Preprocess_Example

August 31, 2015

0.1 Ambient Noise Waveforms Preprocessing Methods Examples

The following notebook contains examples for using the psprocess.py toolbox for preprocessing raw seismic waveforms to a point where they can be used for cross-correlations. Currently the script can only operate with MSEED formats, but additional support for other formats such as: SAC, SEED and SUDS will likely be added in the future. The user specifies the input path to the raw waveform. This waveform should be only one trace for the purposes of this example, e.g. BHZ. A Preprocess object is created with this input, and the output is specified by one of the functions contained.

The theory for the current workflow that this example follows for preprocessing is explained in depth in Bensen et al. (2007).

In [44]:
```
from pysismo.pspreprocess import Preprocess
from obspy import read
from obspy.core import Stream
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

The Preprocess class requires many input parameters to function. Below is a list of examples.

In [45]:
```
# list of example variables for Preprocess class
FREQMAX = 1./1                              # bandpass parameters
FREQMIN = 1/20.0
CORNERS = 2
ZEROPHASE = True
ONEBIT_NORM = False                         # one-bit normalization
PERIOD_RESAMPLE = 0.02                       # resample period to decimate traces, after band-pa
FREQMIN_EARTHQUAKE = 1/75.0                 # earthquakes periods band
FREQMAX_EARTHQUAKE = 1/25.0
WINDOW_TIME = 0.5 * 1./FREQMAX_EARTHQUAKE # time window to calculate time-normalisation weight
WINDOW_FREQ = 0.0002                        # freq window (Hz) to smooth ampl spectrum

# here is a list of all of the functions and variables that the Preprocess class contains
help(Preprocess)
```

```
Help on class Preprocess in module pysismo.pspreprocess:

class Preprocess
 |  Class for performing all possible preprocess steps on a seismic waveform
 |  to then allow for smooth cross-correlation.
 |
 |  @param freqmin: low frequency of the band-pass filter
 |  @param freqmax: high frequency of the band-pass filter
 |  @param freqmin_earthquake: low frequency of the earthquake band
```

```
|  @param freqmax_earthquake: high frequency of the earthquake band
|  @param corners: nb or corners of the band-pass filter
|  @param zerophase: set to True for filter not to shift phase
|  @type zerophase: bool
|  @param period_resample: resampling period in seconds
|  @param onebit_norm: set to True to apply one-bit normalization (else,
|                      running mean normalization is applied)
|  @type onebit_norm: bool
|  @param window_time: width of the window to calculate the running mean
|                      in the earthquake band (for the time-normalization)
|  @param window_freq: width of the window to calculate the running mean
|                      of the amplitude spectrum (for the
|                      spectral whitening)
|
|  Methods defined here:
|
|  __init__(self, freqmin, freqmax, freqmin_earthquake, freqmax_earthquake, corners, zerophase, period_r
|
|  bandpass_filt(self, trace)
|      Function to apply a butterworth bandpass-filter to an obspy
|      trace input object. Note: MUST only be one trace. No streams.
|
|  get_merged_trace(self, station, date, xcorr_interval, skiplocs=[u'50'], minfill=0.8)
|      Returns one trace extracted from selected station, at selected date
|      (+/- 1 hour on each side to avoid edge effects during subsequent
|      processing).
|
|      for 45 minute xcorr interval, change edge effects by 1 minute!
|
|
|
|      Traces whose location belongs to *skiplocs* are discarded, then
|      if several locations remain, only the first is kept. Finally,
|      if several traces (with the same location) remain, they are
|      merged, WITH GAPS FILLED USING LINEAR INTERPOLATION.
|
|      Raises CannotPreprocess exception if:
|          - no trace remain after discarded the unwanted locations
|          - data fill is < *minfill*
|
|      @type station: L{psstation.Station}
|      @type date: L{datetime.date}
|      @param skiplocs: list of locations to discard in station's data
|      @type skiplocs: iterable
|      @param minfill: minimum data fill to keep trace
|      @rtype: L{Trace}
|
|  get_or_attach_response(self, trace, dataless_inventories=(), xml_inventories=())
|      Returns or attach instrumental response, from dataless seed inventories
|      (as returned by psstation.get_dataless_inventories()) and/or StationXML
|      inventories (as returned by psstation.get_stationxml_inventories()).
|      If a response if found in a dataless inventory, then a dict of poles
|      and zeros is returned. If a response is found in a StationXML
|      inventory, then it is directly attached to the trace and nothing is
```

```
|      returned.
|
|      Raises CannotPreprocess exception if no instrumental response is found.
|
|      @type trace: L{Trace}
|      @param dataless_inventories: inventories from dataless seed files
|                                   (as returned by
|                                    psstation.get_dataless_inventories())
|      @type dataless_inventories: list of L{obspy.xseed.parser.Parser}
|      @param xml_inventories: inventories from StationXML files
|                              (as returned by
|                               psstation.get_stationxml_inventories())
|      @type xml_inventories: list of L{obspy.station.inventory.Inventory}
|
|  preprocess_trace(self, trace, paz=None)
|      Preprocesses a trace (so that it is ready to be cross-correlated),
|      by applying the following steps:
|          - removal of instrument response, mean and trend
|          - band-pass filtering between *freqmin*-*freqmax*
|          - downsampling to *period_resample* secs
|          - time-normalization (one-bit normalization or normalization
|            by the running mean in the earthquake frequency band)
|          - spectral whitening (if running mean normalization)
|
|      Raises CannotPreprocess exception if:
|          - trace only contains 0 (happens sometimes...)
|          - a normalization weight is 0 or NaN
|          - a Nan appeared in trace data
|
|      Note that the processing steps are performed in-place.
|
|      @type trace: L{Trace}
|      @param paz: poles and zeros of instrumental response
|                  (set None if response is directly attached to trace)
|
|  remove_resp(self, trace, paz=None)
|
|  spectral_whitening(self, trace)
|      Function that takes an input obspy trace object that has been
|      time-normalised, band-pass filtered and had its repsonse removed,
|      delimited, demeaned and detrended.
|
|  time_norm(self, trace, trcopy)
|
|  trace_downsample(self, trace)
```

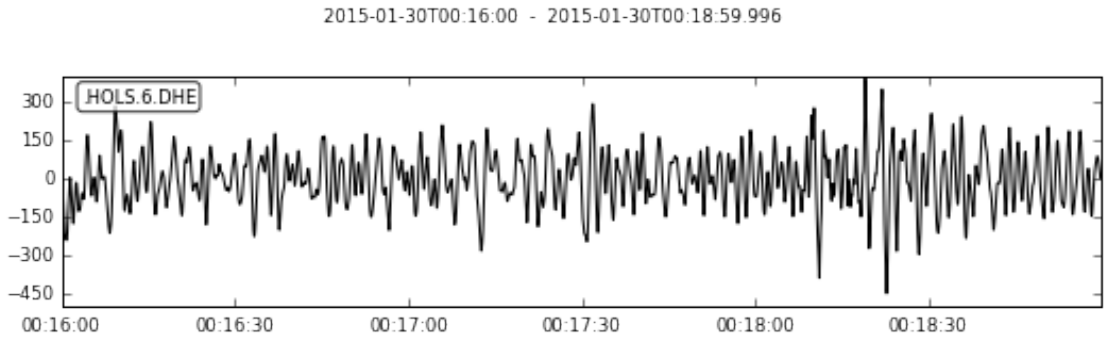
In [46]:
```
# set the path to the desired waveform, the example HOLS.mseed is provided.
example_path = 'tools/examples/HOLS.mseed'

# import a trace from the example waveform
example_trace = read(example_path)[0]
# initialise the Preprocess class
PREPROCESS = Preprocess(FREQMIN, FREQMAX, FREQMIN_EARTHQUAKE,
                        FREQMAX_EARTHQUAKE, CORNERS, ZEROPHASE,
```

```
                                PERIOD_RESAMPLE, WINDOW_TIME, WINDOW_FREQ,
                                ONEBIT_NORM)
```

The following processing examples are in order from Bensen et al. (2007). The final example has all of them combined. - First, the trace has its instrument response removed. - Second, the trace is trimmed, demeaned and detrended. - Third, the trace is passed through a butterworth band-pass filter to remove high amplitude noise and event signals as much as possible. - Fourth, the trace is downsampled to allow for swifter processing. The closer to the original sample rate this is left, the longer overall processing will take! - Fifth, the trace is normalised. This can be specified as either time-normalised or one-bit normalised. - Sixth, the spectrum of the waveform is 'whitened'.

Take note that for the purposes of this example, the instrument response has been kept. This is because the metadata file for this waveform is having some technical difficulties. The resulting waveforms and techniques posed here are still valid for example purposes.

```
In [47]: # process the band-pass filtered trace
         # the bands are set from the above freqmax and freqmin parameters entered when the class is in
         example_trace = PREPROCESS.bandpass_filt(example_trace)
         st = Stream(traces=[example_trace])
         st.plot()
```



Next, downsample the example_trace. The output downsampled trace is dictated by the variable PERIOD_RESAMPLE. The new sample rate is 1/PERIOD_RESAMPLE

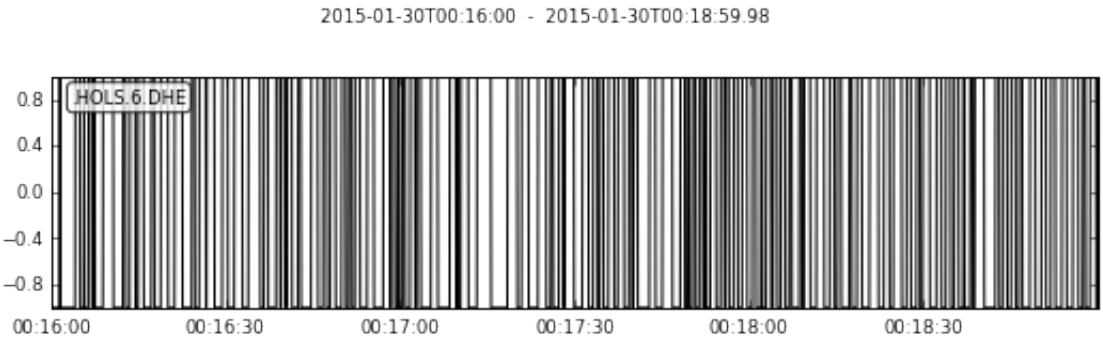
```
In [48]: # Previous trace sample rate:
         print 'Initial trace sample rate: ', example_trace.stats.sampling_rate
         # Downsample trace
         example_trace = PREPROCESS.trace_downsample(example_trace)
         print 'Downsampled trace sample rate: ', example_trace.stats.sampling_rate
```

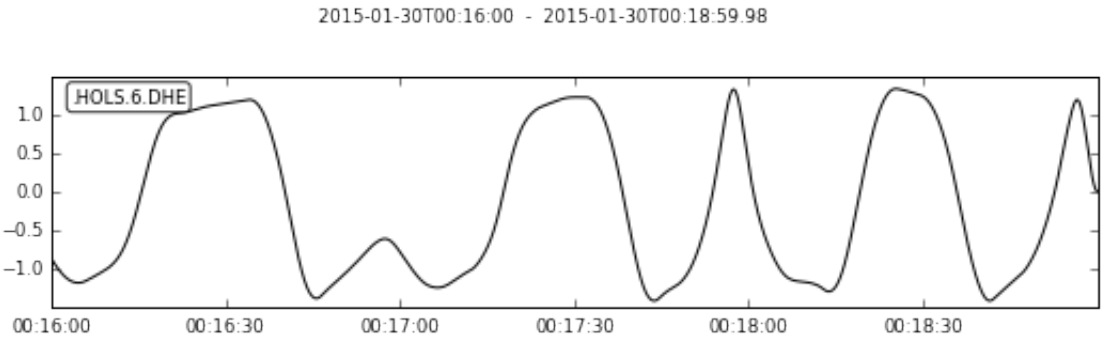
```
Initial trace sample rate:  250.0
<class 'obspy.core.trace.Trace'>
Downsampled trace sample rate:  50.0
```

Normalise the trace, either with respect to time, or the one-bit normalisation procedure.

```
In [49]: example_trace_copy = example_trace
         # one-bit normalization
         example_trace.data = np.sign(example_trace.data)
         st = Stream(traces=[example_trace])
         # plot the one-bit normalised trace
         st.plot()
```

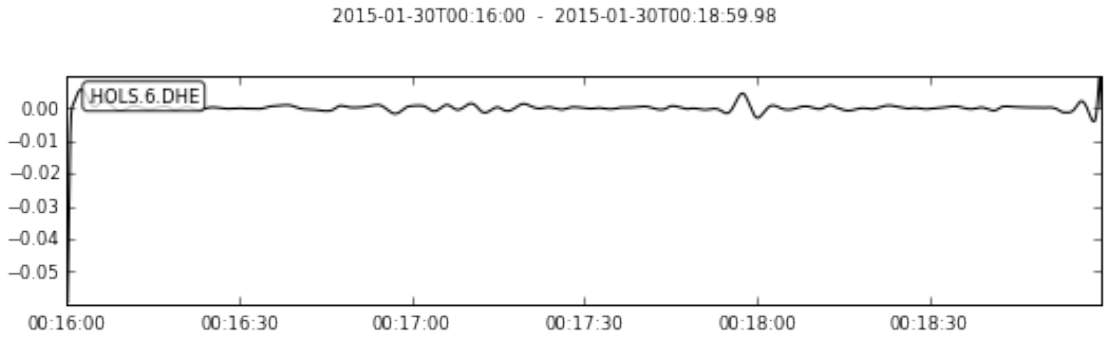
```
# copy the trace for time normalisation
example_trace = example_trace_copy
# process for time normalisation
example_trace = PREPROCESS.time_norm(example_trace, example_trace_copy)
st = Stream(traces=[example_trace])
# plot the time normalised trace
st.plot()
```





Finally spectrally whiten the trace.

In [50]:
```
# process the whitened spectrum for the trace
example_trace = PREPROCESS.spectral_whitening(example_trace)
st = Stream(traces=[example_trace])
# plot the time normalised trace
st.plot()
```



0.2 References

Bensen, G., Ritzwoller, M., Barmin, M., Levshin, A., Lin, F., & Moschetti, M. et al. (2007). Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements. Geophysical Journal International, 169(3), 1239-1260. doi:10.1111/j.1365-246x.2007.03374.x