MNRAS **000**, 1–?? (2020) Preprint 5 October 2020 Compiled using MNRAS LATEX style file v3.0

Seismic attributes

Ross J. Turner[1⋆], Anya Reading[1,2]
[1] *School of Natural Sciences, University of Tasmania, Private Bag 37, Hobart, 7001, Australia*
[2] *Institute for Marine and Antarctic Studies, University of Tasmania, Private Bag 129, Hobart, TAS 7001, Australia*

Accepted XXX. Received YYY; in original form ZZZ.

SEISMIC ATTRIBUTES DOCUMENTATION

This document describes the functions in the *seismic attributes* code and provides examples of accepted usage to download waveforms from online repositories, identify events using one or more seismometers and components, and to derive attributes of the identified signals for use in clustering analyses. The code is written in *python* and imports standard packages including `numpy`, `pandas`, `matplotlib` and `seaborn`, and the specialised package `obspy`. Documentation for private functions is not included in this file.

CONTENTS

get_waveforms

seismic_attributes.get_waveforms(network, station, location, channel, starttime, endtime, event_buffer=3600, waveform_name='waveforms', station_name='stations', client=['IRIS', 'LMU', 'GFZ'], download=True)

******Add the name, observed attributes, and optional fit parameters for a radio AGN to the list of sources.

Parameters:

network : *str*
Select a single SEED or data center defined network code; wildcards are not allowed.

station : *str*
Select a single SEED station code; wildcards are not allowed.

location : *str*
Select a single SEED location identifier; wildcards are not allowed.

channel : *str or list*
Select one or more SEED channel codes. Multiple codes are entered as a list; e.g. ['BHZ', 'HHZ'].

starttime : *UTCDateTime*
Limit results to time series samples on or after the specified start time.

endtime : *UTCDateTime*
Limit results to time series samples on or before the specified end time.

event_buffer : *float, optional*
Specify minimum duration of data to buffer before and after the requested time period; this is used to detect the full length of events that extend beyond the time period. Expected units are seconds.

waveform_name : *str or path, optional*
Specify directory to read and save waveform data. The default location is a folder named *waveforms* in the working directory.

station_name : *str or path, optional*
Specify directory to read and save station data. The default location is a folder named *stations* in the working directory.

client : *str or list, optional*
Specify one or more clients to use to download the requested waveform data if it does not already exist in the specified local directory. Multiple clients are entered as a list; e.g. ['IRIS', 'GFZ'].

download : *bool, optional*
Specify whether days with missing waveform data are to be downloaded from client. Missing data is downloaded by default.

Returns: **stream : *Stream***
Return a stream object with one or more traces for each component at the requested seismometer.

get_events

seismic_attributes.get_events(stream, starttime, endtime, signal_type='amplitude', trigger_type='recstalta', thr_coincidence_sum=-1, thr_event_offset=0.5, thr_on=5, thr_off=1, **options)

******Calculation and calibration of radio continuum redshifts for radio AGNs stored in list. The calibration is only performed if five or more calibrators are included in the list.

Parameters: **stream : *Stream***
Stream containing waveform data for each component from one or more seismometers.

starttime : *UTCDateTime*
Limit results to time series samples on or after the specified start time.

endtime : *UTCDateTime*
Limit results to time series samples on or before the specified end time.

signal_type : *str, optional*
Specify whether event detection algorithm is applied to 'amplitude' (i.e. absolute value) or 'energy' (i.e. amplitude-squared) waveform. The event detection algorithm is applied to the amplitude waveform by default.

trigger_type : *str, optional*
String that specifies which trigger is applied (e.g. 'recstalta'). See e.g. obspy.core.trace.Trace.trigger() for further details. The recursive STA/LTA algorithm is applied by default.

thr_coincidence_sum : *int, optional*
Specify the number of seismometers which an event must be detected for that event to be included in the event list. By default an event must be detected at every seismometer.

thr_event_offset : *float, optional*
Specify the amount to artificially lengthen detected events to remove and join small gaps in triggering of a single events and to handle the difference in signal travel time to proximate seismometers. The events reported in the event list are the actual length.

thr_on : *float, optional*
Threshold for switching single seismometer trigger on.

thr_off : *float, optional*

Threshold for switching single seismometer trigger off.

options

Necessary keyword arguments for the respective trigger that will be passed on. For example 'sta' and 'lta' for any STA/LTA variant (e.g. sta=3, lta=10). Arguments 'sta' and 'lta' (seconds) will be mapped to 'nsta' and 'nlta' (samples) by multiplying with sampling rate of trace. (e.g. sta=3, lta=10 would call the trigger with 3 and 10 seconds average, respectively).

Returns: **events : *DataFrame***

Return pandas dataframe of the events with two columns: the first named 'time' for the start time of the event; and the second named 'duration' for the length of the event. The signal type used in the event detection is included as a third column.

plot_events

seismic_attributes.plot_events(events, stream, starttime, endtime, filename=None)

******Plot of probability density function and Fourier filtered probability density function for radio AGNs in list.

Parameters: **events : *DataFrame***

List of events stored as a pandas dataframe including a column named 'time' for the start time of the event and a column named 'duration' for the length of the event.

stream : *Stream*

Stream containing waveform data for each component from one or more seismometers. The waveform for the first seismometer in the stream is plotted using the signal type (amplitude or energy) specified in the event detection function.

starttime : *UTCDateTime*

Limit results to time series samples on or after the specified start time.

endtime : *UTCDateTime*

Limit results to time series samples on or before the specified end time.

filename : *str or path, optional*

Specify filename and directory to save .pdf document of plots. The default filename is location is the '[station name]_event_plot.pdf' located in the working directory.

Returns: ***None***

get_attributes

seismic_attributes.get_attributes(events, stream, *attributes)

Plot of the correlation between the spectroscopic and radio continuum redshifts for calibrator radio AGNs in list.

Parameters: **events : *DataFrame***

List of events stored as a pandas dataframe including a column named 'time' for the start time of the event and a column named 'duration' for the length of the event.

stream : *Stream*

Stream containing waveform data for each component from one or more seismometers.

attributes

Defined (or user defined) attribute functions to apply to the waveform data. Each function is included separately and comma separated; e.g. get_attributes(..., seismic_attributes.attribute1, user_defined_attribute, seismic_attributes.attribute4).

Returns: **attributes : *DataFrame***

Return pandas dataframe with the attributes of each event. The first two columns named 'start_time' and 'end_time' are for the start and end time of the event; the remaining columns are for the requested attributes and are named as specified in their respective attribute function.

plot_attributes

seismic_attributes.plot_attributes(attributes, attribute_scale=None, filename=None)

******Plot of the correlation between the spectroscopic and radio continuum redshifts for calibrator radio AGNs in list.

Parameters: **attributes : *DataFrame***

List of events and their attributes stored as a pandas dataframe including columns named 'start_time' and 'stop_time' for the start and end time of the event.

attribute_scale : *str or list*

String or list of strings specifying which (if any) attributes need to be plotted using a logarithmic scale; permanent changes to scalings are best made in the attribute functions.

filename : *str or path, optional*

Specify filename and directory to save .pdf plot. The default filename is location is the 'attribute_plot.pdf' located in the working directory.

Returns: ***None***

plot_correlations

seismic_attributes.plot_correlations(attributes, attribute_scale=None, filename=None)

******Plot of the correlation between the spectroscopic and radio continuum redshifts for calibrator radio AGNs in list.

Parameters: **attributes : *DataFrame***

List of events and their attributes stored as a pandas dataframe including columns named 'start_time' and 'stop_time' for the start and end time of the event.

attribute_scale : *str or list*

String or list of strings specifying which (if any) attributes need to be plotted using a logarithmic scale; permanent changes to scalings are best made in the attribute functions.

filename : *str or path, optional*

Specify filename and directory to save .pdf plot. The default filename is location is the 'correlation_plot.pdf' located in the working directory.

Returns: ***None***

ATTRIBUTE FUNCTIONS

The *seismic_attributes* code provides several examples of attributes functions that can be applied directly to waveform data. The general form of the attribute functions is also described to enable the construction of user defined functions.

attribute_1

seismic_attributes.attribute_1(component_stream, event_start, event_stop)

Function to calculate the duration of an event from the waveform data passed on from the get_attributes function.

Parameters: **component_stream : *Stream***

Stream containing waveform data for each component from a single seismometer between the start and end time of the event.

event_start : *UTCDateTime*

Limit results to time series samples on or after the specified start time; the stream is already truncated to this range in the get_attributes function.

event_stop : *UTCDateTime*

Limit results to time series samples on or before the specified end time; the stream is already truncated to this range in the get_attributes function.

Returns: **attribute : *tuple***

Return a tuple containing a the name of the attribute and its value; i.e. 'attribute_1' as a string and the duration of the event as a float.

attribute_4

seismic_attributes.attribute_4(component_stream, event_start, event_stop)

Function to calculate the ratio between ascending and descending time of an event from the waveform data passed on from the get_attributes function.

Parameters: **component_stream : *Stream***

Stream containing waveform data for each component from a single seismometer between the start and end time of the event.

event_start : *UTCDateTime*

Limit results to time series samples on or after the specified start time; the stream is already truncated to this range in the get_attributes function.

event_stop : *UTCDateTime*

Limit results to time series samples on or before the specified end time; the stream is already truncated to this range in the get_attributes function.

Returns: **attribute : *tuple***

Return a tuple containing a the name of the attribute and its value; i.e. 'attribute_4' as a string and the ratio between ascending and descending time of the event as a float.

attribute_10_11_12

seismic_attributes.attribute_10_11_12(component_stream, event_start, event_stop)

Function to calculate the energy in the first third and the remaining part of the autocorrelation function of an event from the waveform data passed on from the get_attributes function.

Parameters: **component_stream : *Stream***

Stream containing waveform data for each component from a single seismometer between the start and end time of the event.

event_start : *UTCDateTime*

Limit results to time series samples on or after the specified start time; the stream is already truncated to this range in the get_attributes function.

event_stop : *UTCDateTime*

Limit results to time series samples on or before the specified end time; the stream is already truncated to this range in the get_attributes function.

Returns: **attribute : *tuple***

Return a tuple containing the names of the attributes and their values; i.e. ['attribute_10', 'attribute_11', 'attribute_12'] as a list of strings and the energy in the first third and the remaining part of the autocorrelation function of the event as a list of floats.

attribute_13_14_15_17

seismic_attributes.attribute_13_14_15_17(component_stream, event_start, event_stop)

Function to calculate the energy filtered in 5 – 10 Hz, 10 – 50 Hz, 5 – 70 Hz and 1 – 10 Hz of an event from the waveform data passed on from the get_attributes function.

Parameters: **component_stream : *Stream***

Stream containing waveform data for each component from a single seismometer between the start and end time of the event.

event_start : *UTCDateTime*

Limit results to time series samples on or after the specified start time; the stream is already truncated to this range in the get_attributes function.

event_stop : *UTCDateTime*

Limit results to time series samples on or before the specified end time; the stream is already truncated to this range in the get_attributes function.

Returns: **attribute : *tuple***

Return a tuple containing the names of the attributes and their values; i.e. ['attribute_13', 'attribute_14', 'attribute_15', 'attribute_17'] as a list of strings and the energy filtered in 5 – 10 Hz, 10 – 50 Hz, 5 – 70 Hz and 1 – 10 Hz of the event as a list of floats.

attribute_68_69_70_71

seismic_attributes.attribute_68_69_70_71(component_stream, event_start, event_stop)

Function to calculate the rectilinearity, azimuth, dip and planarity of an event from the waveform data passed on from the get_attributes function.

Parameters: **component_stream : *Stream***

Stream containing waveform data for each component from a single seismometer between the start and end time of the event.

event_start : *UTCDateTime*

Limit results to time series samples on or after the specified start time; the stream is already truncated to this range in the get_attributes function.

event_stop : *UTCDateTime*

Limit results to time series samples on or before the specified end time; the stream is already truncated to this range in the get_attributes function.

Returns: **attribute : *tuple***

Return a tuple containing the names of the attributes and their values; i.e. ['attribute_68', 'attribute_69', 'attribute_70', 'attribute_71'] as a list of strings and the rectilinearity, azimuth, dip and planarity of the event as a list of floats.

group_components

seismic_attributes.group_components(stream, signal_type='amplitude')

Function to calculate the normal of the waveform amplitude for multiple component measurements. The normal can be returned as an absolute value amplitude waveform or an energy waveform.

Parameters: **stream : *Stream***

Stream containing waveform data for each component from one or more seismometers between the start and end time of the event.

signal_type : *str, optional*

Specify whether components are grouped as an 'amplitude' (i.e. absolute value) or 'energy' (i.e. amplitude-squared) waveform. The components are grouped as an amplitude waveform by default.

Returns: **stream_list : *list***

Return a list of streams for each seismometer with the data representing the amplitude or energy of the waveform measured by taking the normal of the signal from each component. The first stream is accessed as group_components(...)[0] and the trace of that stream as group_components(...)[0][0].

Examples

Create an empty list to add observed attributes of each radio AGNs.

```
>>> import RAiSERed as rr
>>> raisered_list = rr.RAiSERed_list()
```

Add observed attributes for radio AGNs with known spectroscopic redshifts (minimum of five for calibration). Different accepted usage of the input parameters has been included for each source.

```
>>> rr.RAiSERed_add(raisered_list, 'CygnusAE', 0.151, [5960, 450], [58.6, 0.4], 2.8,
        [2.485, 0.009], [9.243, 0.017], specz=0.056075)
>>> rr.RAiSERed_add(raisered_list, '3C20E', 0.178, [21.64, 0.13], [24.55, 0.22], 5.04,
        [2.222, 0.001], [9.759, 0.001], specz=0.174, calib=True)
>>> rr.RAiSERed_add(raisered_list, '3C244.1N', 1.78e8, [12.95, 0.65], [28.45, 0.40], 10.00,
        [2.562, 0.005], [9.930, 0.022], specz=0.428)
>>> rr.RAiSERed_add(raisered_list, 'PKS0529-549', 1.51e8, [2.78, 0.22], [0.6, 0.33],
        [1.6, 5.0, 1000], [2.474, 0.005], [9.160, 0.014], specz=2.57)
>>> rr.RAiSERed_add(raisered_list, 'USS1243+036', 151, [2.23, 0.18], [3, 0.115], [6.9, 0.8],
        [2.739, 0.007], [9.005, 0.014], specz=3.57, calib=True)
```

Add observed attributes for additional radio AGNs with no known spectroscopic redshifts.

```
>>> rr.RAiSERed_add(raisered_list, '3C388W', 178, [14.75, 0.74], [20.5, 0.8], 2.94,
        [2.318, 0.004], [9.723, 0.015], calib=False)
>>> rr.RAiSERed_add(raisered_list, 'USS1707+105', 0.151, [1.44, 0.12], [11.25, 0.115],
        [13.5, 6.25, -1000], [2.528, 0.008], [8.963, 0.016])
```

Calculate redshift probability density function using optimal calibration from RAiSERed manuscript.

```
>>> rr.RAiSERed(raisered_list, b1=0.76, b2=0.01, b3=-0.60, b4=-0.33)
```

Plot correlation between spectroscopic and radio continuum redshifts for calibrators (though we have used a user defined calibration above).

```
>>> rr.RAiSERed_zzplot(raisered_list)
```

Plot redshift probability density functions for all radio AGNs in the list, or just USS1707+105.

```
>>> rr.RAiSERed_plot(raisered_list)
```

```
>>> rr.RAiSERed_plot(raisered_list['USS1707+105'])
```

Find radio continuum redshift and standard deviation for USS1707+105.

```
>>> print(raisered_list['USS1707+105'].mean, raisered_list['USS1707+105'].stdev)
```

Calibrate the RAiSERed model using the calibrators in the list and calculate the redshift probability density function. This may take quite some time to run if numerous calibrators are used.

```
>>> b1, b2, b3, b4 = rr.RAiSERed(raisered_list)
```

Plots of the redshift probability density functions and correlations between the spectroscopic and radio continuum redshifts are made using the `RAiSERed_plot()` and `RAiSERed_zzplot()` functions as before.