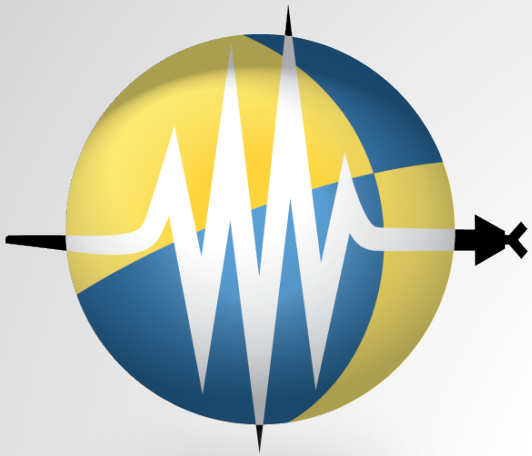


ObsPy

A Python Framework for Seismology

Introduction
Recent Developments
Future Plans

**Tobias Megies, Lion Krischer, Elliott Sales de Andrade,
Robert Barsch, Moritz Beyreuther (...)**

September 2015

Outline

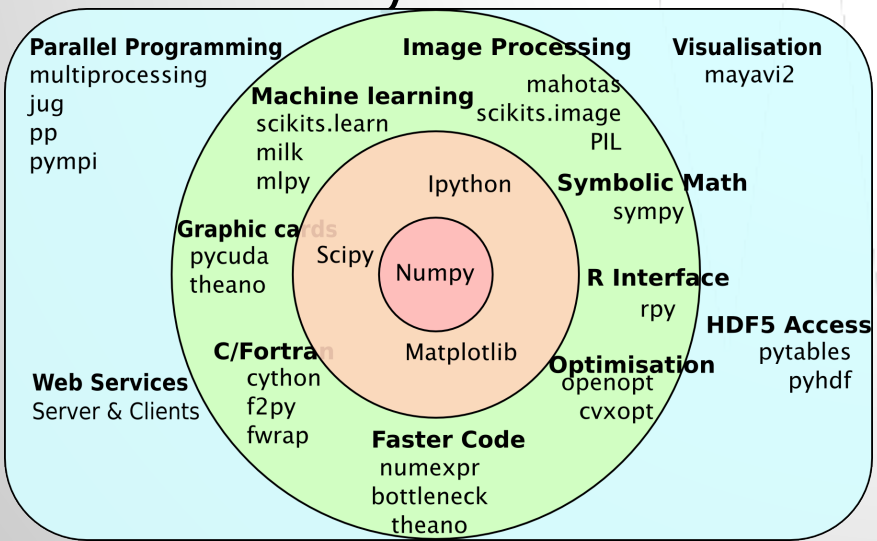
- Python for Seismology
- ObsPy
  - Overview / Functionality
  - Development Goals
  - Impact
  - Use Cases
  - Recent Developments
  - Future Plans

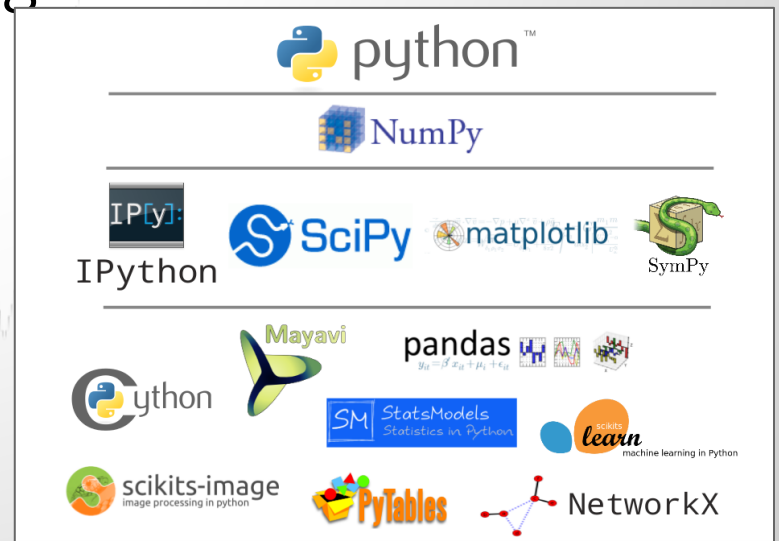
Why Python makes sense for seismology…

- easy to get started
  - one-click, 5 min, all-inclusive installation, binary package distribution (Anaconda)
  - simple, concise, and easy-to-read syntax
  - no compilation steps
  - interactive shell (with help system, tab-completion, …)
  - speed of development over speed of execution

Why Python makes sense for seismology…

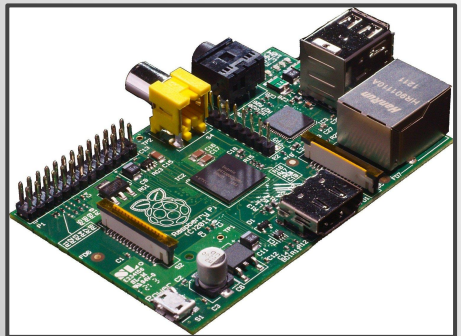
- easy to get started
- high potential
  - general purpose programming language
  - excellent support for scientific purposes
    (big data, signal analysis, visualization, …)
  - tons of small, special purpose scientific packages
  - easy to interact with existing C/Fortran codes

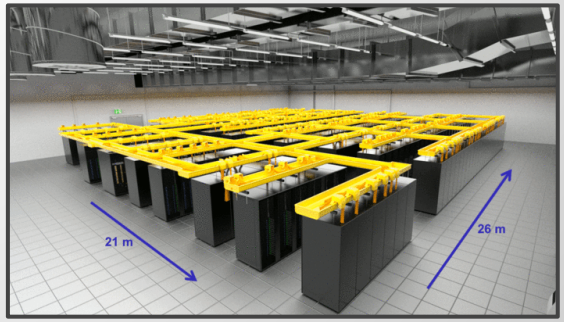




Why Python makes sense for seismology…

- easy to get started
- high potential
- high sustainability/value
  - free and open source
  - readable, reusable, extendable code
  - browser-based notebooks for teaching/sharing
  - cross-platform, cross-architecture
    - Linux, Mac, Windows
    - from Raspberry Pi to SuperMUC





Why Python makes sense for seismology…

- easy to get started
- high potential
- high sustainability/value
- strong, inherent bonds to seismology

Sir Bedevere:

...and that, my liege,
is how we know the Earth
to be banana-shaped.

King Arthur:

This new learning amazes
me, Sir Bedevere.
Explain again how sheep's
bladders may be employed
to prevent earthquakes.

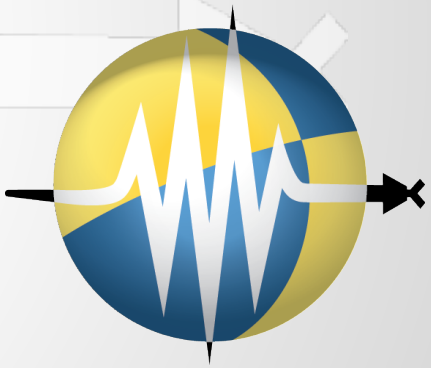
What is ObsPy?

Python library to work with seismological data

- waveform / station / event data

⇒ **teaches Python to “talk seismology”**

⇒ **a bridge for seismologists into the scientific Python ecosystem**



Facilitates development

- from short code snippets
- to complex processing workflows

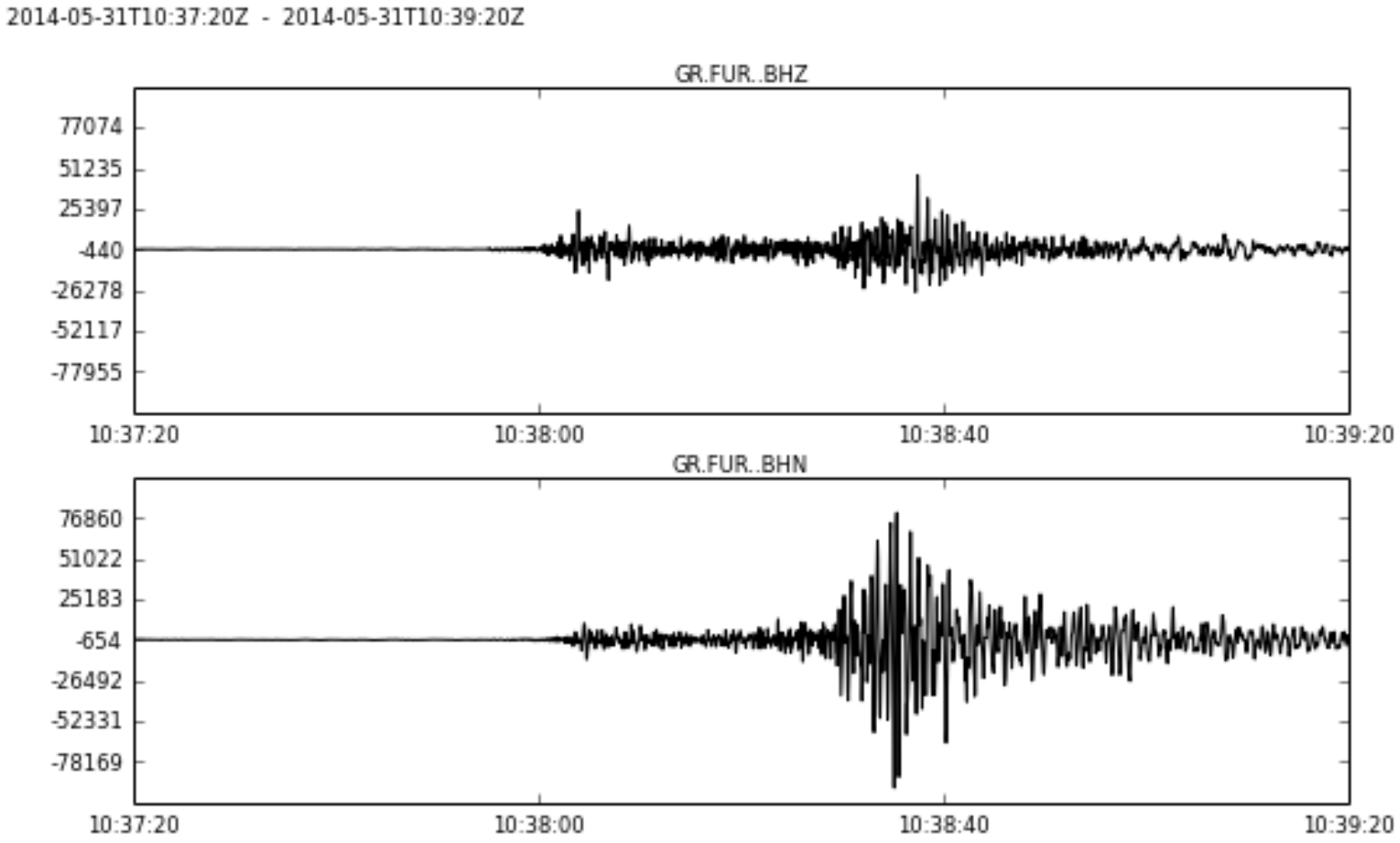
⇒ **makes every day seismology tasks easy, complex things possible**

Functionality?

- read/write support
  - waveforms (MiniSEED, SAC, GSE, SEISAN, SEG-Y, SEG2, Q, CSS, ...)
  - station metadata (StationXML, dataless/full SEED, RESP, SACPZ)
  - event metadata (QuakeML, NDK, NLLOC, ZMAP, CNV, MCHEDR)
- data center access
  (FDSN WS, ArcLink, SeedLink, Earthworm, NEIC, ...)
- unified object classes
- signal processing routines
  - trim, merge, taper, rotate, resample, interpolate, ...
  - filtering, instrument correction, ...
  - theoretical travel times/ray paths, array analysis, CC routines, PPSD, event detection, TF misfits, ...
- visualization capabilities

Functionality?

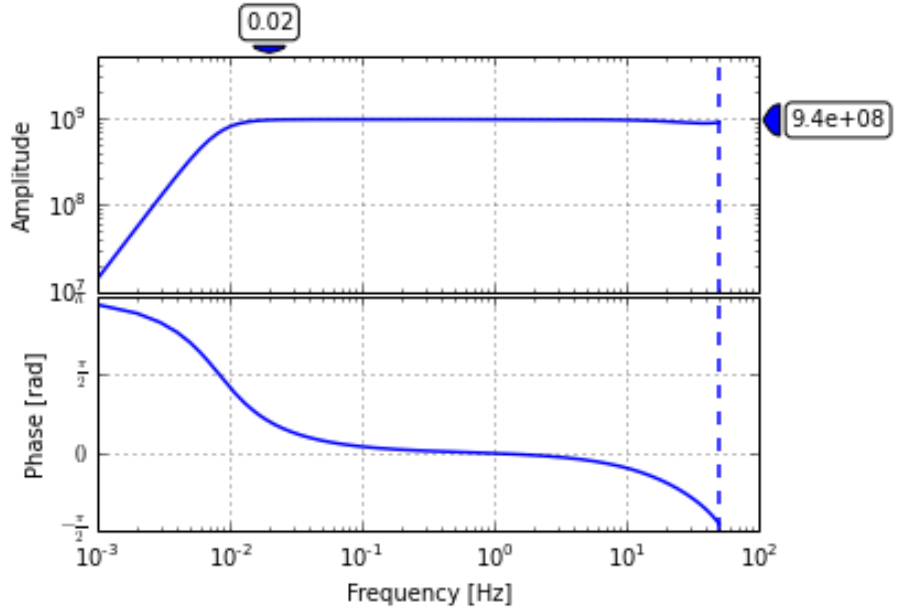
```
In [2]: from obspy import read
        stream = read("waveform.mseed")
        stream.plot()
```



Functionality?

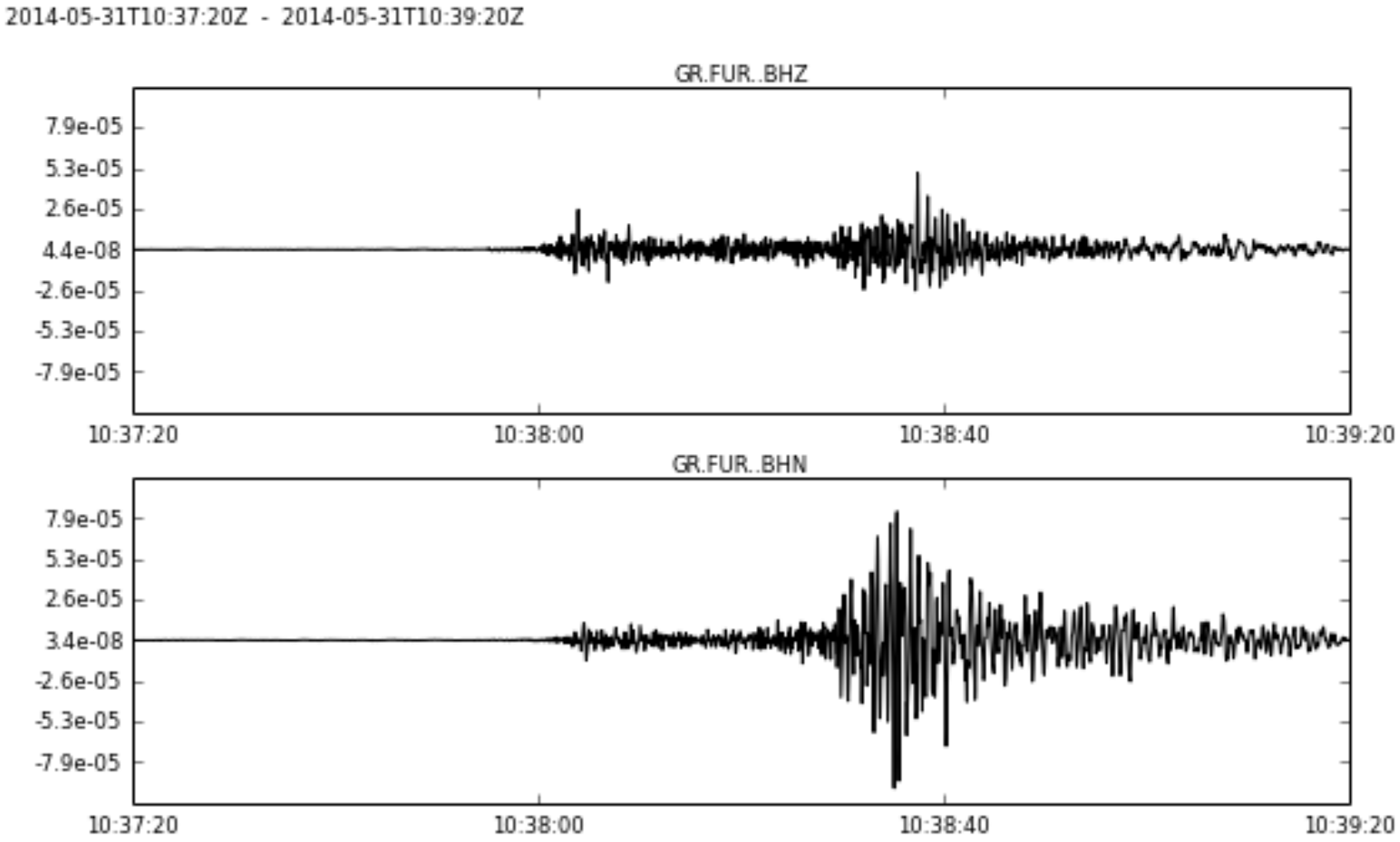
In [3]:
```
from obspy import read_inventory
inventory = read_inventory("station.xml")
channel = inventory[0][0][0]
print channel
channel.plot(0.001)
```

```
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
```

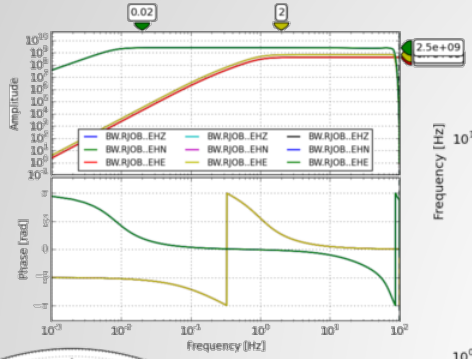


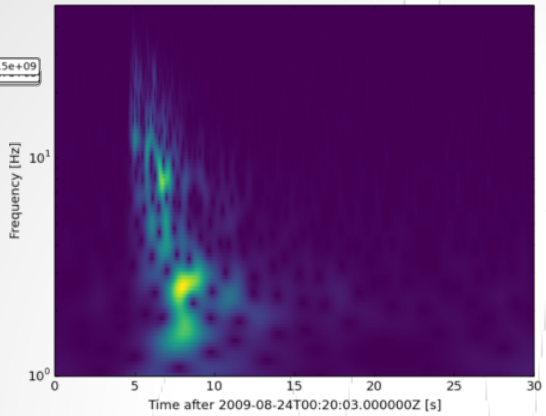
Functionality?

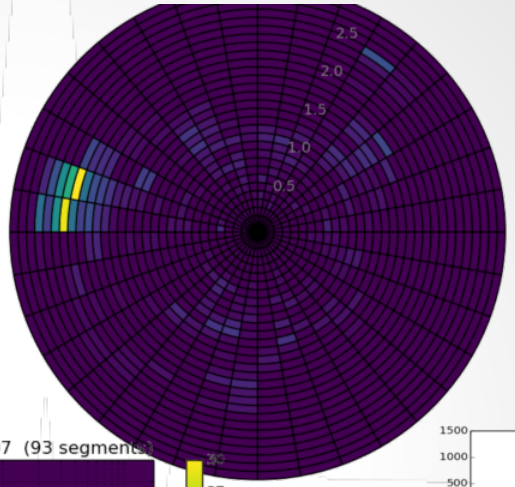
In [4]:
```
stream.attach_response(inventory)
stream.remove_response(output="VEL")
stream.plot()
```

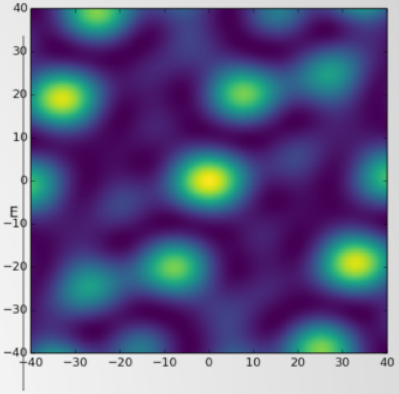


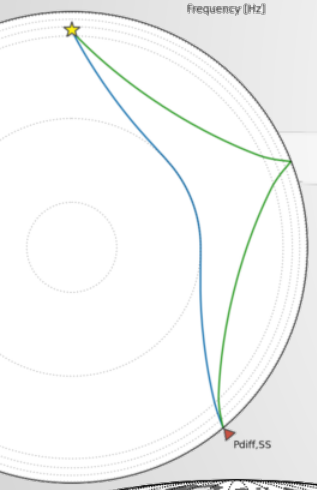
Functionality?

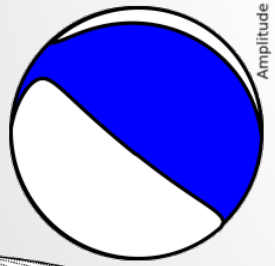


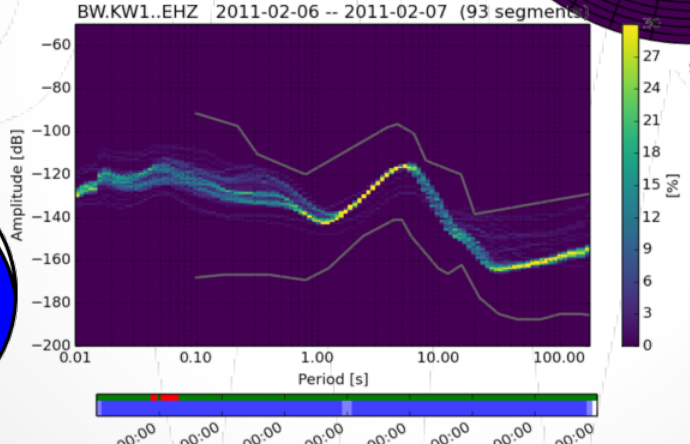


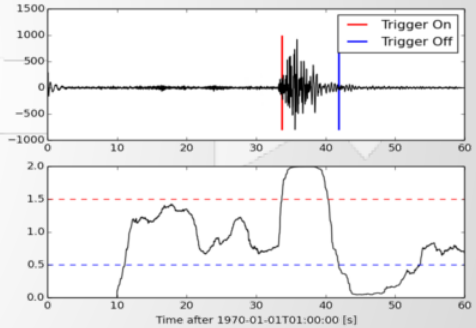


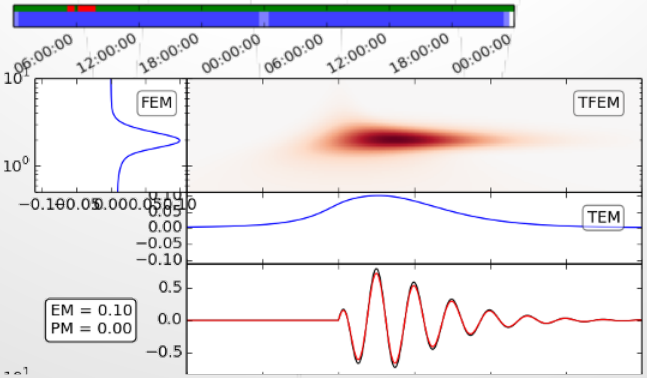


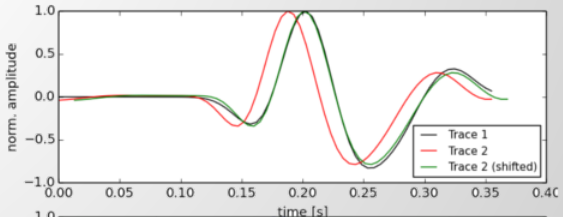


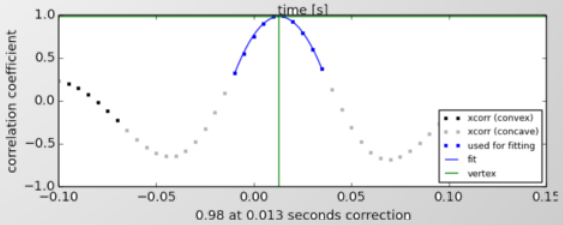


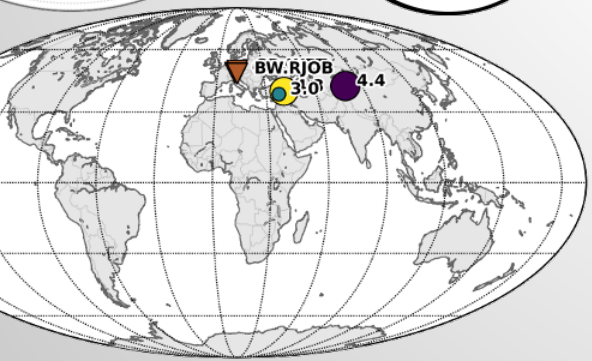












Development goals: try to make it...

...easy to use

- binary packages, system-specific packaging
- documentation: http://docs.obspy.org
- tutorial
- gallery
- mailing list
- workshop documents online

...reliable

- test-driven development
  >1300 tests 86% Test Coverage Travis/AppVeyor CI
- continuous development since 2008

...transparent and accessible to external contributors

- git: distributed version control, branching/merging
- github: central platform for hosting, ticket system, "social coding", CI integration

Development goals: try to make it...

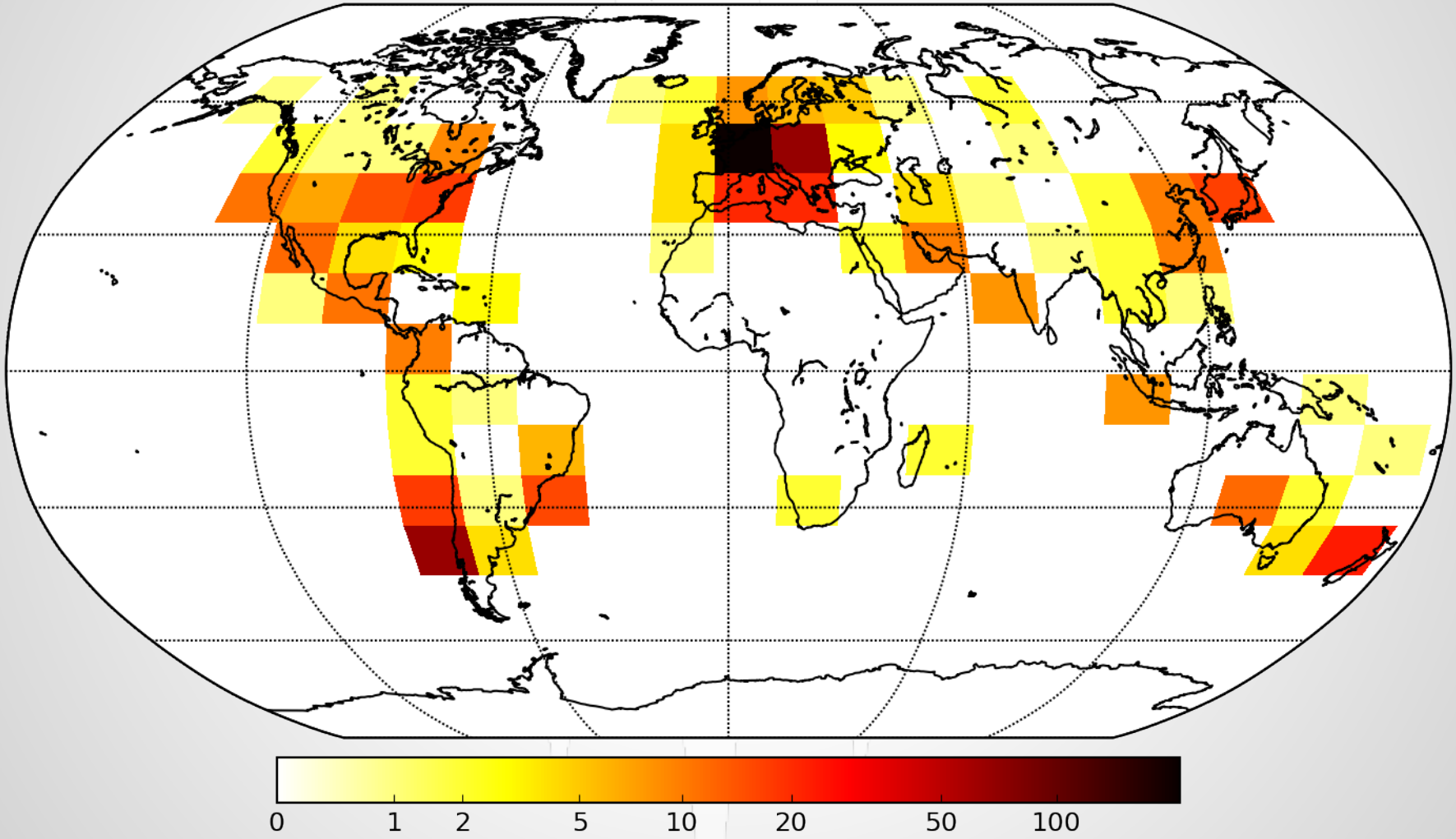
...easy to use

...reliable

...transparent and accessible to external contributors

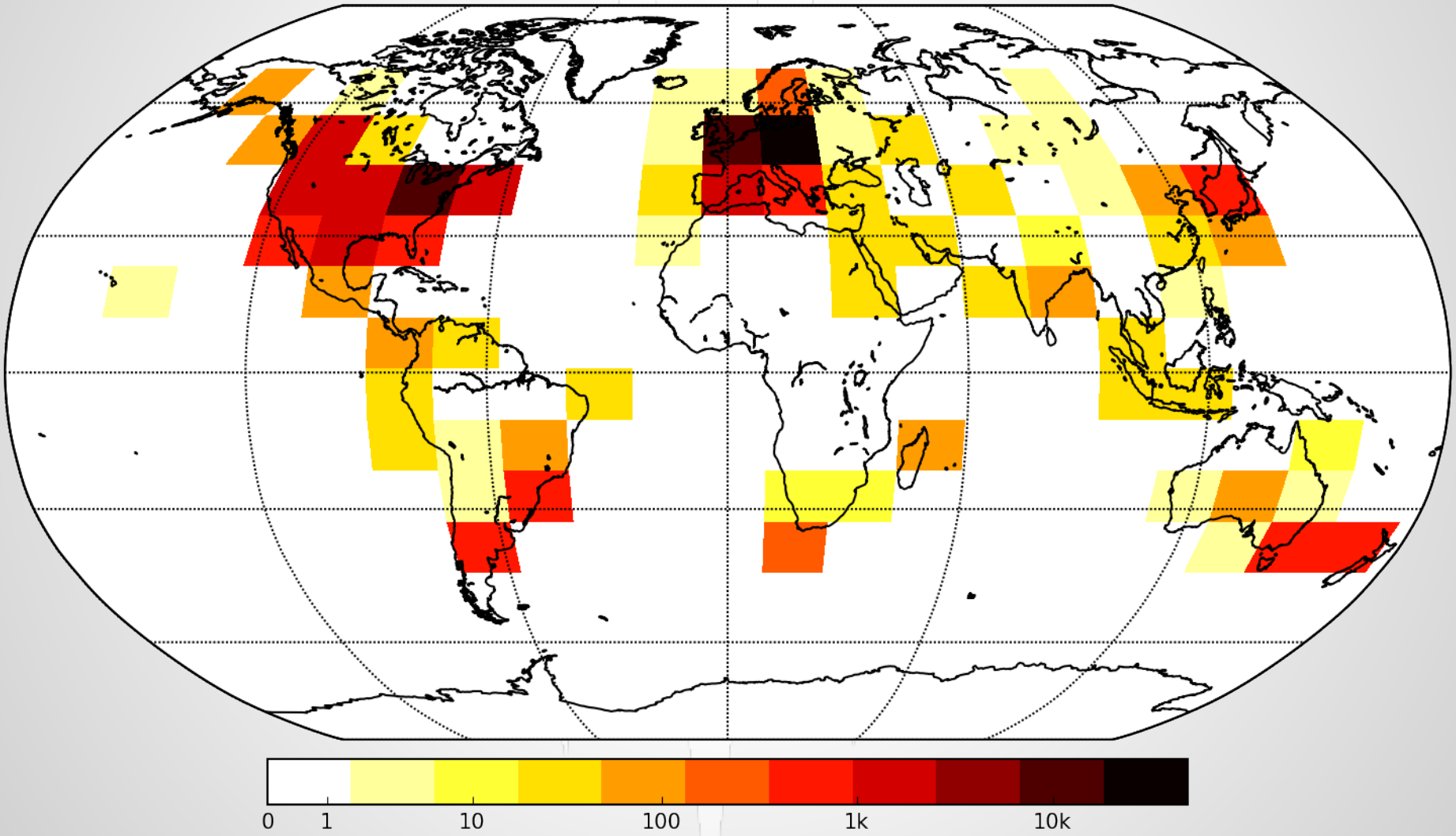
⇒ increase user base
⇒ increase external contributions
⇒ self-sustaining project, no “hero code”

Impact



ObsPy 0.10.2-3 Debian/Ubuntu package downloads (707 unique IPs in 123 days)
This represents only a small subset of all Obspy downloads.

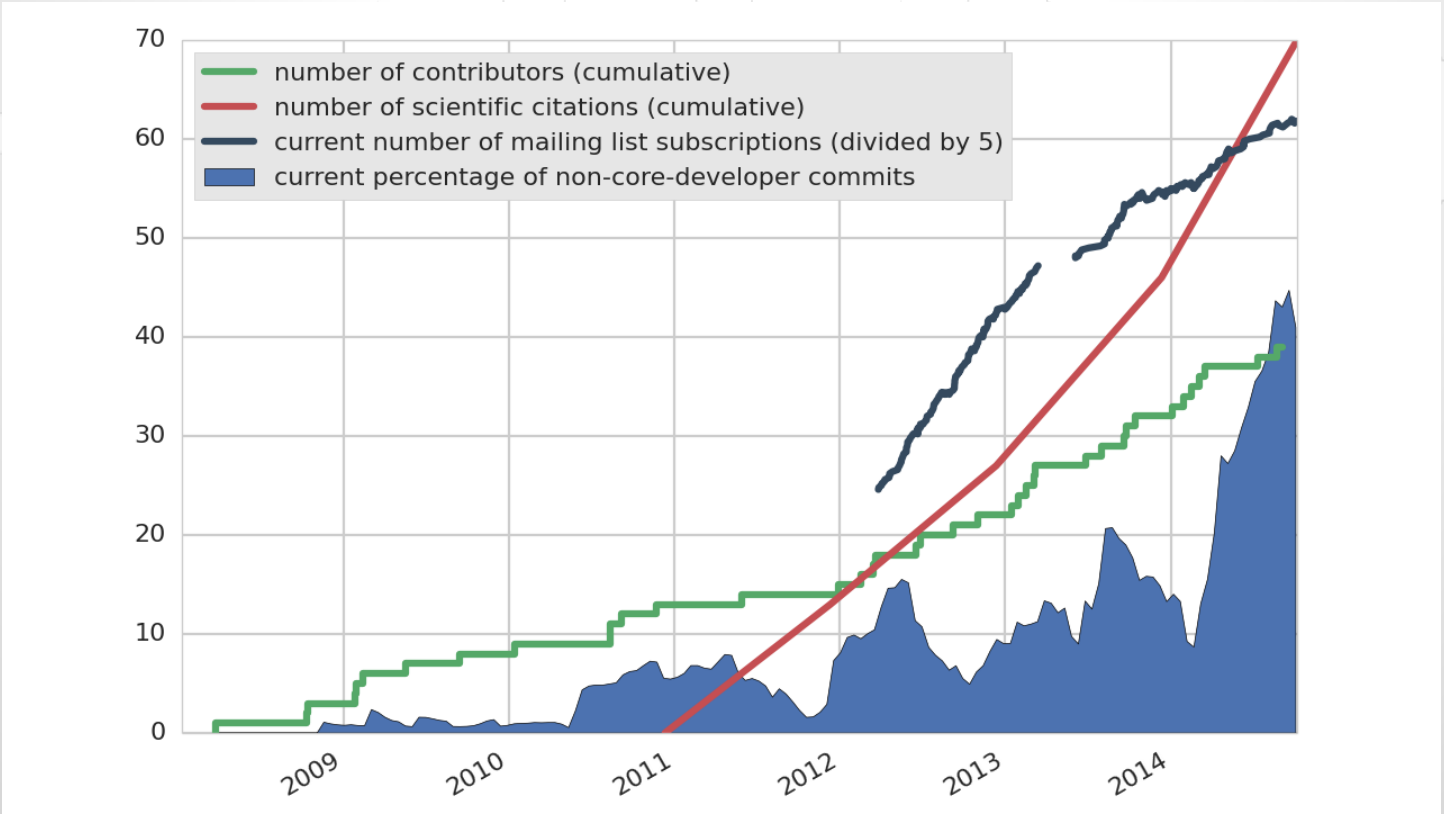
Impact



IRIS webservice requests using ObsPy (2011-01-23 to 2015-05-25)

Impact

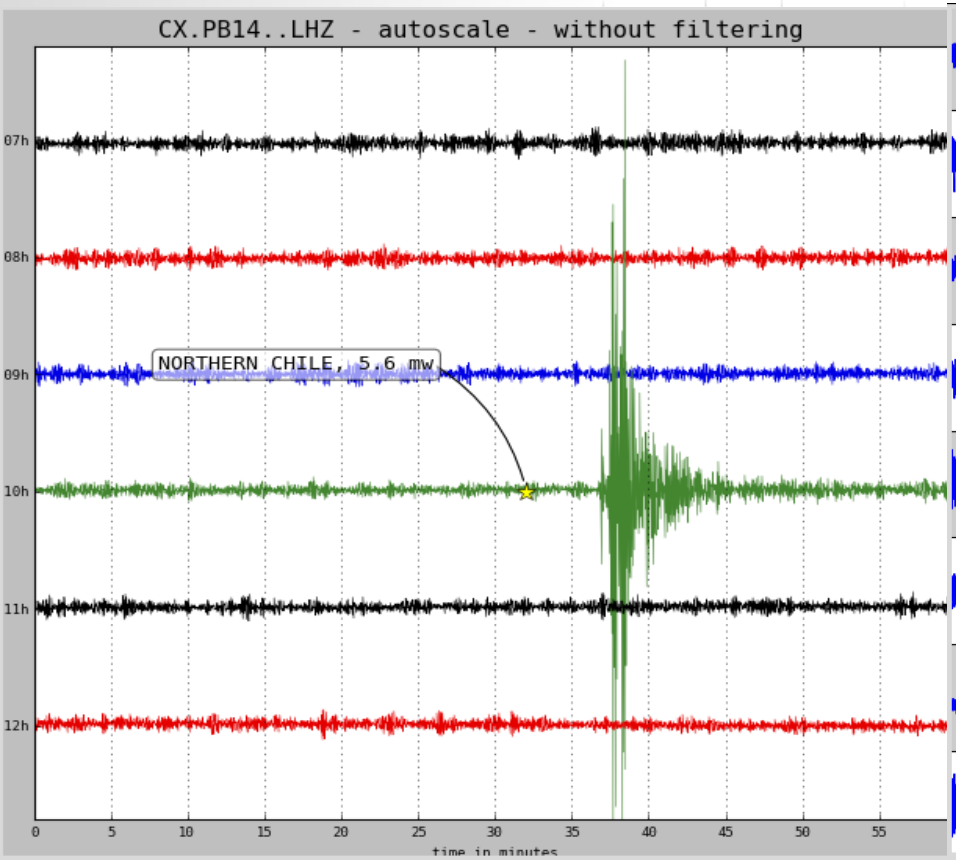
- >350 people on users mailing list
- 132 scientific citations
  Beyreuther et al 2010 Megies et al 2011 Krischer et al 2015 (OA)
- ~40 contributors

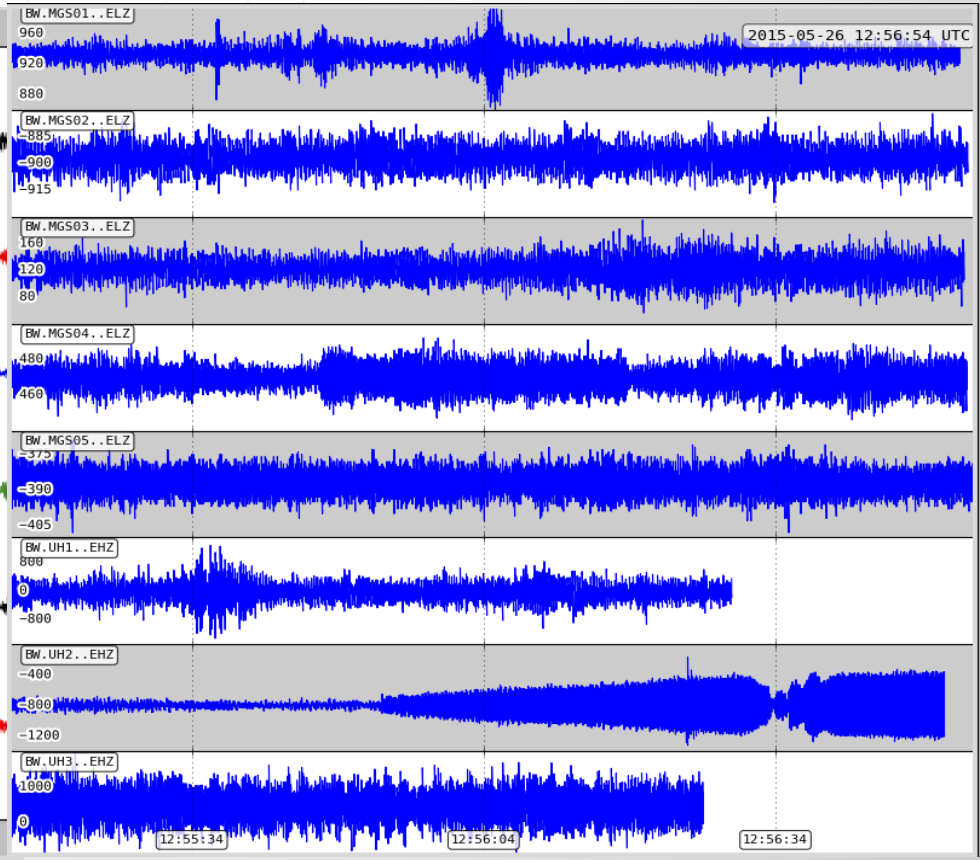


Use cases

- seedlink plotter

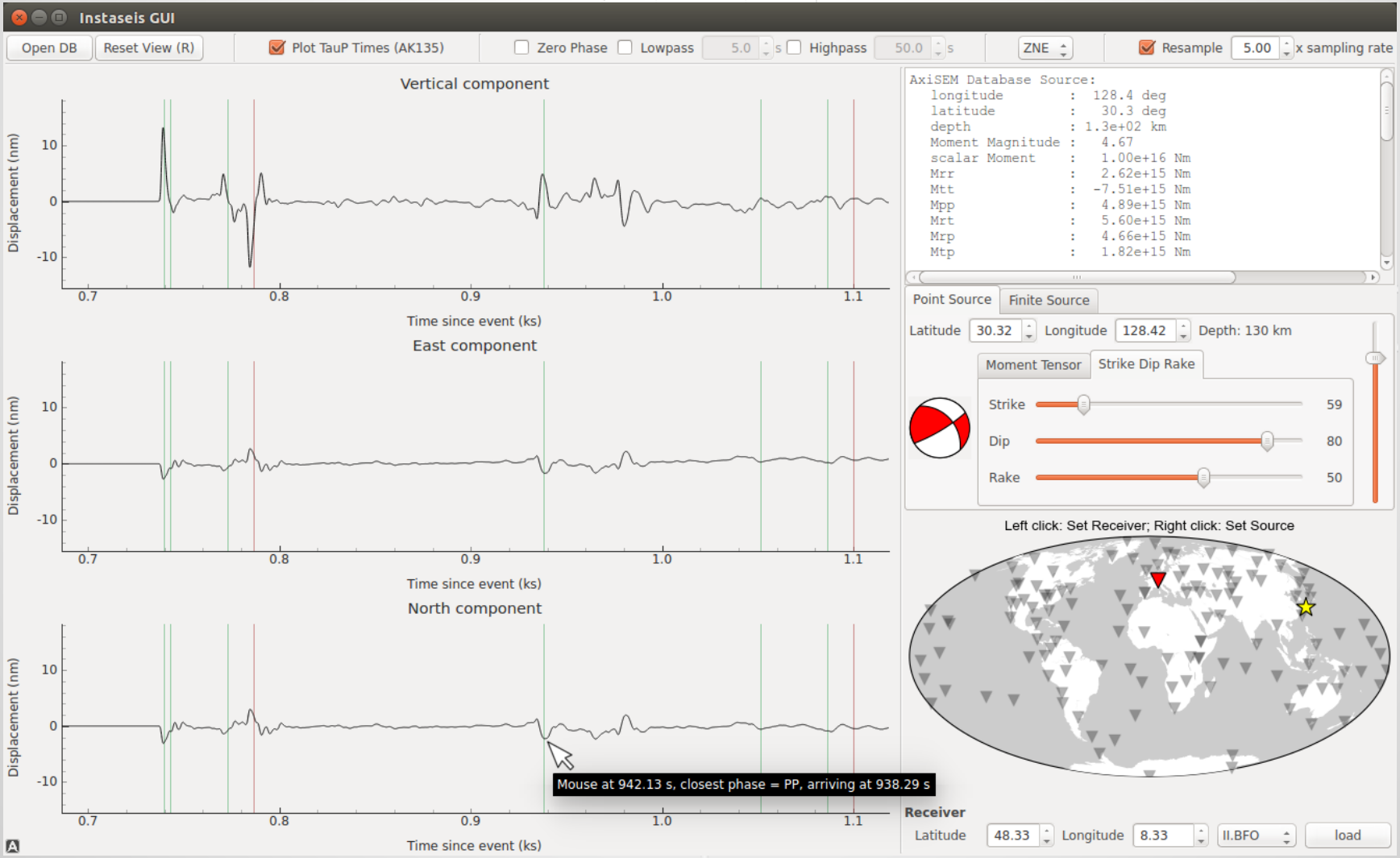
```
$ seedlink-plotter -s "GE_SLIT:HHZ,GE_MORC:HHZ"
 --seedlink_server "geofon.gfz-potsdam.de:18000"
 -b 5m --update_time 1s
```





Use cases

- instaseis



Use cases

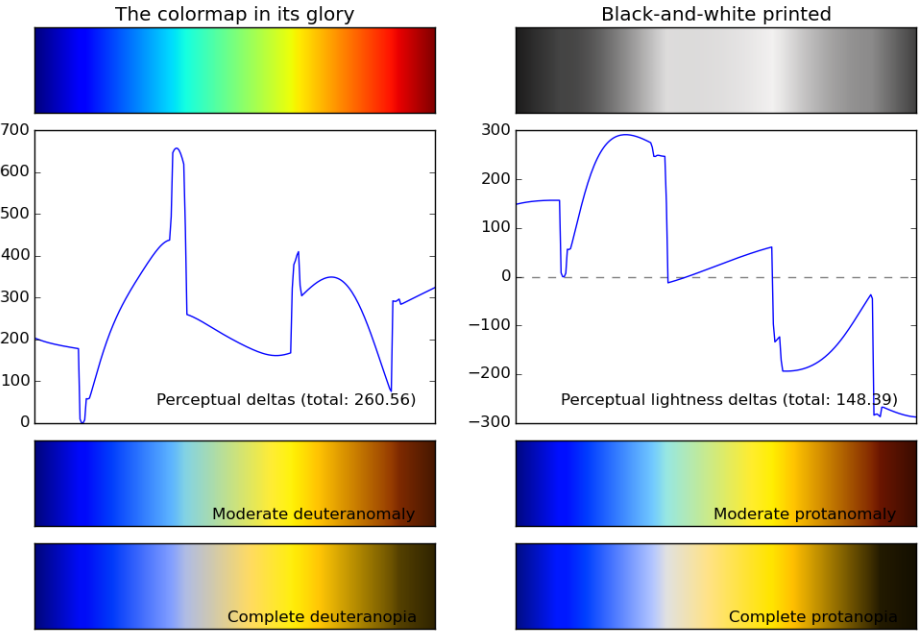
- ObsPyck
- SeismicHandler
- SeisHub
- SeishubExplorer
- HtoVToolbox
- wavePicker
- Antelope Python moment tensor code
- Using ObsPy with py2exe
- Whisper
- Wavesdownloader (on GitHub)
- ADMIRE project
- pSysmon
- ObsPyDMT
- seedlink plotter
- pyTDMT - Python Time Domain Moment Tensor (on GitHub)
- MSNoise - Monitoring Seismic Velocity Changes using Ambient Seismic Noise (on GitHub)
- Pisces: A practical seismological database library in Python (on GitHub)
- HASHpy: Python wrapped fork of HASH first motion focal mechanism code
- waveloc (on GitHub)
- scisola (on GitHub)
- instaseis - Instant Global High Frequency Seismograms
- LASIF - Large-Scale Seismic Inversion Framework
- pyflex - Enhanced port of FLEXWIN
- hypoDDpy - Run hypoDD in a data driven manner.
- wfs_input_generator - Generate input files for many waveform solvers directly from data.
- rf - Calculate receiver functions.
- Qopen - Separation of intrinsic and scattering Q by envelope inversion.
- ...

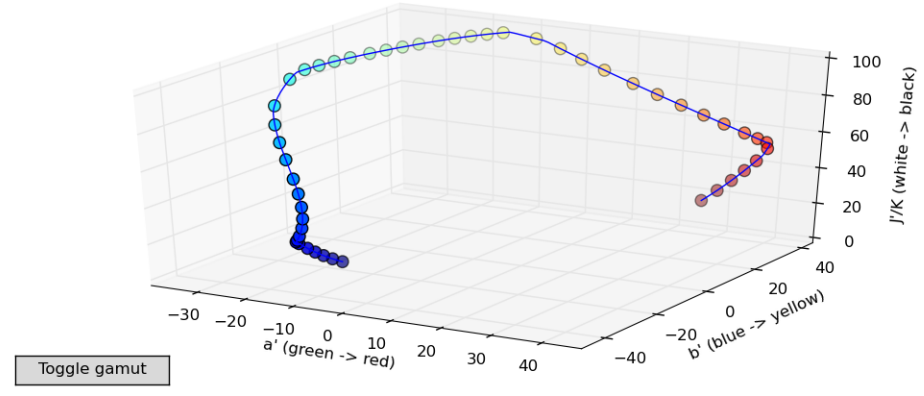
Recent Developments

- Python 2+3 support
- more stable filters
  - backported from latest scipy release
  - cascaded second-order sections
- new default colormaps
  - aka. *“don’t use jet”*
  - "perceptually uniform", etc.
    → youtube: “colormap scipy 2015”
  - backported from upcoming matplotlib release

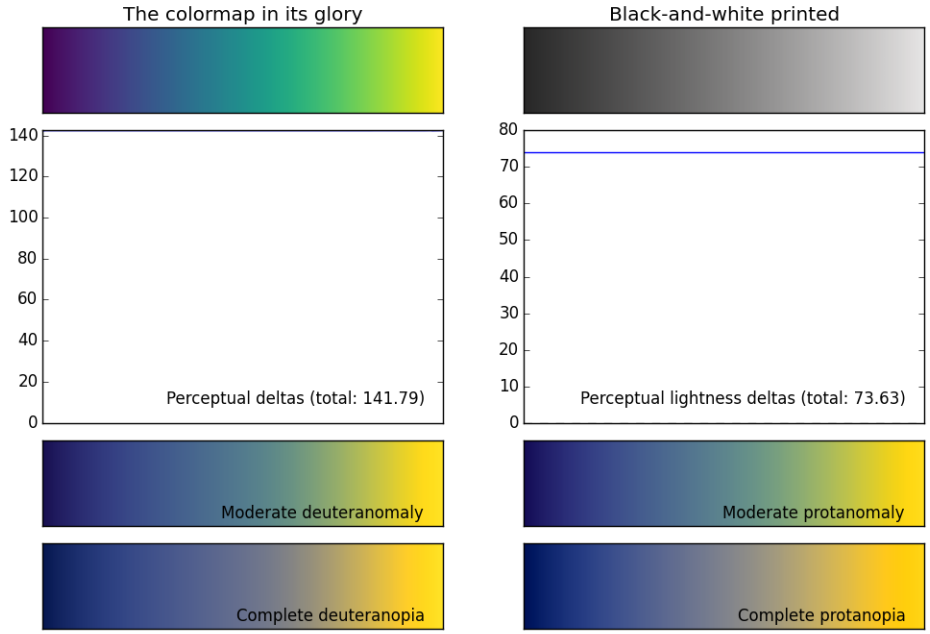
Recent Developments

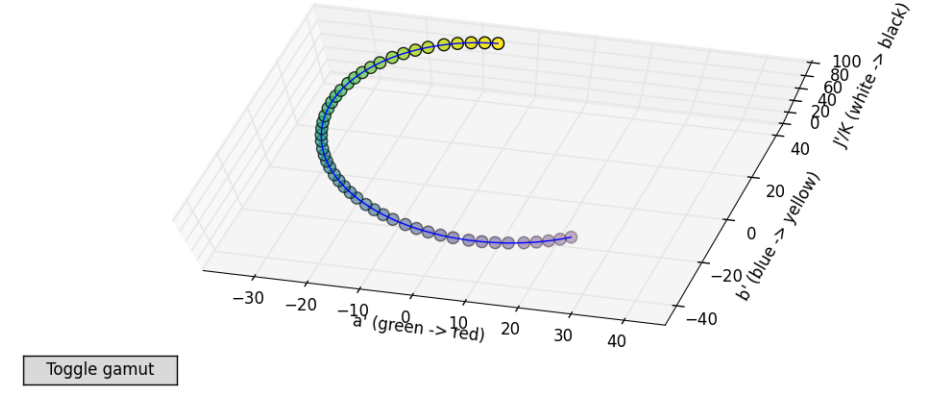
jet





viridis

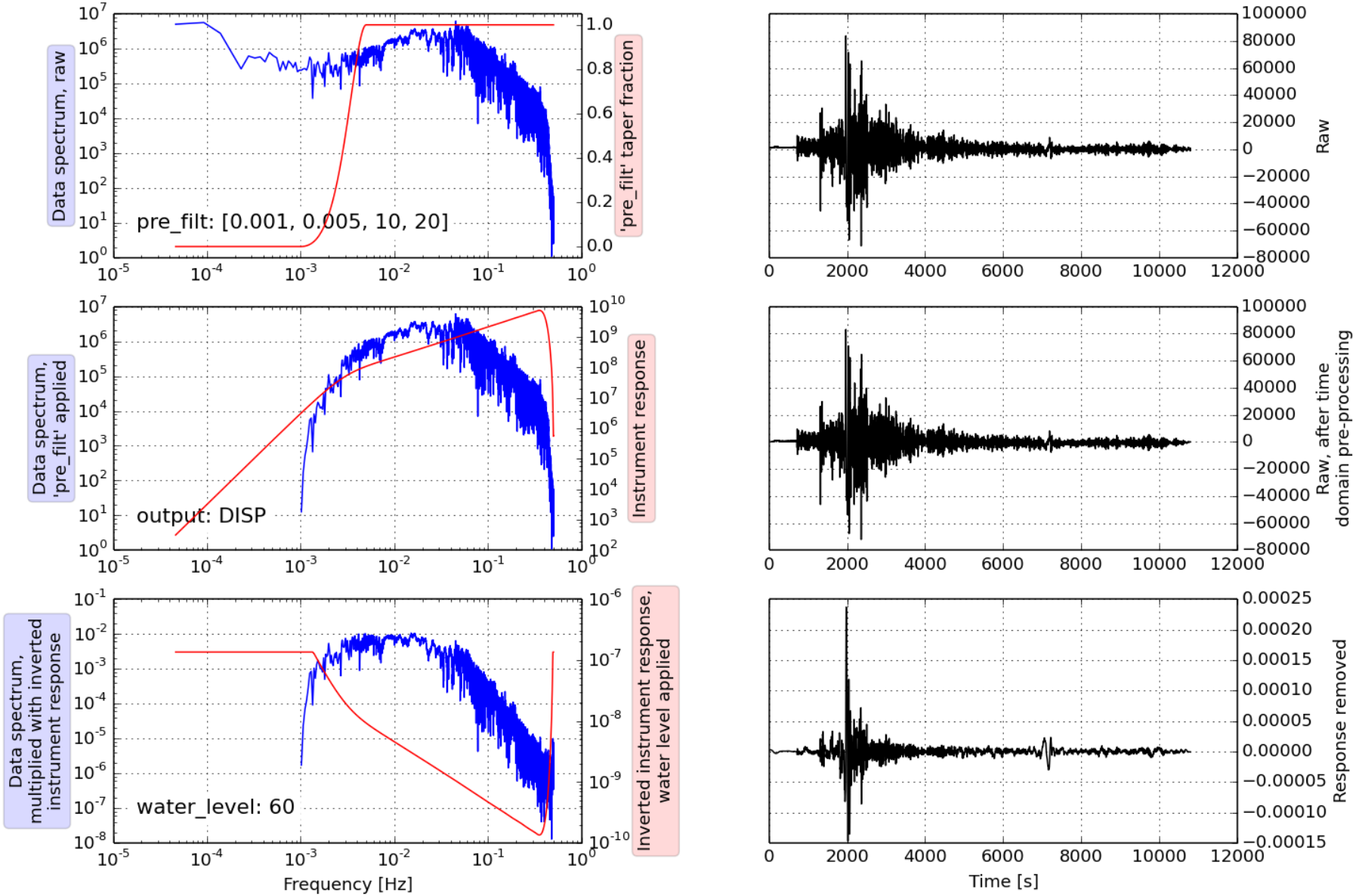




Recent Developments

- Python 2+3 support
- more stable filters
- new default colormaps
- response removal plot
  - less “black box”
  - transparency for what is going on in frequency domain

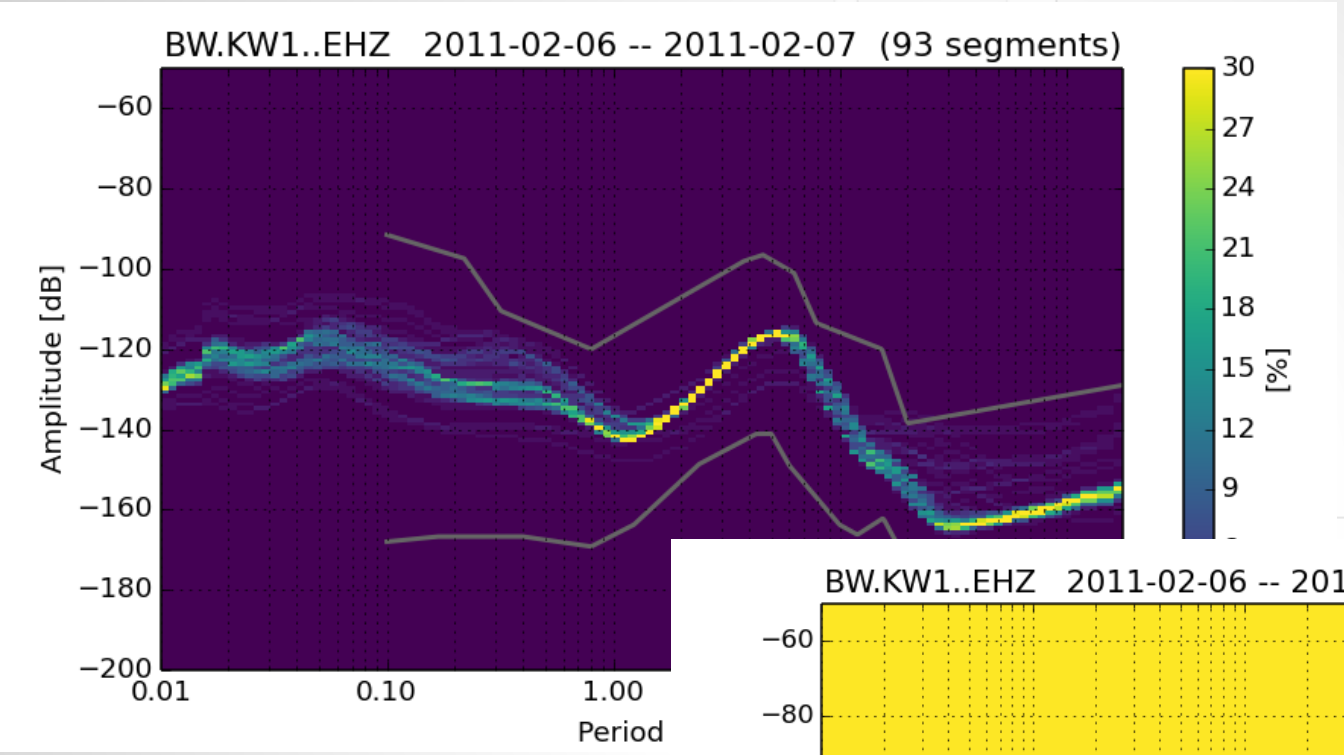
```
>>> st.remove_response(output="DISP",
...                    pre_filt=[0.001, 0.005, 10, 20],
...                    water_level=60, plot=True)
```

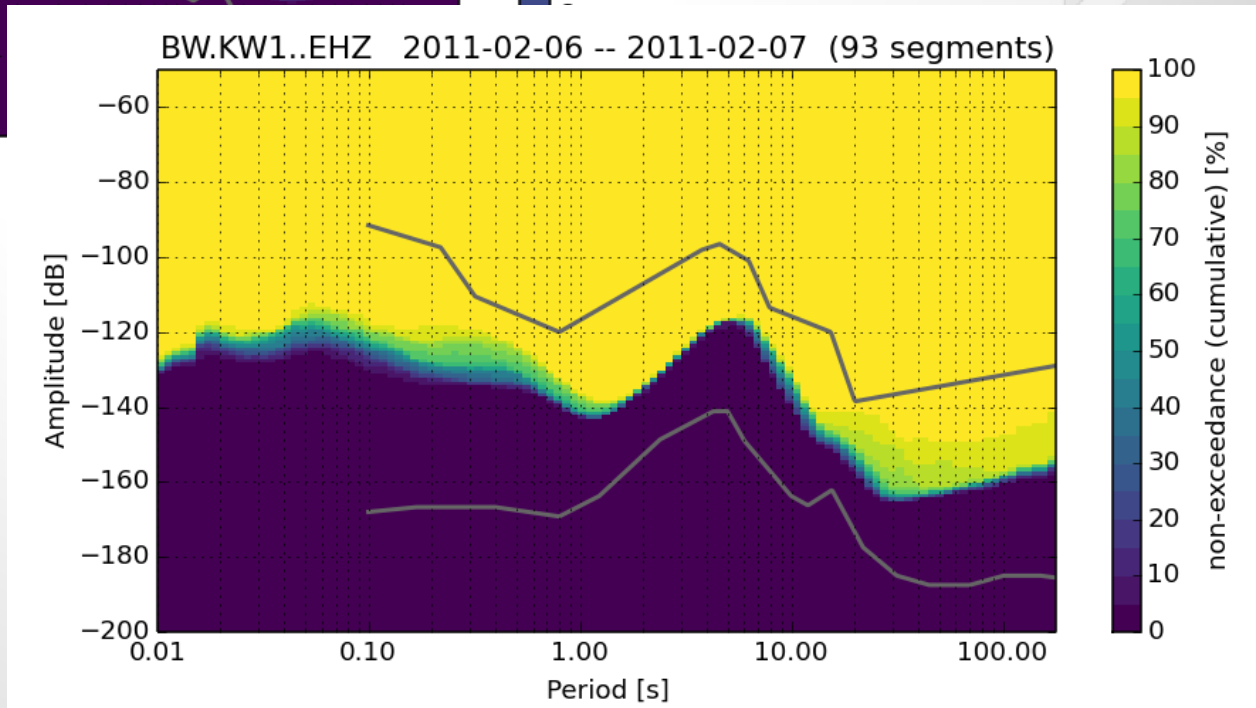


Recent Developments

- Python 2+3 support
- more stable filters
- new default colormaps
- response removal plot
- PPSD changes
  - improved algorithm (esp. affects long periods)
  - remove full response
  - cumulative / “non-exceedance” plot

Recent Developments





Recent Developments

- Python 2+3 support
- more stable filters #1028
- new default colormaps #915
- response removal plot #1116
- PPSD changes #1108
- Lanczos interpolation/resampling #1101
- new input/output plugins
- ...

Recent Developments

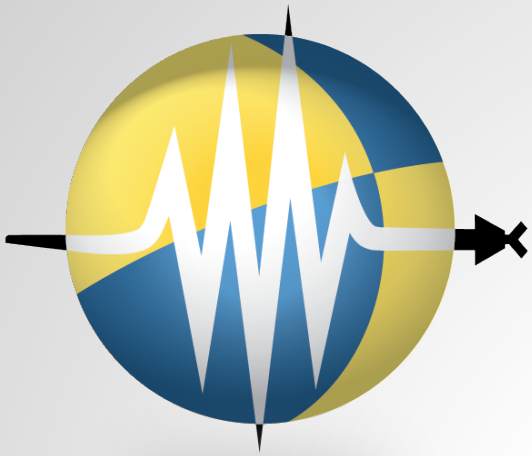
- Python 2+3 support
- more stable filters #1028
- new default colormaps #915

CHANGELOG.txt

```
15    * Support for additional event data formats:
16      - CMTSOLUTION files used by many waveform solvers.
17      - ESRI shapefile write support, useful in GIS applications (see #1066)
18  - obspy.core:
19    * New method for generating sliding windows from Stream/Trace windows.
20      (see #860)
21    * Stream/Trace.slice() now has the optional `nearest_sample` argument from
22      Stream/Trace.trim().
23    * Trace.remove_response() now has `plot` option to show/output a plot of all
24      individual steps of instrument response removal in frequency domain
25      (see #1116).
26  - obspy.clients.neries:
27    * Removed the dedicated client. Data can still be accessed by using the FDSN
28      client.
```

Future Plans

- more convenient, canonical handling for frequency domain operations
  → “FrequencyDomainTrace”
- better interconnection of waveforms / station /event data
  → data set object?
- more fine grained control in PPSD
  → PPSD stack by
    - start/endtime
    - frequency range
    - time of day
- ...



ObsPy

A Python Framework for Seismology

Thanks for your attention!

www.obspy.org

slides online at https://github.com/obspy/docs/