

ObsPy in a Nutshell

Python combines the power of a **fullblown 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, sympy, pandas and scikitlearn, **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 publicationquality 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, Mac OS 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.

STEADY EVOLUTION

What's New?

- New Formats: Guralp GCF, Reftek 130, AH (write support added), NNSK KB Core, Nordic s-file, SCARDEC catalog files, GSE2.0 bulletin, SC3ML, IASPEI ISF ISM 1.0 Bulletin, SEED/RESP (now with inventory integration), Arclink XML, Receiver Gather 1.6, Seismic Handler EVT files
- Full normalization in correlations for template matching
- Client for the Nominal Response Library (NRL)
- Support for two FDSNWS routing services (IRIS federator + EIDAWS routing)
- New QC module, ... and much more!



Tobias Megies¹, Lion Krischer², Derrick JA Chambers³, Tom Eulenfeld⁴, Calum John Chamberlain⁵, Robert Barsch⁶ & the ObsPy Development Team

¹Ludwig Maximilians University of Munich, Germany; ²ETH Zürich, Switzerland; ³Centers for Disease Control and Prevention, WA, USA; ⁴Friedrich Schiller University of Jena, Germany;

⁵Victoria University of Wellington, New Zealand; ⁶EGU Office Munich, Germany

@ devs@obspy.org

obspy

http://obspy.org

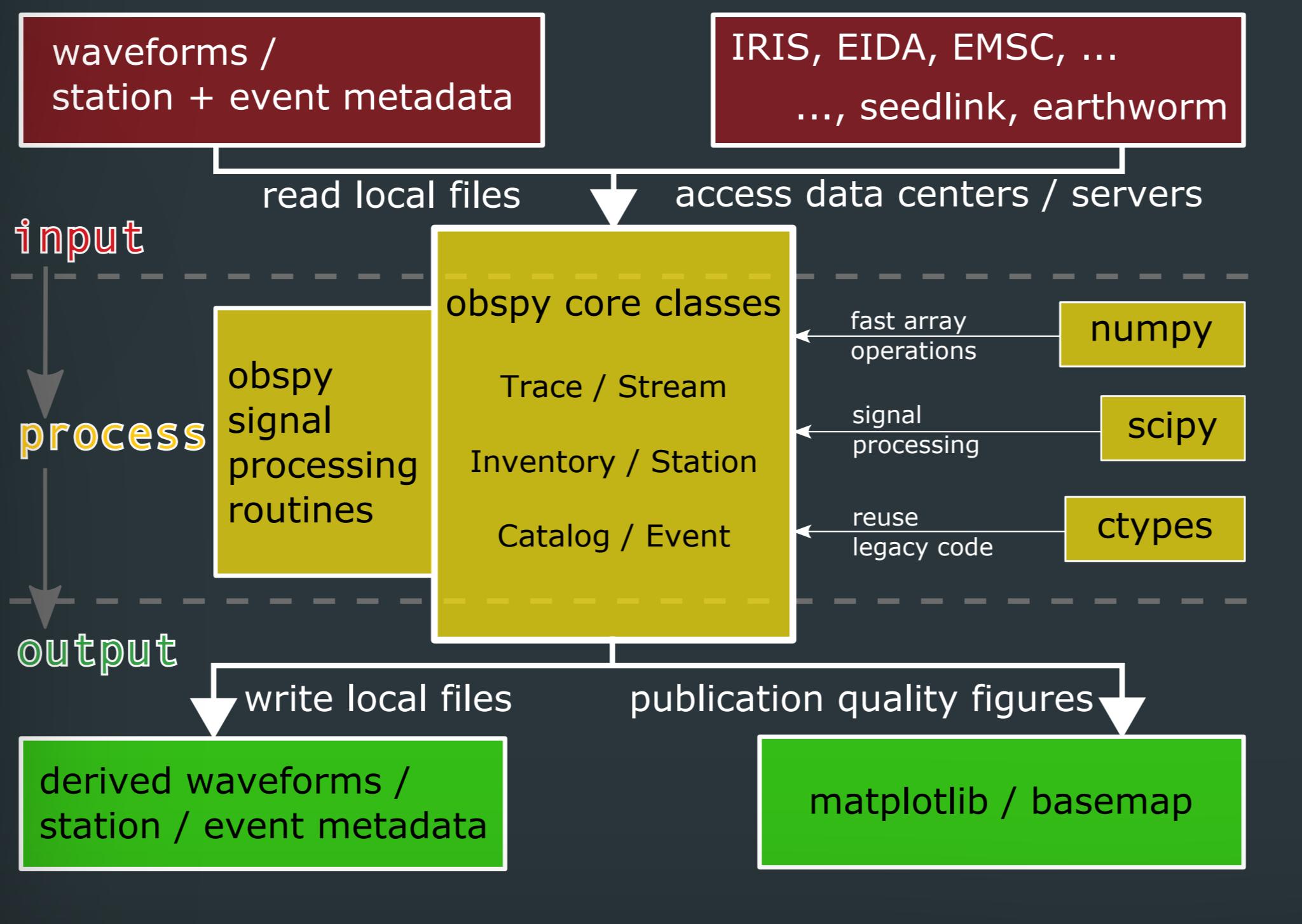
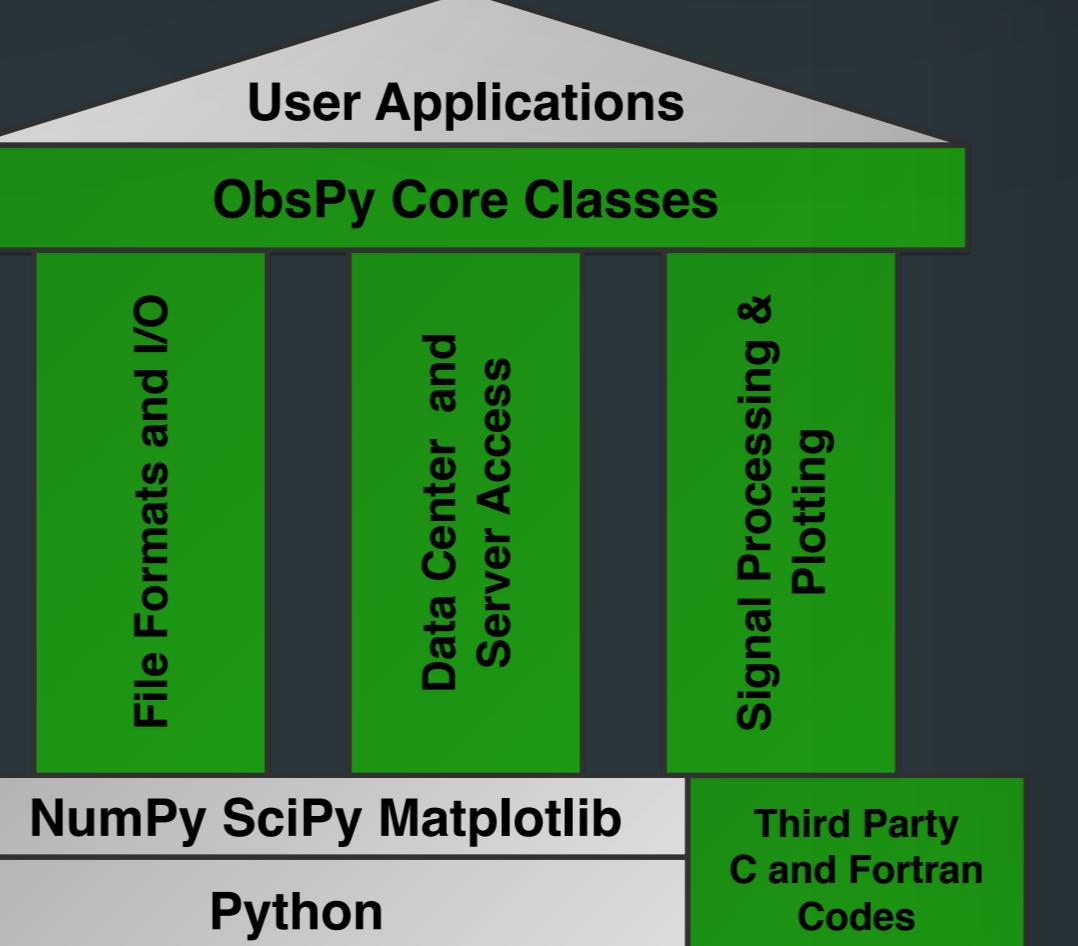
obspy

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 work flows, 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**.



SUPPORTED WAVEFORM FILE FORMATS

AH, ASCII, CSS, Guralp GCF, GSE2, KINEMETRICS, MiniSEED, NIED-KNET, PDAS, RG16, REFTEK 130, SAC, SEISAN, SEG-2, SEG-Y, SH, WAV, WIN, Y

SUPPORTED STATION FILE FORMATS

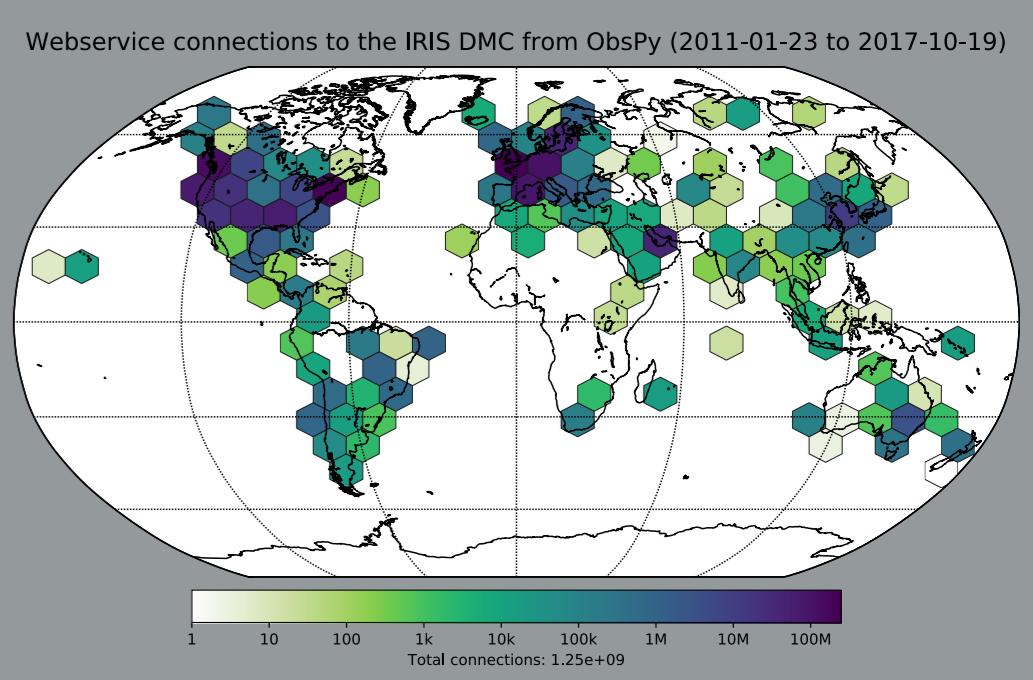
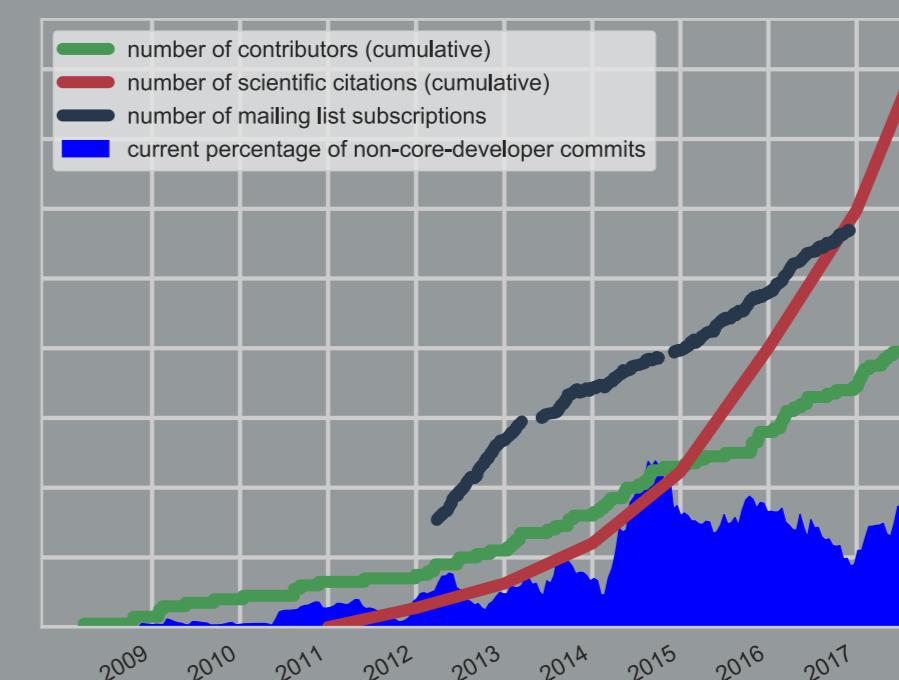
ArcLink, CSS, KML, SACPZ, Seiscomp, Seismic Handler EVT, Shapefiles, StationTXT, StationXML, XSEED, SEED, RESP

SUPPORTED EVENT FILE FORMATS

CMTSOLUTION, CNV, GSE2, IASPEI, JSON, KML, NDK, NIED-F-NETMT, NLLOC, Nordic, PDE, Seiscomp, SCARDEC, Shapefiles, QuakeML, Zmap

Since ObsPy's start by a core developer team of 3 – 5 people at LMU Munich, ObsPy has evolved into a com munity effort with direct **contributions to the code base by over 70 individuals**.

The busy user mailing list currently has around **580 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 about 512 as of Dezember 2018. Applications are numerous and include event (re)locations, ambient seismic noise analysis, seismic tomography, rotational seismology studies, timedeprived seismology, and more. Additionally, a large variety of programs built on top of ObsPy are appearing (full list including URLs at <http://obspy.org>):

scisola

SALVUS
IMAGE EVERYTHING

Instaseis
EQcorna

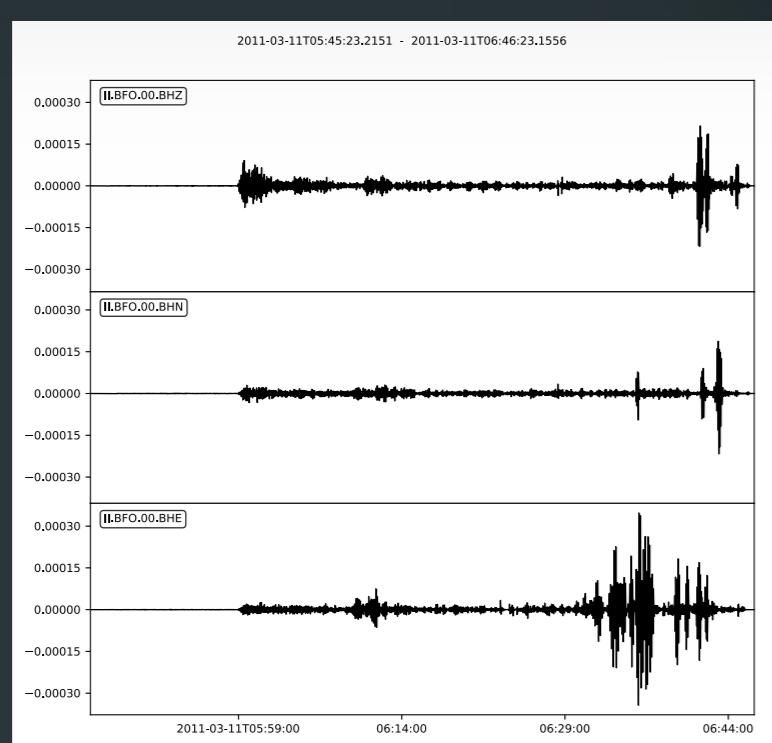
pyasdf
REDPy

Basic Example

```
import obspy
# The 'read()' method can read data from all kinds of sources (files, web,
# archives, ...) and auto-detects the data format and parses it to a common
# structure.
st = obspy.read('data.mseed')

# The 'read_inventory()' method does the same for station meta information.
inv = obspy.read_inventory('station.xml')
# Remove a linear trend.
st.detrend('linear')

# Remove a low pass filter.
st.taper(max_percentage=0.05, type='hann')
# Remove response (inventory-inv, output=VEL,
# pre_filt=(1.0 / 100.0, 1.0 / 50.0, 10.0, 20.0))
# Bandpass filter - note that this is a 20th order filter which is still
# stable.
st.filter('bandpass', freqmin=0.1, freqmax=1.0,
          corners=10, zerophase=True)
# Re-sample to 100 Hz using the most ideal sinc-based reconstruction
st.interpolate(method='lanczos', sampling_rate=100.0, a=12)
# Finally plot it.
st.plot()
```

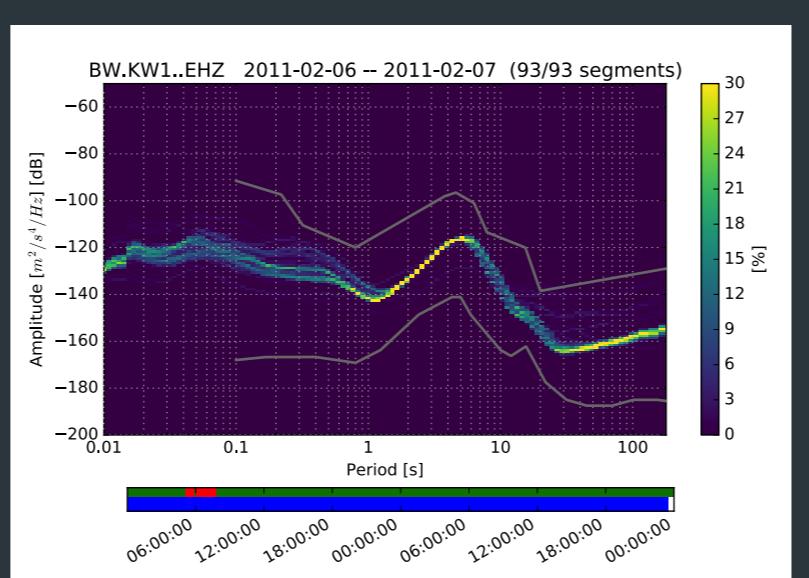


Probabilistic Power Spectral Densities

```
import obspy
from obspy.signal import PSD
from obspy.io.xseed import Parser
# Read data directly from a web address.
st = obspy.read("https://examples.obspy.org/BW.KW1..EHZ.D.2011.037")
# Init PSD object and add meta information.
psd = PSD(st[0].stats, metadata=parser)
psd.add(st)

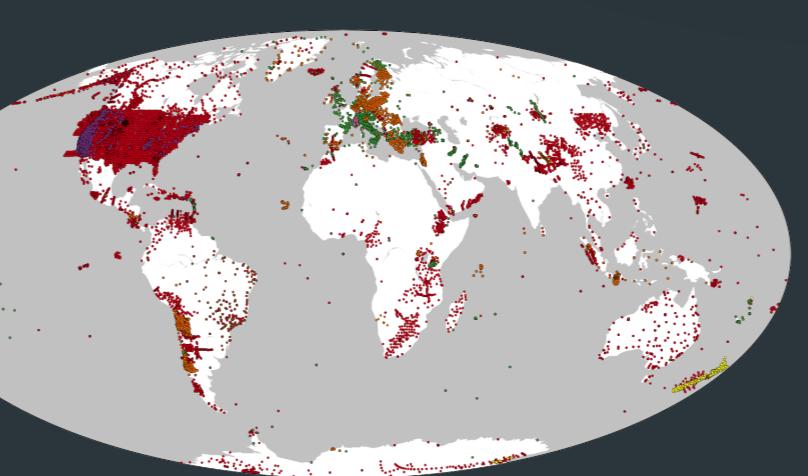
# Add more data.
st = obspy.read("https://examples.obspy.org/BW.KW1..EHZ.D.2011.038")
psd.add(st)

# Plot it.
psd.plot()
```



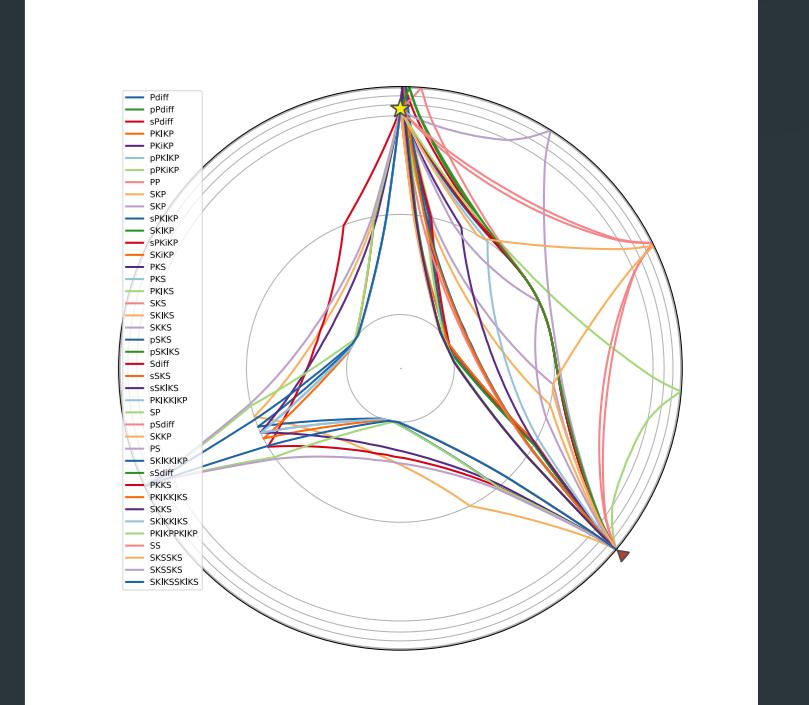
Mass Downloader Example

```
import obspy
from obspy.clients.fdsn.mass_downloader import CircularDomain, \
    Restrictions, MassDownloader
origin_time = obspy.UTCDateTime(2011, 3, 11, 5, 47, 32)
# Circular domain around the epicenter.
domain = CircularDomain(latitude=37.82, longitude=-143.04,
                         minradius=70.0, maxradius=90.0)
restrictions = Restrictions()
# Set time window.
starttime = origin_time - 5 * 60, endtime=origin_time + 3600,
# Reject channels with gaps True, minimum length 0.95.
# No two stations should be closer than 10 km to each other.
minimum_interstation_distance_m=1000,
# Set priority.
channel_priorities = ["HH1[ZNE]", "BH[ZNE]"],
location_priorities = ["*", "*"]
mdl = MassDownloader() # Download from all FDSN web service providers.
# Downloaded to the '/waveforms/' and '/stations/' folders.
mdl.download(domain, restrictions, msid_storage="waveforms",
              station_storage="stations")
```



TauP Example

```
import matplotlib.pyplot as plt
from obspy.taup import TauPModel
# Currently available built-in models are: 1066a, 1066b, ak135, ak135f,
# p135f, no_mu, herring, asp01, jb, prem, pwk, sp6
# TauPModel(model="ak135")
# Get the ray paths. Alternatively one can also get just the arrival times or
# pierce points.
arrivals = mdl.get_ray_paths(source_depth_in_km=500, distance_in_degree=136)
arrivals.plot(show=False)
plt.savefig("taup_plot.pdf")
```



THANK YOU

Ackerley, Nick
Ammon, Charles J.
Antunes, Emanuel
Arnarsson, Olafur St.
Bank, Markus

Barsch, Robert
Behr, Yannik
Bernardi, Fabrizio
Bernaude, Felix
Bes de Berc, Maxime

Beyreuther, Moritz
Bonaime, Sébastien
Carothers, Lloyd
Chamberlain, Calum
Chambers, Derrick

Chen, Zhao
Clark, Adam
Engels, Fabian
Danecic, Peter
Dramsch, Jesper

de Vries, Conny
Eggerdt, Sven
Fee, Jeremy
Ermert, Laura
Grunberg, Marc

Einarsson, Olafur St.
Eulenfeld, Tom
Fabbri, Tommaso
Falco, Nicholas
Heiniger, Lukas
Hoepfner, Gaute
Igel, Heiner

Grellet, Clément
Ketchum, David
Koymans, Mathias
Kremers, Simon

Köhler, Andreas
Isken, Marius
Küller, Daniel
Käufel, Paul

Kress, Victor
Krieger, Lars
Kretschmar, Lion
Legorburu, Paride
Lesage, Philippe

Liu, Yulin
Lomax, Anthony
Leeman, John
Lopes, Rui L.
MacCarthy, Jonathan

Martin, Henri
Megies, Tobias
Meschede, Matthias
Michelini, Alberto
Miller, Nathaniel C.

Morgenstern, Bernhard
Nof, Ran Novitsky
Rothenhäusler, Nicolas
Schneider, Simon
Sullivan, Benjamin
Sales de Andrade, Elliott
Sippl, Christian

Reyes, Celso
Ringler, Adam
Saul, Joachim
Stähler, Simon C.
Stange, Stefan
Uieda, Leonardo
Winkelmann, Andrew
Walther, Marcus

Snoke, Arthur
Tran, Luca
van Wijk, Jasper
Williams, Mark C.
Wassermann, Joachim
Winkelmann, Andrew
Zad, Seyed Kasra Hosseini