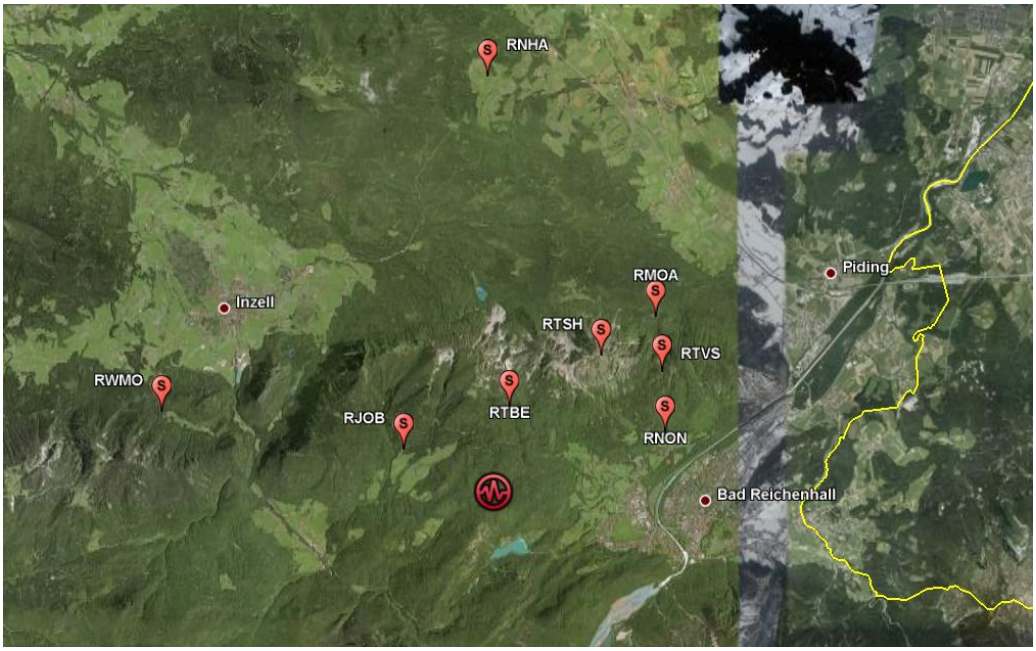
Practical

In this practical the task is to estimate local magnitudes for an earthquake in the Hochstaufen massif in south-eastern Bavaria. Using this example, we will see how to develop an easily readable and extensible, automated processing workflow using ObsPy. We will start with simple programs with many manually specified, hard-coded values and build on them step by step to make the program more flexible and dynamic.
For every part of the exercise there is a Python file with a few comments and tips to build upon, as well as a file with a complete solution. The recommended way to do the exercises is to use a text editor to work on the program and run the program in a IPython shell via `>>> run -i myfile.py`. This lets you continue work interactively after the program is executed. It is best to quit and start a new IPython shell often (to avoid confusion with "old" variables from prior program executions).



[http://earth.google.com]

Exercise 1 (*Estimate Local Magnitude*)
Use the file `RJOB_WA_CUT.MSEED` to read `MiniSEED` waveform data from the earthquake. These data have already been simulated to displacement on a Wood-Anderson seismometer (m) and trimmed to the right time span. Estimate the zero-to-peak amplitude amp_{zp} as half the mean of the maximum minus minimum amplitude on North and East components in the given short time window. Estimate the local magnitude M_l using a hypocentral distance of $d_{hypo} = 7.1$ (km) given the formula (`Bakun&Joyner, BSSA, 1984`):

$$M_l = \log_{10}\left(\frac{amp_{zp}}{1000\frac{\text{mm}}{\text{m}}}\right) + \log_{10}\left(\frac{d_{hypo}}{100\text{km}}\right) + 0.00301 * (d_{hypo} - 100\text{km}) + 3$$

Exercise 2 (*Simulate Wood-Anderson Seismometer*)
Use the file `RJOB.MSEED` to read the original `MiniSEED` waveform data (m/s). Set up two

dictionaries containing the response information of both the original instrument (an `STS-2`) and the Wood-Anderson seismometer in poles-and-zeros (`PAZ`) formulation. Each `PAZ` dictionary needs to contain `sensitivity` (overall sensitivity), `gain` (normalization factor), `poles` and `zeros`. After the instrument simulation, trim the waveform to a short time window around the origin time (`2008-04-17T16:00:32Z`) and calculate M_l like in exc. 1. Use the following values for the `PAZ` dictionaries:

```
STS-2: 'sensitivity': 2516778600.0
       'gain': 60077000.0
       'poles': [-0.037004+0.037016j, -0.037004-0.037016j, -251.33+0j,
                 -131.04-467.29j, -131.04+467.29j]
       'zeros': [0j, 0j]
Wood-Anderson: 'gain': 1
               'sensitivity': 2800
               'poles': [-6.2832-4.7124j, -6.2832+4.7124j]
               'zeros': [0j]
```

Exercise 3 (*Compare to EMSC Catalog*)
Fetch a list of events from NERIES/EMSC for the time of the earthquake in the Hochstaufen region (47.75 N, 12.85 E) using `obspy.neries`. Check if the estimated magnitudes so far roughly match the magnitude information in the catalog.

Exercise 4 (*Fetch Data from WebDC*)
Modify exc. 2 to fetch the waveform data via ArcLink from WebDC using the `Client` provided in `obspy.arclink`. Use option `getPAZ=True` to fetch response information along with the waveform. The `PAZ` information will get attached to the `Stats` object of all traces in the returned `Stream` object. During instrument simulation use option `paz_remove='self'` to use the attached `PAZ` information fetched from WebDC. Calculate M_l like in exc. 2.

Exercise 5 (*Compute Hypocentral Distance*)
Modify exc. 4 to fetch coordinate information for station `RJOB` along with the waveform (in a similar manner to the `PAZ`). Use exc. 3 to fetch the event information from NERIES/EMSC. Use the origin time (stored as `datetime` in the event dictionary) during the data request from WebDC instead of the previously hard-coded value. Also calculate the hypocentral distance dynamically. Use function `utlGeoKm` from module `obspy.signal` to compute horizontal distances from geographic coordinates. Calculate M_l like in exc. 4 using the computed hypocentral distance.

Exercise 6 (*Determine Event Onset Using a Triggering Algorithm*)
Read waveform data from file `RJOB.MSEED` and run a recursive STA/LTA trigger on the Z component of the data. Compute the approximate event onset time from `starttime` and `sampling_rate` of the trace and from the position of the maximum in the triggered data (see built-in methods of `numpy.ndarray` objects: `http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html`). Store this time in an `UTCDateTime` object.

Exercise 7 (*Use Trigger Time for Time Window Selection*)
Modify exc. 5 to use a dynamically determined trigger time like in exc. 6 for the trimming

operations on the waveform data. Calculate M_l like in exc. 5.

Exercise 8 (*Estimate Magnitudes for a List of Stations*)
Modify exc. 7 and use a list of stations (e.g. `RJOB`, `RMOA` and `RNON`) instead of just a single one. Loop over this list and estimate the magnitude for each station individually.

Exercise 9 (*Fetch List of Available Stations from WebDC*)
Fetch a list of available stations in network `BW` (BayernNetz) for the time around the earthquake (`2008-04-17T16:00:32Z`) via ArcLink from WebDC. Print the station code for every station in the list.

Exercise 10 (*Estimate Magnitudes at All Available Stations*)
Modify exc. 8 to fetch a list of stations like in exc. 9. Loop over this list, fetch the data and estimate the magnitude for each of the stations.
Note: Not all reported stations really have waveforms for that time span. You can put the `getWaveform()` call inside a `try/except` statement (see Python API) like this to handle the error and avoid the program execution being interrupted:

```
try:
    client.getWaveform(..., station=sta, ...)
except:
    print "problem with station:", sta
    continue
```

Exercise 11 (*Try Program with Events in Vogtland Swarm Region*)
By now, the program is rather flexible. We can now simply switch to a completely different set of events. Modify exc. 10 and change the event request from EMSC to use events in the Vogtland swarm area (roughly 50.2 N, 12.2 E) at the north-eastern Bavarian border. There are two magnitude 4+ events in the EMSC catalog in 2008 that we can use, for instance. Either use one single event like before or add an additional `for` loop going through all requested events.

Exercise 12 (*Organize Magnitude Estimation in New Python Module*)
At the end we can clean up the program we wrote by extracting the magnitude estimation steps in a separate, new Python module. Make a new Python file `mess_exercise_12_module.py` with a function called `def estimate_magnitude(...)`. This function should expect four arguments: *a*) a `Stream` object with Z, N and E traces with attached `PAZ` and coordinate information, *b*) the longitude of the event, *c*) the latitude of the event and *d*) the depth of the event. Move the signal processing steps to this new module and modify exc. 11 to import and use the function `estimate_magnitude(...)`.