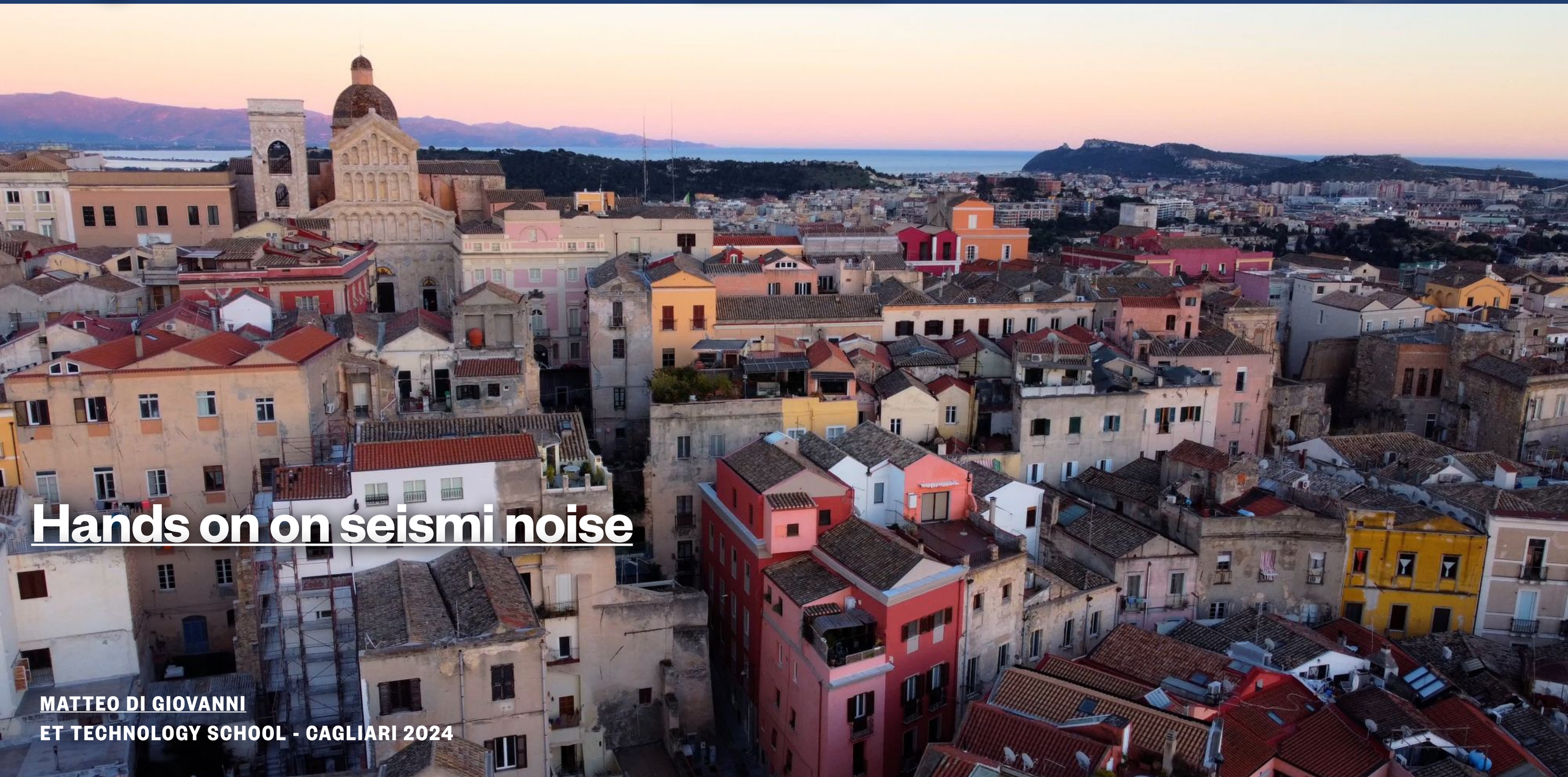
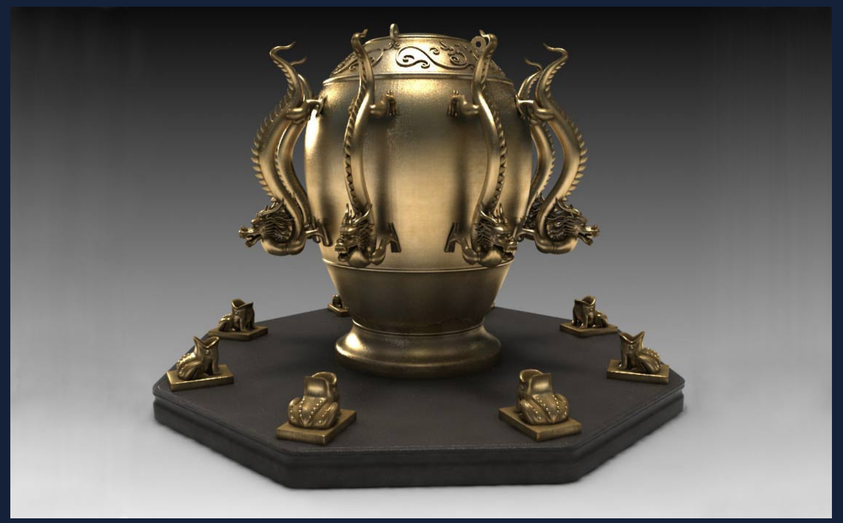


Hands on on seismi noise

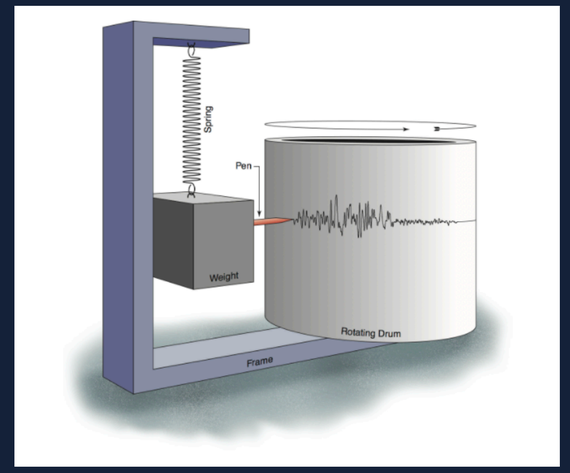
MATTEO DI GIOVANNI

ET TECHNOLOGY SCHOOL - CAGLIARI 2024

How do we measure ground motion?

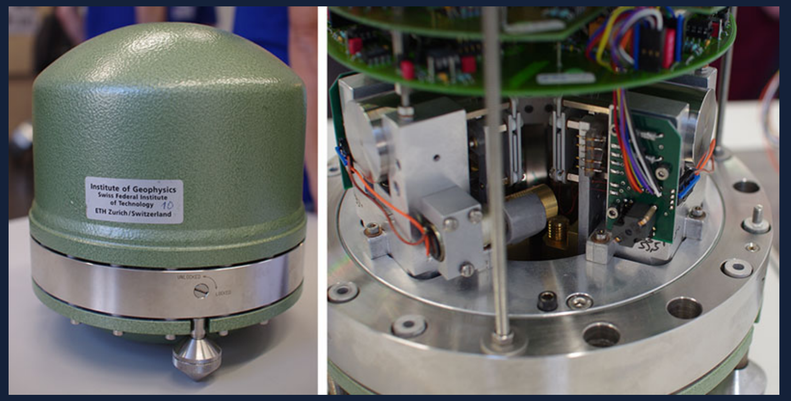


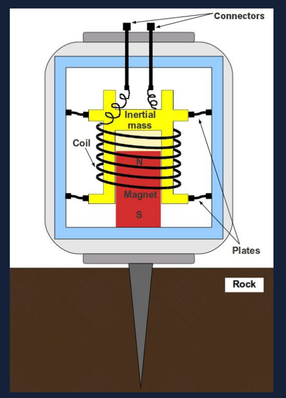
Chinese earthquake detector (II century B.C.)



Seismograph (seismometer + recording system - XIX century)

Modern seismometer (XX century)





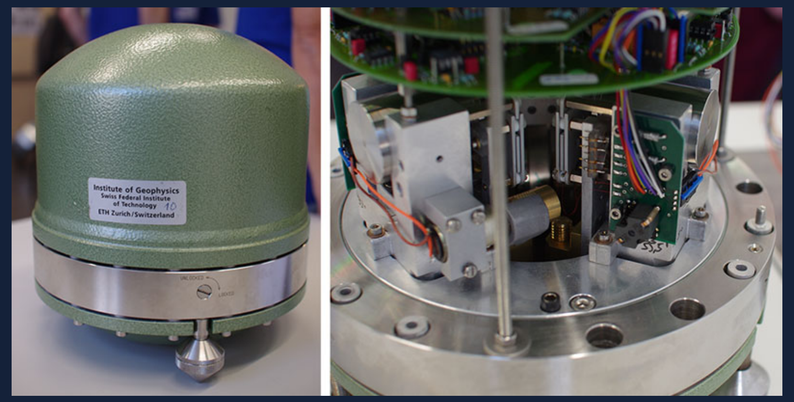
What is a seismometer

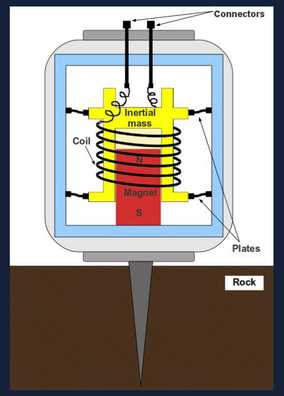
A seismometer is an extremely sensitive electromechanical device used to measure ground movement in a particular area in the order of nanometres. The movement recorded on a time axis called a seismogram.

Seismometers measure ground motion on three axis (N-S, E-W, Z).

Old seismographs used a stylus attached to a mass to measure ground motion. They needed a mass for each axis.

Modern seismometers use a force feedback system that measure the current needed by a magnetic mass as still as possible during ground tremors. They measure the motion on three axes using a single mass.





What is a seismometer

What is a seismometer

Seismometer

Acquisition system

Data storage (local or remote)

Format mSEED

Reading seismic data is very simple, but requires specific software libraries/modules.

Data repository

Data regarding site characterization activities for ET are stored on the ET Site data repository available at the following link: https://etsitedata.pi.infn.it/jupyter/hub

Login specifications:

- USER The ETMD username
- UID 2000 + ETMD ID number
- GID Common GID required for I/O operations to the GPFS volume

When the container is created, a bash start script creates the user and configures its home directory. The user's home directory is NOT persisted by default

Link	Location	Permissions	Description
work	/mnt/user-work/username	700	This is your work directory. Store your code and data here. You are the owner of this directory and its content is only visible to you.
sites-data	/mnt/sites-data	755	This is the location where the data are stored. All users have read-only access to the content of this directory.
shared	/mnt/shared	777	This is a shared directory. All users have read-write access to this directory. You should not put any sensitive data inside it.

Data repository

Enabled OpenSSH service inside the singleuser containers

Authentication via public RSA key to be stored under $HOME/work/.ssh/authorized_keys

```
$ mkdir $HOME/work/.ssh
$ echo "PUT YOUR KEY HERE WITH QUOTES" >> $HOME/work/.ssh/authorized_keys
```

Disabled username/password authentication

Server accessible via a dedicated port (20000 + UID) stored in the **$SSH_PORT** env

```
$ scp -P PORT /path/to/local/file USER@etsitedata.pi.infn.it:/home/USER/work/
```

Full documentation on the wiki: https:/wiki.et-gw.eu/SPB/ETSiteData#Remote_SSH

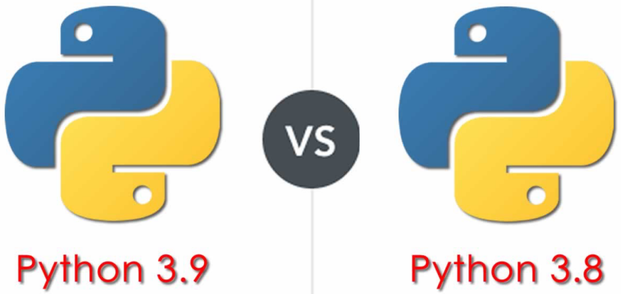
TIPS FOR INSTALLATION

When installing new versions of Python or new modules, my advice is not to mess with your native Python installation but to use Anaconda instead.

Anaconda is a virtual environment (venvs) package manager that lets you create venvs with any available Python verion.

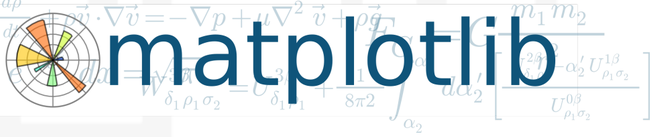
For downloads and tutorials: www.anaconda.com

TIPS FOR INSTALLATION



When you create a Python venv, my advice is to not use the latest Python version available (3.9) but to stick to older releases like 3.7 or 3.8

Python modules may not be up to date with the most recent version and can create conflicts.



TIPS FOR INSTALLATION

Install the ever-present Python modules:

- numpy
- scipy
- matplotlib

plus:

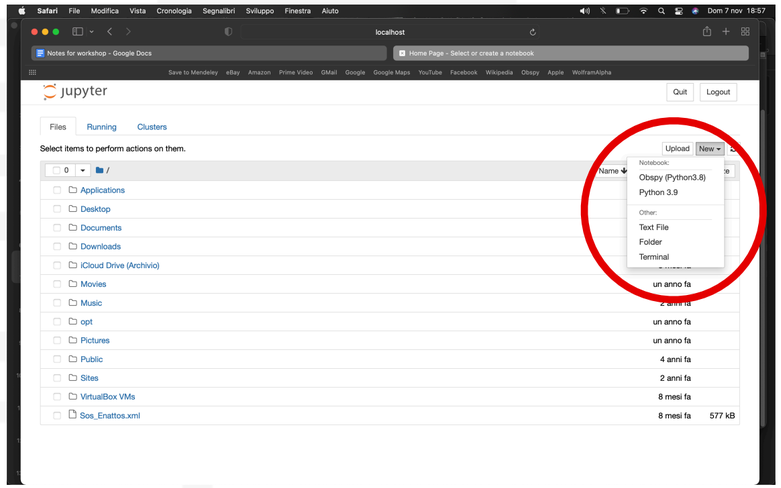
- obspy
- netCDF4
- xarray
- basemap
- motuclient

If you use Anaconda, it is enough to type `conda install module_name` in your venv.
Anaconda delivers 99% of available Python modules.

To simplify things, follow the installation instructions for ObsPy at github.com/obspy/obspy/wiki

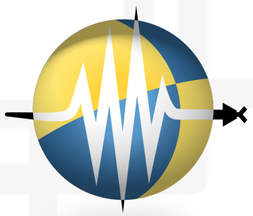
TIPS FOR INSTALLATION

Since some of the provided scripts are Python notebooks, be sure to have Jupyter installed on your computer (https://jupyter.org/install).



Using iPykernel you can add as many kernels as you want to your Jupyter notebooks, including Anaconda kernels.

```
python -m ipykernel install --name venv_name
```



ObsPy

Obspy is a convenient Python module to read, save and manipulate seismic data. It provides lots of useful built-in functions that help facilitate the analysis of seismic data.

```
>conda config --add channels conda-forge
>conda create -n obspy python=3.7
> conda activate obspy
(obspy)> conda install obspy
```

Check the success of the installation by running Python in the venv and importing the obspy module.

Reading and saving data with ObsPy

- Obspy is compatible with the most common seismic data file formats (.sac, .sgy, .mseed…);
- The notebook `read_and_save_with_obspy.ipynb` provides examples of the various ways in which you can read and save seismic data.

Reading and saving data with ObsPy

- Obspy can read data from local and remote locations;
- Keep in mind that, on local directories, Obspy can read data using different procedures:
  - `read(file_name)` function;
  - SDS protocol;

Use of the `read` function

- Useful for small data chunks;
- Saves all the data (traces, times, ecc…) in a stream object.

SDS protocol

- Basic level of standardization and portability to softwares that need direct access to data files;
- Given data from a seismic station from a given year and characterized by the codes `NETWORK, STATION, LOCATION, CHANNEL`, the basic directory and file layouts are defined as:

```
PATH_TO_SDS/Year/NETWORK/STATION/CHANNEL.TYPE/NETWORK.STATION.LOCATION.CHANNEL.TYPE.YEAR.DAY
```

Stream object

Streams are list-like objects which contain multiple Trace objects, i.e. gap-less continuous time series and related header/meta information.

Each Trace object has an attribute called data pointing to a NumPy ndarray of the actual time series and the attribute stats which contains all meta information in a dictionary-like Stats object. Both attributes starttime and endtime of the Stats object are UTCDateTime objects.

Read files from remote servers

ObsPy makes use of the IRIS FDSN webservice to access publicly available seismic data.

The user must know the IRIS client name of the data provider (e.g. INGV), the network (e.g. IV) and station name (e.g. SENA).

To print all available IRIS clients, type the following:

```
from obspy.clients.fdsn.header import URL_MAPPINGS

for key in sorted(URL_MAPPINGS.keys()):
   print("{0:<11} {1}".format(key,  URL_MAPPINGS[key]))
```

Inventory files

Raw seismic data must be deconvoluted with the station’s instrument response. This is done with the use of inventory files (often called dataless files) saved in various formats (.xml, .see, etc…).

```
inventory = read_inventory(file_name.xml)

inv = client.get_stations(network=net, station=sta,
location=loc, channel=cha, level="response")

stream.remove_response(inv = inventory, response = ‘ACC’)
```

Hands-on material

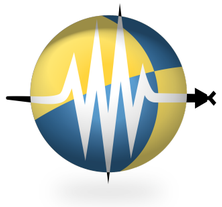
See the notebook `read_and_save_with_obspy.ipynb` *for examples on how to read and save data with ObsPy*

Use the notebook `create_inventory_file.ipynb` *to create your own inventory*

Spectra

Spectra can be calculated in many different ways using an almost infinite variety of functions. I focused on two methods for the analysis:

- ObsPy PPSD function;
- SciPy fft/spectrogram functions

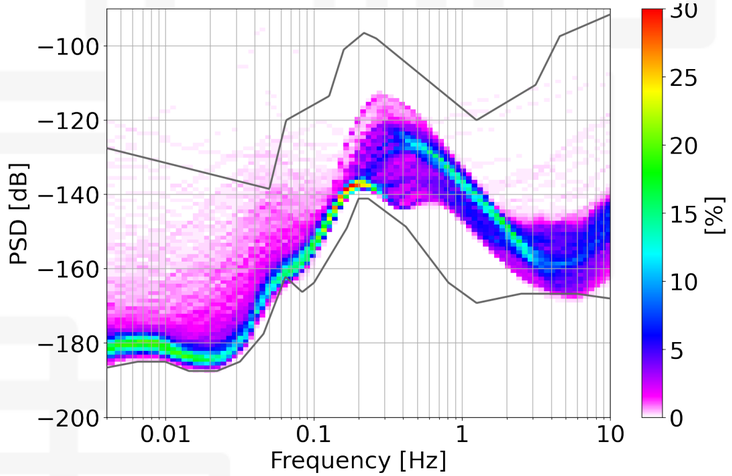


ObsPy

A Python Framework for Seismology

VS.

SciPy



ObsPy PPSD

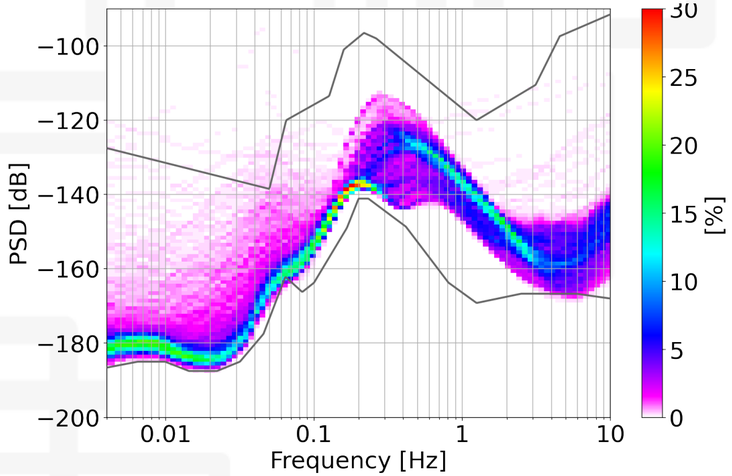
Based on the procedure by McNamara et al. [2004]. Saves and plots PSD histograms.

Also the spectra from the single time segments are saved with their time stamps.

The procedure divides the frequency range into octaves and the in each octave and average is done. This heavily smoothes the spectra.

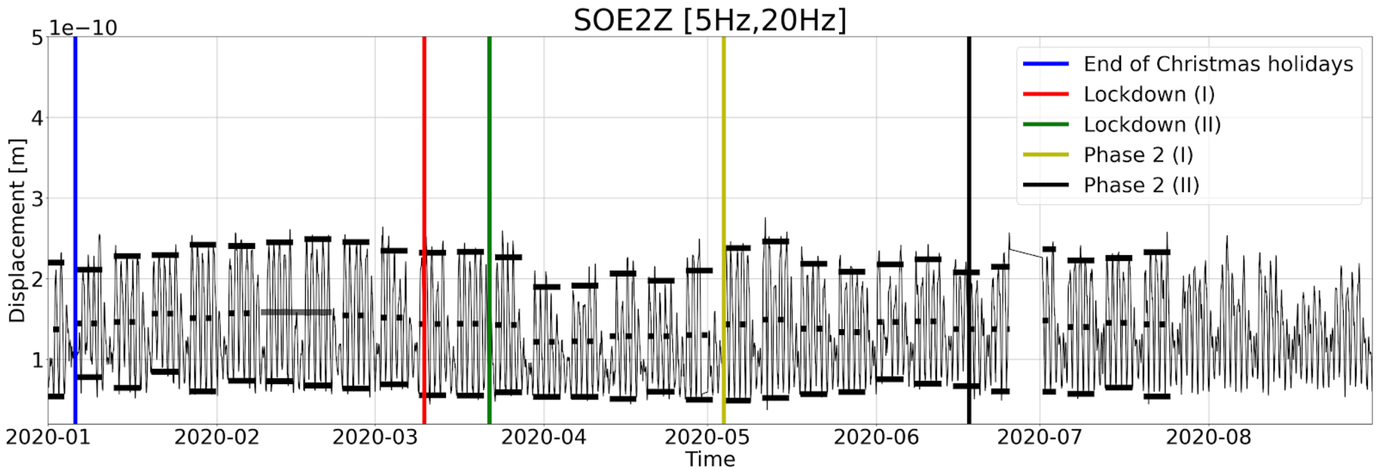
So far I calculated spectra on 1 hr segments with a 50% overlap.

ObsPy PPSD



Using ObsPy, makes very easy also the calculation of the spectra's time series:

```
extract_psd_values(period=T)
```



ObsPy PPSD

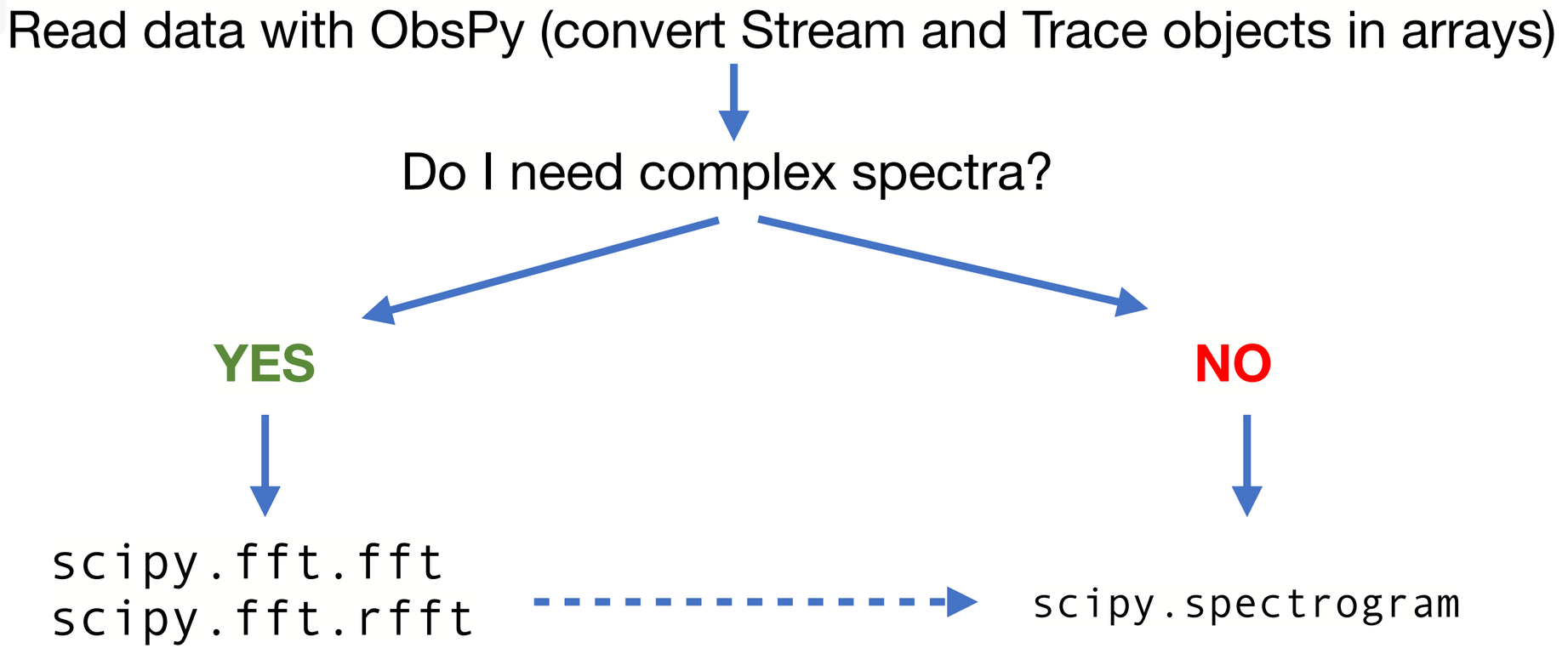
Pros:

- easy to use method (little coding required);
- saves all relevant quantities;
- easy read/save;
- calculation of time series and spectrograms straightforward;

Cons:

- not flexible (closed function);
- spectra smoothed too much (miss all spectral lines), can get only a general understanding of noise levels;

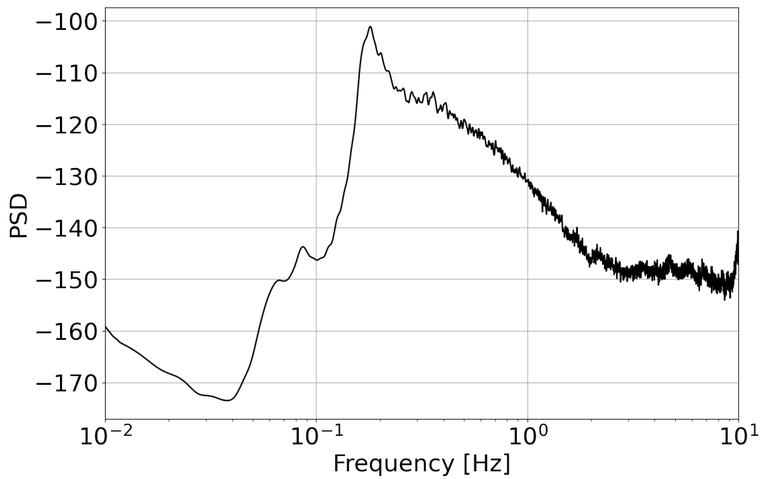
SciPy functions



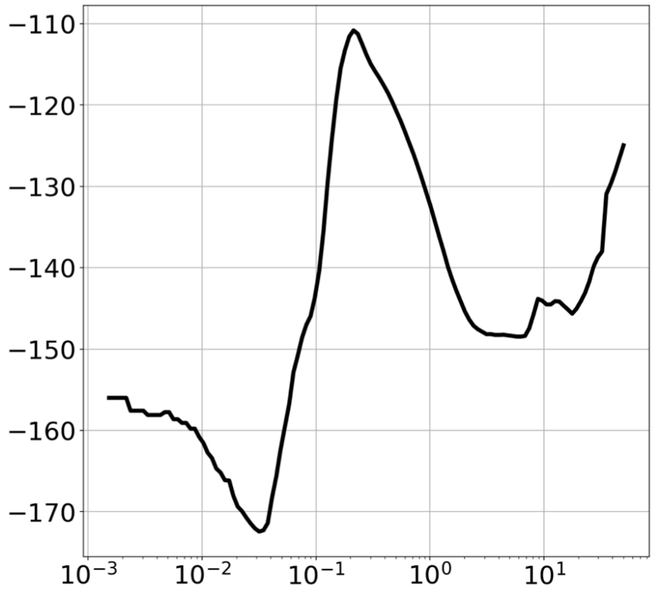
SciPy functions

Compared against ObsPy, the spectra are more full of information

SciPy



ObsPy



SciPy functions

`scipy.fft.fft`
`scipy.fft.rfft`

Pros:

- flexible method;
- reasonably efficient;
- outputs two-sided complex spectra.

Cons:

- Requires more coding;
- calculation of time series cumbersome.

`scipy.spectrogram`

Pros:

- handles the division of the data into segments;
- outputs spectrograms;

Cons:

- Real spectra only;
- requires more coding;
- calculation of time series cumbersome.

Things to do with spectra

- Spectral ratios (surface underground):
  - calculated with ObsPy's spectra;
  - 1 hour long segments, 50% overlap;
  - calculated only on spectra segments at same times;
  - output the mean and percentiles.
- H/V ratio:
  - calculated using procedures used in literature (get raw spectra, apply konno-omachi smoothing, make ratios, output mean and percentiles),