ObsPy: A Python Toolbox for Seismology/Seismological Observatories

Interactive Application and Rapid Program Development

Lion Krischer[a], Moritz Beyreuther[a], Robert Barsch[a], Tobias Megies[a], Yannik Behr[b], Joachim Wassermann[a]
[a] LMU Munich [b] Victoria University of Wellington, New Zealand
Department of Earth and Environmental Sciences (Geophysics), Ludwig-Maximilians-University Munich, Germany
Contact: *krischer@geophysik.uni-muenchen.de*, **http://www.obspy.org**

Abstract

Python enables the user to combine the advantages of a fully grown programming language with the flexibility of an interactive scripting language. Its extensive standard library and many freely available high quality scientific modules enable rapid development.
ObsPy provides the ability to read and write many common waveform file formats as well as seismological signal processing routines.
Furthermore **NumPy** (numpy.scipy.org) and **SciPy** (scipy.org) offer a wide variety of numerical multidimensional array programming methods. Together with **IPython** (ipython.scipy.org) and **Matplotlib** (matplotlib.sourceforge.net) ObsPy offers a powerful, text-based and easy to learn interactive environment consisting of free software packages.
ObsPy has a **modular architecture** which aims at minimizing the dependencies and is available under the **GPL/LGPLv3** licences.

Reading and Writing

ObsPy provides unified access to read and write most commonly used waveform formats such as **MiniSEED, GSE2, SAC, SEISAN and the Seismic Handler formats Q and ASCII**. All data is stored in a Stream object consisting of multiple Traces of contiguous time series. Every Trace keeps track of its own meta information and the data is available as **numpy.ndarrays**.

```
>>> from obspy.core import read
>>> stream = read('BW.FURT..EHZ')
>>> print stream
 1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-02-06T04:53:00.000000Z - 2010-02-06T05:03:00.000000Z | 200.0 Hz, 120001 samples
```

The Stream and Trace objects have many built-in methods, like filtering and plotting:

```
>>> stream.plot()
```

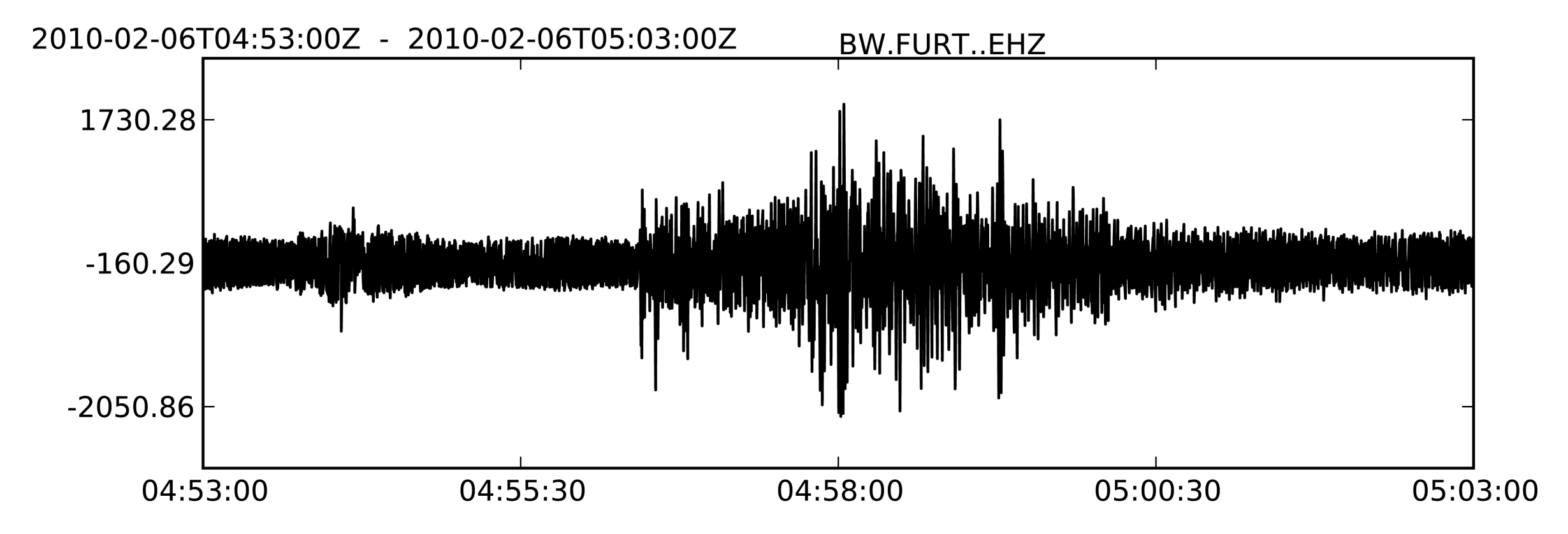


Figure 1: Graphical representation of the Stream object. Created with obspy.imaging.

Data can be exported with the write method.

```
>>> stream.write('BW.FURT..EHZ', format='GSE2')
```

Data Retrieval

ObsPy also has support to retrieve data from ArcLink (www.webdc.eu), Fissures (www.iris.edu/dhi) and SeisHub (www.seishub.org)[2]:

```
from obspy.arclink import Client
client = Client('webdc.eu')
stream = client.getWaveform('CZ', 'PVCC', '', 'BH*', start, end, getPAZ=True, getCoordinates=True)
```

SEED/XML-SEED Support

XML-SEED [4] is a XML representation of the Dataless SEED format [1]. ObsPy is able to convert SEED

```
0500097ANMO +34.946200-106.456700...
```

to XML-SEED and back and furthermore it can extract RESP files from either format.

```
<station_identifier blockette="050">
    <station_call_letters>ANMO</station_call_letters>
    <latitude>+34.946200</latitude>
    <longitude>-106.456700</longitude>
```

obspy.xseed is, to the authors' knowledge, the only freely available and working implementation of XML-SEED.

obspy.signal

Various processing routines, for example:

Instrument Correction

This example shows the correction of a STS2 to a 1 Hz seismometer using **obspy.signal**.
The data is read with **obspy.core**, then the mean value is subtracted.

```
trace = read("http://examples.obspy.org/
   RJOB20090824.ehz")[0]
trace.data = trace.data - trace.data.mean()
```

Now the data is corrected with the simulate() method of the Trace object which calls some functions in obspy.signal. *sts2* and *onehzinst* are Python dictionaries containing information about the instrument responses.

```
trace.simulate(paz_remove=sts2, paz_simulate=
   onehzinst)
```

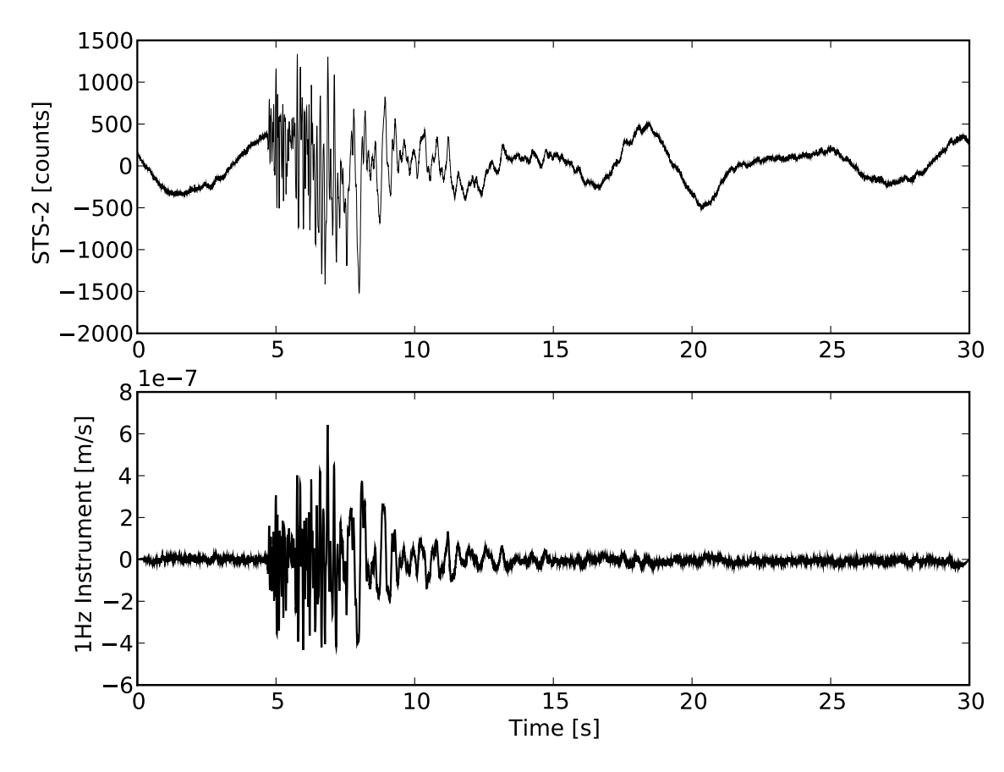


Figure 2: Correction of a STS2 to a 1Hz instrument.

Beamforming/FK Analysis

One of ObsPy's newer developments is the inclusion of FK Analysis. It works on ObsPy's standard Stream and Trace objects.

```
from obspy.signal.array_analysis import sonic
```

The Trace objects can store response information and coordinates for easier handling. After the data is corrected as explained above and setting all the necessary arguments in a dictionary, the analysis can be performed with one single call.

```
out = sonic(stream, **arguments)
```

The output can then be plotted with Matplotlib or further processing can be applied.

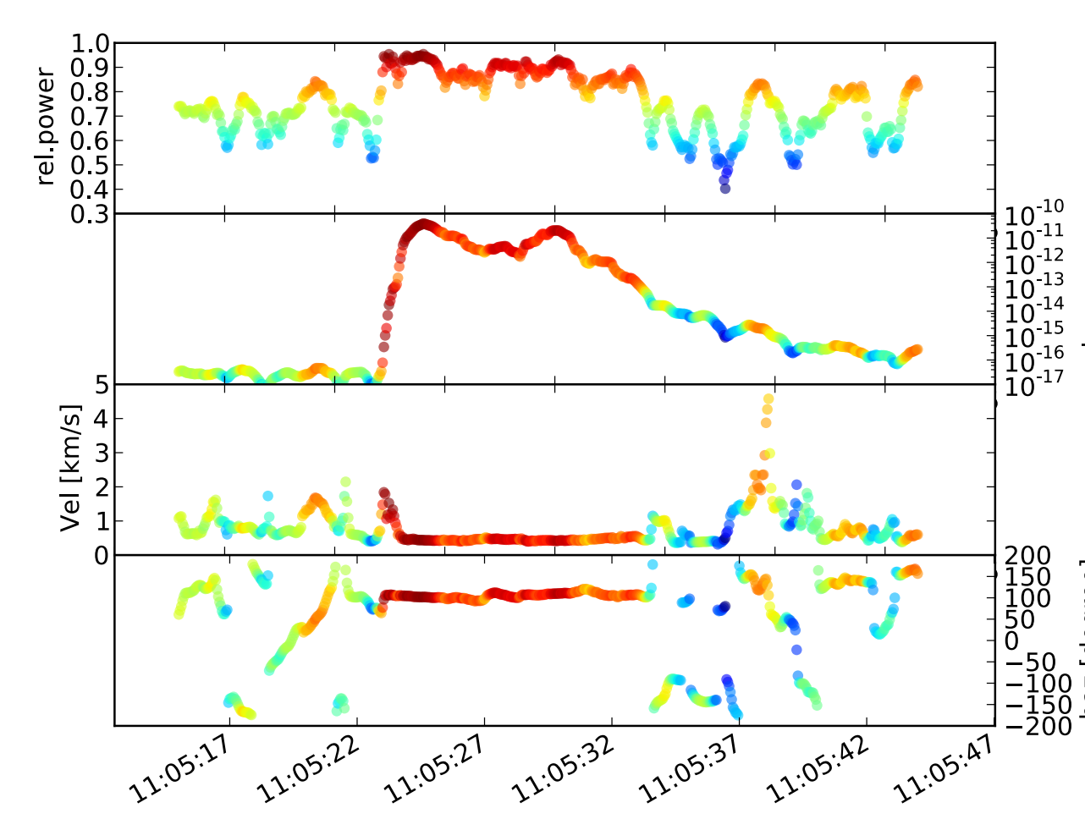


Figure 3: Output of a FK Analysis from blasting the AGFA skyscraper in Munich. Colors correspond to relative power.

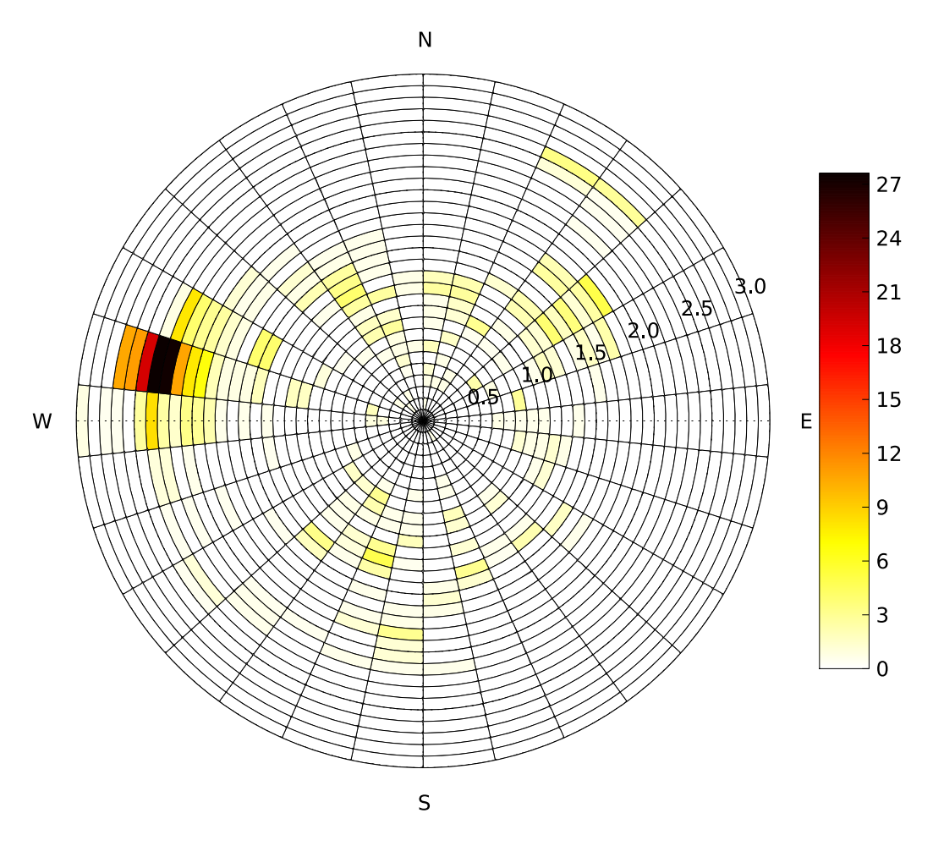
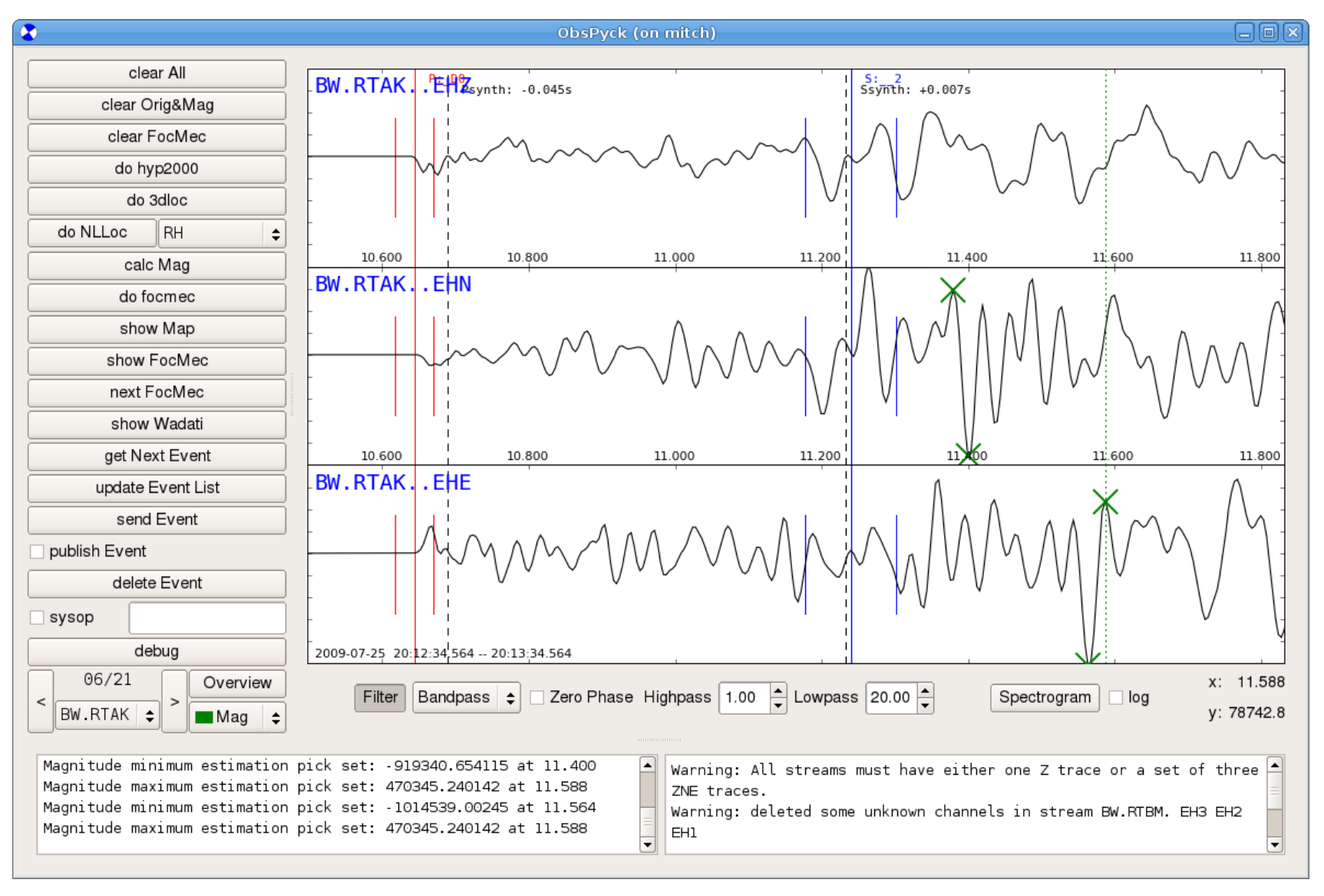


Figure 4: Polar plot of the same FK Analysis.

Applications Using ObsPy

Combining Python and ObsPy facilitates the development of applications ranging from little helper scripts to full blown platform-independent GUI applications with complex workflows.
This section highlights two GUI programs both based on Qt (qt.nokia.com) for the graphical representation. Both were developed in a matter of a few month and make use of ObsPy for the underlying internal data storage which in turn enables them to utilize ObsPy's routines for many commonly used tasks, e.g. filtering and instrument simulation and they instantly gain the ability to work with most commonly used waveform dataformats.
Other advanced third party Python modules that are easily integrated can take care of most processing and plotting needs so the programmer can focus on creating unique new features.
An up to date list of programs making use of ObsPy is available at www.obspy.org. A notable example is the newly developed version of **SeismicHandler** (www.seismic-handler.org) which is also represented at this meeting.



```
Magnitude minimum estimation pick set: -919340.654115 at 11.400
Magnitude maximum estimation pick set: 470345.240142 at 11.588
Magnitude minimum estimation pick set: -1014539.00245 at 11.564
Magnitude maximum estimation pick set: 470345.240142 at 11.588
```

```
Warning: All streams must have either one Z trace or a set of three
ZNE traces.
Warning: deleted some unknown channels in stream BW.RTBM. EH3 EH2
EH1
```

Figure 5: ObsPyck

ObsPyck is able to perform daily routine analysis at an observatory like phase picking and locating events. All data is read and written back to one central SeisHub database server and it just recently was extended to fetch data from ArcLink and Fissures servers to integrate external data into the localization process.

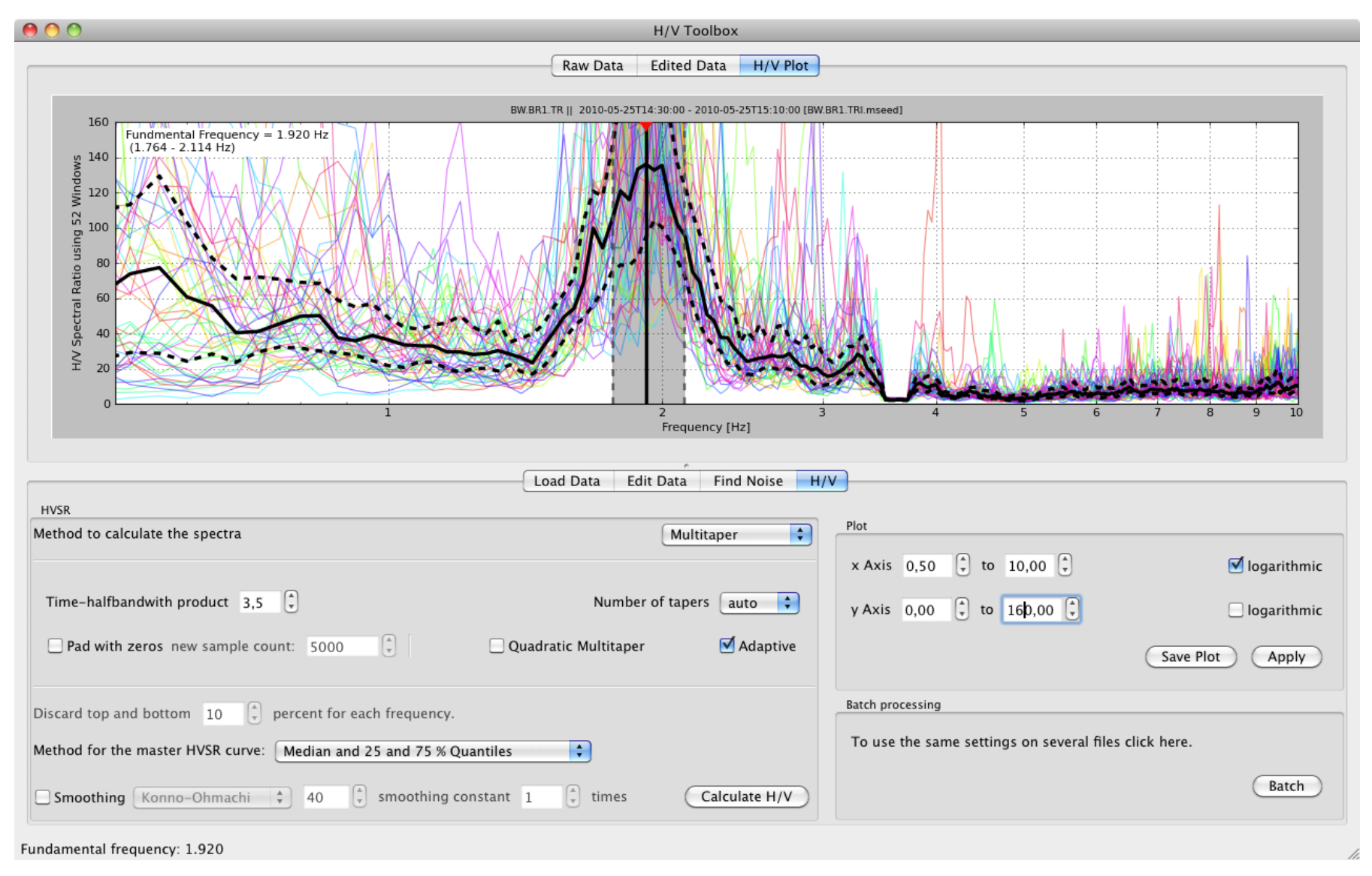


Figure 6: H/V Toolbox

The **H/V Toolbox** handles the whole workflow for calculating horizontal to vertical spectral ratios (HVSR): From reading and preprocessing the data to the automated selection of appropriate time windows with little seismic activity to finally calculating the HVSR in an easy to use interface while giving visual feedback, step by step.

Further Information: http://www.obspy.org

The homepage of the project **http://www.obspy.org** contains extensive **documentation**, tutorials, installation instructions for various platforms, and many more examples.
Contact the developers for support or feedback, to report bugs or to request a new feature. Contributions are more than welcome, see the developer links on the homepage if you are interested.

Further functionality and information:

- **obspy.signal:** Filters, triggers, instrument correction, rotation, array analysis, beamforming.
- **obspy.imaging:** Imaging spectrograms, beachballs and waveforms.
- **Test-driven development (TDD):** TDD and automated unit tests are used throughout the ObsPy code base to ensure a high quality. Automated build bots test the most current ObsPy build on a daily basis on multiple platforms.

Literature

[1] Ahern, Timothy K. ; Dost, Bernard: *SEED Reference Manual - standard for the exchange of earthquake data - SEED format version 2.4*. Incorporated Research Institutions for Seismology DMC, January 2009

[2] Barsch, Robert: *Web-based technology for storage and processing of multi-component data in seismology*, LMU München, Diss., 2009

[3] Beyreuther, Moritz ; Barsch, Robert ; Krischer, Lion ; Megies, Tobias ; Behr, Yannik ; Wassermann, Joachim: ObsPy: A Python Toolbox for Seismology. In: *Seismological Research Letters* 81 (2010), Nr. 3, S. 530

[4] Tsuboi, S. ; Tromp, J. ; Komatitsch, D.: An XML-SEED Format for the Exchange of Synthetic Seismograms. In: *EOS Transactions of American Geophysical Union. SF31B-03*, 2004