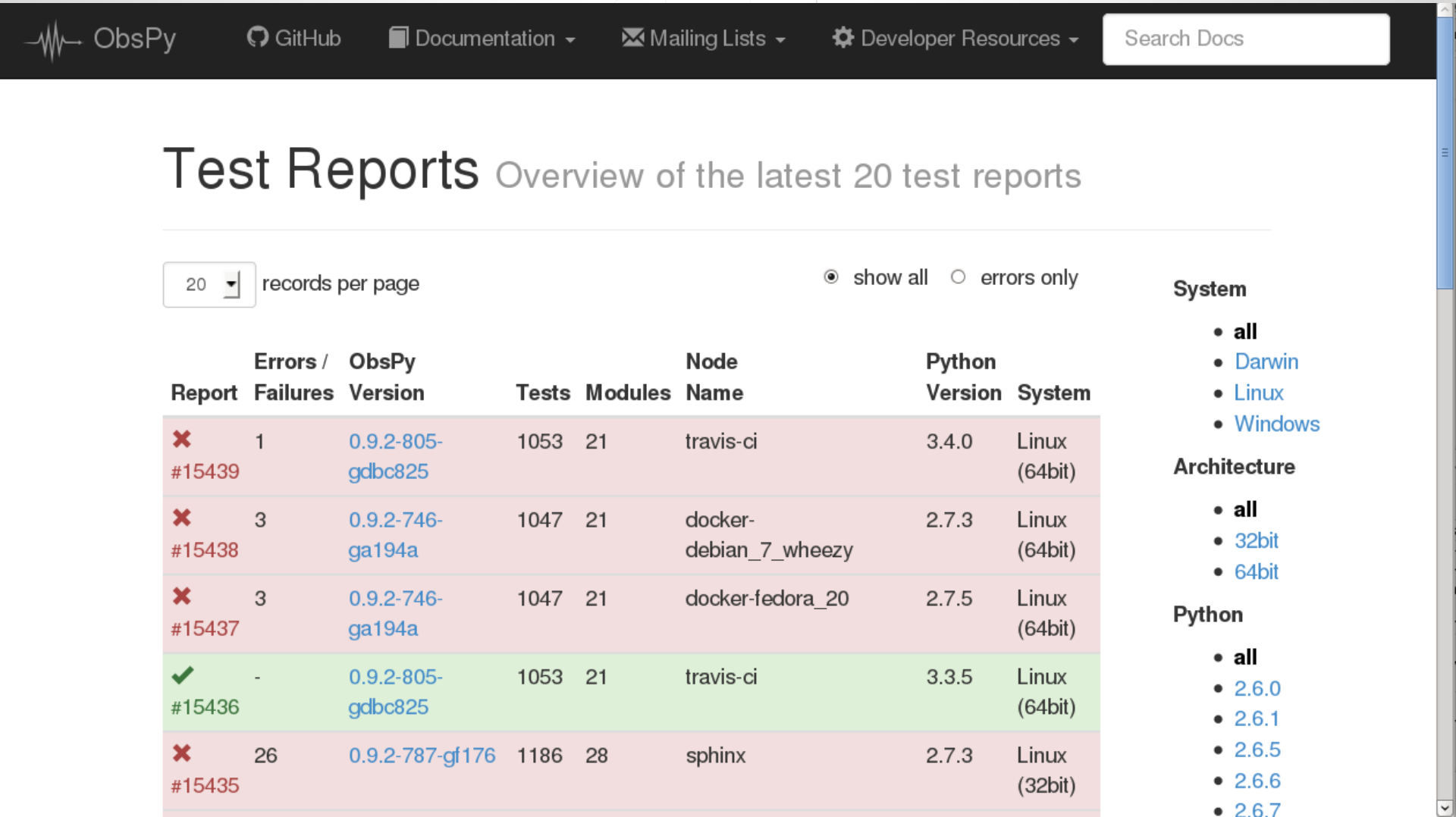
Technical Aspects: Test Reporting



Technical Aspects: Test Reporting



Technical Aspects: Test Reporting



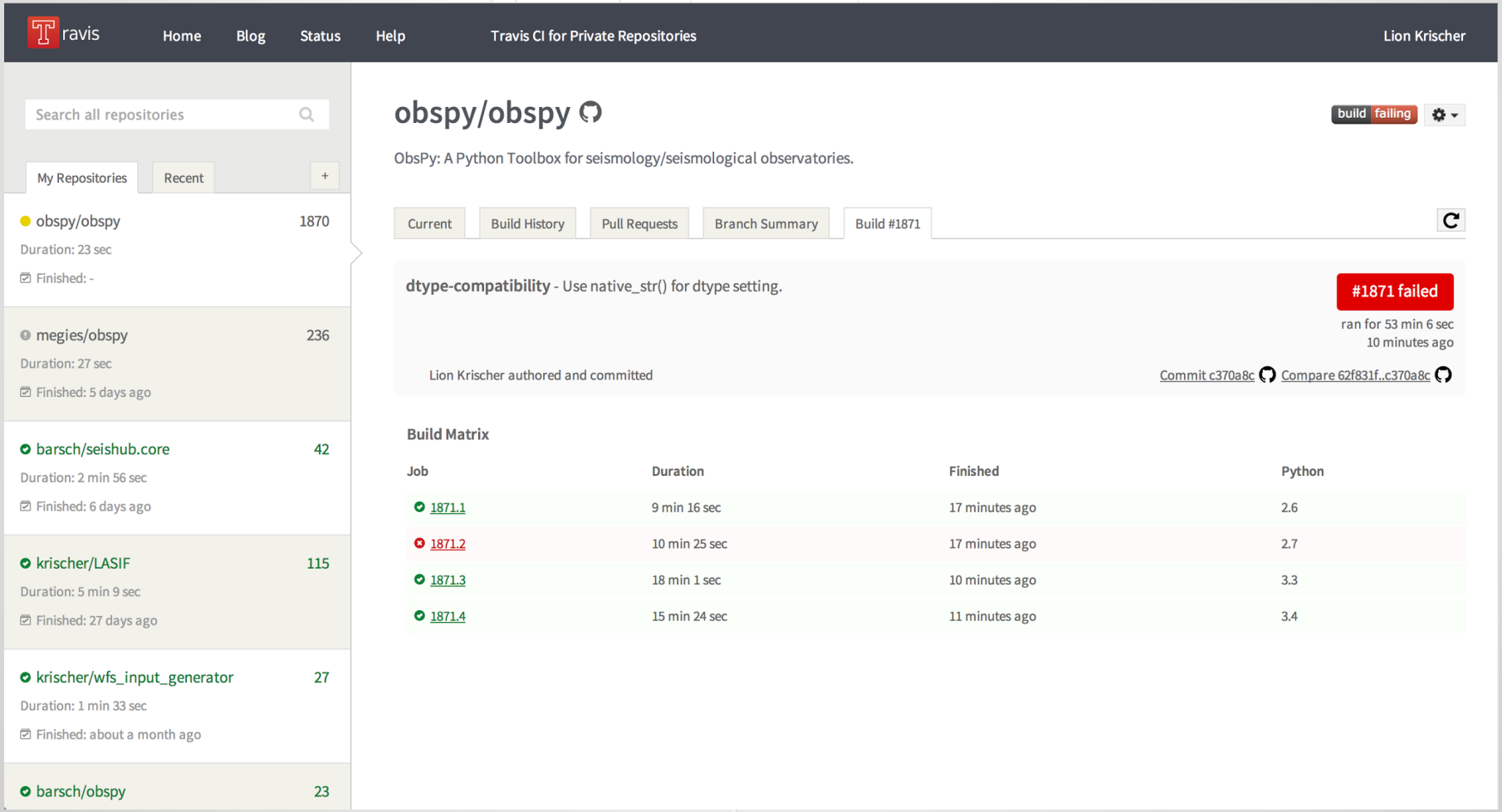
Technical Aspects: Testing

Continuous Integration

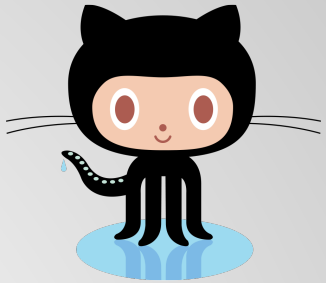
- automate the build (incl. dependencies)
- automate running all tests
- trigger building/testing automatically on changes in the repository
- online overview of build/test results
- keep building/testing fast
- each pull request is automatically tested

Technical Aspects: Testing

Travis: free CI service for open source projects



Technical Aspects: GitHub



GitHub: Free hosting of code repositories

- De facto standard for code hosting
- Worldwide community of developers
- Issue / bug tracker

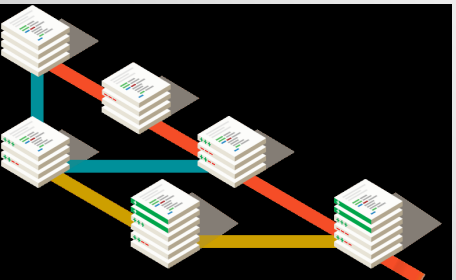
  (add comments, commits, integration with Travis CI)
- **Social coding and communication**
- Forking, pull requests
- Avoids overhead of self-hosting
- Visibility
- New: DOIs for Code => citable

Technical Aspects: git

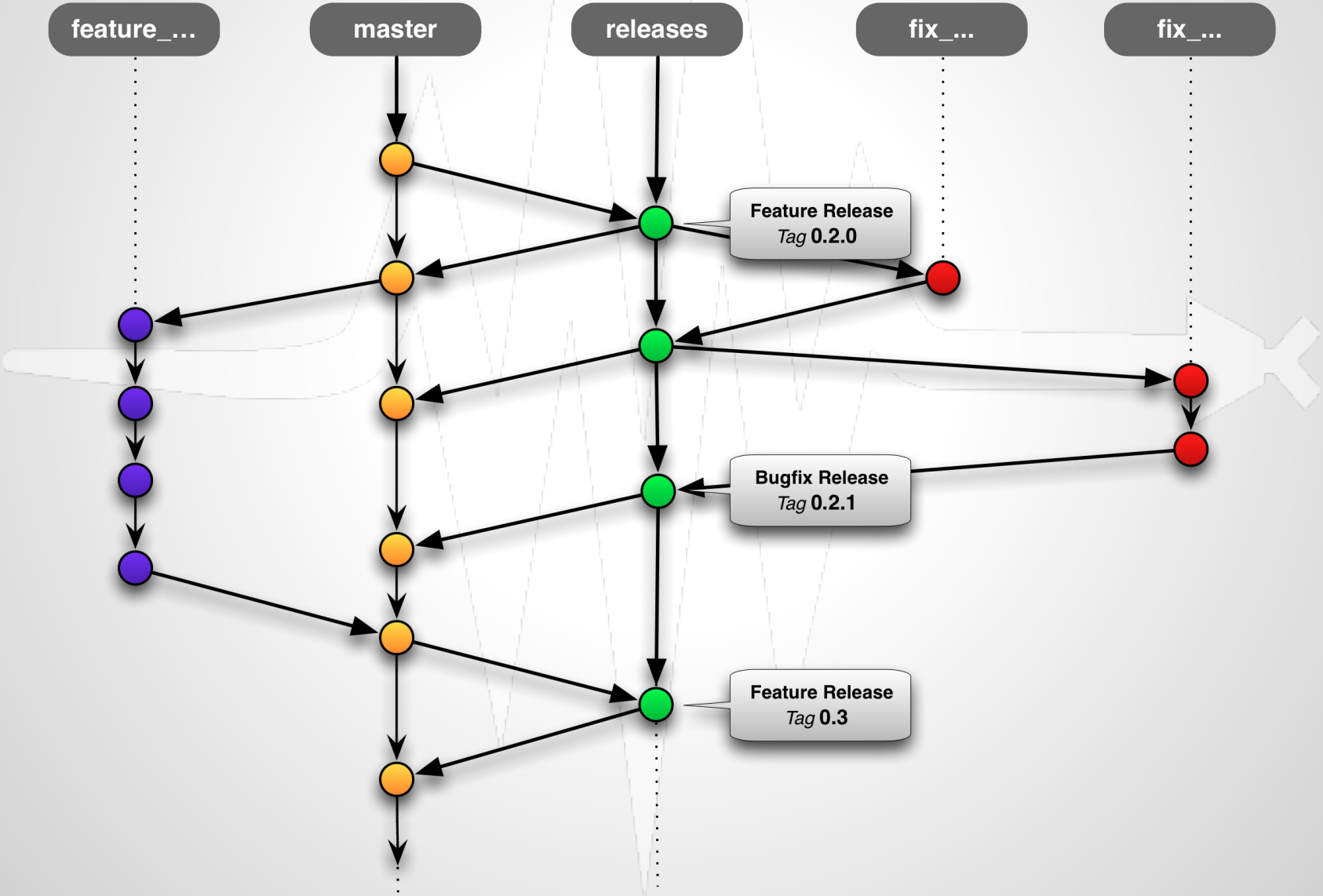
git: **distributed** version control software

- 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

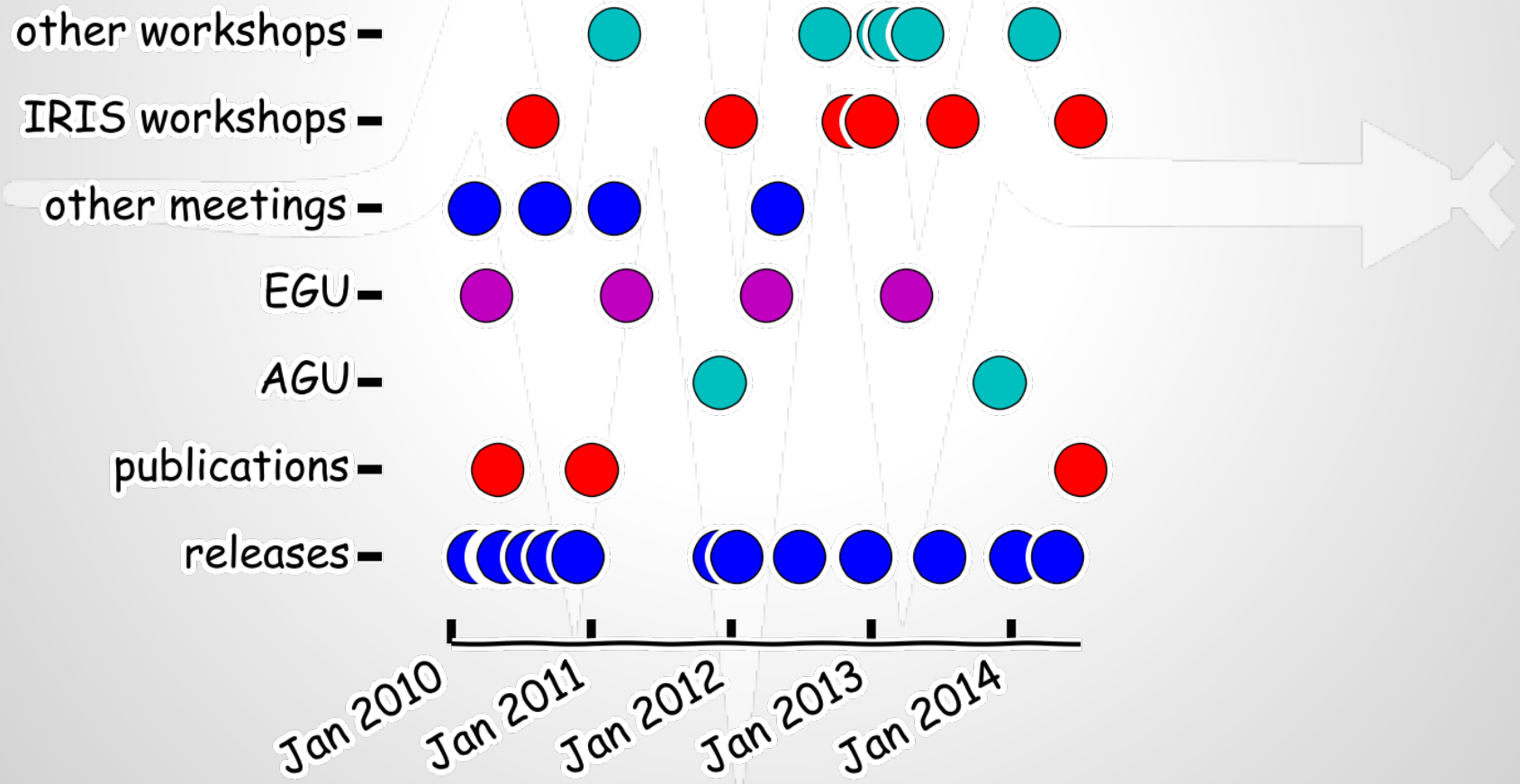
.. reliable (tests!)

.. interactive and responsive

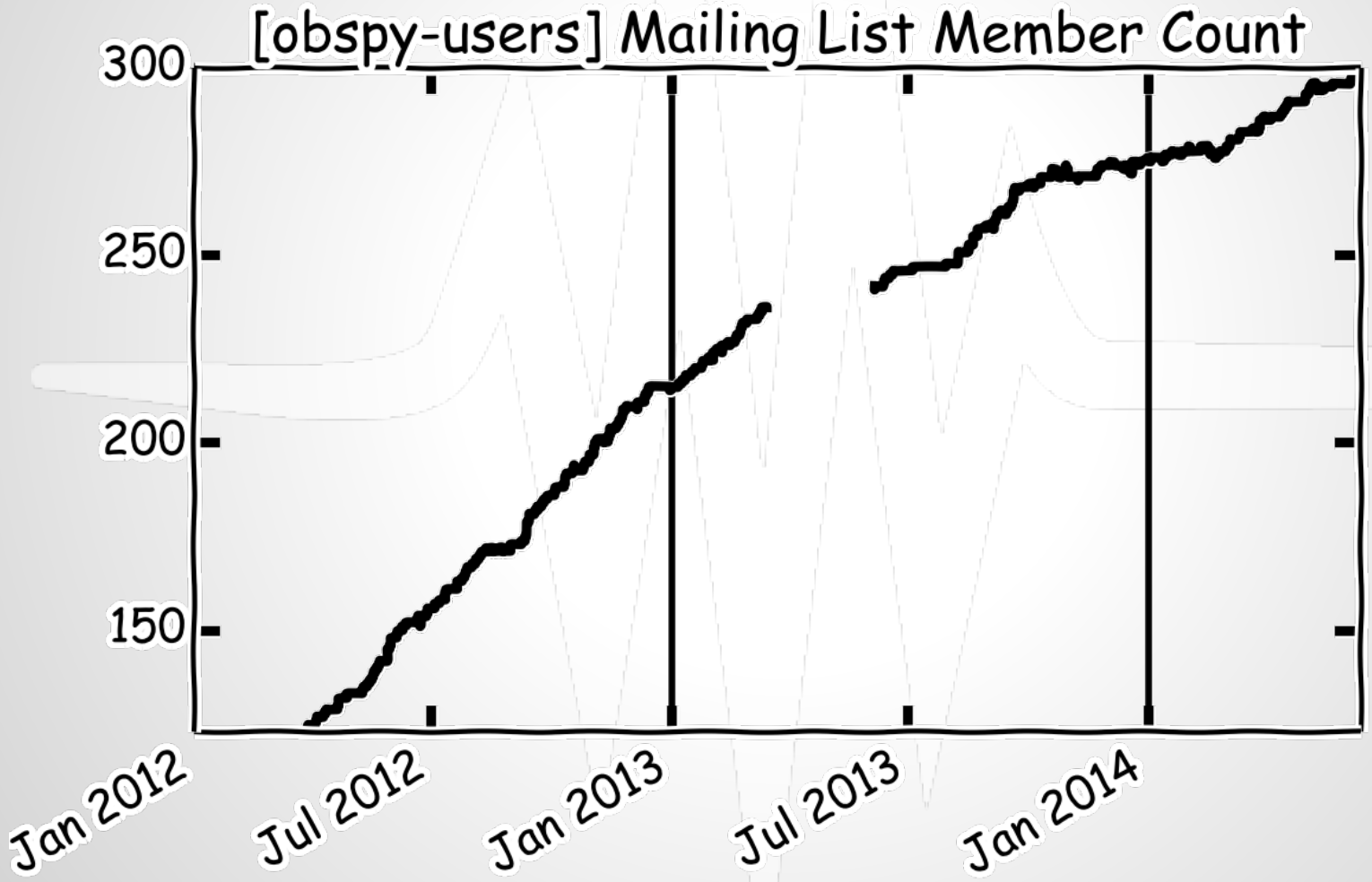
- github, wiki, mailing list

Spreading the word: how?

..wherever we go, we tell people about it

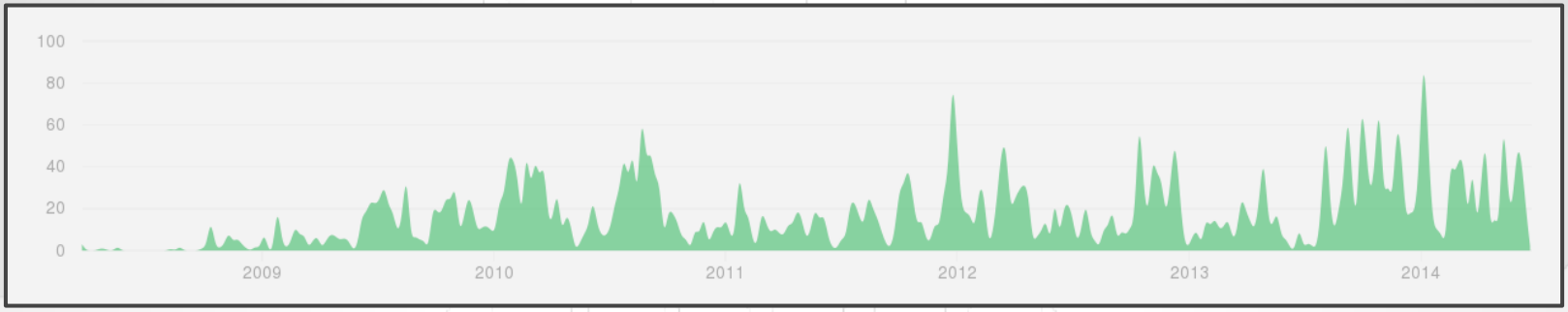


Spreading the word: success so far?



Spreading the word: success so far?

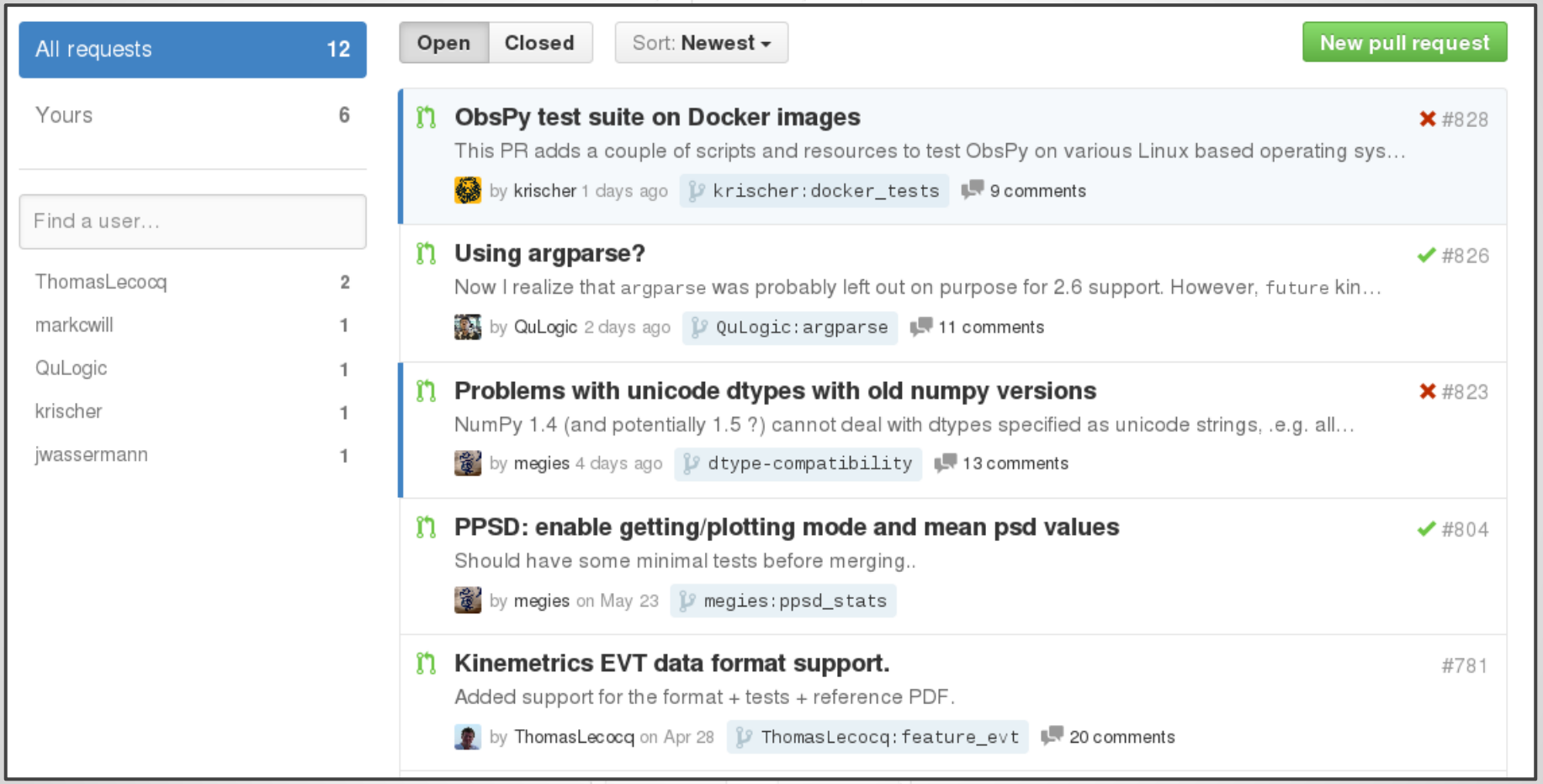
some stats..



- 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?



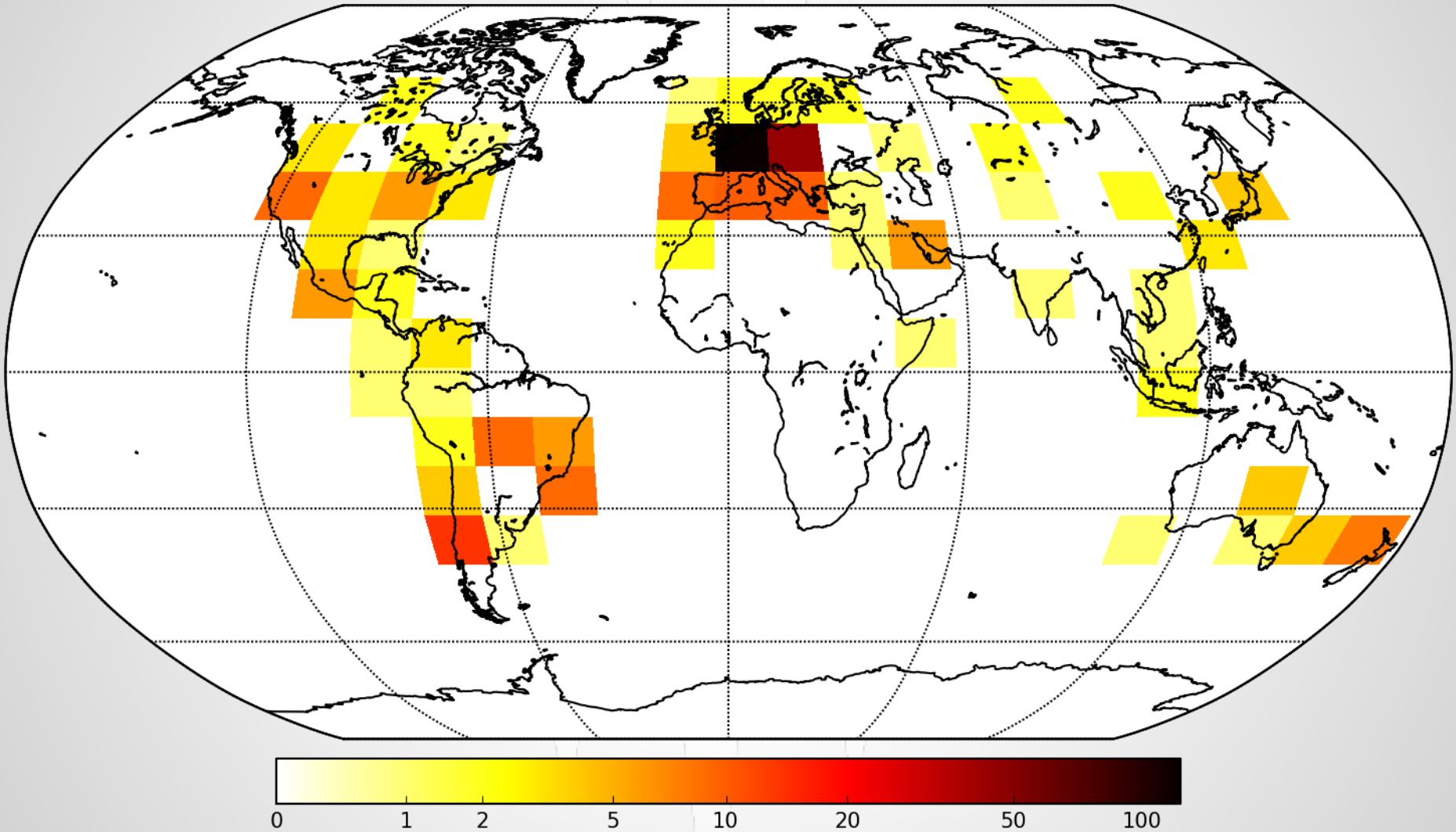
- lot of [active discussions](#) about improvements

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

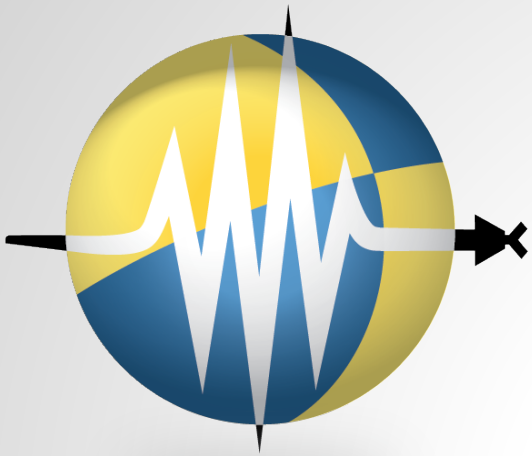
Spreading the word: success so far?



ObsPy 0.9.2 Debian/Ubuntu package downloads
(50 days after release, 348 total)

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