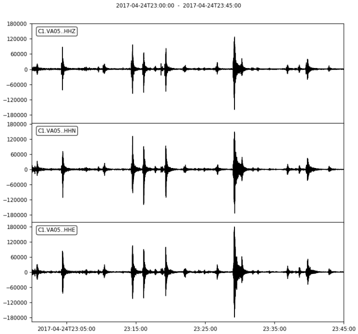
Earthquake detection and location: towards automatization
Day 1

Christian Sippl

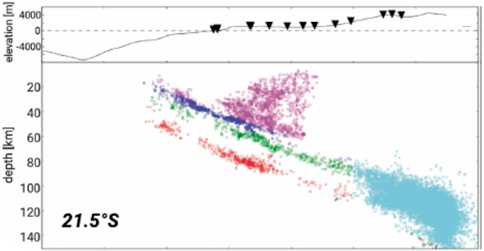
Geofyzikální ústav Akademie věd ČR, v.v.i.

Universidad de Concepcion, January 13-17, 2020

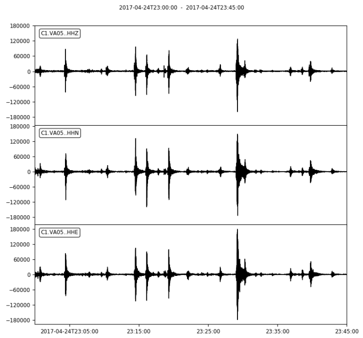


raw waveform data

[many steps are missing...]

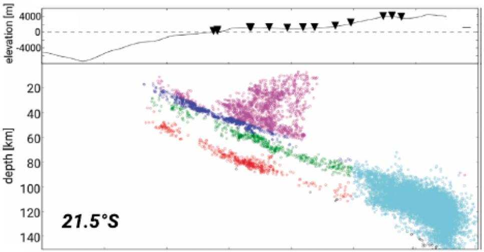


earthquake catalog



raw waveform data

[many steps are missing...]



earthquake catalog

Missing steps

Processing, detection of P and S onsets, event association, location techniques, uncertainty estimation, ...

- **Monday**: Intro and earthquake detection basics (triggers)
- **Tuesday**: Phase pickers and the earthquake association problem
- **Wednesday**: Basic earthquake location techniques, uncertainty estimation
- **Thursday**: Multi-event (re)location techniques, use of cross-correlations
- **Friday**: Putting it all together: how to design an automated approach

- **Monday**: Intro and earthquake detection basics (triggers)
- **Tuesday**: Phase pickers and the earthquake association problem
- **Wednesday**: Basic earthquake location techniques, uncertainty estimation
- **Thursday**: Multi-event (re)location techniques, use of cross-correlations
- **Friday**: Putting it all together: how to design an automated approach

Monday: Intro and earthquake detection basics

- *9:00 - 10:30*: Intro, seismic data handling with python/obspy
- *11:00 - 12:30*: Exercises
- *13:30 - 15:15*: Earthquake detection - basic trigger algorithms
- *15:30 - 17:00*: Exercises

Day 1: Goals

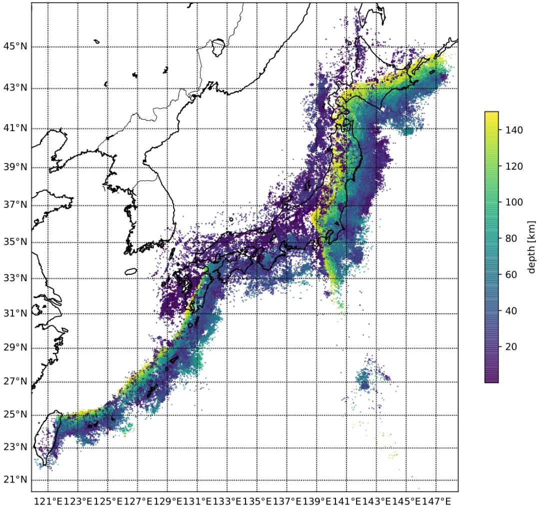
1. You can download seismic data from international networks and process it using python/obspy
2. You have an overview over what approaches for automated earthquake detection exist
3. You have gained some experience in the application of an STA/LTA trigger on real data, its tuning and calibration

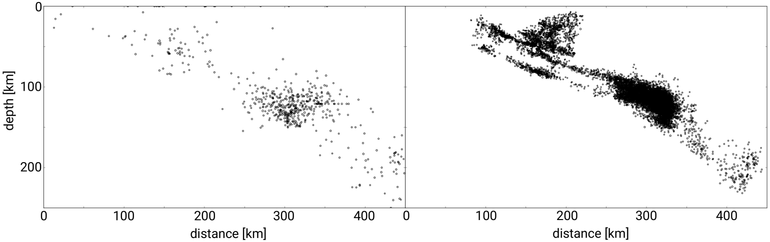
1. Who has worked with seismic data before? Who has handpicked P and S arrivals?
2. Who has programming experience (in general, python)?
3. Who has used obspy for seismology purposes before?

Most importantly

- Is python and obspy working on everyone's computer?
- Any other questions before we start?

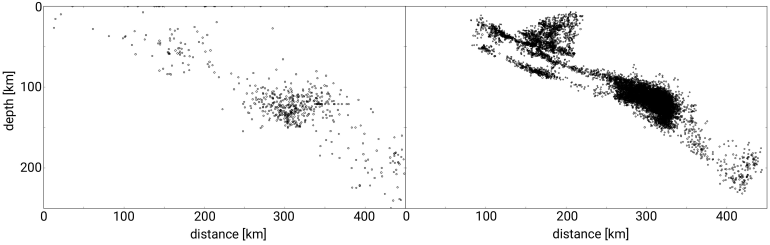
- Probably the most basic seismology resource
- Used as a starting point for nearly any seismological investigation (e.g. moment tensors, analysis of phases, tomography, statistics, tsunami studies)
- Providers: Global (ISC, USGS, Geofon,...) and national agencies (JMA, CSN,...)
- Mostly compiled "by hand" (today often with automatic pre-picker)





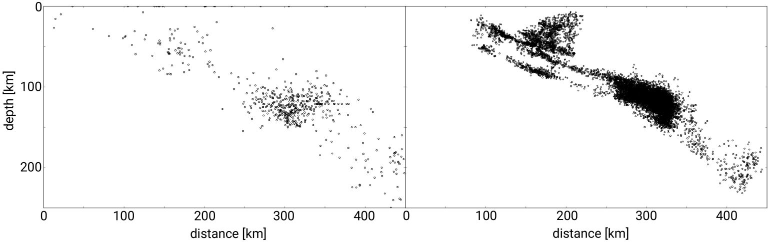
Why should WE do this?

1. Use of temporary network data (higher resolution in small area)
2. Also agency-derived catalogs can be improved (2D or 3D velocity model, relative techniques, etc.)
3. Investigation of areas where no agency-derived catalog exists



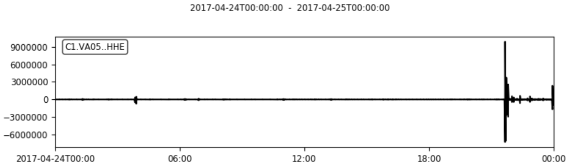
Why should WE do this?

1. Use of temporary network data (higher resolution in small area)
2. Also agency-derived catalogs can be improved (2D or 3D velocity model, relative techniques, etc.)
3. Investigation of areas where no agency-derived catalog exists



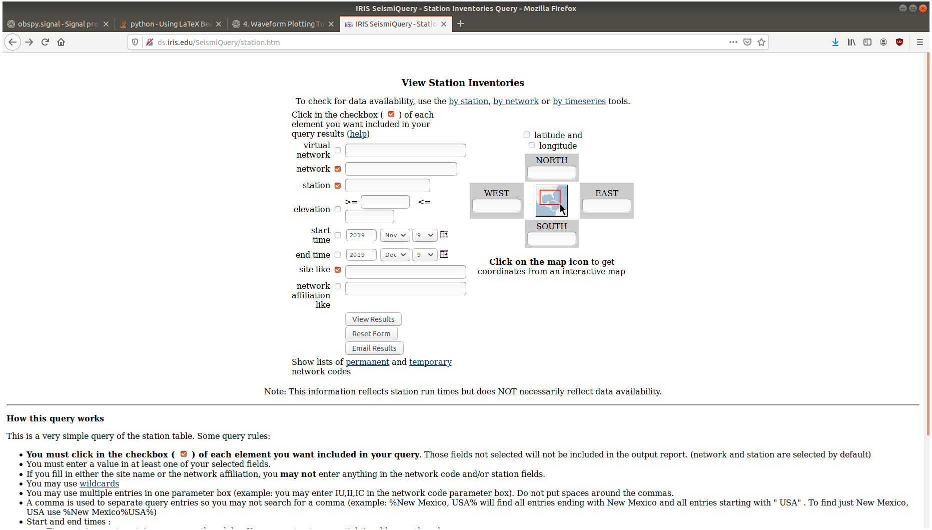
Why should WE do this?

1. Use of temporary network data (higher resolution in small area)
2. Also agency-derived catalogs can be improved (2D or 3D velocity model, relative techniques, etc.)
3. Investigation of areas where no agency-derived catalog exists

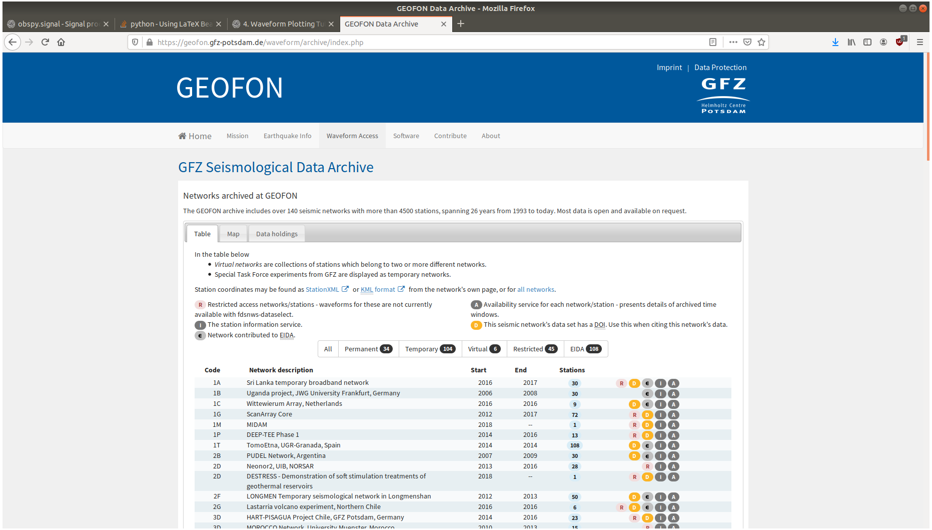


What is a seismogram?

- Series of discrete measurements of ground velocity against time
- Different sampling frequencies (how many measurements) and instrument types (what frequencies can be measured);
  $\rightarrow$ abbreviations LH, BH, HH, SH, EH, CH, ...
- Y-axis is in arbitrary units (counts) that can be converted to m/s using the instrument response
- Data formats: miniSEED, SAC, GSE, GCF, SEG-Y, ...



http://ds.iris.edu/SeismiQuery/station.htm

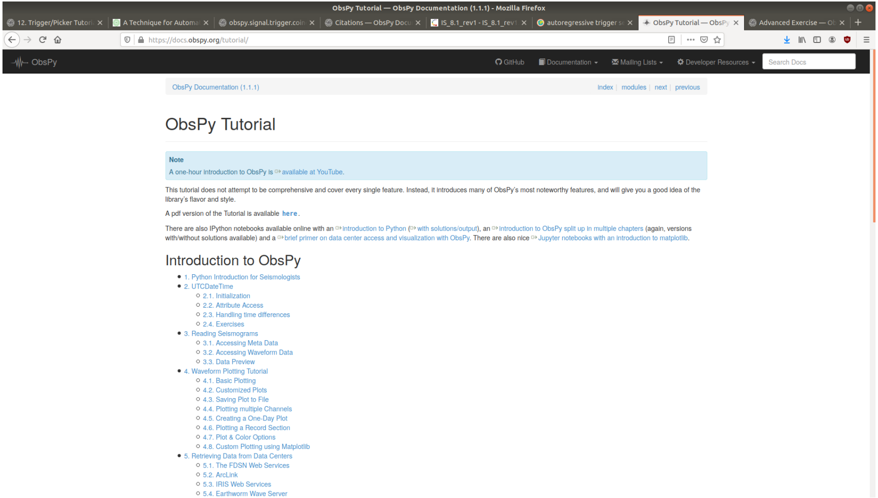


https://geofon.gfz-potsdam.de/waveform/archive/index.php

There is a number of ways to use python:

1. Writing and running scripts (python X.py)
2. Interactive python shell (Ipython)
3. Jupyter notebooks

Up to you what to use. My preference is writing scripts, with Ipython used for trying things out and/or demonstration



ObsPy Tutorial — ObsPy Documentation (1.1.1) - Mozilla Firefox

https://docs.obspy.org/tutorial/

ObsPy | GitHub | Documentation | Mailing Lists | Developer Resources | Search Docs

ObsPy Documentation (1.1.1) — index | modules | next | previous

ObsPy Tutorial

Note

A one-hour introduction to ObsPy is available at YouTube.

This tutorial does not attempt to be comprehensive and cover every single feature. Instead, it introduces many of ObsPy's most noteworthy features, and will give you a good idea of the library's flavor and style.

A pdf version of the Tutorial is available `here`.

There are also IPython notebooks available online with an introduction to Python (with solutions/output), an introduction to ObsPy split up in multiple chapters (again, versions with/without solutions available) and a brief primer on data center access and visualization with ObsPy. There are also nice Jupyter notebooks with an introduction to matplotlib.

Introduction to ObsPy

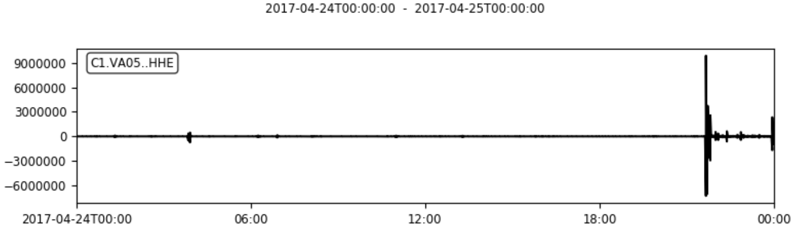
https://docs.obspy.org/tutorial/

```
#Open client connection
from obspy.clients.fdsn import Client
from obspy.core import UTCDateTime

client = Client('iris')

t = UTCDateTime(2018,1,1,0,0,0)

#CSN station Faro Punta Hualpen
data = client.get_waveforms('C1','BI05','--','HHZ',t,t+3600.)
```

```
from obspy.core import read

data = read('quake.mseed')
print data

#Output:
3 Trace(s) in Stream:
C1.VA05..HHE | 2017-04-24T00:00:00.000000Z - 2017-04-25T00:00:00.000000Z | 100.0
    Hz, 8640001 samples
C1.VA05..HHN | 2017-04-24T00:00:00.000000Z - 2017-04-25T00:00:00.000000Z | 100.0
    Hz, 8640001 samples
C1.VA05..HHZ | 2017-04-24T00:00:00.000000Z - 2017-04-25T00:00:00.000000Z | 100.0
    Hz, 8640001 samples
```

```
#how to choose single Trace?
print data[0]

#Output:
C1.VA05..HHE | 2017-04-24T00:00:00.000000Z - 2017-04-25T00:00:00.000000Z | 100.0
    Hz, 8640001 samples

#better option (more flexible)
print data.select(component='Z')

#Output:
1 Trace(s) in Stream:
C1.VA05..HHZ | 2017-04-24T00:00:00.000000Z - 2017-04-25T00:00:00.000000Z | 100.0
    Hz, 8640001 samples
```

```
#only look at data
print data[0].data
#Output
array([ 247,   398,   511, ..., -3021, -3886, -4889], dtype=int32)

#Visualize
data[0].plot()
```



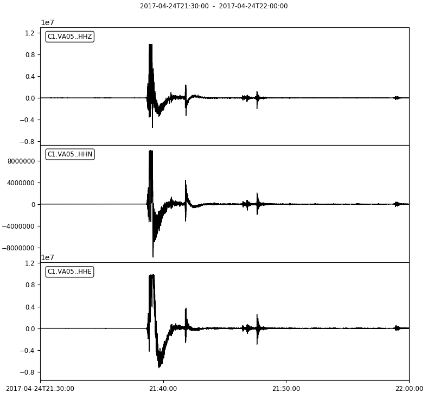
```
#only look at header
print data[0].stats
#Output
         network: C1
         station: VA05
        location:
         channel: HHE
       starttime: 2017-04-24T00:00:00.000000Z
         endtime: 2017-04-25T00:00:00.000000Z
   sampling_rate: 100.0
           delta: 0.01
            npts: 8640001
           calib: 1.0
         _format: MSEED
           mseed: AttribDict({u'record_length': 512, u'encoding': u'STEIM2',
               'filesize': 41568768, u'dataquality': u'M', u'number_of_records':
               26639, u'byteorder': u'>'})
```

```
#write out as a SAC file
data.write('quake.sac',format='SAC')
#this will write three files, because SAC can not have different component records
    in one file

#these can be read in again
data_sac = read('quake01.sac')
#how can this be? Answer: format auto detection

print data_sac[0].stats
```

```
         network: C1
         station: VA05
        location:
         channel: HHE
       starttime: 2017-04-24T00:00:00.000000Z
         endtime: 2017-04-25T00:00:00.000000Z
   sampling_rate: 100.0
           delta: 0.01
            npts: 8640001
           calib: 1.0
         _format: SAC
             sac: AttribDict({u'knetwk': u'C1', u'nzyear': 2017, u'nzjday': 114,
                 u'iztype': 9, u'nzhour': 0, u'lcalda': 0, u'iftype': 1, u'nvhdr':
                 6, u'depmin': -7279245.0, u'kcmpnm': u'HHE', u'nzsec': 0,
                 u'depmen': 110.85744, u'depmax': 9868107.0, u'lovrok': 1, u'scale':
                 1.0, u'delta': 0.0099999998, u'nzmsec': 0, u'lpspol': 1, u'b': 0.0,
                 u'e': 86400.0, u'leven': 1, u'kstnm': u'VA05', u'nzmin': 0,
```

```
from obspy.core import UTCDateTime

time1 = UTCDateTime(2018,5,1,12,35,44.66)
time2 = UTCDateTime(2012,12,28,1,35,55.77)

print (time1 - time2)

#Output
168519588.89
#this is the time difference in seconds

print time1.julday
#Output
121
#accessing year, month, day, hour, minute, second, microsecond works in the same way
```

```
time1 = UTCDateTime(2017,4,24,21,30,00)
time2 = UTCDateTime(2017,4,24,22,00,00)

cut = data.slice(starttime=time1,endtime=time2)
cut.plot()
#now we get a zoom onto the 2017 Valparaiso
    earthquake
```



```
cut[0].data.mean()
#Output
-6855.8640451997489

cut.detrend(type='demean')
cut[0].data.mean()
#Output
6.8544960686362067e-11

#Careful, this is an in-place method which changes your trace. It is often
    recommendable to make a copy before
from copy import deepcopy as cp

cut_copy = cp(cut)
```

```
print cut[0].stats.sampling_rate
#Output: 100.0

cut.decimate(factor=4) #integer, otherwise use cut.resample()
print cut[0].stats.sampling_rate
#Output: 25.0

#Check what was done with trace:
print cut[0].stats.processing
#Output:
[u'ObsPy 1.1.1: trim(endtime=UTCDateTime(2017, 4, 24, 22,
    0)::fill_value=None::nearest_sample=True::pad=False::starttime=UTCDateTime(2017,
    4, 24, 21, 30))', u"ObsPy 1.1.1: detrend(options={}::type='demean')", u"ObsPy
    1.1.1: detrend(options={}::type='linear')", u"ObsPy 1.1.1:
    filter(options={'freq': 12.5, 'maxorder': 12}::type=u'lowpass_cheby_2')",
    u'ObsPy 1.1.1: decimate(factor=4::no_filter=False::strict_length=False)']
```

```
cut.filter('bandpass',freqmin=1,freqmax=5,corners=2)
#or
cut.filter('lowpass',freq=0.2,corners=2)
#or
cut.filter('highpass',freq=3.3,corners=2)

#This modifies traces in-place as well. Careful: often there is a change of data
    type!
print cut[0].data
#Output:
[  7009.8640452  7005.8640452  7059.8640452 ..., 16730.8640452
  15190.8640452 13091.8640452]
```

```
#Open client connection
from obspy.clients.fdsn import Client

client1 = Client('GFZ')
client2 = Client('iris')

t = UTCDateTime(2018,1,1,0,0,0)

#IPOC station Huatacondo
data1 = client1.get_waveforms('CX','PB01','--','HHZ',t,t+3600.)
#CSN station Faro Punta Hualpen
data2 = client2.get_waveforms('C1','BI05','--','HHZ',t,t+3600.)

#data1 and data2 are Stream objects, you can now save, process, filter them etc.
data1.plot()
data2.write('PI05_test.mseed',format='MSEED')
```

Exercises

1. Filter plot

Download seismic data for two time intervals from station VA05 (network C1, available on IRIS):
2017/01/01 18:03:00 - 18:05:00
2017/04/24 04:47:00 - 04:52:00
Plot the waveforms, then apply a series of different filters. Do not forget to copy the raw trace before filtering, demeaning would be good as well. Examples to look at: unfiltered trace, highpass at 1 Hz, highpass at 5 Hz, lowpass at 1 Hz, lowpass at 0.5 Hz, bandpass 0.5-5 Hz, bandpass 0.1-1 Hz, bandpass 5-20 Hz.

2. Record section

The first event from exercise was a magnitude 4.2 event that occurred at lat/lon = -32.115/-71.175 at a depth of 88 km (source: CSN catalog).
Download data for this event from additional stations using the IRIS or GEOFON data holdings. Best choose around 10-15 stations and limit your download to the vertical component data. From this, create a "record section" plot. This means, plot all seismograms into one plot, with distance from the event origin on the Y-axis. Hint: check ObsPy Tutorial (or: obspy.geodetics.gps2distazimuth()); station coordinates can be found at the data source(s).

Question

How can we automatically look for earthquakes in the seismic data?
What parameters could we use for this?

- What else?

Question

How can we automatically look for earthquakes in the seismic data?
What parameters could we use for this?

Earthquakes are usually characterized by:

- What else?

Question

How can we automatically look for earthquakes in the seismic data?
What parameters could we use for this?

Earthquakes are usually characterized by:

- Increase of signal amplitude
- What else?

Question

How can we automatically look for earthquakes in the seismic data?
What parameters could we use for this?

Earthquakes are usually characterized by:

- Increase of signal amplitude
- Change in frequency/spectral content
- What else?

Question

How can we automatically look for earthquakes in the seismic data?
What parameters could we use for this?

Earthquakes are usually characterized by:

- Increase of signal amplitude
- Change in frequency/spectral content
- More focused polarization (we go into that tomorrow)
- What else?

Question

How can we automatically look for earthquakes in the seismic data?
What parameters could we use for this?

Earthquakes are usually characterized by:

- Increase of signal amplitude
- Change in frequency/spectral content
- More focused polarization (we go into that tomorrow)
- What else?

There are a large number of algorithms that exploit these for automatic triggering

Trigger algorithms only report “alerts”; they do not attempt to recognize specific phases or uncertainties, are fast and directly applicable to raw waveform data
Pickers attempt to deliver a phase classification plus an uncertainty estimate; some can also be run on raw data, some need to be supplied with specific time window around the pick to make

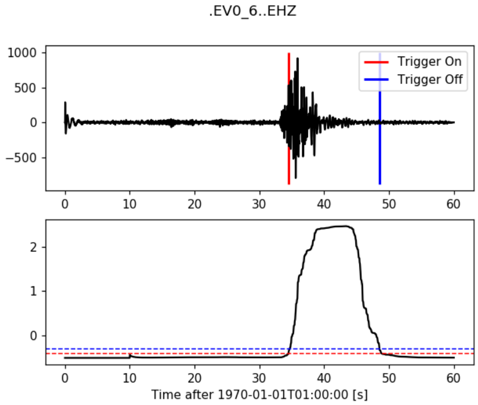
Selection of algorithms

- Z detector
- STA/LTA trigger
- Autoregressive picker

Simple algorithm that looks for sudden amplitude changes

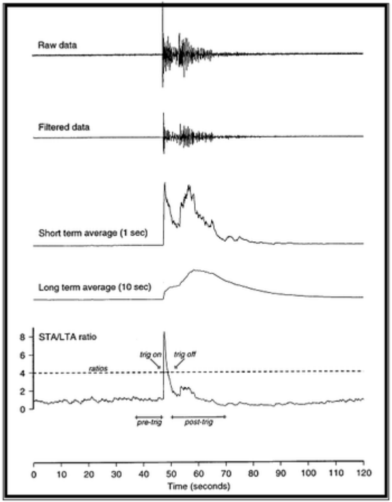
$$Z(x_i) = (x_i - \mu)/\sigma \qquad (1)$$

x_i: data window
μ: mean of that data window
σ: standard deviation in this data window
Only tuning parameter is window length



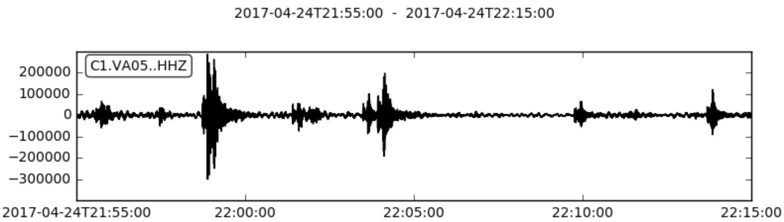
STA/LTA trigger

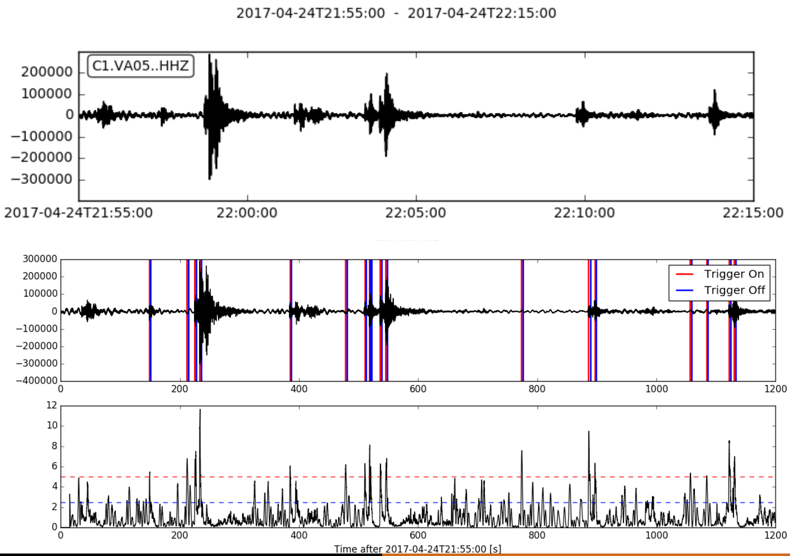
1. Demean, detrend and bandpass filter data
2. Calculate average amplitude in a short and a long window that simultaneously propagate over the data
3. "Characteristic function" is the ratio of the two
4. Set two thresholds for triggering and detriggering



Tuning parameter	Effects
STA window length	*Increase*: suppresses detection of spike-like signals increases trigger delay *Decrease*: Trigger is more accurate detects sharp amplitude changes better than emergent ones
LTA window length	*Increase*: Better detection of emergent onsets Worse at finding secondary onsets (S phases, close events) *Decrease*: Loss of sensitivity
filter parameters	ideally enhances earthquake signals and suppresses noise
threshold value	*too high*: events are missed *too low*: many false alerts
detrigger value	*too high*: many phases per event are triggered *too low*: the second of two events in close succession may be missed

```
#Import trigger module
from obspy.signal import trigger
from obspy.signal.trigger import classic_sta_lta

from obspy.core import read
data = read('C1.VA05..HHZ.D.2017.114')
from obspy.core import UTCDateTime
a = UTCDateTime(2017,4,24,21,55)
b = UTCDateTime(2017,4,24,22,15)
ff = data.slice(starttime=a,endtime=b)

cft = trigger.classic_sta_lta(ff[0].data,100,1500)
plot_trigger(ff[0],cft,5.0,2.5)
```


2017-04-24T21:55:00 - 2017-04-24T22:15:00
C1.VA05..HHZ
200000
100000
0
−100000
−200000
−300000
2017-04-24T21:55:00
22:00:00
22:05:00
22:10:00
22:15:00


2017-04-24T21:55:00 - 2017-04-24T22:15:00
C1.VA05..HHZ
200000
100000
0
−100000
−200000
−300000
2017-04-24T21:55:00
22:00:00
22:05:00
22:10:00
22:15:00
300000
200000
100000
0
−100000
−200000
−300000
−400000
Trigger On
Trigger Off
0
200
400
600
800
1000
1200
12
10
8
6
4
2
0
Time after 2017-04-24T21:55:00 [s]

Classic STA/LTA definition:

$$STA_i = \frac{x_i^2 - x_{i-N_{sta}}^2}{N_{sta}} + STA_{i-a} \qquad (2)$$

$$LTA_i = \frac{x_{i-N_{sta}-1}^2 - x_{i-N_{sta}-N_{lta}-1}^2}{N_{lta}} + LTA_{i-1} \qquad (3)$$

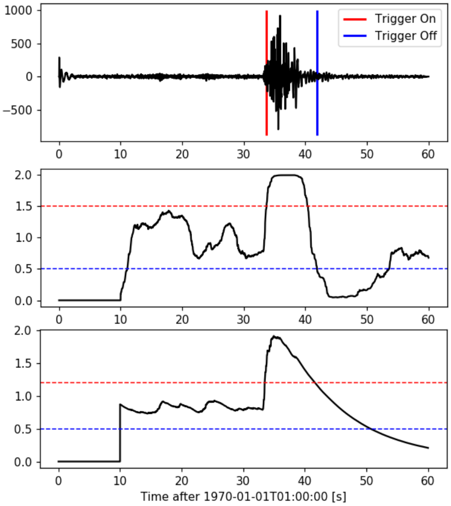
Recursive STA/LTA definition:

$$STA_i = Cx_i + (1-C)STA_{i-1} \qquad (4)$$

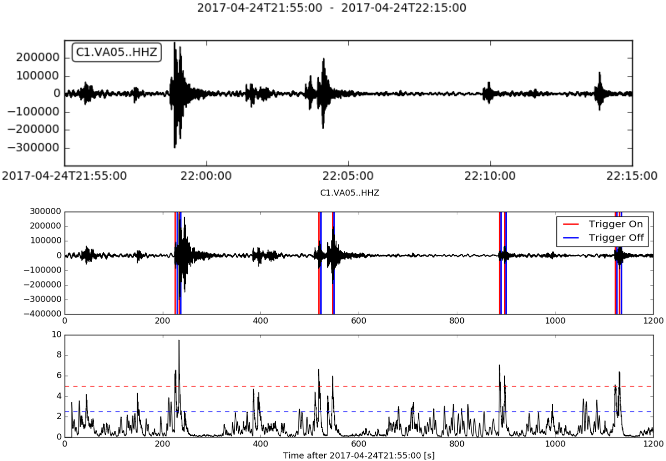
$$C = 1 - e^{-S/T} \qquad (5)$$

LTA analog; usally used: $C = 1/N_{sta}$ or $C = 1/N_{lta}$; N: window lengths; x: data value

Rectangular vs. exponentially decaying impulse response: advantage of shorter "blind window" in recursive STA/LTA



```
cft = trigger.recursive_sta_lta(ff[0].data,100,1500)
plot_trigger(ff[0],cft,5.0,2.5)
```

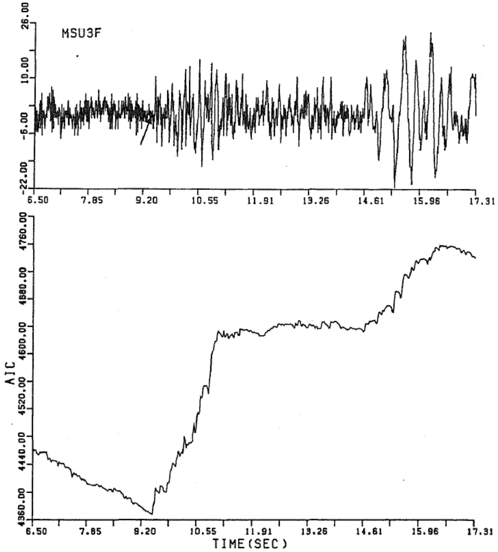


Application is simple and straightforward, the problem is proper calibration:
How do we tune parameters so that:

- Only few events are missed, but number of false alerts is low
- Picks consistently correspond to phases (not mixing P and S)
- it catches large AND small event (different spectra)
- Also: What if there is more than event at the same time? P_n vs P_g?...

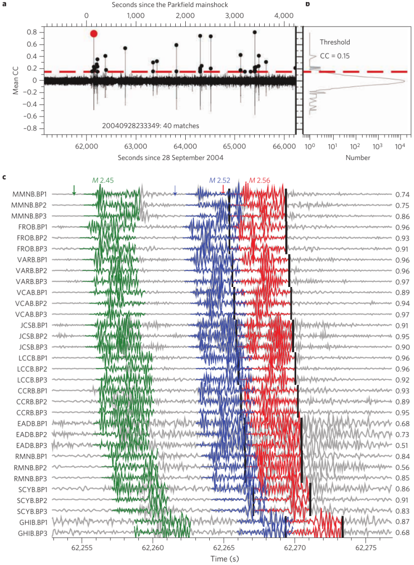
Family of "autoregressive" pickers first applied in Japan in the late 1980s.

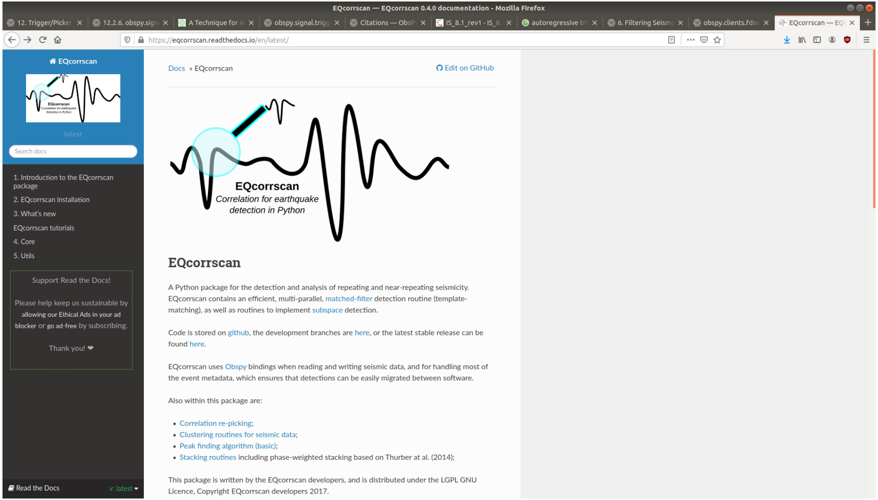
- Investigation of spectral properties
- Assumption that pick is transition from stationary "noise model" to stationary "earthquake model"
- Akaike information criterion (AIC) is statistical measure for likelihood of a point being this transition
- Minimum AIC means highest likelihood
- Obspy: trigger.ar_pick(); this is a method that combines STA/LTA and AR picker



This is a secondary technique designed to find more/smaller events!

- Choice of template events
- Cross-correlation is computed for the raw data (more on this on thursday)
- Selection of events via network threshold



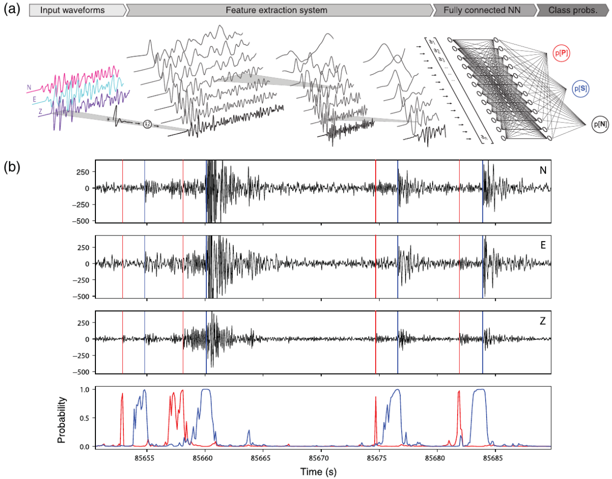


https://eqcorrscan.readthedocs.io/en/latest/

Pros and Cons of template matching

+ Very sensitive method that can identify small events, even if they are buried in noise or coda
+ Applicable to raw data
+ Not prone to false alerts through network detection
– Not applicable as a primary method (i.e. when no previous information exists
– Critically dependent on presence of good/complete set of template events
– Very computationally heavy

Often used for times of increased event numbers (early parts of aftershock series, swarms)



Training based on input of massive amounts of waveform data with picks (signal/noise windows, pick quality/classification); Definition of "features" to evaluate

Pros and Cons of machine learning methods

+ Combinations of many parameters can be evaluated; more sensitive and generalized picker can result
+ Applicable to large volumes of raw data
+ Earthquake catalogs that have been assembled by agencies over the years are great training data
– As with any machine learning method, understanding of how exactly it works is lost
– Critically dependent on good and complete training dataset
– Computationally expensive

Covered today

- Ways to retrieve event “alerts” from raw seismic waveform data
- Good: quick and efficient methods, can be applied to large amounts of data
- But: Calibration is hard, avoiding missed events/false alerts is nearly impossible
- Precision of arrival times is not very good either, discriminating phases is not possible

Tomorrow: How could we improve these things? $\rightarrow$ Use more specific methods, more stations at the same time

Exercises

3. Trigger exploration

Download 1 day of data from IRIS (2017/04/24, station VA05, network C1). This day includes the main event of the Valparaiso sequence (magnitude 6.9) and many early aftershocks.

Use obspy's STA/LTA trigger to detect as many as possible of the earthquakes without creating too many false alerts. Tune the parameters (window sizes, thresholds) and use plot_trigger() to visually check performance. Pre-filtering is another way of tuning that should be explored.

4. Automatization script

Once you are happy with the trigger performance, write a routine that runs the trigger and outputs the alert times into an output file (hint: funtion trigger.trigger_onset()). Try to write the script so that it operates on downloaded data and can loop over many data files/stations (if there is time, try it out for additional stations).

Day 1: Goals

1. You can download seismic data from international networks and process it using python/obspy
2. You have an overview over what approaches for automated earthquake detection exist
3. You have gained some experience in the application of an STA/LTA trigger on real data, its tuning and calibration

Have these goals been reached? Open questions/problems etc.?

Day 1: Goals

1. You can download seismic data from international networks and process it using python/obspy
2. You have an overview over what approaches for automated earthquake detection exist
3. You have gained some experience in the application of an STA/LTA trigger on real data, its tuning and calibration

Have these goals been reached? Open questions/problems etc.?

Tomorrow: Approaches and problems in automated phase picking (especially S); network coincidence