Earthquake Detection in Python

Calum J. Chamberlain
GPHS445 - Observational Earthquake Seismology

12 May 2016

We are going to try and detect earthquakes within a known seismically active period based on the GeoNet national seismic catalogue, using data from the GeoNet national seismic network. We want to:

1. Detect earthquakes as well as GeoNet;
2. Try and do better than they do.

And we want to write re-usable, functional Python code. The main emphasis for this work it to write clear pythonic code. It doesn't have to be the fastest, but it should be pretty. Note that earthquake detection is one of the fundamental steps in observational seismology, there are many steps including quality control following this, given time we would refine the parameters we use here to reduce our false detection rate.

Exercise 1. [8 marks] Write a python function that takes as arguments GeoNet eventIDs in a list, and the number of days of data to download, and a start-date. This function should:

1. download the relevant event data (use obspy.clients.fdsn.Client.get_events, store in Catalog object);
2. download the continuous data for the five sites most commonly used (search for get_waveforms in ObsPy, restrict to only download channels EH*, BH* and HH*).

Focus on writing something re-usable.

Exercise 2. [5 marks] Write a script to call your functions for the sequence that contains the mainshock: **2016p275188**; focus on the day of the mainshock (UT time: 2016-04-11).

- Use the obspy network_coincidence trigger to detect events within the continuous data you downloaded. Begin by using the *recstalta* method with trigger on=10, trigger off=1, short-term average length=0.5 s, long-term average length=10, and require triggers on all five stations to declare a detection.
- Compare your triggers to the known GeoNet events (*hint*, write a function that compares the times of your detected events to those that you downloaded, allow up to 5 seconds difference between detection and origin time).

Exercise 3. [7 marks] Select five earthquakes in the sequence to use as templates, write a function to select the best ones (hint, high signal-to-noise ratios are usually good, and signal-to-noise ratios are stored in event.amplitudes.snr) and use eqcorrscan functions to prepare them (eqcorrscan.core.template_gen). Use the eqcorrscan.utils.pre_processing.dayproc function to prepare the data.

Exercise 4. [5 marks] Use eqcorrscan to detect similar events using the match-filter routines, again focus on the day of the mainshock. Use your function from exercise 2 to compare the detections made here with the original GeoNet events.

You will turn in two python codes, one containing your functions, and one written as a script to call the functions. You will also turn in a brief report outlining your results (I just want the comparison numbers and the parameters you used for detections), and a one paragraph explanation of reasoning for the differences in detections.

The following code can be used for working out what event ids you should download.

```
def get_geonet_ids(minlat, maxlat, minlon, maxlon, startdate, enddate):
    """Generate quakesearch URL query and extract publicID from returned csv

    :type minlat: float
    :param minlat: Southern edge of bounding box.
    :type maxlat: float
    :param maxlat: Northern edge of bounding box.
    :type minlon: float
    :param minlon: Western edge of bounding box.
    :type maxlon: float
    :param maxlon: Eastern edge of bounding box.
    :type startdate: obspy.core.UTCDateTime
    :param startdate: Start to search.
    :type enddate: obspy.core.UTCDateTime
    :param enddate: End of search.

    :returns: list of str of event ids
    """
    import csv
    import urllib2

    base_url = "http://quakesearch.geonet.org.nz/services/1.0.0/csv?"
    bbox_url = "bbox=" + ','.join([str(minlon), str(minlat),
                                   str(maxlon), str(maxlat)])
    url = base_url + bbox_url
    if startdate:
        startdate_url = "&startdate=" + startdate.strftime('%Y-%m-%dT%H:%M:%S')
        url += startdate_url
    if enddate:
        enddate_url = "&enddate=" + enddate.strftime('%Y-%m-%dT%H:%M:%S')
        url += enddate_url
    print("Downloading info from:")
    print(url)
    response = urllib2.urlopen(url)
    quake_search = csv.reader(response)

    header = quake_search.next()
    # Usually publicID is the first column, error if not true
    if not header[0] == 'publicid':
        raise IOError('Unexpected format, first column is not publicid')
    event_ids = [row[0] for row in quake_search]
    return event_ids
```