sprit

Release 1.0.1

Riley Balikian

Apr 25, 2024

CONTENTS:

CHAPTER

ONE

SPRIT PACKAGE

This module analysis ambient seismic data using the Horizontal to Vertical Spectral Ratio (HVSR) technique

class sprit.**HVSRBatch**(**args*, ***kwargs*)

Bases: `object`

HVSRBatch is the data container used for batch processing. It contains several HVSRData objects (one for each site). These can be accessed using their site name, either square brackets (HVSRBatchVariable["SiteName"]) or the dot (HVSRBatchVariable.SiteName) accessor.

The dot accessor may not work if there is a space in the site name.

All of the functions in the sprit.pacakge are designed to perform the bulk of their operations iteratively on the individual HVSRData objects contained in the HVSRBatch object, and do little with the HVSRBatch object itself, besides using it determine which sites are contained within it.

Methods

copy([type])	Make a copy of the HVSRBatch object.
export([export_path, ext])	Method to export HVSRData objects in HVSRBatch container to indivdual .hvsr pickle files.
export_settings([site_name, ...])	Method to export settings from HVSRData object in HVSRBatch object.
get_report(**kwargs)	Method to get report from processed data, in print, graphical, or tabular format.
items()	Method to return both the site names and the HVSRData object as a set of dict_items tuples.
keys()	Method to return the "keys" of the HVSRBatch object.
plot(**kwargs)	Method to plot data, based on the sprit.plot_hvsr() function.
report(**kwargs)	Wrapper of get_report()

copy(*type='shallow'*)

Make a copy of the HVSRBatch object. Uses python copy module.

Parameters

type
[str {'shallow', 'deep'}] Based on input, creates either a shallow or deep copy of the HVSRBatch object. Shallow is equivalent of copy.copy(). Input of 'deep' is equivalent of copy.deepcopy() (still experimental). Defaults to shallow.

export(*export_path=True*, *ext='hvsr'*)

Method to export HVSRData objects in HVSRBatch container to indivdual .hvsr pickle files.

Parameters

export_path
[filepath, default=True] Filepath to save file. Can be either directory (which will assign a filename based on the HVSRData attributes). By default True. If True, it will first try to save each file to the same directory as datapath, then if that does not work, to the current working directory, then to the user's home directory, by default True

ext
[str, optional] The extension to use for the output, by default 'hvsr'. This is still a pickle file that can be read with pickle.load(), but will have .hvsr extension.

export_settings(*site_name=None*, *export_settings_path='default'*, *export_settings_type='all'*, *include_location=False*, *verbose=True*)

Method to export settings from HVSRData object in HVSRBatch object.

Simply calls sprit.export_settings() from specified HVSRData object in the HVSRBatch object. See sprit.export_settings() for more details.

Parameters

site_name
[str, default=None] The name of the site whose settings should be exported. If None, will default to the first site, by default None.

export_settings_path
[str, optional] Filepath to output file. If left as 'default', will save as the default value in the resources directory. If that is not possible, will save to home directory, by default 'default'

export_settings_type
[str, {'all', 'instrument', 'processing'}, optional] They type of settings to save, by default 'all'

include_location
[bool, optional] Whether to include the location information in the instrument settings, if that settings type is selected, by default False

verbose
[bool, optional] Whether to print output (filepath and settings) to terminal, by default True

See also:

export_settings

get_report(***kwargs*)

Method to get report from processed data, in print, graphical, or tabular format.

Returns

Variable
May return nothing, pandas.Dataframe, or pyplot Figure, depending on input.

See also:

get_report

items()

Method to return both the site names and the HVSRData object as a set of dict_items tuples. Functions similar to dict.items().

Returns

type
description

keys()

Method to return the "keys" of the HVSRBatch object. For HVSRBatch objects, these are the site names. Functions similar to dict.keys().

Returns

dict_keys
A dict_keys object listing the site names of each of the HVSRData objects contained in the HVSRBatch object

plot(***kwargs*)

Method to plot data, based on the sprit.plot_hvsr() function.

All the same kwargs and default values apply as plot_hvsr(). For return_fig, returns it to the 'Plot_Report' attribute of each HVSRData object

Returns

type
description

See also:

plot_hvsr

report(***kwargs*)

Wrapper of get_report()

See also:

get_report

class sprit.**HVSRData**(**args*, ***kwargs*)

Bases: object

HVSRData is the basic data class of the sprit package. It contains all the processed data, input parameters, and reports.

These attributes and objects can be accessed using square brackets or the dot accessor. For example, to access the site name, HVSRData['site'] and HVSRData.site will both return the site name.

Some of the methods that work on the HVSRData object (e.g., .plot() and .get_report()) are essentially wrappers for some of the main sprit package functions (sprit.plot_hvsr() and sprit.get_report(), respectively)

Attributes

batch
Whether this HVSRData object is part of an HVSRBatch object.

datastream
A copy of the original obspy datastream read in.

params
Dictionary containing the parameters used to process the data

ppsds
Dictionary copy of the class object obspy.signal.spectral_estimation.PPSD().

ppsds_obspy
The original ppsd information from the obspy.signal.spectral_estimation.PPSD(), so as to keep original if copy is manipulated/changed.

Methods

copy([type])	Make a copy of the HVSRData object.
export([export_path, ext])	Method to export HVSRData objects to .hvsr pickle files.
export_settings([export_settings_path, ...])	Method to export settings from HVSRData object.
get_report(**kwargs)	Method to get report from processed data, in print, graphical, or tabular format.
items()	Method to return the "items" of the HVSRData object.
keys()	Method to return the "keys" of the HVSRData object.
plot(**kwargs)	Method to plot data, wrapper of sprit.plot_hvsr()
report(**kwargs)	Wrapper of get_report()

property batch

Whether this HVSRData object is part of an HVSRBatch object. This is used throughout the code to help direct the object into the proper processing pipeline.

Returns

bool
True if HVSRData object is part of HVSRBatch object, otherwise, False

copy(*type='shallow'*)

Make a copy of the HVSRData object. Uses python copy module.

Parameters

type
[str {'shallow', 'deep'}] Based on input, creates either a shallow or deep copy of the HVSRData object. Shallow is equivalent of copy.copy(). Input of type='deep' is equivalent of copy.deepcopy() (still experimental). Defaults to shallow.

property datastream

A copy of the original obspy datastream read in. This helps to retain the original data even after processing is carried out.

Returns

obspy.core.Stream.stream
Obspy stream

export(*export_path=None*, *ext='hvsr'*)

Method to export HVSRData objects to .hvsr pickle files.

Parameters

export_path
[filepath, default=True] Filepath to save file. Can be either directory (which will assign a filename based on the HVSRData attributes). By default True. If True, it will first try to save each file to the same directory as datapath, then if that does not work, to the current working directory, then to the user's home directory, by default True

ext
[str, optional] The extension to use for the output, by default 'hvsr'. This is still a pickle file that can be read with pickle.load(), but will have .hvsr extension.

`export_settings`(*export_settings_path='default'*, *export_settings_type='all'*, *include_location=False*, *verbose=True*)

Method to export settings from HVSRData object. Simply calls sprit.export_settings() from the HVSRData object. See sprit.export_settings() for more details.

Parameters

export_settings_path
[str, optional] Filepath to output file. If left as 'default', will save as the default value in the resources directory. If that is not possible, will save to home directory, by default 'default'

export_settings_type
[str, {'all', 'instrument', 'processing'}, optional] They type of settings to save, by default 'all'

include_location
[bool, optional] Whether to include the location information in the instrument settings, if that settings type is selected, by default False

verbose
[bool, optional] Whether to print output (filepath and settings) to terminal, by default True

`get_report`(***kwargs*)

Method to get report from processed data, in print, graphical, or tabular format.

Returns

Variable
May return nothing, pandas.Dataframe, or pyplot Figure, depending on input.

See also:

get_report

`items`()

Method to return the "items" of the HVSRData object. For HVSRData objects, this is a dict_items object with the keys and values in tuples. Functions similar to dict.items().

Returns

dict_items
A dict_items object of the HVSRData objects attributes, parameters, etc.

`keys`()

Method to return the "keys" of the HVSRData object. For HVSRData objects, these are the attributes and parameters of the object. Functions similar to dict.keys().

Returns

dict_keys
A dict_keys object of the HVSRData objects attributes, parameters, etc.

property params

Dictionary containing the parameters used to process the data

Returns

dict
Dictionary containing the process parameters

plot(***kwargs*)

Method to plot data, wrapper of sprit.plot_hvsr()

Returns

matplotlib.Figure, matplotlib.Axis (if return_fig=True)

See also:

plot_hvsr
plot_azimuth

property ppsds

Dictionary copy of the class object obspy.signal.spectral_estimation.PPSD(). The dictionary copy allows manipulation of the data in PPSD, whereas that data cannot be easily manipulated in the original Obspy object.

Returns

dict
Dictionary copy of the PPSD information from generate_ppsds()

property ppsds_obspy

The original ppsd information from the obspy.signal.spectral_estimation.PPSD(), so as to keep original if copy is manipulated/changed.

report(***kwargs*)

Wrapper of get_report()

See also:

get_report

sprit.**assert_check**(*var*, *cond=None*, *var_type=None*, *error_message='Output not valid'*, *verbose=False*)

sprit.**batch_data_read**(*input_data*, *batch_type='table'*, *param_col=None*, *batch_params=None*, *verbose=False*, ***readcsv_getMeta_fetch_kwargs*)

Function to read data in data as a batch of multiple data files. This is best used through sprit.fetch_data(*args, source='batch', **other_kwargs).

Parameters

input_data
[filepath or list] Input data information for how to read in data as batch

batch_type
[str, optional] Type of batch read, only 'table' and 'filelist' accepted. If 'table', will read data from a file read in using pandas.read_csv(), by default 'table'

param_col
[None or str, optional] Name of parameter column from batch information file. Only used

if a batch_type='table' and single parameter column is used, rather than one column per parameter (for single parameter column, parameters are formatted with = between keys/values and , between item pairs), by default None

batch_params
[list, dict, or None, default = None] Parameters to be used if batch_type='filelist'. If it is a list, needs to be the same length as input_data. If it is a dict, will be applied to all files in input_data and will combined with extra keyword arguments caught by **readcsv_getMeta_fetch_kwargs.

verbose
[bool, optional] Whether to print information to terminal during batch read, by default False

****readcsv_getMeta_fetch_kwargs**
Keyword arguments that will be read into pandas.read_csv(), sprit.input_params, sprit.get_metadata(), and/or sprit.fetch_data()

Returns

dict
Dictionary with each item representing a different file read in, and which consists of its own parameter dictionary to be used by the rest of the processing steps

Raises

IndexError
description

sprit.**calculate_azimuth**(*hvsr_data*, *azimuth_angle=30*, *azimuth_type='multiple'*, *azimuth_unit='degrees'*, *show_az_plot=False*, *verbose=False*, ***plot_azimuth_kwargs*)

Function to calculate azimuthal horizontal component at specified angle(s). Adds each new horizontal component as a radial component to obspy.Stream object at hvsr_data['stream']

Parameters

hvsr_data
[HVSRData] Input HVSR data

azimuth_angle
[int, default=10] If *azimuth_type='multiple'*, this is the angular step (in unit *azimuth_unit*) of each of the azimuthal measurements. If *azimuth_type='single'* this is the angle (in unit *azimuth_unit*) of the single calculated azimuthal measruement. By default 10.

azimuth_type
[str, default='multiple'] What type of azimuthal measurement to make, by default 'multiple'. If 'multiple' (or {'multi', 'mult', 'm'}), will take a measurement at each angular step of azimuth_angle of unit azimuth_unit. If 'single' (or {'sing', 's'}), will take a single azimuthal measurement at angle specified in azimuth_angle.

azimuth_unit
[str, default='degrees'] Angular unit used to specify *azimuth_angle* parameter. By default 'degrees'. If 'degrees' (or {'deg', 'd'}), will use degrees. If 'radians' (or {'rad', 'r'}), will use radians.

show_az_plot
[bool, default=False] Whether to show azimuthal plot, by default False.

verbose
[bool, default=False] Whether to print terminal output, by default False

Returns

HVSRData
Updated HVSRData object specified in hvsr_data with hvsr_data['stream'] attribute containing additional components (EHR-*), with * being zero-padded (3 digits) azimuth angle in degrees.

sprit.**catch_errors**(*func*)

sprit.**check_gui_requirements**()

sprit.**check_mark**(*incolor=False*, *interminal=False*)

The default Windows terminal is not able to display the check mark character correctly. This function returns another displayable character if platform is Windows

sprit.**check_peaks**(*hvsr_data*, *hvsr_band=[0.4, 40]*, *peak_selection='max'*, *peak_freq_range=[0.4, 40]*, *azimuth='HV'*, *verbose=False*)

Function to run tests on HVSR peaks to find best one and see if it passes quality checks

Parameters

hvsr_data
[dict] Dictionary containing all the calculated information about the HVSR data (i.e., hvsr_out returned from process_hvsr)

hvsr_band
[tuple or list, default=[0.4, 40]] 2-item tuple or list with lower and upper limit of frequencies to analyze

peak_selection
[str or numeric, default='max'] How to select the "best" peak used in the analysis. For peak_selection="max" (default value), the highest peak within peak_freq_range is used. For peak_selection='scored', an algorithm is used to select the peak based in part on which peak passes the most SESAME criteria. If a numeric value is used (e.g., int or float), this should be a frequency value to manually select as the peak of interest.

peak_freq_range
[tuple or list, default=[0.4, 40];] The frequency range within which to check for peaks. If there is an HVSR curve with multiple peaks, this allows the full range of data to be processed while limiting peak picks to likely range.

verbose
[bool, default=False] Whether to print results and inputs to terminal.

Returns

hvsr_data
[HVSRData or HVSRBatch object] Object containing previous input data, plus information about peak tests

sprit.**check_tsteps**(*hvsr_data*)

Check time steps of PPSDS to make sure they are all the same length

sprit.**check_xvalues**(*ppsds*)

Check x_values of PPSDS to make sure they are all the same length

sprit.**checkifpath**(*filepath*, *sample_list=''*, *verbose=False*)

Support function to check if a filepath is a pathlib.Path object and tries to convert if not

Parameters

filepath
[str or pathlib.Path, or anything] Filepath to check. If not a valid filepath, will not convert and raises error

Returns

filepath
[pathlib.Path] pathlib.Path of filepath

sprit.**create_jupyter_ui**()

sprit.**export_data**(*hvsr_data*, *export_path=None*, *ext='hvsr'*, *verbose=False*)

Export data into pickle format that can be read back in using import_data() so data does not need to be processed each time. Default extension is .hvsr but it is still a pickled file that can be read in using pickle.load().

Parameters

hvsr_data
[HVSRData or HVSRBatch] Data to be exported

export_path
[str or filepath object, default = None] String or filepath object to be read by pathlib.Path() and/or a with open(export_path, 'wb') statement. If None, defaults to input datapath directory, by default None

ext
[str, default = 'hvsr'] Filepath extension to use for data file, by default 'hvsr'

sprit.**export_settings**(*hvsr_data*, *export_settings_path='default'*, *export_settings_type='all'*, *include_location=False*, *verbose=True*)

Save settings to json file

Parameters

export_settings_path
[str, default="default"] Where to save the json file(s) containing the settings, by default 'default'. If "default," will save to sprit package resources. Otherwise, set a filepath location you would like for it to be saved to. If 'all' is selected, a directory should be supplied. Otherwise, it will save in the directory of the provided file, if it exists. Otherwise, defaults to the home directory.

export_settings_type
[str, {'all', 'instrument', 'processing'}] What kind of settings to save. If 'all', saves all possible types in their respective json files. If 'instrument', save the instrument settings to their respective file. If 'processing', saves the processing settings to their respective file. By default 'all'

include_location
[bool, default=False, input CRS] Whether to include the location parametersin the exported settings document.This includes xcoord, ycoord, elevation, elev_unit, and input_crs

verbose
[bool, default=True] Whether to print outputs and information to the terminal

sprit.**fetch_data**(*params*, *source='file'*, *trim_dir=None*, *export_format='mseed'*, *detrend='spline'*, *detrend_order=2*, *update_metadata=True*, *plot_input_stream=False*, *verbose=False*, ***kwargs*)

Fetch ambient seismic data from a source to read into obspy stream

Parameters

params
[dict]

Dictionary containing all the necessary params to get data.
Parameters defined using input_params() function.

source
[str, {'raw', 'dir', 'file', 'batch'}]

String indicating where/how data file was created. For example, if raw data, will need to find correct channels.
'raw' finds raspberry shake data, from raw output copied using scp directly from Raspberry Shake, either in folder or subfolders; 'dir' is used if the day's 3 component files (currently Raspberry Shake supported only) are all 3 contained in a directory by themselves. 'file' is used if the params['datapath'] specified in input_params() is the direct filepath to a single file to be read directly into an obspy stream. 'batch' is used to read a list or specified set of seismic files.

Most commonly, a csv file can be read in with all the parameters. Each row in the csv is a separate file. Columns can be arranged by parameter.

trim_dir
[None or str or pathlib obj, default=None] If None (or False), data is not trimmed in this function. Otherwise, this is the directory to save trimmed and exported data.

export_format: str='mseed'
If trim_dir is not None, this is the format in which to save the data

detrend
[str or bool, default='spline'] If False, data is not detrended. Otherwise, this should be a string accepted by the type parameter of the obspy.core.trace.Trace.detrend method: https://docs.obspy.org/packages/autogen/obspy.core.trace.Trace.detrend.html

detrend_order
[int, default=2] If detrend parameter is 'spline' or 'polynomial', this is passed directly to the order parameter of obspy.core.trace.Trace.detrend method.

update_metadata
[bool, default=True] Whether to update the metadata file, used primarily with Raspberry Shake data which uses a generic inventory file.

plot_input_stream
[bool, default=False] Whether to plot the raw input stream. This plot includes a spectrogram (Z component) and the raw (with decimation for speed) plots of each component signal.

verbose
[bool, default=False] Whether to print outputs and inputs to the terminal

****kwargs**
Keywords arguments, primarily for 'batch' and 'dir' sources

Returns

params
[HVSRData or HVSRBatch object] Same as params parameter, but with an additional "stream" attribute with an obspy data stream with 3 traces: Z (vertical), N (North-south), and E (East-west)

sprit.**format_time**(*inputDT*, *tzone='UTC'*)

Private function to format time, used in other functions

Formats input time to datetime objects in utc

Parameters

inputDT
[str or datetime obj] Input datetime. Can include date and time, just date (time inferred to be 00:00:00.00) or just time (if so, date is set as today)

tzone
[str='utc' or int {'utc', 'local'}]

Timezone of data entry.
If string and not utc, assumed to be timezone of computer running the process. If int, assumed to be offset from UTC (e.g., CST in the United States is -6; CDT in the United States is -5)

Returns

outputTimeObj
[datetime object in UTC] Output datetime.datetime object, now in UTC time.

sprit.**generate_ppsds**(*hvsr_data*, *azimuthal_ppsds=False*, *verbose=False*, ***ppsd_kwargs*)

Generates PPSDs for each channel

Channels need to be in Z, N, E order Info on PPSD creation here: https://docs.obspy.org/packages/autogen/obspy.signal.spectral_estimation.PPSD.html

Parameters

hvsr_data
[dict, HVSRData object, or HVSRBatch object] Data object containing all the parameters and other data of interest (stream and paz, for example)

azimuthal_ppsds
[bool, default=False] Whether to generate PPSDs for azimuthal data

verbose
[bool, default=True] Whether to print inputs and results to terminal

****ppsd_kwargs**
[dict] Dictionary with keyword arguments that are passed directly to obspy.signal.PPSD. If the following keywords are not specified, their defaults are amended in this function from the obspy defaults for its PPSD function. Specifically:

- ppsd_length defaults to 60 (seconds) here instead of 3600
- skip_on_gaps defaults to True instead of False
- period_step_octaves defaults to 0.03125 instead of 0.125

Returns

ppsds
[HVSRData object] Dictionary containing entries with ppsds for each channel

sprit.**get_char**(*in_char*)

Outputs character with proper encoding/decoding

sprit.**get_metadata**(*params*, *write_path=''*, *update_metadata=True*, *source=None*, ***read_inventory_kwargs*)

Get metadata and calculate or get paz parameter needed for PPSD

Parameters

params
[dict]

Dictionary containing all the input and other parameters needed for processing
Ouput from input_params() function

write_path
[str]

String with output filepath of where to write updated inventory or metadata file
If not specified, does not write file

update_metadata
[bool] Whether to update the metadata file itself, or just read as-is. If using provided raspberry shake metadata file, select True.

source
[str, default=None] This passes the source variable value to _read_RS_metadata. It is expected that this is passed directly from the source parameter of sprit.fetch_data()

Returns

params
[dict] Modified input dictionary with additional key:value pair containing paz dictionary (key = "paz")

sprit.**get_report**(*hvsr_results*, *report_format=['print', 'csv', 'plot']*, *plot_type='HVSR p ann C+ p ann Spec'*, *azimuth='HV'*, *export_path=None*, *csv_overwrite_opt='append'*, *no_output=False*, *verbose=False*)

Get a report of the HVSR analysis in a variety of formats.

Parameters

hvsr_results
[dict] Dictionary containing all the information about the processed hvsr data

report_format
[{'csv', 'print', plot}] Format in which to print or export the report. The following report_formats return the following items in the following attributes:

- 'plot': hvsr_results['Print_Report'] as a str str
- 'print': hvsr_results['HV_Plot'] - matplotlib.Figure object
- **'csv': hvsr_results['CSV_Report']- pandas.DataFrame object**
  - list/tuple - a list or tuple of the above objects, in the same order they are in the report_format list

plot_type
[str, default = 'HVSR p ann C+ p ann Spec] What type of plot to plot, if 'plot' part of report_format input

azimuth
[str, default = 'HV'] Which azimuth to plot, by default "HV" which is the main "azimuth" combining the E and N components

export_path
[None, bool, or filepath, default = None] If None or False, does not export; if True, will export to same directory as the datapath parameter in the input_params() function. Otherwise, it should be a string or path object indicating where to export results. May be a file or directory. If a directory is specified, the filename will be "<site_name>_<acq_date>_<UTC start time>-<UTC end time>". The suffix defaults to png for report_format="plot", csv for 'csv', and does not export if 'print.'

csv_overwrite_opts
[str, {'append', 'overwrite', 'keep/rename'}] How to handle csv report outputs if the designated csv output file already exists. By default, appends the new information to the end of the existing file.

no_output
[bool, default=False] If True, only reads output to appropriate attribute of data class (ie, print does not print, only reads text into variable). If False, performs as normal.

verbose
[bool, default=True] Whether to print the results to terminal. This is the same output as report_format='print', and will not repeat if that is already selected

Returns

sprit.HVSRData

sprit.**gui**(*kind='default'*)

Function to open a graphical user interface (gui)

Parameters

kind
[str, optional] What type of gui to open. "default" opens regular windowed interface, "widget" opens jupyter widget' "lite" open lite (pending update), by default 'default'

sprit.**has_required_channels**(*stream*)

sprit.**import_data**(*import_filepath*, *data_format='pickle'*)

Function to import .hvsr (or other extension) data exported using export_data() function

Parameters

import_filepath
[str or path object] Filepath of file created using export_data() function. This is usually a pickle file with a .hvsr extension

data_format
[str, default='pickle'] Type of format data is in. Currently, only 'pickle' supported. Eventually, json or other type may be supported, by default 'pickle'.

Returns

HVSRData or HVSRBatch object

sprit.**import_settings**(*settings_import_path*, *settings_import_type='instrument'*, *verbose=False*)

sprit.**input_params**(*datapath*, *site='HVSR Site'*, *network='AM'*, *station='RAC84'*, *loc='00'*, *channels=['EHZ', 'EHN', 'EHE']*, *acq_date='2024-04-25'*, *starttime=UTCDateTime(2024, 4, 25, 0, 0)*, *endtime=UTCDateTime(2024, 4, 25, 23, 59, 59, 999999)*, *tzone='UTC'*, *xcoord=-88.2290526*, *ycoord=40.1012122*, *elevation=755*, *input_crs='EPSG:4326'*, *output_crs='EPSG:4326'*, *elev_unit='feet'*, *depth=0*, *instrument='Raspberry Shake'*, *metapath=None*, *hvsr_band=[0.4, 40]*, *peak_freq_range=[0.4, 40]*, *processing_parameters={}*, *verbose=False*)

Function for designating input parameters for reading in and processing data

Parameters

datapath
[str or pathlib.Path object] Filepath of data. This can be a directory or file, but will need to match with what is chosen later as the source parameter in fetch_data()

site
[str, default="HVSR Site"] Site name as designated by user for ease of reference. Used for plotting titles, filenames, etc.

network
[str, default='AM'] The network designation of the seismometer. This is necessary for data from Raspberry Shakes. 'AM' is for Amateur network, which fits Raspberry Shakes.

station
[str, default='RAC84'] The station name of the seismometer. This is necessary for data from Raspberry Shakes.

loc
[str, default='00'] Location information of the seismometer.

channels
[list, default=['EHZ', 'EHN', 'EHE']] The three channels used in this analysis, as a list of strings. Preferred that Z component is first, but not necessary

acq_date
[str, int, date object, or datetime object] If string, preferred format is 'YYYY-MM-DD'. If int, this will be interpreted as the time_int of year of current year (e.g., 33 would be Feb 2 of current year) If date or datetime object, this will be the date. Make sure to account for time change when converting to UTC (if UTC is the following time_int, use the UTC time_int).

starttime
[str, time object, or datetime object, default='00:00:00.00'] Start time of data stream. This is necessary for Raspberry Shake data in 'raw' form, or for trimming data. Format can be either 'HH:MM:SS.micros' or 'HH:MM' at minimum.

endtime
[str, time obejct, or datetime object, default='23:59:99.99'] End time of data stream. This is necessary for Raspberry Shake data in 'raw' form, or for trimming data. Same format as starttime.

tzone
[str or int, default = 'UTC'] Timezone of input data. If string, 'UTC' will use the time as input directly. Any other string value needs to be a TZ identifier in the IANA database, a wikipedia page of these is available here: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones. If int, should be the int value of the UTC offset (e.g., for American Eastern Standard Time: -5). This is necessary for Raspberry Shake data in 'raw' format.

xcoord
[float, default=-88.2290526] Longitude (or easting, or, generally, X coordinate) of data point, in Coordinate Reference System (CRS) designated by input_crs. Currently only used in csv output, but will likely be used in future for mapping/profile purposes.

ycoord
[float, default=40.1012122] Latitute (or northing, or, generally, X coordinate) of data point, in Coordinate Reference System (CRS) designated by input_crs. Currently only used in csv output, but will likely be used in future for mapping/profile purposes.

input_crs
[str or other format read by pyproj, default='EPSG:4326'] Coordinate reference system of input data, as used by pyproj.CRS.from_user_input()

output_crs
[str or other format read by pyproj, default='EPSG:4326'] Coordinate reference system to which input data will be transformed, as used by pyproj.CRS.from_user_input()

elevation
[float, default=755] Surface elevation of data point. Not currently used (except in csv output), but will likely be used in the future.

depth
[float, default=0] Depth of seismometer. Not currently used, but will likely be used in the future.

instrument
[str or list {'Raspberry Shake')] Instrument from which the data was acquired.

metapath
[str or pathlib.Path object, default=None] Filepath of metadata, in format supported by obspy.read_inventory. If default value of None, will read from resources folder of repository (only supported for Raspberry Shake).

hvsr_band
[list, default=[0.4, 40]] Two-element list containing low and high "corner" frequencies (in Hz) for processing. This can specified again later.

peak_freq_range
[list or tuple, default=[0.4, 40]] Two-element list or tuple containing low and high frequencies (in Hz) that are used to check for HVSR Peaks. This can be a tigher range than hvsr_band, but if larger, it will still only use the hvsr_band range.

processing_parameters={}
[dict or filepath, default={}] If filepath, should point to a .proc json file with processing parameters (i.e, an output from sprit.export_settings()). Note that this only applies to parameters for the functions: 'fetch_data', 'remove_noise', 'generate_ppsds', 'process_hvsr', 'check_peaks', and 'get_report.' If dictionary, dictionary containing nested dictionaries of function names as they key, and the parameter names/values as key/value pairs for each key. If a function name is not present, or if a parameter name is not present, default values will be used. For example:

{ 'fetch_data' : {'source':'batch', 'trim_dir':"/path/to/trimmed/data", 'export_format':'mseed', 'detrend':'spline', 'plot_input_stream':True, 'verbose':False, kwargs:{'kwargskey':'kwargsvalue'}} }

verbose
[bool, default=False] Whether to print output and results to terminal

Returns

params
[sprit.HVSRData] sprit.HVSRData class containing input parameters, including data file path and metadata path. This will be used as an input to other functions. If batch processing, params will be converted to batch type in fetch_data() step.

sprit.**make_it_classy**(*input_data*, *verbose=False*)

sprit.**plot_azimuth**(*hvsr_data*, *fig=None*, *ax=None*, *show_azimuth_peaks=False*, *interpolate_azimuths=True*, *show_azimuth_grid=False*, ***plot_azimuth_kwargs*)

Function to plot azimuths when azimuths are calculated

Parameters

hvsr_data
[HVSRData or HVSRBatch] HVSRData that has gone through at least the sprit.fetch_data() step, and before sprit.generate_ppsds()

show_azimuth_peaks
[bool, optional] Whether to display the peak value at each azimuth calculated on the chart, by default False

interpolate_azimuths
[bool, optional] Whether to interpolate the azimuth data to get a smoother plot. This is just for visualization, does not change underlying data. It takes a lot of time to process the data, but interpolation for vizualization can happen fairly fast. By default True.

show_azimuth_grid
[bool, optional] Whether to display the grid on the chart, by default False

Returns

matplotlib.Figure, matplotlib.Axis
Figure and axis of resulting azimuth plot

sprit.**plot_hvsr**(*hvsr_data*, *plot_type='HVSR ann p C+ ann p SPEC'*, *azimuth='HV'*, *use_subplots=True*, *fig=None*, *ax=None*, *return_fig=False*, *save_dir=None*, *save_suffix=''*, *show_legend=False*, *show=True*, *close_figs=False*, *clear_fig=True*, ***kwargs*)

Function to plot HVSR data

Parameters

hvsr_data
[dict] Dictionary containing output from process_hvsr function

plot_type
[str or list, default = 'HVSR ann p C+ ann p SPEC'] The plot_type of plot(s) to plot. If list, will plot all plots listed - 'HVSR' - Standard HVSR plot, including standard deviation. Options are included below:

- 'p' shows a vertical dotted line at frequency of the "best" peak
- 'ann' annotates the frequency value of of the "best" peak
- 'all' shows all the peaks identified in check_peaks() (by default, only the max is identified)
- 't' shows the H/V curve for all time windows
- 'tp' shows all the peaks from the H/V curves of all the time windows
- **'test' shows a visualization of the results of the peak validity test(s). Examples:**
  - 'tests' visualizes the results of all the peak tests (not the curve tests)
  - **'test12' shows the results of tests 1 and 2.**
    - Append any number 1-6 after 'test' to show a specific test result visualized
- **'COMP' - plot of the PPSD curves for each individual component ("C" also works)**
  - '+' (as a suffix in 'C+' or 'COMP+') plots C on a plot separate from HVSR (C+ is default, but without + will plot on the same plot as HVSR)
  - 'p' shows a vertical dotted line at frequency of the "best" peak
  - 'ann' annotates the frequency value of of the "best" peak
  - 'all' shows all the peaks identified in check_peaks() (by default, only the max is identified)
  - 't' shows the H/V curve for all time windows

- **'SPEC' - spectrogram style plot of the H/V curve over time**
  - 'p' shows a horizontal dotted line at the frequency of the "best" peak
  - 'ann' annotates the frequency value of the "best" peak
  - 'all' shows all the peaks identified in check_peaks()
  - 'tp' shows all the peaks of the H/V curve at all time windows
- **'AZ' - circular plot of calculated azimuthal HV curves, similar in style to SPEC plot.**
  - 'p' shows a point at each calculated (not interpolated) azimuth peak
  - 'g' shows grid lines at various angles
  - **'i' interpolates so that there is an interpolated azimuth at each degree interval (1 degree step)**
    This is the default, so usually 'i' is not needed.
  - '-i' prohibits interpolation (only shows the calculated azimuths, as determined by azimuth_angle (default = 30))

azimuth
: [str, default = 'HV'] What 'azimuth' to plot, default being standard N E components combined

use_subplots
: [bool, default = True] Whether to output the plots as subplots (True) or as separate plots (False)

fig
: [matplotlib.Figure, default = None] If not None, matplotlib figure on which plot is plotted

ax
: [matplotlib.Axis, default = None] If not None, matplotlib axis on which plot is plotted

return_fig
: [bool] Whether to return figure and axis objects

save_dir
: [str or None] Directory in which to save figures

save_suffix
: [str] Suffix to add to end of figure filename(s), if save_dir is used

show_legend
: [bool, default=False] Whether to show legend in plot

show
: [bool] Whether to show plot

close_figs
: [bool, default=False] Whether to close figures before plotting

clear_fig
: [bool, default=True] Whether to clear figures before plotting

****kwargs**
: [keyword arguments] Keyword arguments for matplotlib.pyplot

Returns

fig, ax
[matplotlib figure and axis objects] Returns figure and axis matplotlib.pyplot objects if return_fig=True, otherwise, simply plots the figures

sprit.**process_hvsr**(*hvsr_data*, *method=3*, *smooth=True*, *freq_smooth='konno ohmachi'*, *f_smooth_width=40*, *resample=True*, *outlier_curve_rmse_percentile=False*, *verbose=False*)

Process the input data and get HVSR data

This is the main function that uses other (private) functions to do the bulk of processing of the HVSR data and the data quality checks.

Parameters

hvsr_data
[HVSRData or HVSRBatch] Data object containing all the parameters input and generated by the user (usually, during sprit.input_params(), sprit.fetch_data(), sprit.generate_ppsds() and/or sprit.remove_noise()).

method
[int or str, default=3]

Method to use for combining the horizontal components

0) (not used)

1) Diffuse field assumption, or 'DFA' (not currently implemented)

2) 'Arithmetic Mean': H (HN + HE)/2

3) 'Geometric Mean': H HN · HE, recommended by the SESAME project (2004)

4) 'Vector Summation': H H2 N + H2 E

5) 'Quadratic Mean': H (H2 N + H2 E)/2

6) 'Maximum Horizontal Value': H max {HN, HE}

smooth
[bool, default=True]

bool or int may be used.
If True, default to smooth H/V curve to using savgoy filter with window length of 51 (works well with default resample of 1000 pts) If int, the length of the window in the savgoy filter.

freq_smooth
[str {'konno ohmachi', 'constant', 'proportional'}]

Which frequency smoothing method to use. By default, uses the 'konno ohmachi' method.

- The Konno & Ohmachi method uses the obspy.signal.konnoohmachismoothing.konno_ohmachi_smoothing() function: https://docs.obspy.org/packages/autogen/obspy.signal.konnoohmachismoothing.konno_ohmachi_smoothing.html
- The constant method uses a window of constant length f_smooth_width
- The proportional method uses a window the percentage length of the frequncy steps/range (f_smooth_width now refers to percentage)

See here for more information: https://www.geopsy.org/documentation/geopsy/hv-processing.html

f_smooth_width
[int, default = 40]

- For 'konno ohmachi': passed directly to the bandwidth parameter of the konno_ohmachi_smoothing() function, determines the width of the smoothing peak, with lower values resulting in broader peak. Must be > 0.
- For 'constant': the size of a triangular smoothing window in the number of frequency steps
- For 'proportional': the size of a triangular smoothing window in percentage of the number of frequency steps (e.g., if 1000 frequency steps/bins and f_smooth_width=40, window would be 400 steps wide)

resample
[bool, default = True]

bool or int.
If True, default to resample H/V data to include 1000 frequency values for the rest of the analysis If int, the number of data points to interpolate/resample/smooth the component psd/HV curve data to.

outlier_curve_rmse_percentile
[bool, float, default = False] If False, outlier curve removal is not carried out here. If True, defaults to 98 (98th percentile). Otherwise, float of percentile used as rmse_thresh of remove_outlier_curve().

verbose
[bool, defualt=False] Whether to print output to terminal

Returns

hvsr_out
[dict] Dictionary containing all the information about the data, including input parameters

sprit.**read_from_RS**(*dest*, *src='SHAKENAME@HOSTNAME:/opt/data/archive/YEAR/AM/STATION/'*, *opts='az'*, *username='myshake'*, *password='shakeme'*, *hostname='rs.local'*, *year='2023'*, *sta='RAC84'*, *sleep_time=0.1*, *verbose=True*, *save_progress=True*, *method='scp'*)

sprit.**read_tromino_files**(*datapath*, *params*, *sampling_rate=128*, *start_byte=24576*, *verbose=False*, ***kwargs*)

Function to read data from tromino. Specifically, this has been lightly tested on Tromino 3G+ machines

Parameters

datapath
[str, pathlib.Path()] The input parameter _datapath_ from sprit.input_params()

params
[HVSRData or HVSRBatch] The parameters as read in from input_params() and and fetch_data()

verbose
[bool, optional] Whether to print results to terminal, by default False

Returns

obspy.Stream
An obspy.Stream object containing the trace data from the Tromino instrument

sprit.**remove_noise**(*hvsr_data*, *remove_method='auto'*, *sat_percent=0.995*, *noise_percent=0.8*, *sta=2*, *lta=30*, *stalta_thresh=[8, 16]*, *warmup_time=0*, *cooldown_time=0*, *min_win_size=1*, *remove_raw_noise=False*, *show_stalta_plot=False*, *verbose=False*)

Function to remove noisy windows from data, using various methods.

Methods include - Manual window selection (by clicking on a chart with spectrogram and stream data), - Auto window selection, which does the following two in sequence (these can also be done indepently):

- A sta/lta "antitrigger" method (using stalta values to automatically remove triggered windows where there appears to be too much noise)
- A noise threshold method, that cuts off all times where the noise threshold equals more than (by default) 80% of the highest amplitude noise sample for the length specified by lta (in seconds)
- A saturation threshold method, that cuts off all times where the noise threshold equals more than (by default) 99.5% of the highest amplitude noise sample.

Parameters

hvsr_data
: [dict, obspy.Stream, or obspy.Trace] Dictionary containing all the data and parameters for the HVSR analysis

remove_method
: [str, {'auto', 'manual', 'stalta'/'antitrigger', 'saturation threshold', 'noise threshold', 'warmup'/'cooldown'/'buffer'/'warm_cool'}] The different methods for removing noise from the dataset. A list of strings will also work, in which case, it should be a list of the above strings. See descriptions above for what how each method works. By default 'auto.' If remove_method='auto', this is the equivalent of remove_method=['noise threshold', 'antitrigger', 'saturation threshold', 'warm_cool']

sat_percent
: [float, default=0.995] Percentage (between 0 and 1), to use as the threshold at which to remove data. This is used in the saturation method. By default 0.995. If a value is passed that is greater than 1, it will be divided by 100 to obtain the percentage.

noise_percent
: [float, default = 0.8] Percentage (between 0 and 1), to use as the threshold at which to remove data, if it persists for longer than time (in seconds (specified by min_win_size)). This is used in the noise threshold method. By default 0.8. If a value is passed that is greater than 1, it will be divided by 100 to obtain the percentage.

sta
: [int, optional] Short term average (STA) window (in seconds), by default 2. For use with sta/lta antitrigger method.

lta
: [int, optional] Long term average (STA) window (in seconds), by default 30. For use with sta/lta antitrigger method.

stalta_thresh
: [list, default=[0.5,5]] Two-item list or tuple with the thresholds for the stalta antitrigger. The first value (index [0]) is the lower threshold, the second value (index [1] is the upper threshold), by default [0.5,5]

warmup_time
: [int, default=0] Time in seconds to allow for warmup of the instrument (or while operator is still near instrument). This will renove any data before this time, by default 0.

cooldown_time
: [int, default=0] Time in seconds to allow for cooldown of the instrument (or for when operator is nearing instrument). This will renove any data before this time, by default 0.

min_win_size
[float, default=1] The minumum size a window must be over specified threshold (in seconds) for it to be removed

remove_raw_noise
[bool, default=False] If remove_raw_noise=True, will perform operation on raw data ('input_stream'), rather than potentially already-modified data ('stream').

verbose
[bool, default=False] Whether to print status of remove_noise

Returns

output
[dict] Dictionary similar to hvsr_data, but containing modified data with 'noise' removed

sprit.**remove_outlier_curves**(*hvsr_data*, *rmse_thresh=98*, *use_percentile=True*, *use_hv_curve=False*, *show_outlier_plot=False*, *verbose=False*)

Function used to remove outliers curves using Root Mean Square Error to calculate the error of each windowed Probabilistic Power Spectral Density (PPSD) curve against the median PPSD value at each frequency step for all times. It calculates the RMSE for the PPSD curves of each component individually. All curves are removed from analysis.

Some abberant curves often occur due to the remove_noise() function, so this should be run some time after remove_noise(). In general, the recommended workflow is to run this immediately following the generate_ppsds() function.

Parameters

hvsr_data
[dict] Input dictionary containing all the values and parameters of interest

rmse_thresh
[float or int, default=98] The Root Mean Square Error value to use as a threshold for determining whether a curve is an outlier. This averages over each individual entire curve so that curves with very abberant data (often occurs when using the remove_noise() method), can be identified. Otherwise, specify a float or integer to use as the cutoff RMSE value (all curves with RMSE above will be removed)

use_percentile
[float, default=True] Whether rmse_thresh should be interepreted as a raw RMSE value or as a percentile of the RMSE values.

use_hv_curve
[bool, default=False] Whether to use the calculated HV Curve or the individual components. This can only be True after process_hvsr() has been run.

show_plot
[bool, default=False] Whether to show a plot of the removed data

verbose
[bool, default=False] Whether to print output of function to terminal

Returns

hvsr_data
[dict] Input dictionary with values modified based on work of function.

sprit.**run**(*datapath*, *source='file'*, *azimuth_calculation=False*, *noise_removal=False*, *outlier_curves_removal=False*, *verbose=False*, ***kwargs*)

The sprit.run() is the main function that allows you to do all your HVSR processing in one simple step (sprit.run() is how you would call it in your code, but it may also be called using sprit.sprit_hvsr.run())

The datapath parameter of sprit.run() is the only required parameter. This can be either a single file, a list of files (one for each component, for example), a directory (in which case, all obspy-readable files will be added to an HVSRBatch instance), a Rasp. Shake raw data directory, or sample data.

Parameters

datapath
[str or filepath object that can be read by obspy] Filepath to data to be processed. This may be a file or directory, depending on what kind of data is being processed (this can be specified with the source parameter). For sample data, The following can be specified as the datapath parameter:

- Any integer 1-6 (inclusive), or the string (e.g., datapath="1" or datapath=1 will work)
- The word "sample" before any integer (e.g., datapath="sample1")
- The word "sample" will default to "sample1" if source='file'.
- If source='batch', datapath should be datapath='sample' or datapath='batch'. In this case, it will read and process all the sample files using the HVSRBatch class. Set verbose=True to see all the information in the sample batch csv file.

source
[str, optional] _description_, by default 'file'

azimuth
[bool, optional] Whether to perform azimuthal analysis, by default False.

verbose
[bool, optional] _description_, by default False

****kwargs**
Keyword arguments for the functions listed above. The keyword arguments are unique, so they will get parsed out and passed into the appropriate function.

input_params
[function name (not an actual parameter)] Function for designating input parameters for reading in and processing data See API documentation: [input_params()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.input_params)

datapath: any, default = '<no default>'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

site: any, default = 'HVSR Site'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

network: any, default = 'AM'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

station: any, default = 'RAC84'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

loc: any, default = '00'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

channels: any, default = ['EHZ', 'EHN', 'EHE']
See API documentation at link above or at *help(input_params)* for specifics.

acq_date: any, default = '2024-04-25'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

starttime: any, default = 2024-04-25T00:00:00.000000Z
See API documentation at link above or at *help(input_params)* for specifics.

endtime: any, default = 2024-04-25T23:59:59.999999Z
See API documentation at link above or at *help(input_params)* for specifics.

tzone: any, default = 'UTC'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

xcoord: any, default = -88.2290526
See API documentation at link above or at *help(input_params)* for specifics.

ycoord: any, default = 40.1012122
See API documentation at link above or at *help(input_params)* for specifics.

elevation: any, default = 755
See API documentation at link above or at *help(input_params)* for specifics.

input_crs: any, default = 'EPSG:4326'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

output_crs: any, default = 'EPSG:4326'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

elev_unit: any, default = 'feet'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

depth: any, default = 0
See API documentation at link above or at *help(input_params)* for specifics.

instrument: any, default = 'Raspberry Shake'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

metapath: any, default = None
See API documentation at link above or at *help(input_params)* for specifics.

hvsr_band: any, default = [0.4, 40]
See API documentation at link above or at *help(input_params)* for specifics.

peak_freq_range: any, default = [0.4, 40]
See API documentation at link above or at *help(input_params)* for specifics.

processing_parameters: any, default = {}
See API documentation at link above or at *help(input_params)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(input_params)* for specifics.

fetch_data
[function name (not an actual parameter)] Fetch ambient seismic data from a source to read into obspy stream See API documentation: [fetch_data()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.fetch_data)

params: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

source: any, default = 'file'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

trim_dir: any, default = None
See API documentation at link above or at *help(fetch_data)* for specifics.

export_format: any, default = 'mseed'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

detrend: any, default = 'spline'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

detrend_order: any, default = 2
See API documentation at link above or at *help(fetch_data)* for specifics.

update_metadata: any, default = True
See API documentation at link above or at *help(fetch_data)* for specifics.

plot_input_stream: any, default = False
See API documentation at link above or at *help(fetch_data)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(fetch_data)* for specifics.

kwargs: any, default = {}
See API documentation at link above or at *help(fetch_data)* for specifics.

calculate_azimuth
[function name (not an actual parameter)] Function to calculate azimuthal horizontal component at specified angle(s). Adds each new horizontal See API documentation: [calculate_azimuth()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.calculate_azimuth)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.calculate_azimuth)* for specifics.

azimuth_angle: any, default = 30
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

azimuth_type: any, default = 'multiple'
See API documentation at link above or at *help(sprit.calculate_azimuth)* for specifics.

azimuth_unit: any, default = 'degrees'
See API documentation at link above or at *help(sprit.calculate_azimuth)* for specifics.

show_az_plot: any, default = False
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

plot_azimuth_kwargs: any, default = {}
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

remove_noise
[function name (not an actual parameter)] Function to remove noisy windows from data, using various methods. See API documentation: [remove_noise()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.remove_noise)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.remove_noise)* for specifics.

remove_method: any, default = 'auto'
See API documentation at link above or at *help(sprit.remove_noise)* for specifics.

sat_percent: any, default = 0.995
See API documentation at link above or at *help(remove_noise)* for specifics.

noise_percent: any, default = 0.8
See API documentation at link above or at *help(remove_noise)* for specifics.

sta: any, default = 2
See API documentation at link above or at *help(remove_noise)* for specifics.

lta: any, default = 30
See API documentation at link above or at *help(remove_noise)* for specifics.

stalta_thresh: any, default = [8, 16]
See API documentation at link above or at *help(remove_noise)* for specifics.

warmup_time: any, default = 0
See API documentation at link above or at *help(remove_noise)* for specifics.

cooldown_time: any, default = 0
See API documentation at link above or at *help(remove_noise)* for specifics.

min_win_size: any, default = 1
See API documentation at link above or at *help(remove_noise)* for specifics.

remove_raw_noise: any, default = False
See API documentation at link above or at *help(remove_noise)* for specifics.

show_stalta_plot: any, default = False
See API documentation at link above or at *help(remove_noise)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(remove_noise)* for specifics.

generate_ppsds
[function name (not an actual parameter)] Generates PPSDs for each channel See API documentation: [generate_ppsds()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.generate_ppsds)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.generate_ppsds)* for specifics.

azimuthal_ppsds: any, default = False
See API documentation at link above or at *help(generate_ppsds)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(generate_ppsds)* for specifics.

ppsd_kwargs: any, default = {}
See API documentation at link above or at *help(generate_ppsds)* for specifics.

process_hvsr
[function name (not an actual parameter)] Process the input data and get HVSR data See API documentation: [process_hvsr()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.process_hvsr)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.process_hvsr)* for specifics.

method: any, default = 3
See API documentation at link above or at *help(process_hvsr)* for specifics.

smooth: any, default = True
See API documentation at link above or at *help(process_hvsr)* for specifics.

freq_smooth: any, default = 'konno ohmachi'
See API documentation at link above or at *help(sprit.process_hvsr)* for specifics.

f_smooth_width: any, default = 40
See API documentation at link above or at *help(process_hvsr)* for specifics.

resample: any, default = True
See API documentation at link above or at *help(process_hvsr)* for specifics.

outlier_curve_rmse_percentile: any, default = False
See API documentation at link above or at *help(process_hvsr)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(process_hvsr)* for specifics.

remove_outlier_curves
[function name (not an actual parameter)] Function used to remove outliers curves using Root Mean Square Error to calculate the error of each See API documentation: [remove_outlier_curves()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.remove_outlier_curves)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.remove_outlier_curves)* for specifics.

rmse_thresh: any, default = 98
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

use_percentile: any, default = True
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

use_hv_curve: any, default = False
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

show_outlier_plot: any, default = False
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

check_peaks
[function name (not an actual parameter)] Function to run tests on HVSR peaks to find best one and see if it passes quality checks See API documentation: [check_peaks()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.check_peaks)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.check_peaks)* for specifics.

hvsr_band: any, default = [0.4, 40]
See API documentation at link above or at *help(check_peaks)* for specifics.

peak_selection: any, default = 'max'
See API documentation at link above or at *help(sprit.check_peaks)* for specifics.

peak_freq_range: any, default = [0.4, 40]
See API documentation at link above or at *help(check_peaks)* for specifics.

azimuth: any, default = 'HV'
See API documentation at link above or at *help(sprit.check_peaks)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(check_peaks)* for specifics.

get_report
[function name (not an actual parameter)] Get a report of the HVSR analysis in a variety of formats. See API documentation: [get_report()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.get_report)

hvsr_results: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

report_format: any, default = ['print', 'csv', 'plot']
See API documentation at link above or at *help(get_report)* for specifics.

plot_type: any, default = 'HVSR p ann C+ p ann Spec'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

azimuth: any, default = 'HV'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

export_path: any, default = None
See API documentation at link above or at *help(get_report)* for specifics.

csv_overwrite_opt: any, default = 'append'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

no_output: any, default = False
See API documentation at link above or at *help(get_report)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(get_report)* for specifics.

export_data
[function name (not an actual parameter)] Export data into pickle format that can be read back in using import_data() so data does not need to See API documentation: [export_data()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.export_data)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.export_data)* for specifics.

export_path: any, default = None
See API documentation at link above or at *help(export_data)* for specifics.

ext: any, default = 'hvsr'
See API documentation at link above or at *help(sprit.export_data)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(export_data)* for specifics.

Returns

hvsr_results
[sprit.HVSRData or sprit.HVSRBatch object] If a single file/data point is being processed, a HVSRData object will be returned. Otherwise, it will be a HVSRBatch object. See their documention for more information.

Raises

RuntimeError
If the input parameter may not be read correctly. This is raised if the input_params() function fails. This raises an error since no other data processing or reading steps will be able to carried out correctly.

RuntimeError
If the data is not read/fetched correctly using fetch_data(), an error will be raised. This is raised if the fetch_data() function fails. This raises an error since no other data processing steps will be able to carried out correctly.

RuntimeError
If the data being processed is a single file, an error will be raised if generate_ppsds() does not work correctly. No errors are raised for remove_noise() errors (since that is an optional step) and the process_hvsr() step (since that is the last processing step) .

Notes

The sprit.run() function calls the following functions. This is the recommended order/set of functions to run to process HVSR using SpRIT. See the API documentation for these functions for more information: - input_params(): The datapath parameter of input_params() is the only required variable, though others may also need to be called for your data to process correctly. - fetch_data(): the source parameter of fetch_data() is the only explicit variable in the sprit.run() function aside from datapath and verbose. Everything else gets delivered to the correct function via the kwargs dictionary - remove_noise(): by default, the kind of noise removal is remove_method='auto'. See the remove_noise() documentation for more information. If remove_method is set to anything other than one of the explicit options in remove_noise, noise removal will not be carried out. - generate_ppsds(): generates ppsds for each component, which will be combined/used later. Any parameter of obspy.signal.spectral_estimation.PPSD() may also be read into this function. - remove_outlier_curves(): removes any outlier ppsd curves so that the data quality for when curves are combined will be enhanced. See the remove_outlier_curves() documentation for more information. - process_hvsr(): this is the main function processing the hvsr curve and statistics. See process_hvsr() documentation for more details. The hvsr_band parameter sets the frequency spectrum over which these calculations occur. - check_peaks(): this is the main function that will find and 'score' peaks to get a best peak. The parameter peak_freq_range can be set to limit the frequencies within which peaks are checked and scored. - get_report(): this is the main function that will print, plot, and/or save the results of the data. See the get_report() API documentation for more information. - export_data(): this function exports the final data output as a pickle file (by default, this pickle object has a .hvsr extension). This can be used to read data back into SpRIT without having to reprocess data.

sprit.**time_it**(*_t*, *proc_name=''*, *verbose=True*)

Computes elapsed time since the last call.

sprit.**x_mark**(*incolor=False*, *inTerminal=False*)

The default Windows terminal is not able to display the check mark character correctly. This function returns another displayable character if platform is Windows

1.1 Submodules

1.1.1 sprit.sprit_cli module

This module/script is used to run sprit from the command line.

The arguments here should correspond to any of the keyword arguments that can be used with sprit.run() (or sprit_hvsr.run()). See the run() function's documentation for more information, or the individual functions that are run within it.

For list inputs, you should pass the argument multiple times(e.g., –report_format "csv" –report_format "print" –report_format "plot"). (In the case of –report_format, you can also just use "all" to get csv, print, and plot report types)

The datapath parameter of input_params() is the only required argument, though for your data processing to work correctly and to be formatted correctly, you may need to pass others as well.

sprit.sprit_cli.**get_param_docstring**(*func*, *param_name*)

sprit.sprit_cli.**main**()

1.1.2 sprit.sprit_gui module

This script contains all the functions, classes, etc. to create a tkinter app for graphical user interface.

class sprit.sprit_gui.**SPRIT_App**(*master*)

Bases: object

Methods

log_errorMsg(logMsg)
manual_label_update()

create_menubar
create_tabs

create_menubar()

create_tabs()

log_errorMsg(*logMsg*)

manual_label_update()

sprit.sprit_gui.**catch_errors**(*func*)

sprit.sprit_gui.**on_closing**()

sprit.sprit_gui.**reboot_app**()

Restarts the current program. Note: this function does not return. Any cleanup action (like saving data) must be done before calling this function.

1.1.3 sprit.sprit_hvsr module

This module is the main SpRIT module that contains all the functions needed to run HVSR analysis.

The functions defined here are read both by the SpRIT graphical user interface and by the command-line interface to run HVSR analysis on input data.

See documentation for individual functions for more information.

class sprit.sprit_hvsr.**HVSRBatch**(**args*, ***kwargs*)

Bases: object

HVSRBatch is the data container used for batch processing. It contains several HVSRData objects (one for each site). These can be accessed using their site name, either square brackets (HVSRBatchVariable["SiteName"]) or the dot (HVSRBatchVariable.SiteName) accessor.

The dot accessor may not work if there is a space in the site name.

All of the functions in the sprit.pacakge are designed to perform the bulk of their operations iteratively on the individual HVSRData objects contained in the HVSRBatch object, and do little with the HVSRBatch object itself, besides using it determine which sites are contained within it.

Methods

copy([type])	Make a copy of the HVSRBatch object.
export([export_path, ext])	Method to export HVSRData objects in HVSRBatch container to indivdual .hvsr pickle files.
export_settings([site_name, ...])	Method to export settings from HVSRData object in HVSRBatch object.
get_report(**kwargs)	Method to get report from processed data, in print, graphical, or tabular format.
items()	Method to return both the site names and the HVSRData object as a set of dict_items tuples.
keys()	Method to return the "keys" of the HVSRBatch object.
plot(**kwargs)	Method to plot data, based on the sprit.plot_hvsr() function.
report(**kwargs)	Wrapper of get_report()

copy(*type='shallow'*)

Make a copy of the HVSRBatch object. Uses python copy module.

Parameters

type
[str {'shallow', 'deep'}] Based on input, creates either a shallow or deep copy of the HVSRBatch object. Shallow is equivalent of copy.copy(). Input of 'deep' is equivalent of copy.deepcopy() (still experimental). Defaults to shallow.

export(*export_path=True*, *ext='hvsr'*)

Method to export HVSRData objects in HVSRBatch container to indivdual .hvsr pickle files.

Parameters

export_path
[filepath, default=True] Filepath to save file. Can be either directory (which will assign a filename based on the HVSRData attributes). By default True. If True, it will first try to save each file to the same directory as datapath, then if that does not work, to the current working directory, then to the user's home directory, by default True

ext
[str, optional] The extension to use for the output, by default 'hvsr'. This is still a pickle file that can be read with pickle.load(), but will have .hvsr extension.

export_settings(*site_name=None*, *export_settings_path='default'*, *export_settings_type='all'*, *include_location=False*, *verbose=True*)

Method to export settings from HVSRData object in HVSRBatch object.

Simply calls sprit.export_settings() from specified HVSRData object in the HVSRBatch object. See sprit.export_settings() for more details.

Parameters

site_name
[str, default=None] The name of the site whose settings should be exported. If None, will default to the first site, by default None.

export_settings_path
[str, optional] Filepath to output file. If left as 'default', will save as the default

value in the resources directory. If that is not possible, will save to home directory, by default 'default'

export_settings_type
[str, {'all', 'instrument', 'processing'}, optional] They type of settings to save, by default 'all'

include_location
[bool, optional] Whether to include the location information in the instrument settings, if that settings type is selected, by default False

verbose
[bool, optional] Whether to print output (filepath and settings) to terminal, by default True

See also:

export_settings

`get_report`(***kwargs*)

Method to get report from processed data, in print, graphical, or tabular format.

Returns

Variable
May return nothing, pandas.Dataframe, or pyplot Figure, depending on input.

See also:

get_report

`items`()

Method to return both the site names and the HVSRData object as a set of dict_items tuples. Functions similar to dict.items().

Returns

type
description

`keys`()

Method to return the "keys" of the HVSRBatch object. For HVSRBatch objects, these are the site names. Functions similar to dict.keys().

Returns

dict_keys
A dict_keys object listing the site names of each of the HVSRData objects contained in the HVSRBatch object

`plot`(***kwargs*)

Method to plot data, based on the sprit.plot_hvsr() function.

All the same kwargs and default values apply as plot_hvsr(). For return_fig, returns it to the 'Plot_Report' attribute of each HVSRData object

Returns

type
description

See also:

plot_hvsr

`report`(***kwargs*)

Wrapper of get_report()

See also:

get_report

class sprit.sprit_hvsr.**HVSRData**(**args*, ***kwargs*)

Bases: object

HVSRData is the basic data class of the sprit package. It contains all the processed data, input parameters, and reports.

These attributes and objects can be accessed using square brackets or the dot accessor. For example, to access the site name, HVSRData['site'] and HVSRData.site will both return the site name.

Some of the methods that work on the HVSRData object (e.g., .plot() and .get_report()) are essentially wrappers for some of the main sprit package functions (sprit.plot_hvsr() and sprit.get_report(), respectively)

Attributes

batch
Whether this HVSRData object is part of an HVSRBatch object.

datastream
A copy of the original obspy datastream read in.

params
Dictionary containing the parameters used to process the data

ppsds
Dictionary copy of the class object obspy.signal.spectral_estimation.PPSD().

ppsds_obspy
The original ppsd information from the obspy.signal.spectral_estimation.PPSD(), so as to keep original if copy is manipulated/changed.

Methods

copy([type])	Make a copy of the HVSRData object.
export([export_path, ext])	Method to export HVSRData objects to .hvsr pickle files.
export_settings([export_settings_path, ...])	Method to export settings from HVSRData object.
get_report(**kwargs)	Method to get report from processed data, in print, graphical, or tabular format.
items()	Method to return the "items" of the HVSRData object.
keys()	Method to return the "keys" of the HVSRData object.
plot(**kwargs)	Method to plot data, wrapper of sprit.plot_hvsr()
report(**kwargs)	Wrapper of get_report()

property batch

Whether this HVSRData object is part of an HVSRBatch object. This is used throughout the code to help direct the object into the proper processing pipeline.

Returns

bool
True if HVSRData object is part of HVSRBatch object, otherwise, False

`copy`(*type='shallow'*)

Make a copy of the HVSRData object. Uses python copy module.

Parameters

type
[str {'shallow', 'deep'}] Based on input, creates either a shallow or deep copy of the HVSRData object. Shallow is equivalent of copy.copy(). Input of type='deep' is equivalent of copy.deepcopy() (still experimental). Defaults to shallow.

`property datastream`

A copy of the original obspy datastream read in. This helps to retain the original data even after processing is carried out.

Returns

obspy.core.Stream.stream
Obspy stream

`export`(*export_path=None*, *ext='hvsr'*)

Method to export HVSRData objects to .hvsr pickle files.

Parameters

export_path
[filepath, default=True] Filepath to save file. Can be either directory (which will assign a filename based on the HVSRData attributes). By default True. If True, it will first try to save each file to the same directory as datapath, then if that does not work, to the current working directory, then to the user's home directory, by default True

ext
[str, optional] The extension to use for the output, by default 'hvsr'. This is still a pickle file that can be read with pickle.load(), but will have .hvsr extension.

`export_settings`(*export_settings_path='default'*, *export_settings_type='all'*, *include_location=False*, *verbose=True*)

Method to export settings from HVSRData object. Simply calls sprit.export_settings() from the HVSRData object. See sprit.export_settings() for more details.

Parameters

export_settings_path
[str, optional] Filepath to output file. If left as 'default', will save as the default value in the resources directory. If that is not possible, will save to home directory, by default 'default'

export_settings_type
[str, {'all', 'instrument', 'processing'}, optional] They type of settings to save, by default 'all'

include_location
[bool, optional] Whether to include the location information in the instrument settings, if that settings type is selected, by default False

verbose
[bool, optional] Whether to print output (filepath and settings) to terminal, by default True

`get_report`(***kwargs*)

Method to get report from processed data, in print, graphical, or tabular format.

Returns

Variable
May return nothing, pandas.Dataframe, or pyplot Figure, depending on input.

See also:

get_report

`items()`

Method to return the "items" of the HVSRData object. For HVSRData objects, this is a dict_items object with the keys and values in tuples. Functions similar to dict.items().

Returns

dict_items
A dict_items object of the HVSRData objects attributes, parameters, etc.

`keys()`

Method to return the "keys" of the HVSRData object. For HVSRData objects, these are the attributes and parameters of the object. Functions similar to dict.keys().

Returns

dict_keys
A dict_keys object of the HVSRData objects attributes, parameters, etc.

`property params`

Dictionary containing the parameters used to process the data

Returns

dict
Dictionary containing the process parameters

`plot(**kwargs)`

Method to plot data, wrapper of sprit.plot_hvsr()

Returns

matplotlib.Figure, matplotlib.Axis (if return_fig=True)

See also:

plot_hvsr
plot_azimuth

`property ppsds`

Dictionary copy of the class object obspy.signal.spectral_estimation.PPSD(). The dictionary copy allows manipulation of the data in PPSD, whereas that data cannot be easily manipulated in the original Obspy object.

Returns

dict
Dictionary copy of the PPSD information from generate_ppsds()

`property ppsds_obspy`

The original ppsd information from the obspy.signal.spectral_estimation.PPSD(), so as to keep original if copy is manipulated/changed.

`report(**kwargs)`

Wrapper of get_report()

See also:

get_report

sprit.sprit_hvsr.**batch_data_read**(*input_data*, *batch_type='table'*, *param_col=None*, *batch_params=None*, *verbose=False*, ***readcsv_getMeta_fetch_kwargs*)

Function to read data in data as a batch of multiple data files. This is best used through sprit.fetch_data(*args, source='batch', **other_kwargs).

Parameters

input_data
[filepath or list] Input data information for how to read in data as batch

batch_type
[str, optional] Type of batch read, only 'table' and 'filelist' accepted. If 'table', will read data from a file read in using pandas.read_csv(), by default 'table'

param_col
[None or str, optional] Name of parameter column from batch information file. Only used if a batch_type='table' and single parameter column is used, rather than one column per parameter (for single parameter column, parameters are formatted with = between keys/values and , between item pairs), by default None

batch_params
[list, dict, or None, default = None] Parameters to be used if batch_type='filelist'. If it is a list, needs to be the same length as input_data. If it is a dict, will be applied to all files in input_data and will combined with extra keyword arguments caught by **readcsv_getMeta_fetch_kwargs.

verbose
[bool, optional] Whether to print information to terminal during batch read, by default False

****readcsv_getMeta_fetch_kwargs**
Keyword arguments that will be read into pandas.read_csv(), sprit.input_params, sprit.get_metadata(), and/or sprit.fetch_data()

Returns

dict
Dictionary with each item representing a different file read in, and which consists of its own parameter dictionary to be used by the rest of the processing steps

Raises

IndexError
description

sprit.sprit_hvsr.**calculate_azimuth**(*hvsr_data*, *azimuth_angle=30*, *azimuth_type='multiple'*, *azimuth_unit='degrees'*, *show_az_plot=False*, *verbose=False*, ***plot_azimuth_kwargs*)

Function to calculate azimuthal horizontal component at specified angle(s). Adds each new horizontal component as a radial component to obspy.Stream object at hvsr_data['stream']

Parameters

hvsr_data
[HVSRData] Input HVSR data

azimuth_angle
[int, default=10] If *azimuth_type='multiple'*, this is the angular step (in unit *azimuth_unit*) of each of the azimuthal measurements. If *azimuth_type='single'* this is the angle (in unit *azimuth_unit*) of the single calculated azimuthal measruement. By default 10.

azimuth_type
[str, default='multiple'] What type of azimuthal measurement to make, by default 'multiple'. If 'multiple' (or {'multi', 'mult', 'm'}), will take a measurement at each angular step of azimuth_angle of unit azimuth_unit. If 'single' (or {'sing', 's'}), will take a single azimuthal measurement at angle specified in azimuth_angle.

azimuth_unit
[str, default='degrees'] Angular unit used to specify *azimuth_angle* parameter. By default 'degrees'. If 'degrees' (or {'deg', 'd'}), will use degrees. If 'radians' (or {'rad', 'r'}), will use radians.

show_az_plot
[bool, default=False] Whether to show azimuthal plot, by default False.

verbose
[bool, default=False] Whether to print terminal output, by default False

Returns

HVSRData
Updated HVSRData object specified in hvsr_data with hvsr_data['stream'] attribute containing additional components (EHR-***), with *** being zero-padded (3 digits) azimuth angle in degrees.

sprit.sprit_hvsr.**check_instance**(*init*)

sprit.sprit_hvsr.**check_peaks**(*hvsr_data*, *hvsr_band=[0.4, 40]*, *peak_selection='max'*, *peak_freq_range=[0.4, 40]*, *azimuth='HV'*, *verbose=False*)

Function to run tests on HVSR peaks to find best one and see if it passes quality checks

Parameters

hvsr_data
[dict] Dictionary containing all the calculated information about the HVSR data (i.e., hvsr_out returned from process_hvsr)

hvsr_band
[tuple or list, default=[0.4, 40]] 2-item tuple or list with lower and upper limit of frequencies to analyze

peak_selection
[str or numeric, default='max'] How to select the "best" peak used in the analysis. For peak_selection="max" (default value), the highest peak within peak_freq_range is used. For peak_selection='scored', an algorithm is used to select the peak based in part on which peak passes the most SESAME criteria. If a numeric value is used (e.g., int or float), this should be a frequency value to manually select as the peak of interest.

peak_freq_range
[tuple or list, default=[0.4, 40];] The frequency range within which to check for peaks. If there is an HVSR curve with multiple peaks, this allows the full range of data to be processed while limiting peak picks to likely range.

verbose
[bool, default=False] Whether to print results and inputs to terminal.

Returns

hvsr_data
[HVSRData or HVSRBatch object] Object containing previous input data, plus information about peak tests

sprit.sprit_hvsr.**export_data**(*hvsr_data*, *export_path=None*, *ext='hvsr'*, *verbose=False*)

Export data into pickle format that can be read back in using import_data() so data does not need to be processed each time. Default extension is .hvsr but it is still a pickled file that can be read in using pickle.load().

Parameters

hvsr_data
[HVSRData or HVSRBatch] Data to be exported

export_path
[str or filepath object, default = None] String or filepath object to be read by pathlib.Path() and/or a with open(export_path, 'wb') statement. If None, defaults to input datapath directory, by default None

ext
[str, default = 'hvsr'] Filepath extension to use for data file, by default 'hvsr'

sprit.sprit_hvsr.**export_settings**(*hvsr_data*, *export_settings_path='default'*, *export_settings_type='all'*, *include_location=False*, *verbose=True*)

Save settings to json file

Parameters

export_settings_path
[str, default="default"] Where to save the json file(s) containing the settings, by default 'default'. If "default," will save to sprit package resources. Otherwise, set a filepath location you would like for it to be saved to. If 'all' is selected, a directory should be supplied. Otherwise, it will save in the directory of the provided file, if it exists. Otherwise, defaults to the home directory.

export_settings_type
[str, {'all', 'instrument', 'processing'}] What kind of settings to save. If 'all', saves all possible types in their respective json files. If 'instrument', save the instrument settings to their respective file. If 'processing', saves the processing settings to their respective file. By default 'all'

include_location
[bool, default=False, input CRS] Whether to include the location parametersin the exported settings document.This includes xcoord, ycoord, elevation, elev_unit, and input_crs

verbose
[bool, default=True] Whether to print outputs and information to the terminal

sprit.sprit_hvsr.**fetch_data**(*params*, *source='file'*, *trim_dir=None*, *export_format='mseed'*, *detrend='spline'*, *detrend_order=2*, *update_metadata=True*, *plot_input_stream=False*, *verbose=False*, ***kwargs*)

Fetch ambient seismic data from a source to read into obspy stream

Parameters

params
[dict]

Dictionary containing all the necessary params to get data.
Parameters defined using input_params() function.

source
[str, {'raw', 'dir', 'file', 'batch'}]

String indicating where/how data file was created. For example, if raw data, will need to find correct channels.
'raw' finds raspberry shake data, from raw output copied using scp directly from

Raspberry Shake, either in folder or subfolders; 'dir' is used if the day's 3 component files (currently Raspberry Shake supported only) are all 3 contained in a directory by themselves. 'file' is used if the params['datapath'] specified in input_params() is the direct filepath to a single file to be read directly into an obspy stream. 'batch' is used to read a list or specified set of seismic files.

Most commonly, a csv file can be read in with all the parameters. Each row in the csv is a separate file. Columns can be arranged by parameter.

trim_dir
[None or str or pathlib obj, default=None] If None (or False), data is not trimmed in this function. Otherwise, this is the directory to save trimmed and exported data.

export_format: str='mseed'
If trim_dir is not None, this is the format in which to save the data

detrend
[str or bool, default='spline'] If False, data is not detrended. Otherwise, this should be a string accepted by the type parameter of the obspy.core.trace.Trace.detrend method: https://docs.obspy.org/packages/autogen/obspy.core.trace.Trace.detrend.html

detrend_order
[int, default=2] If detrend parameter is 'spline' or 'polynomial', this is passed directly to the order parameter of obspy.core.trace.Trace.detrend method.

update_metadata
[bool, default=True] Whether to update the metadata file, used primarily with Raspberry Shake data which uses a generic inventory file.

plot_input_stream
[bool, default=False] Whether to plot the raw input stream. This plot includes a spectrogram (Z component) and the raw (with decimation for speed) plots of each component signal.

verbose
[bool, default=False] Whether to print outputs and inputs to the terminal

****kwargs**
Keywords arguments, primarily for 'batch' and 'dir' sources

Returns

params
[HVSRData or HVSRBatch object] Same as params parameter, but with an additional "stream" attribute with an obspy data stream with 3 traces: Z (vertical), N (North-south), and E (East-west)

sprit.sprit_hvsr.**generate_ppsds**(*hvsr_data*, *azimuthal_ppsds=False*, *verbose=False*, ***ppsd_kwargs*)

Generates PPSDs for each channel

Channels need to be in Z, N, E order Info on PPSD creation here: https://docs.obspy.org/packages/autogen/obspy.signal.spectral_estimation.PPSD.html

Parameters

hvsr_data
[dict, HVSRData object, or HVSRBatch object] Data object containing all the parameters and other data of interest (stream and paz, for example)

azimuthal_ppsds
[bool, default=False] Whether to generate PPSDs for azimuthal data

verbose
[bool, default=True] Whether to print inputs and results to terminal

****ppsd_kwargs**
[dict] Dictionary with keyword arguments that are passed directly to obspy.signal.PPSD. If the following keywords are not specified, their defaults are amended in this function from the obspy defaults for its PPSD function. Specifically:

- ppsd_length defaults to 60 (seconds) here instead of 3600
- skip_on_gaps defaults to True instead of False
- period_step_octaves defaults to 0.03125 instead of 0.125

Returns

ppsds
[HVSRData object] Dictionary containing entries with ppsds for each channel

sprit.sprit_hvsr.**get_metadata**(*params*, *write_path=''*, *update_metadata=True*, *source=None*, ***read_inventory_kwargs*)

Get metadata and calculate or get paz parameter needed for PPSD

Parameters

params
[dict]

Dictionary containing all the input and other parameters needed for processing
Ouput from input_params() function

write_path
[str]

String with output filepath of where to write updated inventory or metadata file
If not specified, does not write file

update_metadata
[bool] Whether to update the metadata file itself, or just read as-is. If using provided raspberry shake metadata file, select True.

source
[str, default=None] This passes the source variable value to _read_RS_metadata. It is expected that this is passed directly from the source parameter of sprit.fetch_data()

Returns

params
[dict] Modified input dictionary with additional key:value pair containing paz dictionary (key = "paz")

sprit.sprit_hvsr.**get_report**(*hvsr_results*, *report_format=['print', 'csv', 'plot']*, *plot_type='HVSR p ann C+ p ann Spec'*, *azimuth='HV'*, *export_path=None*, *csv_overwrite_opt='append'*, *no_output=False*, *verbose=False*)

Get a report of the HVSR analysis in a variety of formats.

Parameters

hvsr_results
[dict] Dictionary containing all the information about the processed hvsr data

report_format
[{'csv', 'print', plot}] Format in which to print or export the report. The following report_formats return the following items in the following attributes:

- 'plot': hvsr_results['Print_Report'] as a str str

- 'print': hvsr_results['HV_Plot'] - matplotlib.Figure object
- **'csv': hvsr_results['CSV_Report']- pandas.DataFrame object**
  - list/tuple - a list or tuple of the above objects, in the same order they are in the report_format list

plot_type
[str, default = 'HVSR p ann C+ p ann Spec] What type of plot to plot, if 'plot' part of report_format input

azimuth
[str, default = 'HV'] Which azimuth to plot, by default "HV" which is the main "azimuth" combining the E and N components

export_path
[None, bool, or filepath, default = None] If None or False, does not export; if True, will export to same directory as the datapath parameter in the input_params() function. Otherwise, it should be a string or path object indicating where to export results. May be a file or directory. If a directory is specified, the filename will be "<site_name>_<acq_date>_<UTC start time>-<UTC end time>". The suffix defaults to png for report_format="plot", csv for 'csv', and does not export if 'print.'

csv_overwrite_opts
[str, {'append', 'overwrite', 'keep/rename'}] How to handle csv report outputs if the designated csv output file already exists. By default, appends the new information to the end of the existing file.

no_output
[bool, default=False] If True, only reads output to appropriate attribute of data class (ie, print does not print, only reads text into variable). If False, performs as normal.

verbose
[bool, default=True] Whether to print the results to terminal. This is the same output as report_format='print', and will not repeat if that is already selected

Returns

sprit.HVSRData

sprit.sprit_hvsr.**gui**(*kind='default'*)

Function to open a graphical user interface (gui)

Parameters

kind
[str, optional] What type of gui to open. "default" opens regular windowed interface, "widget" opens jupyter widget' "lite" open lite (pending update), by default 'default'

sprit.sprit_hvsr.**gui_test**()

sprit.sprit_hvsr.**import_data**(*import_filepath*, *data_format='pickle'*)

Function to import .hvsr (or other extension) data exported using export_data() function

Parameters

import_filepath
[str or path object] Filepath of file created using export_data() function. This is usually a pickle file with a .hvsr extension

data_format
[str, default='pickle'] Type of format data is in. Currently, only 'pickle' supported. Eventually, json or other type may be supported, by default 'pickle'.

Returns

HVSRData or HVSRBatch object

sprit.sprit_hvsr.**import_settings**(*settings_import_path*, *settings_import_type='instrument'*, *verbose=False*)

sprit.sprit_hvsr.**input_params**(*datapath*, *site='HVSR Site'*, *network='AM'*, *station='RAC84'*, *loc='00'*, *channels=['EHZ', 'EHN', 'EHE']*, *acq_date='2024-04-25'*, *starttime=UTCDateTime(2024, 4, 25, 0, 0)*, *endtime=UTCDateTime(2024, 4, 25, 23, 59, 59, 999999)*, *tzone='UTC'*, *xcoord=-88.2290526*, *ycoord=40.1012122*, *elevation=755*, *input_crs='EPSG:4326'*, *output_crs='EPSG:4326'*, *elev_unit='feet'*, *depth=0*, *instrument='Raspberry Shake'*, *metapath=None*, *hvsr_band=[0.4, 40]*, *peak_freq_range=[0.4, 40]*, *processing_parameters={}*, *verbose=False*)

Function for designating input parameters for reading in and processing data

Parameters

datapath
[str or pathlib.Path object] Filepath of data. This can be a directory or file, but will need to match with what is chosen later as the source parameter in fetch_data()

site
[str, default="HVSR Site"] Site name as designated by user for ease of reference. Used for plotting titles, filenames, etc.

network
[str, default='AM'] The network designation of the seismometer. This is necessary for data from Raspberry Shakes. 'AM' is for Amateur network, which fits Raspberry Shakes.

station
[str, default='RAC84'] The station name of the seismometer. This is necessary for data from Raspberry Shakes.

loc
[str, default='00'] Location information of the seismometer.

channels
[list, default=['EHZ', 'EHN', 'EHE']] The three channels used in this analysis, as a list of strings. Preferred that Z component is first, but not necessary

acq_date
[str, int, date object, or datetime object] If string, preferred format is 'YYYY-MM-DD'. If int, this will be interpreted as the time_int of year of current year (e.g., 33 would be Feb 2 of current year) If date or datetime object, this will be the date. Make sure to account for time change when converting to UTC (if UTC is the following time_int, use the UTC time_int).

starttime
[str, time object, or datetime object, default='00:00:00.00'] Start time of data stream. This is necessary for Raspberry Shake data in 'raw' form, or for trimming data. Format can be either 'HH:MM:SS.micros' or 'HH:MM' at minimum.

endtime
[str, time obejct, or datetime object, default='23:59:99.99'] End time of data stream. This is necessary for Raspberry Shake data in 'raw' form, or for trimming data. Same format as starttime.

tzone
[str or int, default = 'UTC'] Timezone of input data. If string, 'UTC' will use the

time as input directly. Any other string value needs to be a TZ identifier in the IANA database, a wikipedia page of these is available here: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones. If int, should be the int value of the UTC offset (e.g., for American Eastern Standard Time: -5). This is necessary for Raspberry Shake data in 'raw' format.

xcoord
[float, default=-88.2290526] Longitude (or easting, or, generally, X coordinate) of data point, in Coordinate Reference System (CRS) designated by input_crs. Currently only used in csv output, but will likely be used in future for mapping/profile purposes.

ycoord
[float, default=40.1012122] Latitute (or northing, or, generally, X coordinate) of data point, in Coordinate Reference System (CRS) designated by input_crs. Currently only used in csv output, but will likely be used in future for mapping/profile purposes.

input_crs
[str or other format read by pyproj, default='EPSG:4326'] Coordinate reference system of input data, as used by pyproj.CRS.from_user_input()

output_crs
[str or other format read by pyproj, default='EPSG:4326'] Coordinate reference system to which input data will be transformed, as used by pyproj.CRS.from_user_input()

elevation
[float, default=755] Surface elevation of data point. Not currently used (except in csv output), but will likely be used in the future.

depth
[float, default=0] Depth of seismometer. Not currently used, but will likely be used in the future.

instrument
[str or list {'Raspberry Shake')] Instrument from which the data was acquired.

metapath
[str or pathlib.Path object, default=None] Filepath of metadata, in format supported by obspy.read_inventory. If default value of None, will read from resources folder of repository (only supported for Raspberry Shake).

hvsr_band
[list, default=[0.4, 40]] Two-element list containing low and high "corner" frequencies (in Hz) for processing. This can specified again later.

peak_freq_range
[list or tuple, default=[0.4, 40]] Two-element list or tuple containing low and high frequencies (in Hz) that are used to check for HVSR Peaks. This can be a tigher range than hvsr_band, but if larger, it will still only use the hvsr_band range.

processing_parameters={}
[dict or filepath, default={}] If filepath, should point to a .proc json file with processing parameters (i.e, an output from sprit.export_settings()). Note that this only applies to parameters for the functions: 'fetch_data', 'remove_noise', 'generate_ppsds', 'process_hvsr', 'check_peaks', and 'get_report.' If dictionary, dictionary containing nested dictionaries of function names as they key, and the parameter names/values as key/value pairs for each key. If a function name is not present, or if a parameter name is not present, default values will be used. For example:

{ 'fetch_data' : {'source':'batch', 'trim_dir':"/path/to/trimmed/data", 'export_format':'mseed', 'detrend':'spline', 'plot_input_stream':True, 'ver-

bose':False, kwargs:{'kwargskey':'kwargsvalue'}} }

verbose
[bool, default=False] Whether to print output and results to terminal

Returns

params
[sprit.HVSRData] sprit.HVSRData class containing input parameters, including data file path and metadata path. This will be used as an input to other functions. If batch processing, params will be converted to batch type in fetch_data() step.

sprit.sprit_hvsr.**plot_azimuth**(*hvsr_data*, *fig=None*, *ax=None*, *show_azimuth_peaks=False*, *interpolate_azimuths=True*, *show_azimuth_grid=False*, ***plot_azimuth_kwargs*)

Function to plot azimuths when azimuths are calculated

Parameters

hvsr_data
[HVSRData or HVSRBatch] HVSRData that has gone through at least the sprit.fetch_data() step, and before sprit.generate_ppsds()

show_azimuth_peaks
[bool, optional] Whether to display the peak value at each azimuth calculated on the chart, by default False

interpolate_azimuths
[bool, optional] Whether to interpolate the azimuth data to get a smoother plot. This is just for visualization, does not change underlying data. It takes a lot of time to process the data, but interpolation for vizualization can happen fairly fast. By default True.

show_azimuth_grid
[bool, optional] Whether to display the grid on the chart, by default False

Returns

matplotlib.Figure, matplotlib.Axis
Figure and axis of resulting azimuth plot

sprit.sprit_hvsr.**plot_hvsr**(*hvsr_data*, *plot_type='HVSR ann p C+ ann p SPEC'*, *azimuth='HV'*, *use_subplots=True*, *fig=None*, *ax=None*, *return_fig=False*, *save_dir=None*, *save_suffix=''*, *show_legend=False*, *show=True*, *close_figs=False*, *clear_fig=True*, ***kwargs*)

Function to plot HVSR data

Parameters

hvsr_data
[dict] Dictionary containing output from process_hvsr function

plot_type
[str or list, default = 'HVSR ann p C+ ann p SPEC'] The plot_type of plot(s) to plot. If list, will plot all plots listed - 'HVSR' - Standard HVSR plot, including standard deviation. Options are included below:

- 'p' shows a vertical dotted line at frequency of the "best" peak
- 'ann' annotates the frequency value of of the "best" peak
- 'all' shows all the peaks identified in check_peaks() (by default, only the max is identified)
- 't' shows the H/V curve for all time windows

- 'tp' shows all the peaks from the H/V curves of all the time windows
- **'test' shows a visualization of the results of the peak validity test(s). Examples:**
  - 'tests' visualizes the results of all the peak tests (not the curve tests)
  - **'test12' shows the results of tests 1 and 2.**
    - Append any number 1-6 after 'test' to show a specific test result visualized
- **'COMP' - plot of the PPSD curves for each individual component ("C" also works)**
  - '+' (as a suffix in 'C+' or 'COMP+') plots C on a plot separate from HVSR (C+ is default, but without + will plot on the same plot as HVSR)
  - 'p' shows a vertical dotted line at frequency of the "best" peak
  - 'ann' annotates the frequency value of of the "best" peak
  - 'all' shows all the peaks identified in check_peaks() (by default, only the max is identified)
  - 't' shows the H/V curve for all time windows
- **'SPEC' - spectrogram style plot of the H/V curve over time**
  - 'p' shows a horizontal dotted line at the frequency of the "best" peak
  - 'ann' annotates the frequency value of the "best" peak
  - 'all' shows all the peaks identified in check_peaks()
  - 'tp' shows all the peaks of the H/V curve at all time windows
- **'AZ' - circular plot of calculated azimuthal HV curves, similar in style to SPEC plot.**
  - 'p' shows a point at each calculated (not interpolated) azimuth peak
  - 'g' shows grid lines at various angles
  - **'i' interpolates so that there is an interpolated azimuth at each degree interval (1 degree step)**
    This is the default, so usually 'i' is not needed.
  - '-i' prohibits interpolation (only shows the calculated azimuths, as determined by azimuth_angle (default = 30))

azimuth
: [str, default = 'HV'] What 'azimuth' to plot, default being standard N E components combined

use_subplots
: [bool, default = True] Whether to output the plots as subplots (True) or as separate plots (False)

fig
: [matplotlib.Figure, default = None] If not None, matplotlib figure on which plot is plotted

ax
: [matplotlib.Axis, default = None] If not None, matplotlib axis on which plot is plotted

return_fig
[bool] Whether to return figure and axis objects

save_dir
[str or None] Directory in which to save figures

save_suffix
[str] Suffix to add to end of figure filename(s), if save_dir is used

show_legend
[bool, default=False] Whether to show legend in plot

show
[bool] Whether to show plot

close_figs
[bool, default=False] Whether to close figures before plotting

clear_fig
[bool, default=True] Whether to clear figures before plotting

****kwargs**
[keyword arguments] Keyword arguments for matplotlib.pyplot

Returns

fig, ax
[matplotlib figure and axis objects] Returns figure and axis matplotlib.pyplot objects if return_fig=True, otherwise, simply plots the figures

sprit.sprit_hvsr.**plot_stream**(*stream*, *params*, *fig=None*, *axes=None*, *show_plot=False*, *ylim_std=0.75*, *return_fig=True*)

Function to plot a stream of data with Z, E, N components using matplotlib. Similar to obspy.Stream.Plot(), but will be formatted differently and eventually more customizable. This is also used in various functions throughout the package.

Parameters

stream
[obspy.core.Stream.stream] Obpsy stream of data with Z, E, N componenents

params
[HVSRData or HVSRBatch] Data object with parameters relevant for creating plot

fig
[matplotlib.Figure, default=None] Optional: if not None, matplotlib.Figure in which to plot the resulting figure (i.e., can be plotted in existing figure)

axes
[matplotlib.Axis, default=None] Optional: if not None, matplotlib.Axis in which to plot the resulting figure (i.e., can be plotted in existing axis)

show_plot
[bool, default=False] Whether to do matplotlib.pylot.show(), by default False

ylim_std
[float, default = 0.75] Optional: the standard deviation of the data at which to clip the chart, by default 0.75

return_fig
[bool, default=True] Optional: whether to return the figure, by default True

Returns

(matplotlib.Figure, matplotlib.Axes)
Tuple containing the figure and axes of the resulting plot, only returned if return_fig = True

sprit.sprit_hvsr.**process_hvsr**(*hvsr_data*, *method=3*, *smooth=True*, *freq_smooth='konno ohmachi'*, *f_smooth_width=40*, *resample=True*, *outlier_curve_rmse_percentile=False*, *verbose=False*)

Process the input data and get HVSR data

This is the main function that uses other (private) functions to do the bulk of processing of the HVSR data and the data quality checks.

Parameters

hvsr_data
[HVSRData or HVSRBatch] Data object containing all the parameters input and generated by the user (usually, during sprit.input_params(), sprit.fetch_data(), sprit.generate_ppsds() and/or sprit.remove_noise()).

method
[int or str, default=3]

Method to use for combining the horizontal components

0) (not used)

1) Diffuse field assumption, or 'DFA' (not currently implemented)

2) 'Arithmetic Mean': H (HN + HE)/2

3) 'Geometric Mean': H HN · HE, recommended by the SESAME project (2004)

4) 'Vector Summation': H H2 N + H2 E

5) 'Quadratic Mean': H (H2 N + H2 E)/2

6) 'Maximum Horizontal Value': H max {HN, HE}

smooth
[bool, default=True]

bool or int may be used.
If True, default to smooth H/V curve to using savgoy filter with window length of 51 (works well with default resample of 1000 pts) If int, the length of the window in the savgoy filter.

freq_smooth
[str {'konno ohmachi', 'constant', 'proportional'}]

Which frequency smoothing method to use. By default, uses the 'konno ohmachi' method.

- The Konno & Ohmachi method uses the obspy.signal.konnoohmachismoothing.konno_ohmachi_smoothing() function: https://docs.obspy.org/packages/autogen/obspy.signal.konnoohmachismoothing.konno_ohmachi_smoothing.html
- The constant method uses a window of constant length f_smooth_width
- The proportional method uses a window the percentage length of the frequncy steps/range (f_smooth_width now refers to percentage)

See here for more information: https://www.geopsy.org/documentation/geopsy/hv-processing.html

f_smooth_width
[int, default = 40]

- For 'konno ohmachi': passed directly to the bandwidth parameter of the konno_ohmachi_smoothing() function, determines the width of the smoothing peak, with lower values resulting in broader peak. Must be > 0.
- For 'constant': the size of a triangular smoothing window in the number of frequency steps
- For 'proportional': the size of a triangular smoothing window in percentage of the number of frequency steps (e.g., if 1000 frequency steps/bins and f_smooth_width=40, window would be 400 steps wide)

resample
[bool, default = True]

bool or int.
If True, default to resample H/V data to include 1000 frequency values for the rest of the analysis If int, the number of data points to interpolate/resample/smooth the component psd/HV curve data to.

outlier_curve_rmse_percentile
[bool, float, default = False] If False, outlier curve removal is not carried out here. If True, defaults to 98 (98th percentile). Otherwise, float of percentile used as rmse_thresh of remove_outlier_curve().

verbose
[bool, defualt=False] Whether to print output to terminal

Returns

hvsr_out
[dict] Dictionary containing all the information about the data, including input parameters

sprit.sprit_hvsr.**read_tromino_files**(*datapath*, *params*, *sampling_rate=128*, *start_byte=24576*, *verbose=False*, ***kwargs*)

Function to read data from tromino. Specifically, this has been lightly tested on Tromino 3G+ machines

Parameters

datapath
[str, pathlib.Path()] The input parameter _datapath_ from sprit.input_params()

params
[HVSRData or HVSRBatch] The parameters as read in from input_params() and and fetch_data()

verbose
[bool, optional] Whether to print results to terminal, by default False

Returns

obspy.Stream
An obspy.Stream object containing the trace data from the Tromino instrument

sprit.sprit_hvsr.**remove_noise**(*hvsr_data*, *remove_method='auto'*, *sat_percent=0.995*, *noise_percent=0.8*, *sta=2*, *lta=30*, *stalta_thresh=[8, 16]*, *warmup_time=0*, *cooldown_time=0*, *min_win_size=1*, *remove_raw_noise=False*, *show_stalta_plot=False*, *verbose=False*)

Function to remove noisy windows from data, using various methods.

Methods include - Manual window selection (by clicking on a chart with spectrogram and stream data), - Auto window selection, which does the following two in sequence (these can also be done indepently):

- A sta/lta "antitrigger" method (using stalta values to automatically remove triggered windows where there appears to be too much noise)
- A noise threshold method, that cuts off all times where the noise threshold equals more than (by default) 80% of the highest amplitude noise sample for the length specified by lta (in seconds)
- A saturation threshold method, that cuts off all times where the noise threshold equals more than (by default) 99.5% of the highest amplitude noise sample.

Parameters

hvsr_data
[dict, obspy.Stream, or obspy.Trace] Dictionary containing all the data and parameters for the HVSR analysis

remove_method
[str, {'auto', 'manual', 'stalta'/'antitrigger', 'saturation threshold', 'noise threshold', 'warmup'/'cooldown'/'buffer'/'warm_cool'}] The different methods for removing noise from the dataset. A list of strings will also work, in which case, it should be a list of the above strings. See descriptions above for what how each method works. By default 'auto.' If remove_method='auto', this is the equivalent of remove_method=['noise threshold', 'antitrigger', 'saturation threshold', 'warm_cool']

sat_percent
[float, default=0.995] Percentage (between 0 and 1), to use as the threshold at which to remove data. This is used in the saturation method. By default 0.995. If a value is passed that is greater than 1, it will be divided by 100 to obtain the percentage.

noise_percent
[float, default = 0.8] Percentage (between 0 and 1), to use as the threshold at which to remove data, if it persists for longer than time (in seconds (specified by min_win_size)). This is used in the noise threshold method. By default 0.8. If a value is passed that is greater than 1, it will be divided by 100 to obtain the percentage.

sta
[int, optional] Short term average (STA) window (in seconds), by default 2. For use with sta/lta antitrigger method.

lta
[int, optional] Long term average (STA) window (in seconds), by default 30. For use with sta/lta antitrigger method.

stalta_thresh
[list, default=[0.5,5]] Two-item list or tuple with the thresholds for the stalta antitrigger. The first value (index [0]) is the lower threshold, the second value (index [1] is the upper threshold), by default [0.5,5]

warmup_time
[int, default=0] Time in seconds to allow for warmup of the instrument (or while operator is still near instrument). This will renove any data before this time, by default 0.

cooldown_time
[int, default=0] Time in seconds to allow for cooldown of the instrument (or for when operator is nearing instrument). This will renove any data before this time, by default 0.

min_win_size
[float, default=1] The minumum size a window must be over specified threshold (in

seconds) for it to be removed

remove_raw_noise
[bool, default=False] If remove_raw_noise=True, will perform operation on raw data ('input_stream'), rather than potentially already-modified data ('stream').

verbose
[bool, default=False] Whether to print status of remove_noise

Returns

output
[dict] Dictionary similar to hvsr_data, but containing modified data with 'noise' removed

sprit.sprit_hvsr.**remove_outlier_curves**(*hvsr_data*, *rmse_thresh=98*, *use_percentile=True*, *use_hv_curve=False*, *show_outlier_plot=False*, *verbose=False*)

Function used to remove outliers curves using Root Mean Square Error to calculate the error of each windowed Probabilistic Power Spectral Density (PPSD) curve against the median PPSD value at each frequency step for all times. It calculates the RMSE for the PPSD curves of each component individually. All curves are removed from analysis.

Some abberant curves often occur due to the remove_noise() function, so this should be run some time after remove_noise(). In general, the recommended workflow is to run this immediately following the generate_ppsds() function.

Parameters

hvsr_data
[dict] Input dictionary containing all the values and parameters of interest

rmse_thresh
[float or int, default=98] The Root Mean Square Error value to use as a threshold for determining whether a curve is an outlier. This averages over each individual entire curve so that curves with very abberant data (often occurs when using the remove_noise() method), can be identified. Otherwise, specify a float or integer to use as the cutoff RMSE value (all curves with RMSE above will be removed)

use_percentile
[float, default=True] Whether rmse_thresh should be interepreted as a raw RMSE value or as a percentile of the RMSE values.

use_hv_curve
[bool, default=False] Whether to use the calculated HV Curve or the individual components. This can only be True after process_hvsr() has been run.

show_plot
[bool, default=False] Whether to show a plot of the removed data

verbose
[bool, default=False] Whether to print output of function to terminal

Returns

hvsr_data
[dict] Input dictionary with values modified based on work of function.

sprit.sprit_hvsr.**run**(*datapath*, *source='file'*, *azimuth_calculation=False*, *noise_removal=False*, *outlier_curves_removal=False*, *verbose=False*, ***kwargs*)

The sprit.run() is the main function that allows you to do all your HVSR processing in one simple step (sprit.run() is how you would call it in your code, but it may also be called using sprit.sprit_hvsr.run())

The datapath parameter of sprit.run() is the only required parameter. This can be either a single file, a list of files (one for each component, for example), a directory (in which case, all obspy-readable files will be added to an HVSRBatch instance), a Rasp. Shake raw data directory, or sample data.

Parameters

datapath
[str or filepath object that can be read by obspy] Filepath to data to be processed. This may be a file or directory, depending on what kind of data is being processed (this can be specified with the source parameter). For sample data, The following can be specified as the datapath parameter:

- Any integer 1-6 (inclusive), or the string (e.g., datapath="1" or datapath=1 will work)
- The word "sample" before any integer (e.g., datapath="sample1")
- The word "sample" will default to "sample1" if source='file'.
- If source='batch', datapath should be datapath='sample' or datapath='batch'. In this case, it will read and process all the sample files using the HVSRBatch class. Set verbose=True to see all the information in the sample batch csv file.

source
[str, optional] _description_, by default 'file'

azimuth
[bool, optional] Whether to perform azimuthal analysis, by default False.

verbose
[bool, optional] _description_, by default False

****kwargs**
Keyword arguments for the functions listed above. The keyword arguments are unique, so they will get parsed out and passed into the appropriate function.

input_params
[function name (not an actual parameter)] Function for designating input parameters for reading in and processing data See API documentation: [input_params()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.input_params)

datapath: any, default = '<no default>'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

site: any, default = 'HVSR Site'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

network: any, default = 'AM'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

station: any, default = 'RAC84'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

loc: any, default = '00'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

channels: any, default = ['EHZ', 'EHN', 'EHE']
See API documentation at link above or at *help(input_params)* for specifics.

acq_date: any, default = '2024-04-25'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

starttime: any, default = 2024-04-25T00:00:00.000000Z
See API documentation at link above or at *help(input_params)* for specifics.

endtime: any, default = 2024-04-25T23:59:59.999999Z
See API documentation at link above or at *help(input_params)* for specifics.

tzone: any, default = 'UTC'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

xcoord: any, default = -88.2290526
See API documentation at link above or at *help(input_params)* for specifics.

ycoord: any, default = 40.1012122
See API documentation at link above or at *help(input_params)* for specifics.

elevation: any, default = 755
See API documentation at link above or at *help(input_params)* for specifics.

input_crs: any, default = 'EPSG:4326'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

output_crs: any, default = 'EPSG:4326'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

elev_unit: any, default = 'feet'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

depth: any, default = 0
See API documentation at link above or at *help(input_params)* for specifics.

instrument: any, default = 'Raspberry Shake'
See API documentation at link above or at *help(sprit.input_params)* for specifics.

metapath: any, default = None
See API documentation at link above or at *help(input_params)* for specifics.

hvsr_band: any, default = [0.4, 40]
See API documentation at link above or at *help(input_params)* for specifics.

peak_freq_range: any, default = [0.4, 40]
See API documentation at link above or at *help(input_params)* for specifics.

processing_parameters: any, default = {}
See API documentation at link above or at *help(input_params)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(input_params)* for specifics.

fetch_data
[function name (not an actual parameter)] Fetch ambient seismic data from a source to read into obspy stream See API documentation: [fetch_data()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.fetch_data)

params: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

source: any, default = 'file'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

trim_dir: any, default = None
See API documentation at link above or at *help(fetch_data)* for specifics.

export_format: any, default = 'mseed'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

detrend: any, default = 'spline'
See API documentation at link above or at *help(sprit.fetch_data)* for specifics.

detrend_order: any, default = 2
See API documentation at link above or at *help(fetch_data)* for specifics.

update_metadata: any, default = True
See API documentation at link above or at *help(fetch_data)* for specifics.

plot_input_stream: any, default = False
See API documentation at link above or at *help(fetch_data)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(fetch_data)* for specifics.

kwargs: any, default = {}
See API documentation at link above or at *help(fetch_data)* for specifics.

calculate_azimuth
[function name (not an actual parameter)] Function to calculate azimuthal horizontal component at specified angle(s). Adds each new horizontal See API documentation: [calculate_azimuth()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.calculate_azimuth)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.calculate_azimuth)* for specifics.

azimuth_angle: any, default = 30
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

azimuth_type: any, default = 'multiple'
See API documentation at link above or at *help(sprit.calculate_azimuth)* for specifics.

azimuth_unit: any, default = 'degrees'
See API documentation at link above or at *help(sprit.calculate_azimuth)* for specifics.

show_az_plot: any, default = False
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

plot_azimuth_kwargs: any, default = {}
See API documentation at link above or at *help(calculate_azimuth)* for specifics.

remove_noise
[function name (not an actual parameter)] Function to remove noisy windows from data, using various methods. See API documentation: [remove_noise()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.remove_noise)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.remove_noise)* for specifics.

remove_method: any, default = 'auto'
See API documentation at link above or at *help(sprit.remove_noise)* for specifics.

sat_percent: any, default = 0.995
See API documentation at link above or at *help(remove_noise)* for specifics.

noise_percent: any, default = 0.8
See API documentation at link above or at *help(remove_noise)* for specifics.

sta: any, default = 2
See API documentation at link above or at *help(remove_noise)* for specifics.

lta: any, default = 30
See API documentation at link above or at *help(remove_noise)* for specifics.

stalta_thresh: any, default = [8, 16]
See API documentation at link above or at *help(remove_noise)* for specifics.

warmup_time: any, default = 0
See API documentation at link above or at *help(remove_noise)* for specifics.

cooldown_time: any, default = 0
See API documentation at link above or at *help(remove_noise)* for specifics.

min_win_size: any, default = 1
See API documentation at link above or at *help(remove_noise)* for specifics.

remove_raw_noise: any, default = False
See API documentation at link above or at *help(remove_noise)* for specifics.

show_stalta_plot: any, default = False
See API documentation at link above or at *help(remove_noise)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(remove_noise)* for specifics.

generate_ppsds
[function name (not an actual parameter)] Generates PPSDs for each channel See API documentation: [generate_ppsds()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.generate_ppsds)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.generate_ppsds)* for specifics.

azimuthal_ppsds: any, default = False
See API documentation at link above or at *help(generate_ppsds)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(generate_ppsds)* for specifics.

ppsd_kwargs: any, default = {}
See API documentation at link above or at *help(generate_ppsds)* for specifics.

process_hvsr
[function name (not an actual parameter)] Process the input data and get HVSR data See API documentation: [process_hvsr()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.process_hvsr)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.process_hvsr)* for specifics.

method: any, default = 3
See API documentation at link above or at *help(process_hvsr)* for specifics.

smooth: any, default = True
See API documentation at link above or at *help(process_hvsr)* for specifics.

freq_smooth: any, default = 'konno ohmachi'
See API documentation at link above or at *help(sprit.process_hvsr)* for specifics.

f_smooth_width: any, default = 40
See API documentation at link above or at *help(process_hvsr)* for specifics.

resample: any, default = True
See API documentation at link above or at *help(process_hvsr)* for specifics.

outlier_curve_rmse_percentile: any, default = False
See API documentation at link above or at *help(process_hvsr)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(process_hvsr)* for specifics.

remove_outlier_curves
[function name (not an actual parameter)] Function used to remove outliers curves using Root Mean Square Error to calculate the error of each See API documentation: [remove_outlier_curves()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.remove_outlier_curves)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.remove_outlier_curves)* for specifics.

rmse_thresh: any, default = 98
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

use_percentile: any, default = True
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

use_hv_curve: any, default = False
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

show_outlier_plot: any, default = False
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(remove_outlier_curves)* for specifics.

check_peaks
[function name (not an actual parameter)] Function to run tests on HVSR peaks to find best one and see if it passes quality checks See API documentation: [check_peaks()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.check_peaks)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.check_peaks)* for specifics.

hvsr_band: any, default = [0.4, 40]
See API documentation at link above or at *help(check_peaks)* for specifics.

peak_selection: any, default = 'max'
See API documentation at link above or at *help(sprit.check_peaks)* for specifics.

peak_freq_range: any, default = [0.4, 40]
See API documentation at link above or at *help(check_peaks)* for specifics.

azimuth: any, default = 'HV'
See API documentation at link above or at *help(sprit.check_peaks)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(check_peaks)* for specifics.

get_report
[function name (not an actual parameter)] Get a report of the HVSR analysis in a variety of formats. See API documentation: [get_report()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.get_report)

hvsr_results: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

report_format: any, default = ['print', 'csv', 'plot']
See API documentation at link above or at *help(get_report)* for specifics.

plot_type: any, default = 'HVSR p ann C+ p ann Spec'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

azimuth: any, default = 'HV'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

export_path: any, default = None
See API documentation at link above or at *help(get_report)* for specifics.

csv_overwrite_opt: any, default = 'append'
See API documentation at link above or at *help(sprit.get_report)* for specifics.

no_output: any, default = False
See API documentation at link above or at *help(get_report)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(get_report)* for specifics.

export_data
[function name (not an actual parameter)] Export data into pickle format that can be read back in using import_data() so data does not need to See API documentation: [export_data()](https://sprit.readthedocs.io/en/latest/sprit.html#sprit.export_data)

hvsr_data: any, default = '<output of previous function>'
See API documentation at link above or at *help(sprit.export_data)* for specifics.

export_path: any, default = None
See API documentation at link above or at *help(export_data)* for specifics.

ext: any, default = 'hvsr'
See API documentation at link above or at *help(sprit.export_data)* for specifics.

verbose: any, default = False
See API documentation at link above or at *help(export_data)* for specifics.

Returns

hvsr_results
[sprit.HVSRData or sprit.HVSRBatch object] If a single file/data point is being processed, a HVSRData object will be returned. Otherwise, it will be a HVSRBatch object. See their documention for more information.

Raises

RuntimeError
If the input parameter may not be read correctly. This is raised if the input_params() function fails. This raises an error since no other data processing or reading steps will be able to carried out correctly.

RuntimeError
If the data is not read/fetched correctly using fetch_data(), an error will be raised. This is raised if the fetch_data() function fails. This raises an error since no other data processing steps will be able to carried out correctly.

RuntimeError
If the data being processed is a single file, an error will be raised if generate_ppsds() does not work correctly. No errors are raised for remove_noise() errors (since that is an optional step) and the process_hvsr() step (since that is the last processing step) .

Notes

The sprit.run() function calls the following functions. This is the recommended order/set of functions to run to process HVSR using SpRIT. See the API documentation for these functions for more information: - input_params(): The datapath parameter of input_params() is the only required variable, though others may also need to be called for your data to process correctly. - fetch_data(): the source parameter of fetch_data() is the only explicit variable in the sprit.run() function aside from datapath and verbose. Everything else gets delivered to the correct function via the kwargs dictionary - remove_noise(): by default, the kind of noise removal is remove_method='auto'. See the remove_noise() documentation for more information. If remove_method is set to anything other than one of the explicit options in remove_noise, noise removal will not be carried out. - generate_ppsds(): generates ppsds for each component, which will be combined/used later. Any parameter of obspy.signal.spectral_estimation.PPSD() may also be read into this function. - remove_outlier_curves(): removes any outlier ppsd curves so that the data quality for when curves are combined will be enhanced. See the remove_outlier_curves() documentation for more information. - process_hvsr(): this is the main function processing the hvsr curve and statistics. See process_hvsr() documentation for more details. The hvsr_band parameter sets the frequency spectrum over which these calculations occur. - check_peaks(): this is the main function that will find and 'score' peaks to get a best peak. The parameter peak_freq_range can be set to limit the frequencies within which peaks are checked and scored. - get_report(): this is the main function that will print, plot, and/or save the results of the data. See the get_report() API documentation for more information. - export_data(): this function exports the final data output as a pickle file (by default, this pickle object has a .hvsr extension). This can be used to read data back into SpRIT without having to reprocess data.

sprit.sprit_hvsr.**test_function**()

1.1.4 sprit.sprit_jupyter_UI module

Functions to create jupyter notebook widget UI

sprit.sprit_jupyter_UI.**create_jupyter_ui**()

sprit.sprit_jupyter_UI.**get_default**(*func*, *param*)

1.1.5 sprit.sprit_utils module

sprit.sprit_utils.**assert_check**(*var*, *cond=None*, *var_type=None*, *error_message='Output not valid'*, *verbose=False*)

sprit.sprit_utils.**check_gui_requirements**()

sprit.sprit_utils.**check_mark**(*incolor=False*, *interminal=False*)

The default Windows terminal is not able to display the check mark character correctly. This function returns another displayable character if platform is Windows

sprit.sprit_utils.**check_tsteps**(*hvsr_data*)

Check time steps of PPSDS to make sure they are all the same length

sprit.sprit_utils.**check_xvalues**(*ppsds*)

Check x_values of PPSDS to make sure they are all the same length

sprit.sprit_utils.**checkifpath**(*filepath*, *sample_list=''*, *verbose=False*)

Support function to check if a filepath is a pathlib.Path object and tries to convert if not

Parameters

filepath
[str or pathlib.Path, or anything] Filepath to check. If not a valid filepath, will not convert and raises error

Returns

filepath
[pathlib.Path] pathlib.Path of filepath

sprit.sprit_utils.**format_time**(*inputDT*, *tzone='UTC'*)

Private function to format time, used in other functions

Formats input time to datetime objects in utc

Parameters

inputDT
[str or datetime obj] Input datetime. Can include date and time, just date (time inferred to be 00:00:00.00) or just time (if so, date is set as today)

tzone
[str='utc' or int {'utc', 'local'}]

Timezone of data entry.
If string and not utc, assumed to be timezone of computer running the process. If int, assumed to be offset from UTC (e.g., CST in the United States is -6; CDT in the United States is -5)

Returns

outputTimeObj
[datetime object in UTC] Output datetime.datetime object, now in UTC time.

sprit.sprit_utils.**get_char**(*in_char*)

Outputs character with proper encoding/decoding

sprit.sprit_utils.**has_required_channels**(*stream*)

sprit.sprit_utils.**make_it_classy**(*input_data*, *verbose=False*)

sprit.sprit_utils.**read_from_RS**(*dest*, *src='SHAKENAME@HOSTNAME:/opt/data/archive/YEAR/AM/STATION/'*, *opts='az'*, *username='myshake'*, *password='shakeme'*, *hostname='rs.local'*, *year='2023'*, *sta='RAC84'*, *sleep_time=0.1*, *verbose=True*, *save_progress=True*, *method='scp'*)

sprit.sprit_utils.**time_it**(*_t*, *proc_name=''*, *verbose=True*)

Computes elapsed time since the last call.

sprit.sprit_utils.**x_mark**(*incolor=False*, *inTerminal=False*)

The default Windows terminal is not able to display the check mark character correctly. This function returns another displayable character if platform is Windows

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

H

I

K

L

M

O

P

R

S

T

X