summary

ObsPy in a Nutshell

Python combines the power of a **full-blown programming** language with the flexibility and fast code development of an **interactive scripting language**. Its extensive standard library and large variety of freely available high quality scientific modules cover most needs in **developing scientific processing workflows**.

ObsPy extends Python's capabilities to fit the specific needs that arise when working with seismological data. It a) provides **read and write support for all of the most important waveform, station and event metadata formats** b) enables **direct access to all important data centers, web services and databases** to easily retrieve waveform data and station/event metadata and c) comes with a continuously growing **powerful signal processing toolbox** that covers all everyday tasks in seismological analysis.

In combination with mature and free Python packages like NumPy, SciPy, Matplotlib, IPython/Jupyter, Pandas and PyQt, **ObsPy makes it possible to develop complete seismological processing workflows**, ranging from reading locally stored data or requesting data from one or more different data centers via signal analysis and data processing to visualization in GUI and web applications, output of modified/derived data and the creation of publication-quality figures. All functionality is **extensively documented** and the online **ObsPy Tutorial and Gallery** give a good impression of the wide range of possible use cases. ObsPy is tested and **running on Linux, MacOS X and Windows** and comes with installation routines for these systems. ObsPy is developed in a test-driven approach and is available under the **LGPLv3 open source licence**.

Users are welcome to request help, report bugs, propose enhancements or contribute code via either the user mailing list or the **project page on GitHub**.

obspy as a bridge

The Scientific Python Ecosystem

Over the last decade, Python grew a vast and rich ecosystem of scientific third party libraries. **ObsPy serves as a bridge** and enables its users to effortlessly tie into this system giving access to large amounts of tools designed to process and analyze data. This box quickly introduces the core packages needed for scientific analysis.

Jupyter: Web Notebooks and Interactive Console

- Powerful interactive shells (terminal and Qt-based)
- A browser-based notebook with support for code, text, mathematical expressions, inline plots and other rich media. Recently featured in Nature.
- Easy to use, high performance tools for parallel computing

NumPy: Modern Array Processing

NumPy is the fundamental package needed for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

SciPy: Fundamental Library for Scientific Computing

SciPy is open-source software for mathematics, science, and engineering. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and

Matplotlib: Comprehensive, high-quality Plotting

2D and 3D plotting library for Python that produces high quality figures that can be used in various hardcopy and interactive environments.

SymPy: Symbolic Mathematics

SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible and easily extensible. SymPy is written entirely in Python and does not require any external libraries.

pandas: Data Structure & Analysis

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

scikit-learn: Machine Learning in Python

Simple and efficient tools for data mining and data analysis. Includes classification, regression, clustering, dimensionality reduction, model selection, preprocessing and feature extraction, and a lot more.

ObsPy

A Python Framework for Seismology

Try it now: seismo-live.org

Sustainability, New Features, and Applications

Tobias Megies[1], Lion Krischer[1,2], Elliott Sales de Andrade[3], Robert Barsch[4], Jonathan MacCarthy[5] and the Obspy Development Team

[1]Ludwig Maximilian University of Munich
[2]ETH Zurich
[3]University of Toronto
[4]EGU Executive Office
[5]Los Alamos National Lab

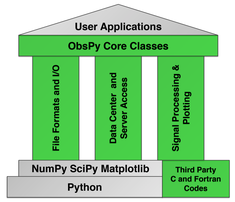
Contact: devs@obspy.org @obspy **http://www.obspy.org**

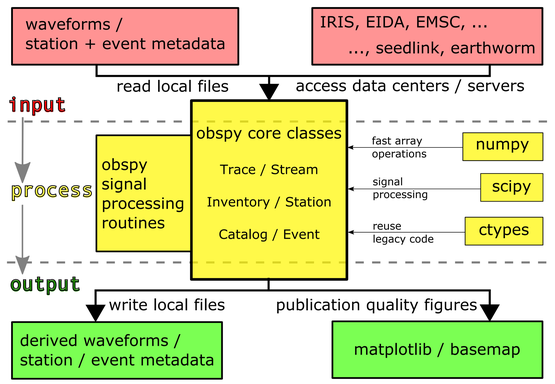
What Can I Do With ObsPy?

Easy to use helper functions to access local files and online data centers give **quick access to all data necessary for seismological data analysis**.

All acquired information is exposed to the user in **ObsPy's core classes that handle waveform data, station and event metadata in a unified, consistent fashion, regardless of the data source**. This makes it easy to combine data from different sources in unified workflows, both interactively and automated.

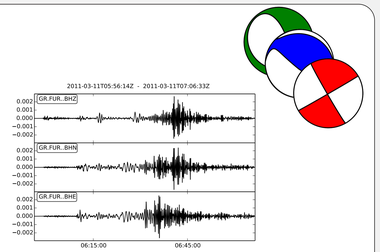
ObsPy's core classes have many convenience routines for signal processing directly attached for **quick, reproducible and well tested execution of common processing tasks**.

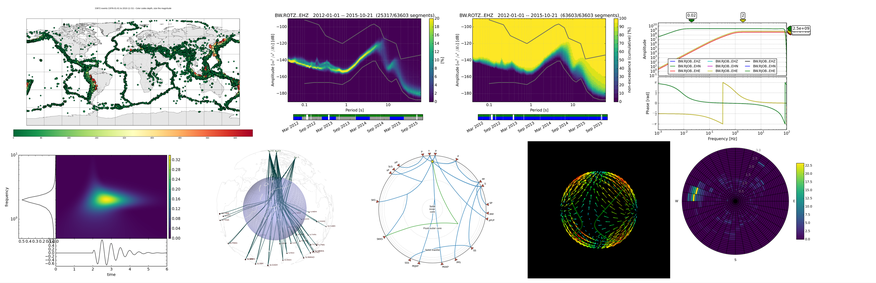




Basic Example

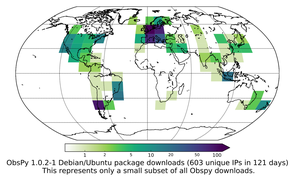
```
from obspy import UTCDateTime
from obspy.fdsn import Client

# connect to an FDSN webservice
client = Client("http://erde.geophysik.uni-muenchen.de:8080")

# use origin time of devastating Japan earthquake
start = UTCDateTime("2011-03-11 05:46:23") + 10 * 60
end = start + 70 * 60

# download waveform and station metadata of station FUR
stream = client.get_waveforms(
    network="GR", station="FUR", location="", channel="BH*",
    starttime=start, endtime=end, attach_response=True)

# do basic signal processing and plot the data! ---->
stream.remove_response()
stream.filter("bandpass", freqmin=0.01, freqmax=1)
stream.plot()
```





impact, sustainability, and applications

Is It Used?



Seven years after the beginnings of the project, ObsPy is used by seismologists all around the world. With **more than 600 downloads for Debian/Ubuntu Linux** alone and **6000 conda downloads for the latest version alone**, we estimate — including Mac, Windows and other Linux/Unix users — an **active user base of several thousand people**.

Since ObsPy's start by a **core developer team of 3 – 5 people** at LMU Munich, ObsPy has evolved into a community effort with **contributions to the code base by 57 individuals**.

The busy user mailing list currently has around **450 subscribers** and serves as a place for discussions and asking for help from more experienced users.

The **impact of ObsPy and the appreciation within the seismological community** finds expression in the **increasing number of scientific citations, which stands at over 212 as of December 2016**. Applications are numerous and include event (re)locations, ambient seismic noise analysis, seismic tomography, rotational seismology studies, time-dependent seismology, and more. Additionally, a large variety of programs built on top of ObsPy are appearing:

EQcorrscan — Correlation for earthquake detection in Python · MSNoise · instaseis · scisola · pyasdf

recent developments

What's New?

- Support all active Python versions (2.7 - 3.5)
- New formats: Guralp Compressed Format (GCF), Reftek 130 (rt130), Nordic format (s-file), GSE2.0 bulletin, NNSA KB Core
- Quality control module (on its way to being standardized)
- Much improved io.sac module (including a SACTrace class for full header control)
- Iterative reading of arbitrarily large SEG-Y and SU files
- Mass downloader for FDSN web services and SDS file system client
- Tested support for essentially all common Linux distributions
- Several hundred bugfixes, stability improvements and other small enhancements

obspy's online resources

Where Can I Learn More?

- Source Code, Bug Tracker, and Wiki
- Extensive documentation and example gallery
- Tutorial on how to get started
- **[obspy-users]** mailing list
- Installation instructions for various platforms
- Use cases and user application showcases

http://obspy.org

@obspy
https://github.com/obspy/obspy

thank you

References and Contributors

Beyreuther, M., R. Barsch, L. Krischer, T. Megies, Y. Behr and J. Wassermann (2010)
ObsPy: A Python Toolbox for Seismology
Seismological Research Letters, 81(3):530-533, doi:10.1785/gssrl.81.3.530

Megies, T., M. Beyreuther, R. Barsch, L. Krischer and J. Wassermann (2011)
ObsPy – What can it do for data centers and observatories?
Annals Of Geophysics, 54(1), 47-58, doi:10.4401/ag-4838

Krischer, L., T. Megies, R. Barsch, M. Beyreuther, T. Lecocq, C. Caudron and J. Wassermann (2015)
ObsPy: A Bridge for Seismology into the Scientific Python Ecosystem
Computational Science & Discovery, 8, doi:10.1088/1749-4699/8/1/014003

We would like to thank our contributors, whose efforts make this software what it is. These people have helped by writing code and documentation, and by testing. They have created and maintained this product, its associated libraries and applications, our build tools and our web sites.

- Ammon, Charles J.
- Antunes, Emanuel
- Arnarsson, Ólafur St.
- Bank, Markus
- Barsch, Robert
- Behr, Yannik
- Bernardi, Fabrizio
- Bernauer, Felix
- Beyreuther, Moritz
- Bonaime, Sébastien
- Carothers, Lloyd
- Chamberlain, Calum
- Danecek, Peter
- van Driel, Martin
- Egdorf, Sven
- Engels, Fabian
- Ermert, Laura
- Eulenfeld, Tom
- Fabbri, Tommaso
- Falco, Nicholas
- Fee, Jeremy
- Grellier, Clément
- Grunberg, Marc
- Hammer, Conny
- Heimann, Sebastian
- Heiniger, Lukas
- Hope, Gaute
- Igel, Heiner
- Inza, Adolfo
- Isken, Martius
- Ketchum, David
- Koymans, Mathijs
- Kremers, Simon
- Kress, Victor
- Krieger, Lars
- Krischer, Lion
- Käufl, Paul
- Köhler, Andreas
- Lecocq, Thomas
- Leeman, John
- Lesage, Philippe
- Lomax, Anthony
- Lopes, Rui L.
- MacCarthy, Jonathan
- Maggi, Alessia
- Martin, Henri
- Megies, Tobias
- Meschede, Matthias
- Michelini, Alberto
- Miller, Nathaniel C.
- Morgenstern, Bernhard
- Nof, Ran Novitsky
- Panning, Mark P.
- Rapagnani, Giovanni
- Reyes, Celso
- Ringler, Adam
- Rothenhäusler, Nicolas
- Russo, Emiliano
- Sales de Andrade, Elliott
- Satriano, Claudio
- Saul, Joachim
- Scheingraber, Chris
- Sippl, Christian
- Snoke, Arthur
- Stange, Stefan
- Sullivan, Benjamin
- Trabant, Chad
- Ulecla, Leonardo
- Walker, Andrew
- Walther, Marcus
- Wassermann, Joachim
- Williams, Mark C.
- Winkelman, Andrew
- Zad, Seyed Kasra Hosseini