Package 'eseis'

November 24, 2024

Type Package

Title Environmental Seismology Toolbox

Version 0.8.0

Date 2024-11-24

Maintainer Michael Dietze <michael.dietze@uni-goettingen.de>

Description Environmental seismology is a scientific field that studies the seismic signals, emitted by Earth surface processes. This package provides all relevant functions to read/write seismic data files, prepare, analyse and visualise seismic data, and generate reports of the processing history.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

LinkingTo Rcpp

Imports terra, caTools, signal, fftw, matrixStats, graphics, methods, XML, shiny, rmarkdown, colorspace, reticulate, extraDistr, minpack.lm, Rcpp

Suggests plot3D, rgl, seewave

SystemRequirements gipptools dataselect

RoxygenNote 7.3.2

NeedsCompilation yes

Author Michael Dietze [cre, aut, trl],
Christoph Burow [ctb],
Sophie Lagarde [ctb, trl],
Clement Hibert [ctb, aut]

Repository CRAN

Date/Publication 2024-11-24 13:40:02 UTC

Contents

`eseis-package` *eseis: Environmental Seismology Toolbox*

Description

Environmental seismoloy studies the seismic signals, emitted by Earth surface processes. This package eseis provides all relevant functions to read/write seismic data files, prepare, analyse and visualise seismic data, and generate reports of the processing history.

Author(s)

Maintainer: Michael Dietze <michael.dietze@uni-goettingen.de> [translator]

Authors:

- Clement Hibert [contributor]

Other contributors:

- Christoph Burow [contributor]
- Sophie Lagarde [contributor, translator]

`aux_checkfiles` *Check structured seismic files for consistency*

Description

The function checks seismic files organised by `aux_organisecubefiles` or `aux_organisecentaurfiles` for completeness. The tests include agreement of file name and seismic file meta data.

Usage

```
aux_checkfiles(
  dir,
  station,
  component = "BHZ",
  method = "thorough",
  period = "weekly",
  format = "sac",
  duration_set = 3600,
  plot = TRUE
)
```

Arguments

dir	Character value, path to directorythat contains the seismic files to check.
station	Character value, ID of the station of which the files will be checked. ID must match the IDs as used in the file names.
component	Character value, seismic component, which must correspond to the component name in the file name of the data directory structure (cf. aux_organisecubefiles). Default is "BHZ" (vertical component of a sac file).
method	Character value, method used to check files. One out of "fast" (only file names are used to check the files, the actual files are not read and examined) and "thorough" (files are imported to R and their meta data will be used for checking, as well). Option "thorough" is many times slower. Default is "thorough".
period	Character value, aggregation period, i.e., the time period used to generate plots. One out of "total","yearly", "monthly", "weekly", "daily". Default is "weekly".
format	Character value, seismic data format. One out of "sac" and "mseed". Default is "sac".
duration_set	Numeric value, anticipated length of the seismic time series of the files to test in seconds. Default is 3600 (one hour).
plot	Logical value, option to visualise the output of the function. Default is TRUE.

Value

Data frame containing check results and meta data.

Author(s)

Michael Dietze

Examples

```
## Not run:

## set seismic data directory
dir_data <- paste0(system.file("extdata", package="eseis"), "/")

## check data archive for record completeness
chk <- aux_checkfiles(dir = dir_data,
                      station = "RUEG1",
                      component = "BHZ",
                      plot = TRUE)


## End(Not run)
```

aux_commondt *Identify highest common sampling interval*

Description

The function compares the sampling intervals of a list of eseis objects and identifies the highest common sampling interval (dt) as well as the aggregation factors for each eseis object needed to reach this common sampling interval.

Usage

```
aux_commondt(data, dt)
```

Arguments

data	list of eseis objects or vector of sampling intervals to be checked for highest common sampling interval
dt	Numeric vector of length one, user-defined common sampling frequency for which aggregation factors shall be computed.

Value

list object with elements dt (highest common sampling interval) and agg (aggregation factors for each of the input data sets to reach the common sampling interval)

Author(s)

Michael Dietze

Examples

```
## Not run:

## TO BE WRITTEN

## End(Not run)
```

aux_cubeinfo *Get cube file information*

Description

This is a simple wrapper for the Gipptools program `cubeinfo`, providing a short summary of the cube file meta data, in a coherent data frame structure.

Usage

```
aux_cubeinfo(file, gipptools)
```

Arguments

`file`	`Characater` value, cube file to be processes
`gipptools`	`Character` value, path to gipptools or Gipptools directory.

Value

`data frame` with cube meta data

Author(s)

Michael Dietze

Examples

```
## Not run:

## get cube info
x = aux_cubeinfo(file = "data/cube/example.ATB",
                 gipptools = "/software/gipptools-2019.332/")


## End(Not run)
```

aux_eseisobspy *Convert eseis object to ObsPy stream object*

Description

The function converts an eseis object to an ObsPy stream object. The functionality is mainly useful when running ObsPy through R using the package 'reticulate'. Currently, only single traces (i.e., single eseis objects) can be converted. Thus, to convert multiple traces, these need to be converted individually and added to the first trace using ObsPy functionalities.

Usage

```
aux_eseisobspy(data)
```

Arguments

data	eseis object, list element.

Value

ObsPy stream object as defined by the architecture of package 'reticulate'.

Author(s)

Michael Dietze

Examples

```
## Not run:

## load ObsPy library with package 'reticulate'
## (requires ObsPy to be installed on the computer)
obspy <- reticulate::import("obspy")

## load example data set
data(rockfall)

## convert example eseis object to ObsPy stream object
x <- aux_eseisobspy(data = rockfall_eseis)

## filter data set using ObsPy
x_filter <- obspy$traces[[1]]$filter(type = "bandpass",
                                     freqmin = 0.5,
                                     freqmax = 1.0)

## plot filtered trace using ObsPy plotting routine
x$traces[[1]]$plot()


## End(Not run)
```

---

aux_fixmseed *Fix corrupt miniseed files*

Description

This function is a wrapper for the library 'dataselect' from IRIS. It reads a corrupt mseed file and saves it in fixed state. Therefore, the function requires dataselect being installed (see details).

Usage

```
aux_fixmseed(file, input_dir, output_dir, software)
```

Arguments

`file`	`Character` vector, seismic file to process.
`input_dir`	`Character` value, path to input directory, i.e., the directory where the files to process are located.
`output_dir`	`Character` value, path to output directory, i.e., the directory where the processed files are written to. This must be different from `input_dir`.
`software`	`Character` value, path to the 'dataselect' library, required unless the path to the library is made gobally visible.

Details

The library 'dataselect' can be downloaded at https://github.com/iris-edu/dataselect and requires compilation (see README file in dataselect directory). The function goes back to an email discussion with Gillian Sharer (IRIS team), many thanks for pointing me at this option to process corrupt mseed files.

Value

a set of mseed files written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:

aux_fixmseed(file = list.files(path = "~/data/mseed",
                               pattern = "miniseed"),
                    input_dir = "~/data/mseed",
                    software = "~/software/dataselect-3.17")


## End(Not run)
```

aux_getevent *Load seismic data of a user-defined event*

Description

The function loads seismic data from a data directory structure (see aux_organisecubefiles()) based on the event start time, duration, component and station ID.

Usage

```
aux_getevent(
  start,
  duration,
  station,
  component = "BHZ",
  format,
  dir,
  simplify = TRUE,
  eseis = TRUE,
  try = FALSE,
  verbose = FALSE
)
```

Arguments

start	POSIXct value, start time of the data to import. If lazy users only submit a text string instead of a POSIXct obejct, the function will try to convert that text string.
duration	Numeric value, duration of the data to import, in seconds.
station	Character value, seismic station ID, which must correspond to the ID in the file name of the data directory structure (cf. aux_organisecubefiles).
component	Character value, seismic component, which must correspond to the component name in the file name of the data directory structure (cf. aux_organisecubefiles). Default is "BHZ" (vertical component of a sac file).
format	Character value, seismic data format. One out of "sac" and "mseed". If omitted, the function will try to identify the right format automatically.
dir	Character value, path to the seismic data directory. See details for further info on data structure.
simplify	Logical value, option to simplify output when possible. This basically means that if only data from one station is loaded, the list object will have one level less. Default is TRUE.
eseis	Logical value, option to read data to an eseis object (recommended, see documentation of aux_initiateeseis), default is TRUE
try	Logical value, option to run the function in try-mode, i.e., to let it return NA in case an error occurs during data import. Default is FALSE.

verbose — Logical value, option to show messages during function execution. Default is FALSE.

Details

The data to be read needs to be adequately structured. The data directory must contain SAC files organised by year (e.g.2022) then by Julian Day in full three digits (e.g. 001) and then by a dedicated SAC file name, containing the station ID, two-digit year, three-digit Julian Day, start time hour, minute and second, three channel ID and the file extension SAC. All these items need to be separated by stops (e.g. sac/2022/001/LAU01.22.001.08.00.00. BHZ.SAC). This data structure will be most conveniently created by the functions `aux_organisecubefiles()` or `aux_organisecentaurfiles()`, or by manually written R code.

The function assumes complete data sets, i.e., not a single hourly data set must be missing. The time vector is loaded only once, from the first station and its first component. Thus, it is assumed that all loaded seismic signals are of the same sampling frequency and length.

Value

A `list` object containing either a set of `eseis` objects or a data set with the time vector (`$time`) and a list of seismic stations (`$station_ID`) with their seismic signals as data frame (`$signal`). If `simplify = TRUE` (the default option) and only one seismic station is provided, the output object containseither just one eseis object or the vectors for `$time` and `$signal`.

Author(s)

Michael Dietze

Examples

```
## set seismic data directory
dir_data <- paste0(system.file("extdata", package="eseis"), "/")

## load the z component data from a station
data <- aux_getevent(start = as.POSIXct(x = "2017-04-09 01:20:00",
                                        tz = "UTC"),
                     duration = 120,
                     station = "RUEG1",
                     component = "BHZ",
                     dir = dir_data)
## plot signal
plot_signal(data = data)

## load data from two stations
data <- aux_getevent(start = as.POSIXct(x = "2017-04-09 01:20:00",
                                        tz = "UTC"),
                    duration = 120,
                    station = c("RUEG1", "RUEG2"),
                    component = "BHZ",
                    dir = dir_data)

## plot both signals
```

```
par(mfcol = c(2, 1))
lapply(X = data, FUN = plot_signal)
```

---

aux_getFDSNdata *Download seismic data from FDSN data base*

Description

The function accesses the specified FDSN internet data base(s) and downloads seismic data based on the network and station IDs and time constraints.

Usage

```
aux_getFDSNdata(
  start,
  duration,
  channel = "BHZ",
  network,
  station,
  url,
  link_only = FALSE,
  eseis = TRUE
)
```

Arguments

start	POSIXct value, start time of the data to query.
duration	Numeric value, length of the data to query, in seconds.
channel	Character value, seismic channel to get. Default is "BHZ".
network	Character vector, two-character FDSN network ID.
station	Character vector, FDSN station ID.
url	Character vector, FDSN URL.
link_only	Logical vector, return only FDSN link instead of downloading and importing the data.
eseis	Logical scalar, option to read data to an eseis object (recommended, see documentation of aux_initiateeseis), default is TRUE

Details

A convenient way to get all the required input data is using the function aux_getFDSNstation before. It will return all the information in a structured way.

It is possible to use the function to process more than one data set. In this case, the arguments network, station and url must match pairwise. The arguments start, duration and channel will be treated as constants if not also provided as vectors.

Value

List object with imported seismic data for each provided set of input arguments.

Author(s)

Michael Dietze

See Also

aux_get_FDSNstation, read_mseed

Examples

```
## Not run:

## get stations < 0.6 degrees away from Piz Chengalo collapse
x <- aux_getFDSNstation(centre = c(46.3, 9.6),
                        radius = 0.6,
                        access = TRUE)

## sort statiions by distance
x <- x[order(x$distance),]

## download available data
d <- aux_getFDSNdata(start = as.POSIXct(x = "2017-08-23 07:30:00",
                                        tz = "UTC"),
                     duration = 180,
                     network = x$network_ID,
                     station = x$station_code,
                     url = x$network_url)

## remove stations without available data
x <- x[!unlist(lapply(d, is.null)),]
d <- d[!unlist(lapply(d, is.null))]

## generate plots of the three nearest stations
par(mfcol = c(3, 1))

for(i in 1:3) {

  plot_signal(data = d[[i]],
              main = paste(x$ID[i],
                           " | ",
                           round(x$distance[i], 2),
                           "distance (DD)"))
}

## End(Not run)
```

aux_getFDSNstation *Query FDSN data base for stations*

Description

This function queries as series of data bases for seismic stations that match a set of criteria for seismic data. The criteria include signal time stamp and location, and component. The returned data can be used to download data using the function aux_FDSNdata.

Usage

```
aux_getFDSNstation(centre, radius, start, access, url)
```

Arguments

centre	Numeric vector of length two, center coordinates of the location to search data for (c(latitude, longitude)). Units must be decimal degrees.
radius	Numeric value, radius within which to search for seismic stations. Unit must be decimal degrees.
start	POSIXct value, start time of the data to query. If omitted, stations are queried for the full time available.
access	Logical value, access type of the data. If omitted, all data sets are returned, if set TRUE, only data with access flag "open" are returned.
url	Character vector, optional other FDSN base web addresses to search for stations. See details for default addresses and their format.

Details

The function requires a working internet connection to perform the query. It uses the following FDSN data bases by default:

- orfeus "http://www.orfeus-eu.org"
- geofon "http://geofon.gfz-potsdam.de/"
- bgr "http://eida.bgr.de"
- sss "http://eida.ethz.ch"

Other FDSN data base addresses can be provided in the same way as the addresses in the above list.
They need to be provided as character vector. For a list of addresses see "http://www.fdsn.org/webservices/datacenter and "http://docs.obspy.org/packages/obspy.clients.fdsn.html#module-obspy.clients.fdsn".

Value

Data frame with query results. The data frame contains information for all seismic stations fulfilling the defined criteria.

Author(s)

Michael Dietze

See Also

aux_get_FDSNdata, aux_getIRISstation

Examples

```
## Not run:

x <- aux_getFDSNstation(start = as.POSIXct(x = "2010-01-01 22:22:22",
                                           tz = "UTC"),
                        centre = c(45, 10),
                        radius = 1)

## optionally plot station locations on a map (requires RgoogleMaps)
center <- c(mean(x$station_latitude),
            mean(x$station_longitude))

zoom <- min(RgoogleMaps::MaxZoom(range(x$station_latitude),
                                 range(x$station_longitude)))

Map <- RgoogleMaps::GetMap(center = center,
                           zoom = zoom,
                           maptype = "terrain")

RgoogleMaps::PlotOnStaticMap(MyMap = Map,
                             lat = x$station_latitude,
                             lon = x$station_longitude,
                             pch = 15,
                             col = 4)

## End(Not run)
```

---

aux_gettemperature *Extract temperature data from cube files.*

Description

This function reads auxiliary information stored in Omnirecs/Digos Datacube files and extracts the temperature data that is stored along with each GPS tag. Optionally, the data is interpolated to equal intervals.

Usage

```
aux_gettemperature(input_dir, logger_ID, interval, cpu, gipptools)
```

Arguments

input_dir	Character value, path to directory where all cube files to be processed as stored. Each set of files from one logger must be stored in a separate sub-directory named after the cube ID.
logger_ID	Character vector, logger ID.
interval	Numeric value, time interval (minutes) to which temperature data is interpolated. No interpolation is performed if this argument is omitted.
cpu	Numeric value, fraction of CPUs to use for parallel processing. If omitted, one CPU is used.
gipptools	Character value, path to gipptools or cubetools directory.

Details

This feature is ony available for Omnirecs/Digos Datacube that were produced since 2015, i.e., whose GPS output files also record the temperature inside the logger. Generating an ACSII GPS tag file using the gipptools software requires a few minutes time per daily file.

Value

A list of data frames with time and temperature values for each cube data logger.

Author(s)

Michael Dietze

Examples

```
## uncomment to use
# t <- aux_gettemperature(input_dir = "input",
#                         logger_ID = c("ANN", "ABT"),
#                         interval = 15,
#                         gipptools = "~/software/gipptools-2015.225/")
```

---

aux_getxml *Download and/or read station XML file*

Description

This function either downloads an online station XML file and (optionally) saves it and/or reads an already existing local station XML file.

Usage

```
aux_getxml(
  xml,
  start,
  duration,
  network,
  station,
  component,
  url,
  level = "response"
)
```

Arguments

xml	Character value, local XML file to import. If omitted, the function expects information for the other arguments listed below to download (into a temporary directory) and read the station XML file. In case both, the below information and a XML path is provided, the downloaded data will be saved under the provided XML file name.
start	POSIXct value, start time for which the XML information shall be collected. If omitted, XML information for all available time spans will be downloaded.
duration	Numeric value, duration until when the XML information shall be collected. If omitted, XML information for all available time spans will be downloaded.
network	Character value, seismic network to which the station belongs, for which the XML data will be downloaded.
station	Character value, seismic station ID for which the XML data will be downloaded. If omitted, XML information for all stations of the network will be collected.
component	Character value, component for which the XML data will be downloaded. If omitted, XML data will be collected for all components available for the specified stations.
url	Chracter vector, URL of the FDSN data provider. Should be of the form "http://service.iris.edu", i.e., without further URL parts. URLs can be submitted as a vector.
level	Character value, level of station XML information depth. Default is "response".

Details

Currently, the function uses Obspy python code. Hence, both python and the package 'obspy' need to be installed in order to use the function.

Value

Obspy object with station meta info inventory.

Author(s)

Michael Dietze

Examples

```
## Not run:

x <- aux_getxml(start = "2010-10-10",
                duration = 60,
                network = "GE",
                station = "BRNL",
                component = "BHZ",
                url = "http://service.iris.edu")


## End(Not run)
```

`aux_hvanalysis` *Perform H-V-spectral ratio analysis of a seismic data set*

Description

This function cuts a three component seismic data set into time windows that may or may not overlap and calculates the spectral ratio for each of these windows. It returns a matrix with the ratios for each time slice. Thus, it is a wrapper for the function `signal_hvratio`. For further information about the technique and function arguments see the description of `signal_hvratio`.

Usage

```
aux_hvanalysis(
  data,
  time,
  window,
  overlap = 0,
  dt,
  method = "periodogram",
  kernel,
  order = "xyz",
  plot = FALSE,
  ...
)
```

Arguments

`data`	`List`, `data frame` or `matrix`, seismic componenents to be processed. If `data` is a matrix, the components must be organised as columns. Also, `data` can be a list of `eseis` objects.
`time`	`POSIXct` vector with time values. If omitted, an synthetic time vector will be created, based on `dt`.

window	Numeric scalar, time window length in seconds used to calculate individual spectral ratios. Set to 10 percent of the time series length by default.
overlap	Numeric value, fraction of window overlap.
dt	Numeric value, sampling period.
method	Character value, method for calculating the spectra. One out of "periodogram" , "autoregressive" and "multitaper", default is "periodogram".
kernel	Numeric value, window size (defined by number of samples) of the moving window used for smoothing the spectra. By default no smoothing is performed.
order	Character value, order of the seismic components. Describtion must contain the letters "x","y" and "z" in the order according to the input data set. Default is "xyz" (NW-SE-vertical).
plot	Logical value, toggle plot output. Default is FALSE.
...	Additional arguments passed to the plot function.

Value

A matrix with the h-v-frequency ratios for each time slice.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(earthquake)

## ATTENTION, THIS EXAMPLE DATA SET IS FAR FROM IDEAL FOR THIS PURPOSE

## detrend data
s <- signal_detrend(data = s)

## calculate the HV ratios straightforward
HV <- aux_hvanalysis(data = s,
                     dt = 1 / 200,
                     kernel = 100)

## calculate the HV ratios with plot output, userdefined palette
plot_col <- colorRampPalette(colors = c("grey", "darkblue", "blue", "orange"))
HV <- aux_hvanalysis(data = s,
                     dt = 1 / 200,
                     kernel = 100,
                     plot = TRUE,
                     col = plot_col(100))

## calculate the HV ratios with optimised method settings
HV <- aux_hvanalysis(data = s,
                     time = t,
                     dt = 1 / 200,
```

```
                    window = 10,
                    overlap = 0.9,
                    method = "autoregressive",
                    plot = TRUE,
                    col = plot_col(100),
                    xlab = "Time (UTC)",
                    ylab = "f (Hz)")

## calculate and plot stack (mean and sd) of all spectral ratios
HV_mean <- apply(X = HV, MARGIN = 1, FUN = mean)
HV_sd <- apply(X = HV, MARGIN = 1, FUN = sd)
HV_f <- as.numeric(rownames(HV))

plot(x = HV_f, y = HV_mean, type = "l", ylim = c(0, 50))
lines(x = HV_f, y = HV_mean + HV_sd, col = 2)
lines(x = HV_f, y = HV_mean - HV_sd, col = 2)
```

`aux_initiateeseis` *Initiate an eseis object*

Description

The function generates an empty eseis object, starting with processing step 0. The object contains no data and the history only contains the system information.

Usage

```
aux_initiateeseis()
```

Details

The S3 object class `eseis` contains the data vector (`$signal`), a meta information list (`$meta`) with all essential seismic meta data - such as sampling interval, station ID, component, start time of the stream or file name of the input file - a list with header data of the seismic source file (`$header`), and a history list (`$history`), which records all data manipulation steps of an (`eseis`) object. The element (`$meta`) will be used by functions of the package to look for essential information to perform data manipulations (e.g., the sampling interval). Thus, working with (`eseis`) objects is convenient and less prone to user related errors/bugs, given that the meta information is correct and does not change during the processing chain; package functions will update the meta information whenever necessary (e.g., `signal_aggregate`). The element `$header` will only be present if a seismic source file has been imported.

The history element is the key feature for transparent and reproducable research using this R package. An `eseis` object records a history of every function it has been subject to, including the time stamp, the function call, all used function arguments and their associated values, and the overall processing duration in seconds. The history is updated whenever an `eseis` object is manipulated with one of the functions of this package (with a few exceptions, mainly from the aux_... category).

Value

S3 list object of class eseis.

Author(s)

Michael Dietze

Examples

```
## initiate eseis object
aux_initiateeseis()
```

---

aux_obspyeseis *Convert ObsPy object to eseis object*

Description

The function converts an ObsPy stream object to an eseis object. The functionality is mainly useful when running ObsPy through R using the package 'reticulate'.

Usage

```
aux_obspyeseis(data, simplify = TRUE)
```

Arguments

data	obspy stream object, list element, created by running ObsPy through R using the package 'reticulate'.
simplify	Logical value, option to simplify output when possible. This basically means that if there is only trace object in the ObsPy stream, the list object will have one level less. Default is TRUE.

Details

Initiation of the reticulate-based python toolbox support can be cumbersome. The following suggestions from Guthub (https://github.com/rstudio/reticulate/issues/578) helped in a case study:

library(reticulate) use_condaenv("r-reticulate") py_install("obspy", pip = TRUE)

Value

eseis object.

Author(s)

Michael Dietze

Examples

```
## Not run:

## load ObsPy library with package 'reticulate'
## (requires ObsPy to be installed on the computer)
obspy <- reticulate::import("obspy")

## set seismic data directory
dir_data <- paste0(system.file("extdata", package="eseis"), "/")

## read miniseed file to stream object via ObsPy
x <- obspy$read(pathname_or_url = "dir_data/2017/99/RUEG1.17.99.00.00.00.BHZ.SAC")

## convert ObsPy stream object to eseis object
y <- aux_obspyeseis(data = x)

## plot eseis object
plot_signal(y)

## End(Not run)
```

---

aux_organisecentaurfiles

Reorganise seismic files recorded by Nanometrics Centaur loggers

---

Description

This function optionally converts mseed files to sac files and organises these in a coherent directory structure, by year, Julian day, (station, hour and channel). It depends on the cubetools or gipptools software package (see details). The function is at an experimental stage and only used for data processing at the GFZ Geomorphology section, currently.

Usage

```
aux_organisecentaurfiles(
  stationfile,
  input_dir,
  output_dir,
  format = "sac",
  channel_name = "bh",
  cpu,
  gipptools,
  file_key = "miniseed",
  network
)
```

Arguments

stationfile	Character value, file name of the station info file, with extension. See aux_stationinfofile.
input_dir	Character value, path to directory where all files to be processed as stored. Each set of files from one logger must be stored in a separate sub-directory named after the logger ID (which in turn must be the four digit number of the logger).
output_dir	Character value, path to directory where output data is written to.
format	Character value, output file format. One out of "mseed" and "sac". Default is "sac".
channel_name	Character value, output file extension. One out of "bh" and "p". Default is "bh".
cpu	Numeric value, fraction of CPUs to use for parallel processing. If omitted, one CPU is used.
gipptools	Character value, path to gipptools or cubetools directory.
file_key	Character value, file name extension of the files to process. Only files with this extension will be processed. Default is "miniseed".
network	Character value, optional seismic network code.

Details

The function assumes that the Nanometrics Centaur data logger directory contains only hourly mseed files. These hourly files are organised in a coherent directory structure which is organised by year and Julian day. In each Julian day directory the hourly files are placed and named according to the following scheme: STATIONID.YEAR.JULIANDAY.HOUR.MINUTE.SECOND.CHANNEL.
The function requires that the software cubetools (http://www.omnirecs.de/documents.html) or gipptools (http://www.gfz-potsdam.de/en/section/geophysical-deep-sounding/infrastructure/geophysical are installed.
Specifying an input directory (input_dir) is mandatory. This input directory must only contain the subdirectories with mseed data for each Centaur logger. The subdirectory must be named after the four digit Centaur ID and contain only mseed files, regardless if further subdirectories are used (e.g., for calendar days).

In the case a six-channel Centaur is used to record signals from two sensors, in the station info file (cf. aux_stationinfofile()) the logger ID field must contain the four digit logger ID and the channel qualifiers, e.g., "AH" (first three channels) or "BH" (last three channels), separated by an underscore.

Value

A set of hourly seismic files written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:

## basic example with minimum effort
aux_organisecentaurfiles(stationfile = "output/stationinfo.txt",
                         input_dir = "input",
                         gipptools = "software/gipptools-2015.225/")


## End(Not run)
```

---

aux_organisecubefiles *Convert Datacube files and organise them in directory structure*

Description

The function converts Omnirecs/Digos Datacube files to mseed or sac files and organises these in a coherent directory structure (see details) for available structures. The conversion depends on the gipptools software package (see details) provided externally.

Usage

```
aux_organisecubefiles(
  station,
  input,
  output,
  gipptools,
  format = "sac",
  pattern = "eseis",
  component = "BH",
  mode = "dir-wise",
  fringe = "constant",
  cpu,
  verbose = TRUE
)
```

Arguments

station	data frame with seismic station information See aux_stationinfofile. This data frame can also be provided manually, in which case it must contain two elements: a first vector that contains the Cube IDs, and a second that contains the corresponding station IDs that will be used in the meta info and file names. If the argument is omitted, the Cube IDs will be used as station IDs.
input	Character value, path to directory where the Cube files to be processed are stored.
output	Character value, path to directory where output data is written to.

gipptools	Character value, path to gipptools or cubetools directory.
format	Character value, output file format. One out of "mseed" and "sac". Default is "sac".
pattern	Character value, file organisation scheme keyword. One out of "eseis" and "seiscomp". Default is "eseis". See details and read_data for further information.
component	Character vector, component code and output file extension prefix. It is assumed that this prefix comprises two characters, the first describing the band code, the second the instrument code. See details for further information. Default is "BH", hence broadband and high gain sensor. The spatial component ("E", "N", "Z") will be added automatically. See details for a tabular overview of common band codes and instrument codes.
mode	Character value, mode of file conversion. One out of "file-wise" and "dir-wise". Default is "file-wise". See details for further important information.
fringe	Character value, option to handle data outside the GPS-tagged time span. One out of "skip", "nominal" or "constant". Default is "constant".
cpu	Numeric value, fraction of CPUs to use for parallel processing. If omitted, one CPU is used.
verbose	Logical value, option to enable extended screen output of cubetools operations. Default is FALSE. This option might not work with Windows operating systems.

Details

The function converts seismic data from the binary cube file format to mseed (cf. read_mseed) or sac (cf. read_sac) and organises the resulting files into a consistent structure, expected by 'eseis' for convenient data handling (cf. read_data).

Currently, there are two data structure schemes supported, "eseis" and "seiscomp". In the "eseis" case, the daily cube files are cut to hourly files and organised in directories structured by four digit year and three digit Julian day numbers. In each Julian day directory, the hourly files are placed and named after the following scheme: STATION.YEAR.JULIANDAY.HOUR.MINUTE.SECOND.COMPONENT.

The "seiscomp" case will yield daily files, which are organised by four digit year, seismic network, seismic station, and seismic component, each building a separate directory. In the deepest subdirectory, files are named by: NETWORK.STATION.LOCATION.COMPONENT.TYPE.YEAR.JULIANDAY.

The component naming scheme defines the codes for the sensor's band code (first letter) and instrument code (second letter). The third letter, defining the spatial component, will be added automatically. For definitions of channel codes see https://migg-ntu.github.io/SeisTomo_Tutorials/seismology/seismic-d

The function requires that the software gipptools (http://www.gfz-potsdam.de/en/section/geophysical-deep-soundi is installed. Note that the gipptools are provided at regular update intervals, including an up to date GPS leap second table, essential to convert recently recorded files.

The Cube files will be imported in place but a series of temporary files will be created in a temporary directory in the specified output directory. Hence, if the routine stops due to a processing issue, one needs to delete the temporary data manually. The path to the temporary directory will be provided as screen output when the argument verbose = TRUE.

The Cube files can be converted in two modes: "file-wise" and "dir-wise". In "file-wise" mode, each Cube file will be converted individually. This option has the advantage that if one file

in a month-long sequence of records is corrupt, the conversion will not stop, but only discard the part from the corrupted section until the file end. The disadvantage is however, that the data before the first and after the last GPS tags will not be converted unless the option `fringe = "constant"` (by default this is the case) is used.

In `"dir-wise"` mode, the fringe sample issue reduces to the margins of the total sequence of daily files but the corrupt file issue will become a more severe danger to the success when converting a large number of files.

Specifying an input directory (`input`) is mandatory. That directory should only contain the directories with the cube files to process. Files downloaded from a Cube are usually contained in one or more further directories, which should be moved into a single one before running this function.

Each set of cube files from a given logger should be located in a separate directory per logger and these directories should have the same name as the logger IDs (`logger_ID`). An appropriate structure for files from two loggers, A1A and A1B, would be something like:

1. input
   (a) A1A
      i. file1.A1A
      ii. file2.A1A
   (b) A1B
      i. file1.A1B
      ii. file2.A1B

The component definition can follow the typical keywords and key letters defined in seismology: `https://migg-ntu.github.io/SeisTomo_Tutorials/seismology/seismic-data/seismic-time-series-data.html` hence the first letter indicating the instrument's band type and the second letter indicating the instrument code or instrument type.

Band code	Explanation band type
E	Extremely short period
S	Short period
H	High broad band
B	Broad band
M	Mid band
L	Long band
V	Very long band

Instrument code	Explanation
H	High gain seismometer
L	Low gain seismometer
G	Gravimeter
M	Mass position seismometer
N	Accelerometer
P	Geophone

Value

A set of converted and organised seismic files written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:

## basic example with minimum effort
aux_organisecubefiles(stationfile = data.frame(logger = c("A1A", "A1B"),
                                               station = c("ST1", "ST2")),
                      input = "input",
                      gipptools = "software/gipptools-2023.352")


## End(Not run)
```

`aux_picknetwork` *Pick seismic events with a sensor network approach*

Description

The function allows detection of discrete seismic events using the STA-LTA picker approach and including several stations of a seismic network. It will pick events at all input stations and identify events picked by more than a user defined minimum number of stations, within a given time window. It further allows rejecting events that are too short or too long, or occur to short after a previous event. The function can be used to process long data archives organised in a consistent file structure.

Usage

```
aux_picknetwork(
  start,
  stop,
  res,
  buffer,
  station,
  component,
  dir,
  f,
  envelope = TRUE,
  sta,
  lta,
  on,
  off,
```

```
  freeze,
  dur_min,
  dur_max,
  n_common,
  t_common,
  t_pause,
  verbose = FALSE,
  ...
)
```

Arguments

start	POSIXct value, start time for event detection analysis. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
stop	POSIXct value, start time for event detection analysis. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
res	Numeric value, time resolution of the snippets used to detect events, in seconds. A useful value would be one hour (res = 3600).
buffer	Numeric vector, of length two. Size of the two-sided buffer to account for signal pre-processing effects, in seconds. The first value is used for the left side, the second for the right side. Events that happen to be detected during the buffer time will be removed from the output data set.
station	Character vector, seismic station IDs to be processed.
component	Character value or vector, seismic component to be used. Information can be restricted to a single character, denoting the last character of the station component code, i.e. "Z" would use the first and last component of c("BHZ", "HHE", "HHZ"). If omitted, the function will use "BHZ" by default.
dir	Character value, path to directory that contains the seismic files. See read_data for details and constraints on the data structure.
f	Numeric Numeric vector of length two, cutoff frequencies (Hz) for the bandpass filter to be applied. If omitted, no filtering will be performed.
envelope	Logical value, option to calculate signal envelopes before detecting events. Default is TRUE.
sta	Numeric value, duration of short-term window. Attention, this definition differs from the one in pick_stalta!
lta	Numeric value, duration of long-term window. Attention, this definition differs from the one in pick_stalta!
on	Numeric value, threshold value for event onset.
off	Numeric value, threshold value for event end.
freeze	Logical value, option to freeze lta value at start of an event. Useful to avoid self-adjustment of lta for long-duration events.
dur_min	Numeric value, minimum duration of an event (s). Picks with a STA-LTA duration below this threshold will be discarded.
dur_max	Numeric value, maximum duration of an event (s). Picks with a STA-LTA duration longer than this threshold will be discarded.

n_common	Numeric value, minimum number of stations at which an event has to be registered.
t_common	Numeric value, time (s) within which the picks of an event need to be registered by seismic stations.
t_pause	Numeric value, time period after the onset of an event during which no other event is allowed to happen.
verbose	Logical value, option to allow detailed screen output of information on the picking progress and results. Default is FALSE.
...	Further arguments passed to the function, i.e. keywords for the signal deconvolution part. See details for further information.

Details

The data will be imported time slice by time slice. Optionally, a signal deconvolution can be done. The data will be filtered to isolate the frequency range of interest, and tapered to account for edge effects. The taper size is automatically set be the two-sided buffer (see below). Optionally, a signal envelope can be calculated as part of the preprocessing work. For all successfully imported and preprocessed signal time snippets, events will be detected by the STL-LTA approach. Events from all input signals will be checked for a minium and maximum allowed event duration, set as thresholds by the user. All remaining events are checked for the number of stations of the network that did consistently detect an event. That means an event must be found in each signal within a user defined time window, a window that corresponds to the maximum travel time of a seismic signal through the network. The minimum number of stations that detect the event can be set by the user. In addition, events can be rejected if they occur during a user defined ban time after the onset of a previous event, which assumes that no subsequent events are allowed during the implementation of long duration events happening.

The time period to process is defined by `start`, `end` and the length of time snippets `res` (s). In addition, a left sided and right sided buffer can and should be added `buffer = c(60, 60)` (s), to account for artefacts during signal preprocessing and events that occur across the time snippets. The `start` and `end` time should be provided in `POSIXct` format. If it is provided as text string, the function will try to convert to POSIXct assuming the time zone is UTC.

There must be at least two seismic stations. The seismic components `component` must be a vector of the same length as `station` and contain the corresponding information for each station.

Deconvolution of the input signals can be done, although usually this is not essential for just picking events if the filter frequencies are in the flat response range of the sensors. If wanted, one needs to provide the additional vectors (all of the same length as `station`) `sensor`, `logger` and optionally `gain` (otherwise assumed to be `1`). See `signal_deconvolve()` for further information on the deconvolution routine and the supported keywords for sensors and loggers.

STA-LTA picking is usually done on signal envelopes `enevelope = TRUE`. However, for array seismology or other specific cases it might be useful to work with the waveforms directly, instead of with their envelopes.

The STA-LTA routine requires information on the length of the STA window, the LTA window, the on-threshold to define the start of an event, the off-threshold to define the end of an event, and the option to freeze the STA-LTA ratio at the onset of and event. See the documentation of the function `pick_stalta()` for further information.

Events that last shorter than dur_min or longer than dur_max are rejected. This criterion is applied before the test of how many stations have commonly detected an event. Hence, the minimum and maximum duration criteria need to be fulfilled by an event for all seismic stations, or at least as many as defined as minimum number of stations n_common.

A valid event should be detected by more than one station. At least two stations (n_common = 2) should detect it to confirm it is not just a local effect like rain drop impacts or animal activity. At least three stations (n_common = 3) are needed to locate an event. Detection at more stations increases the validity of an event as proper signal. However, the likelihood of stations not operating correctly or that some stations may be too far to detect a smaller event limits the number of stations that are be set as (n_common).

An event that happens just outside a seismic network will generate seismic waves that travel through the network and will require a given time for that. The time is set by the longest inter-station distance and the apparent seismic wave velocity. For example, an event who's signal propagated with 1000 m/s through a network with a maximum aperture of 1000 m will need up to 1 s. Hence, the parameter t_common should be set to 1, in this case. This parameter is good to reject events that show a longer travel time, e.g. because they are pressure waves that travel through the atmosphere, like helicopter signals or lightning strikes.

Value

data frame, detected events (start time, duration in seconds)

Author(s)

Michael Dietze

Examples

```
picks <- aux_picknetwork(start = "2017-04-09 01:00:00 UTC",
                         stop = "2017-04-09 02:00:00 UTC",
                         res = 1800,
                         buffer = c(90, 90),
                         station = c("RUEG1","RUEG2"),
                         component = rep("BHZ", 2),
                         dir = paste0(path.package("eseis"), "/extdata/"),
                         f = c(0.1, 0.2),
                         envelope = TRUE,
                         sta = 0.5,
                         lta = 300,
                         on = 3,
                         off = 1,
                         freeze = TRUE,
                         dur_min = 2,
                         dur_max = 90,
                         n_common = 2,
                         t_common = 5,
                         t_pause = 30,
                         verbose = TRUE)
```

```
print(picks)
```

`aux_picknetworkparallel` *Pick seismic events with a sensor network approach, parallel version*

Description

The function allows detection of discrete seismic events using the STA-LTA picker approach and including several stations of a seismic network. It will pick events at all input stations and identify events picked by more than a user defined minimum number of stations, within a given time window. It further allows rejecting events that are too short or too long, or occur to short after a previous event. The function can be used to process long data archives organised in a consistent file structure. This is the parallel computation version of the routine. For the single CPU version, allowing to show verbose progress information, use `aux_picknetwork()`.

Usage

```
aux_picknetworkparallel(
  start,
  stop,
  res,
  buffer,
  station,
  component,
  dir,
  f,
  envelope = TRUE,
  sta,
  lta,
  on,
  off,
  freeze,
  dur_min,
  dur_max,
  n_common,
  t_common,
  t_pause,
  cpu,
  ...
)
```

Arguments

`start`	`POSIXct` value, start time for event detection analysis. If a character string (or vector) is provided, the function will try to convert it to POSIXct.

stop	POSIXct value, start time for event detection analysis. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
res	Numeric value, time resolution of the snippets used to detect events, in seconds. A useful value would be one hour (res = 3600).
buffer	Numeric vector, of length two. Size of the two-sided buffer to account for signal pre-processing effects, in seconds. The first value is used for the left side, the second for the right side. Events that happen to be detected during the buffer time will be removed from the output data set.
station	Character vector, seismic station IDs to be processed.
component	Character value or vector, seismic component to be used. Information can be restricted to a single character, denoting the last character of the station component code, i.e. "Z" would use the first and last component of c("BHZ", "HHE", "HHZ"). If omitted, the function will use "BHZ" by default.
dir	Character value, path to directory that contains the seismic files. See read_data for details and constraints on the data structure.
f	Numeric Numeric vector of length two, cutoff frequencies (Hz) for the bandpass filter to be applied. If omitted, no filtering will be performed.
envelope	Logical value, option to calculate signal envelopes before detecting events. Default is TRUE.
sta	Numeric value, duration of short-term window. Attention, this definition differs from the one in pick_stalta!
lta	Numeric value, duration of long-term window. Attention, this definition differs from the one in pick_stalta!
on	Numeric value, threshold value for event onset.
off	Numeric value, threshold value for event end.
freeze	Logical value, option to freeze lta value at start of an event. Useful to avoid self-adjustment of lta for long-duration events.
dur_min	Numeric value, minimum duration of an event (s). Picks with a STA-LTA duration below this threshold will be discarded.
dur_max	Numeric value, maximum duration of an event (s). Picks with a STA-LTA duration longer than this threshold will be discarded.
n_common	Numeric value, minimum number of stations at which an event has to be registered.
t_common	Numeric value, time (s) within which the picks of an event need to be registered by seismic stations.
t_pause	Numeric value, time period after the onset of an event during which no other event is allowed to happen.
cpu	Numeric value, fraction of CPUs to use. If omitted, only one CPU will be used.
...	Further arguments passed to the function, i.e. keywords for the signal deconvolution part. See details for further information.

Details

The data will be imported time slice by time slice. Optionally, a signal deconvolution can be done. The data will be filtered to isolate the frequency range of interest, and tapered to account for edge effects. The taper size is automatically set be the two-sided buffer (see below). Optionally, a signal envelope can be calculated as part of the preprocessing work. For all successfully imported and preprocessed signal time snippets, events will be detected by the STL-LTA approach. Events from all input signals will be checked for a minium and maximum allowed event duration, set as thresholds by the user. All remaining events are checked for the number of stations of the network that did consistently detect an event. That means an event must be found in each signal within a user defined time window, a window that corresponds to the maximum travel time of a seismic signal through the network. The minimum number of stations that detect the event can be set by the user. In addition, events can be rejected if they occur during a user defined ban time after the onset of a previous event, which assumes that no subsequent events are allowed during the implementation of long duration events happening.

The time period to process is defined by `start`, `end` and the length of time snippets `res` (s). In addition, a left sided and right sided buffer can and should be added `buffer = c(60, 60)` (s), to account for artefacts during signal preprocessing and events that occur across the time snippets. The `start` and `end` time should be provided in `POSIXct` format. If it is provided as text string, the function will try to convert to POSIXct assuming the time zone is UTC.

There must be at least two seismic stations. The seismic components `component` must be a vector of the same length as `station` and contain the corresponding information for each station.

Deconvolution of the input signals can be done, although usually this is not essential for just picking events if the filter frequencies are in the flat response range of the sensors. If wanted, one needs to provide the additional vectors (all of the same length as `station`) `sensor`, `logger` and optionally `gain` (otherwise assumed to be `1`). See `signal_deconvolve()` for further information on the deconvolution routine and the supported keywords for sensors and loggers.

STA-LTA picking is usually done on signal envelopes `enevelope = TRUE`. However, for array seismology or other specific cases it might be useful to work with the waveforms directly, instead of with their envelopes.

The STA-LTA routine requires information on the length of the STA window, the LTA window, the on-threshold to define the start of an event, the off-threshold to define the end of an event, and the option to freeze the STA-LTA ratio at the onset of and event. See the documentation of the function `pick_stalta()` for further information.

Events that last shorter than `dur_min` or longer than `dur_max` are rejected. This criterion is applied before the test of how many stations have commonly detected an event. Hence, the minimum and maximum duration criteria need to be fulfilled by an event for all seismic stations, or at least as many as defined as minimum number of stations `n_common`.

A valid event should be detected by more than one station. At least two stations (`n_common = 2`) should detect it to confirm it is not just a local effect like rain drop impacts or animal activity. At least three stations (`n_common = 3`) are needed to locate an event. Detection at more stations increases the validity of an event as proper signal. However, the likelihood of stations not operating correctly or that some stations may be too far to detect a smaller event limits the number of stations that are be set as (`n_common`).

An event that happens just outside a seismic network will generate seismic waves that travel through the network and will require a given time for that. The time is set by the longest inter-station

distance and the apparent seismic wave velocity. For example, an event who's signal propagated with 1000 m/s through a network with a maximum aperture of 1000 m will need up to 1 s. Hence, the parameter `t_common` should be set to 1, in this case. This parameter is good to reject events that show a longer travel time, e.g. because they are pressure waves that travel through the atmosphere, like helicopter signals or lightning strikes.

Value

`data frame`, detected events (start time, duration in seconds)

Author(s)

Michael Dietze

Examples

```
picks <- aux_picknetwork(start = "2017-04-09 01:00:00 UTC",
                         stop = "2017-04-09 02:00:00 UTC",
                         res = 1800,
                         buffer = c(90, 90),
                         station = c("RUEG1","RUEG2"),
                         component = rep("BHZ", 2),
                         dir = paste0(path.package("eseis"), "/extdata/"),
                         f = c(0.1, 0.2),
                         envelope = TRUE,
                         sta = 0.5,
                         lta = 300,
                         on = 3,
                         off = 1,
                         freeze = TRUE,
                         dur_min = 2,
                         dur_max = 90,
                         n_common = 2,
                         t_common = 5,
                         t_pause = 30)

print(picks)
```

---

aux_psdsummary *Calculate aggregated PSDs over long time periods*

Description

The function generates a long time PSD with aggregated time and frequency resolution.

Usage

```
aux_psdsummary(
  start,
  stop,
  ID,
  component = "BHZ",
  dir,
  window,
  sensor,
  logger,
  gain = 1,
  hours_skip,
  res = 1000,
  n = 100,
  cpu,
  verbose = FALSE
)
```

Arguments

start	POSIXct value, start time for PSD calculation. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
stop	POSIXct value, stop time for PSD calculation. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
ID	Character vector, station ID to be processed
component	Character value or vector, seismic component to be used. If omitted, the function will use "BHZ" by default.
dir	Character value, path to directory that contains the seismic files. See read_data for details and constraints on the data structure.
window	Numeric value, time window size for the PSD calculation. Should be appropriately large to avoid extensive calculation times, usually at the order of 0.1 percent of the total PSD duration.
sensor	Character value, sensor keyword for the deconvolution step (see signal_deconvolve for details and keywords). If omitted, no deconvolution will be attempted.
logger	Character value, logger keyword for the deconvolution step (see signal_deconvolve for details and keywords). If omitted, no deconvolution will be attempted.
gain	Numeric value, signal preamplification factor used for the deconvolution step (see signal_deconvolve for details. Default value is 1.
hours_skip	Integer vector, one or more full hours that will be excluded from the PSD generation. This optional value is useful if one wants to omit noisy daytime hours.
res	Numeric value, frequency resolution of the PSD, by default set to 1000. This is used to reduce the resulting data size.
n	Numeric value, number of times to try to find a time snippet with data to estimate the frequency vector length. By default set to 100.

cpu	Numeric value, fraction of CPUs to use. If omitted, only one CPU will be used.
verbose	Logical value, option to show extended function information as the function is running. Default is FALSE.

Details

The function will calculate PSDs using the Welch method (see signal_spectrogram), with no overlap of the main time windows. The sub-windows will be automatically set to 10 the overlap of sub-windows to 0.5.

Value

eseis object, a spectrogram

Author(s)

Michael Dietze

Examples

```
## Not run:

p <- aux_psdsummary(start = "2017-04-15 19:00:00 UTC",
                    stop = "2017-04-15 22:00:00 UTC",
                    ID = "RUEG1",
                    component = "BHE",
                    dir = "~/data/sac/",
                    sensor = "TC120s",
                    logger = "Cube3ext",
                    window = 600,
                    res = 1000,
                    verbose = TRUE)

plot(p)


## End(Not run)
```

---

aux_sonifysignal *Convert seismic signal to sound (sonification)*

Description

The function converts a seismic signal to sound and saves it as a wav file.

Usage

```
aux_sonifysignal(
  data,
  file,
  aggregate = 1,
  amplification = 10^6,
  speed = 1,
  dt
)
```

Arguments

data	eseis object to be converted to sound file
file	Character value, file name under which the sonified signal is saved.
aggregate	Numeric value, factor by which the seismic file is aggregated before conversion. Aggregation is performed by linear interpolation.
amplification	Numeric value, amplification factor. Default is 10^6.
speed	Numeric value, factor by which sampling rate is increased to make sound sensible. The higher the speed value, the higher is the tone. Default is 1 (100 Hz seismic signal becomes 100 Hz sound signal).
dt	Numeric value, samplig rate. Only needed if data is not an eseis object.

Value

Sound file in wav format, written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:

## load example data
data(rockfall)

## deconvolve and taper signal
s <- signal_deconvolve(data = rockfall_eseis)
s <- signal_taper(data = s, p = 0.05)

## sonify as is (barely sensible, due to too low frequency)
aux_sonifysignal(data = s,
                 file = "~/Downloads/r1.wav")

## sonify at 20-fold speed
aux_sonifysignal(data = s,
                 file = "~/Downloads/r1.wav",
                 speed = 20)
```

```
## End(Not run)
```

`aux_splitcubechannels` *Create multiple single-component files from multi-component files*

Description

The function changes the meta data and file names of seismic data sets (organised in an eseis-compatible data structure). Data cubes in combination with splitter break-out boxes allow to record vertical component signals from three single-channel geophones, each connected by an individual cable. The logger interprets the data as E, N and Z components, regardless of the actual sensor and component connected to the respective cable. Hence, it is possible to record data from three independent (vertical) sensors with a single Data cube. However, the channels will be named according to the internal scheme (E, N, Z). To correct for that effect, the function will look up the right combination of station ID and component, and change meta data and name of each file in the input directory accordingly, before saving the updated file in the output directory.

Usage

```
aux_splitcubechannels(
  input,
  output,
  ID_in,
  component_in,
  ID_out,
  component_out,
  gipptools
)
```

Arguments

`input`	`Character` value, path to directory where the Cube files to be processed are stored.
`output`	`Character` value, path to directory where output data is written. It is not recommended to use the same directory for output and input, as files may be overwritten unintentionally.
`ID_in`	`Character` vector with the IDs of the stations that will be changed. Note that for every channel of the input data, the corresponding ID must be repeated. See details for further information.
`component_in`	`Character` vector with the components of the input stations that will be changed. The length must match the length of `ID_in` as well as `ID_out` and `component_out`.
`ID_out`	`Character` vector with the IDs of the output files corresponding to the input stations and components.
`component_out`	`Character` vector with the components of the output stations.
`gipptools`	`Character` value, path to gipptools or cubetools directory. Only needed when mseed files are processed

Value

A set of converted and organised seismic files written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:

## the example will convert the E, N and Z components of station RUEG1
## to the station IDs RUEGA, RUEGB, RUEGC, all with BHZ as component

aux_splitcubechannels(input =  paste0(system.file("extdata",
                                      package="eseis"), "/"),
                      output = tempdir(),
                      ID_in = c("RUEG1", "RUEG1", "RUEG1"),
                      component_in = c("BHE", "BHN", "BHZ"),
                      ID_out = c("RUEGA", "RUEGB", "RUEGC"),
                      component_out = c("BHZ", "BHZ", "BHZ"))


## End(Not run)
```

`aux_stationinfofile` *Create station info file from cube files.*

Description

This function reads GPS tags from Omnirecs/Digos Datacube files and creates a station info file from additional input data. It depends on the cubetools or gipptools software package (see details).

Usage

```
aux_stationinfofile(
  name,
  input_dir,
  output_dir,
  station_ID,
  station_name,
  station_z,
  station_d,
  sensor_type,
  logger_type,
  sensor_ID,
  logger_ID,
```

```
  unit = "dd",
  n,
  quantile = 0.95,
  random = TRUE,
  cpu,
  gipptools,
  write_file = TRUE,
  write_raw = FALSE,
  write_data = FALSE
)
```

Arguments

name	Character value, file name of the output station info file, with extension.
input_dir	Character value, path to directory where all cube files to be processed as stored. Each set of files from one logger must be stored in a separate sub-directory named after the cube ID.
output_dir	Character value, path to directory where output data is written to.
station_ID	Character vector, seismic station ID. Each value must not contain more than 5 characters. Longer entries will be clipped. If omitted, a default ID will be created.
station_name	Character vector, seismic station name. If omitted, the station ID is used as name.
station_z	Numeric vector, elevation of the seismic stations.
station_d	Numeric vector, deployment depth of the seismic sensor.
sensor_type	Character vector, sensor type.
logger_type	Character vector, logger type.
sensor_ID	Character vector, sensor ID.
logger_ID	Character vector, logger ID.
unit	Character value, coordinates unit of the location. One out of "dd" (decimal degrees) and "utm" (metric in UTM zone). Default is "dd".
n	Numeric value, number of cube file to process for GPS coordinate extraction. If omitted, all files are processed.
quantile	Numeric value, quantile size to which the extracted coordinate sample size is restricted. This is mainly used to remove coordinate outliers, due to spurious GPS signals. Default is 0.95. Set to 1 to omit any sample rejection.
random	Logical value, option to draw n cube files randomly instead of ordered by date. Default is TRUE.
cpu	Numeric value, fraction of CPUs to use for parallel processing. If omitted, one CPU is used.
gipptools	Character value, path to gipptools or cubetools directory.
write_file	Logical value, option to write station info file to disk. Default is TRUE.
write_raw	Logical value, option to write (keep) raw ASCII GPS data. Default is FALSE.
write_data	Logical value, option to write gps raw data as rda-file. File name will be the same as for file. Default is FALSE.

Details

A station info file is an ASCII file that contains all relevant information about the individual stations of a seismic network. The variables contain a station ID (containing not more than 5 characters), station name, latitude, longitude, elevation, deployment depth, sensor type, logger type, sensor ID and logger ID.
The function requires that the software cubetools (`http://www.omnirecs.de/documents.html`) or gipptools (`http://www.gfz-potsdam.de/en/section/geophysical-deep-sounding/infrastructure/geophysical`) are installed. Note that GPS tag extraction may take several minutes per cube file. Hence, depending on the number of files and utilised CPUs the processing may take a while.
Specifying an input directory (`input_dir`) is mandatory. This input directory must only contain the subdirectories with the cube files to process, each set of cube files must be located in a separate subdirectory and these subdiretories must have the same name as specified by the logger IDs (`logger_ID`). An appropriate structure would be something like:

1. input
   (a) A1A
      i. file1.A1A
      ii. file2.A1A
   (b) A1B
      i. file1.A1B
      ii. file2.A1B

Value

A set of files written to disk and a data frame with seismic station information.

Author(s)

Michael Dietze

Examples

```
## Not run:

## basic example with minimum effort
aux_stationinfofile(name = "stationinfo.txt",
                    input_dir = "input",
                    logger_ID = c("864", "876", "AB1"),
                    gipptools = "software/gipptools-2015.225")

## example with more adjustments
aux_stationinfofile(name = "stationinfo.txt",
                    input_dir = "input",
                    logger_ID = c("864", "876", "AB1"),
                    station_name = c("KTZ01", "KTZ02", "KTZ03"),
                    station_z = c(30, 28, 29),
                    station_d = rep(0.5, 3),
                    sensor_type = rep("TC120s", 3),
```

```
                    logger_type = rep("Cube3ext", 3),
                    unit = "utm",
                    n = 1,
                    cpu = 0.9,
                    gipptools = "software/gipptools-2015.225",
                    write_raw = TRUE,
                    write_data = TRUE)


## End(Not run)
```

`earthquake` *Seismic traces of a small earthquake*

Description

The dataset comprises the seismic signal (all three components) of a small earthquake. The data have been recorded at 200 Hz sampling frequency with an Omnirecs Cube ext 3 data logger.

The dataset comprises the time vector associated with the data set `earthquake`.

Usage

```
s

t
```

Format

The format is: List of 3 $ BHE: num [1:8001] -3.95e-07 ... $ BHN: num [1:8001] -2.02e-07 ... $ BHZ: num [1:8001] -1.65e-07 ...

The format is: POSIXct[1:98400], format: "2015-04-06 13:16:54" ...

Examples

```
## load example data set
data(earthquake)

## plot signal vector
plot(x = t, y = s$BHZ, type = "l")


## load example data set
data(earthquake)

## show range of time vector
range(t)
```

`fmi_inversion` *Invert fluvial data set based on reference spectra catalogue*

Description

The fluvial model inversion (FMI) routine uses a predefined look-up table with reference spectra to identify those spectra that fit the empirical data set best, and returns the corresponding target variables and fit errors.

Usage

```
fmi_inversion(reference, data, n_cores = 1)
```

Arguments

`reference`	`List` containing lists with precalculated model spectra.
`data`	`eseis` object or `numeric` matrix (spectra organised by columns), empiric spectra which are used to identify the best matching target parameters of the reference data set.
`n_cores`	`Numeric` value, number of CPU cores to use. Disabled by setting to 1. Default is 1.

Details

Note that the frequencies of the empiric and modelled data sets must match.

Value

`List` object containing the inversion results.

Author(s)

Michael Dietze

Examples

```
## NOTE THAT THE EXAMPLE IS OF BAD QUALITY BECAUSE ONLY 10 REFERENCE
## PARAMETER SETS AND SPECTRA ARE CALCULATED, DUE TO COMPUATATION TIME
## CONSTRAINTS. SET THE VALUE TO 1000 FOR MORE MEANINGFUL RESULTS.

## create 100 example reference parameter sets
ref_pars <- fmi_parameters(n = 10,
                           h_w = c(0.02, 1.20),
                           q_s = c(0.001, 8.000) / 2650,
                           d_s = 0.01,
                           s_s = 1.35,
                           r_s = 2650,
                           w_w = 6,
```

```
                              a_w = 0.0075,
                              f_min = 5,
                              f_max = 80,
                              r_0 = 6,
                              f_0 = 1,
                              q_0 = 10,
                              v_0 = 350,
                              p_0 = 0.55,
                              e_0 = 0.09,
                              n_0_a = 0.6,
                              n_0_b = 0.8,
                              res = 100)

## create corresponding reference spectra
ref_spectra <- fmi_spectra(parameters = ref_pars)

## define water level and bedload flux time series
h <- c(0.01, 1.00, 0.84, 0.60, 0.43, 0.32, 0.24, 0.18, 0.14, 0.11)
q <- c(0.05, 5.00, 4.18, 3.01, 2.16, 1.58, 1.18, 0.89, 0.69, 0.54) / 2650
hq <- as.list(as.data.frame(rbind(h, q)))

## calculate synthetic spectrogram
psd <- do.call(cbind, lapply(hq, function(hq) {

  psd_turbulence <- eseis::model_turbulence(h_w = hq[1],
                                            d_s = 0.01,
                                            s_s = 1.35,
                                            r_s = 2650,
                                            w_w = 6,
                                            a_w = 0.0075,
                                            f = c(10, 70),
                                            r_0 = 5.5,
                                            f_0 = 1,
                                            q_0 = 18,
                                            v_0 = 450,
                                            p_0 = 0.34,
                                            e_0 = 0.0,
                                            n_0 = c(0.5, 0.8),
                                            res = 100,
                                            eseis = FALSE)$power

  psd_bedload <- eseis::model_bedload(h_w = hq[1],
                                      q_s = hq[2],
                                      d_s = 0.01,
                                      s_s = 1.35,
                                      r_s = 2650,
                                      w_w = 6,
                                      a_w = 0.0075,
                                      f = c(10, 70),
                                      r_0 = 5.5,
                                      f_0 = 1,
                                      q_0 = 18,
                                      v_0 = 450,
```

```
                                        x_0 = 0.34,
                                        e_0 = 0.0,
                                        n_0 = 0.5,
                                        res = 100,
                                        eseis = FALSE)$power

  ## combine spectra
  psd_sum <- psd_turbulence + psd_bedload

  ## return output
  return(10 * log10(psd_sum))
}))

graphics::image(t(psd))

## invert empiric data set
X <- fmi_inversion(reference = ref_spectra,
                   data = psd)

## plot model results
plot(X$parameters$q_s * 2650,
     type = "l")
plot(X$parameters$h_w,
     type = "l")
```

---

fmi_parameters *Create reference model reference parameter catalogue*

Description

In order to run the fluvial model inversion (FMI) routine, a set of randomised target parameter combinations needs to be created. This function does this job.

Usage

```
fmi_parameters(
  n,
  d_s,
  s_s,
  r_s,
  q_s,
  h_w,
  w_w,
  a_w,
  f_min,
  f_max,
  r_0,
  f_0,
```

```
  q_0,
  v_0,
  p_0,
  e_0,
  n_0_a,
  n_0_b,
  res
)
```

Arguments

n	Numeric value, number of output reference spectra.
d_s	Numeric value, mean sediment grain diameter (m). Alternative to gsd.
s_s	Numeric value, standard deviation of sediment grain diameter (m). Alternative to gsd.
r_s	Numeric value, specific sediment density (kg / m^3)
q_s	Numeric value, unit sediment flux (m^2 / s)
h_w	Numeric value, fluid flow depth (m)
w_w	Numeric value, fluid flow width (m)
a_w	Numeric value, fluid flow inclination angle (radians)
f_min	Numeric value, lower boundary of the frequency range to be modelled.
f_max	Numeric value, upper boundary of the frequency range to be modelled.
r_0	Numeric value, distance of seismic station to source
f_0	Numeric value, reference frequency (Hz)
q_0	Numeric value, ground quality factor at f_0. "Reasonable value may be 20" (Tsai et al. 2012).
v_0	Numeric value, phase speed of the Rayleigh wave at f_0 (m/s). Assuming a shear wave velocity of about 2200 m/s, Tsai et al. (2012) yield a value of 1295 m/s for this parameter.
p_0	Numeric value, variation exponent of Rayleigh wave velocities with frequency (dimensionless)
e_0	Numeric value, exponent characterizing quality factor increase with frequency (dimensionless). "Reasonable value may be 0" (Tsai et al. 2012).
n_0_a	Numeric value, lower Greens function displacement amplitude coefficients. Cf. N_ij in eq. 36 in Gimbert et al. (2014)
n_0_b	Numeric value, lower Greens function displacement amplitude coefficients. Cf. N_ij in eq. 36 in Gimbert et al. (2014)
res	Numeric value, output resolution, i.e. length of the spectrum vector.

Details

All parameters must be provided as single values, except for those parameters that shall be randomised, which must be provided as a vector of length two. This vector defines the range within which uniformly distributed random values will be generated and assigned.

Value

List object with model reference parameters.

Author(s)

Michael Dietze

Examples

```
## create two parameter sets where h_w (water level) and q_s (sediment
## flux) are randomly varied.

ref_pars <- fmi_parameters(n = 2,
                           h_w = c(0.02, 2.00),
                           q_s = c(0.001, 50.000) / 2650,
                           d_s = 0.01,
                           s_s = 1.35,
                           r_s = 2650,
                           w_w = 6,
                           a_w = 0.0075,
                           f_min = 5,
                           f_max = 80,
                           r_0 = 6,
                           f_0 = 1,
                           q_0 = 10,
                           v_0 = 350,
                           p_0 = 0.55,
                           e_0 = 0.09,
                           n_0_a = 0.6,
                           n_0_b = 0.8,
                           res = 100)
```

---

fmi_spectra *Create reference model spectra catalogue*

Description

In order to run the fluvial model inversion (FMI) routine, a look-up table with reference spectra needs to be created (based on predefined model parameters). This function does this job.

Usage

```
fmi_spectra(parameters, n_cores = 1)
```

Arguments

parameters	List containing lists with model parameters for which the spectra shall be calculated.
n_cores	Numeric value, number of CPU cores to use. Disabled by setting to 1. Default is 1.

Value

List object containing the calculated reference spectra and the corresponding input parameters. Note that the spectra are given in dB for a seamless comparison with the empirical PSD data, while the original output of the models are in linear scale.

Author(s)

Michael Dietze

Examples

```
## create 2 example reference parameter sets
ref_pars <- fmi_parameters(n = 2,
                           h_w = c(0.02, 2.00),
                           q_s = c(0.001, 50.000) / 2650,
                           d_s = 0.01,
                           s_s = 1.35,
                           r_s = 2650,
                           w_w = 6,
                           a_w = 0.0075,
                           f_min = 5,
                           f_max = 80,
                           r_0 = 6,
                           f_0 = 1,
                           q_0 = 10,
                           v_0 = 350,
                           p_0 = 0.55,
                           e_0 = 0.09,
                           n_0_a = 0.6,
                           n_0_b = 0.8,
                           res = 100)

## create corresponding reference spectra
ref_spectra <- fmi_spectra(parameters = ref_pars)
```

---

gui_models *Start GUI with seismic models*

Description

This function starts a browser-based graphic user interface to explore the parameter space of seismic models that predict the spectra of turbulent water flow and bedload flux. Empirical spectra can be added if they are present in the current environment as eseis objects. Note that those spectra should be made from deconvolved input data in order to have the same units as the model spectra (dB). ATTENTION, due to an unresolved issue, there must be at least one eseis object present in the working environment.

Usage

```
gui_models(...)
```

Arguments

...	further arguments to pass to runApp

Author(s)

Michael Dietze

See Also

runApp

Examples

```
## Not run:
# Start the GUI
gui_models()

## End(Not run)
```

---

list_logger *List library with data logger information.*

Description

The function returns the list of supported data loggers to extract signal deconvolution parameters.

Usage

```
list_logger()
```

Details

The value AD is the analogue-digital conversion factor.

Value

List object, supported loggers with their parameters.

Author(s)

Michael Dietze

Examples

```
## show documented loggers
list_logger()

## show names of loggers in list
names(list_logger())
```

`list_sacparameters` *List all header parameters of a sac file.*

Description

The function returns a data frame with all header values of a sac file. It may be used for advanced modifications by the user.

Usage

```
list_sacparameters()
```

Value

List object, parameters supported by a sac file.

Author(s)

Michael Dietze

Examples

```
## show sac parameters
list_sacparameters()
```

list_sensor *List sensor library.*

Description

The function returns the list of supported sensors to extract signal deconvolution parameters.

Usage

```
list_sensor()
```

Details

Poles and zeros must be given in rad/s. Characteristics of further sensors can be added manually. See examples of `signal_deconvolve` for further information. The value s is the generator constant (sensitivity) given in Vs/m. The value k is the normalisation factor of the sensor.

Value

`List` object, supported sensors with their parameters.

Author(s)

Michael Dietze

Examples

```
## show sensors
list_sensor()
```

model_amplitude *Model source amplitude by amplitude-distance model fitting*

Description

The function fits one of several models of signal amplitude attenuation and returns a set of model parameters, including the source amplitude (a_0).

Usage

```
model_amplitude(
  data,
  model = "SurfSpreadAtten",
  distance,
  source,
  d_map,
  coupling,
  v,
  q,
  f,
  k,
  a_0
)
```

Arguments

data	Numeric matrix or eseis object, seismic signals to work with. Since the function will calculate the maxima of the data it is usually the envolopes of the data that should be used here. In an extreme case, a vector with just the maximum amplitudes recorded at each station can be provided, as well.
model	Character value, model to fit the data. One out of the list in the details section. Default is "SurfSpreadAtten".
distance	Numeric vector with distance of station locations to source. Alternatively, the distance can be calculated by providing the source coordinates (xy) and distance maps (d_map)
source	Numeric vector of length two, location of the seismic source to model (x and y coordinates).
d_map	List object, distance maps for each station (i.e., SpatialGridDataFrame objects). Output of distance_map.
coupling	Numeric vector, coupling efficiency factors for each seismic station. The best coupled station (or the one with the highest amplification) must receive 1, the others must be scaled relatively to this one.
v	Numeric value, mean velocity of seismic waves (m/s). Only relevant for models accounting for unelastic attenuation (see details).
q	Numeric value, quality factor of the ground. Only relevant for models accounting for unelastic attenuation (see details).
f	Numeric value, frequency for which to model the attenuation. Only relevant for models accounting for unelastic attenuation (see details).
k	Numeric value, fraction of surface wave contribution to signals. Only relevant for models that include mixture of surface and body waves (see details).
a_0	Logical value, start parameter of the source amplitude, if not provided, a best guess is made as 100 times the maximum amplitude value of the data set.

Details

Depending on the choice of the model to fit, several parameters can (or should) be provided, e.g. `f`,`q`, `v`, `k`, and most importantly, `a_0`. If more signals than free parameters are available, the missing parameters may be estimated during the fit, but without any checks of quality and meaningfulness. The parameter `a_0` will be defined as 100 times the maximum input amplitude, by default. The parameters `f` will be set to 10 Hz, `q` to 50, `v` to 1000 m/s and `k` to 0.5.

ISSUES: account for non-fixed parameters, especially k

The following amplitude-distance models are available:

- `"SurfSpreadAtten"`, Surface waves including geometric spreading and unelastic attenuation
- `"BodySpreadAtten"`, Body waves including geometric spreading and unelastic attenuation
- `"SurfBodySpreadAtten"`, Surface and body waves including geometric spreading and unelastic attenuation
- `"SurfSpread"`, Surface waves including geometric spreading, only
- `"BodySpread"`, Body waves including geometric spreading, only
- `"SurfBodySpread"`, Surface and body waves including geometric spreading, only

SurfSpreadAtten The model is based on Eq. 17 from Burtin et al. (2016):

$$a_d = a_0/sqrt(d) * exp(-(pi * f * d)/(q * v))$$

where a_0 is the source amplitude, a_d the amplitude as recorded by a sensor at distance d, f is the center frequency of the signal, q the ground quality factor and v the seismic wave velocity.

BodySpreadAtten The model is based on Eq. 16 from Burtin et al. (2016):

$$a_d = a_0/d * exp(-(pi * f * d)/(q * v))$$

where a_0 is the source amplitude, a_d the amplitude as recorded by a sensor at distance d, f is the center frequency of the signal, q the ground quality factor and v the seismic wave velocity.

SurfBodySpreadAtten The model based on Eqs. 16 and 17 from Burtin et al. (2016):

$$a_d = k * a_0/sqrt(d) * exp(-(pi * f * d)/(q * v)) + (1 - k) * a_0/d * exp(-(pi * f * d)/(q * v))$$

where a_0 is the source amplitude, a_d the amplitude as recorded by a sensor at distance d, f is the center frequency of the signal, q the ground quality factor, v the seismic wave velocity, and n and m two factors determining the relative contributions of the two wave types, thus summing to 1.

BodySpread The model is simply accounting for geometric spreading

$$a_d = a_0/d$$

where a_0 is the source amplitude, a_d the amplitude as recorded by a sensor at distance d.

SurfSpread The model is simply accounting for geometric spreading

$$a_d = a_0/sqrt(d)$$

where a_0 is the source amplitude, a_d the amplitude as recorded by a sensor at distance d.

SurfBodySpread The model is simply accounting for geometric spreading

$$a_d = k * (a_0/d) + (1 - k) * a_d/sqrt(d)$$

where a_0 is the source amplitude, a_d the amplitude as recorded by a sensor at distance d, and n and m two factors determining the relative contributions of the two wave types, thus summing to 1.

References - Burtin, A., Hovius, N., and Turowski, J. M.: Seismic monitoring of torrential and fluvial processes, Earth Surf. Dynam., 4, 285–307, https://doi.org/10.5194/esurf-4-285-2016, 2016.

Value

`List` with model results, including `a_0` (source amplitude), `residuals` (model residuals), `coefficients` model coefficients.

Author(s)

Michael Dietze

Examples

```
## Not run:

 ## create synthetic DEM
dem <- terra::rast(nrows = 20, ncols = 20,
                   xmin = 0, xmax = 10000,
                   ymin= 0, ymax = 10000,
                   vals = rep(0, 400))

## define station coordinates
sta <- data.frame(x = c(1000, 9000, 5000, 9000),
                  y = c(1000, 1000, 9000, 9000),
                  ID = c("A", "B", "C", "D"))

## create synthetic signal (source in towards lower left corner of the DEM)
s <- rbind(dnorm(x = 1:1000, mean = 500, sd = 50) * 50,
           dnorm(x = 1:1000, mean = 500, sd = 50) * 2,
           dnorm(x = 1:1000, mean = 500, sd = 50) * 1,
           dnorm(x = 1:1000, mean = 500, sd = 50) * 0.5)

## calculate spatial distance maps and inter-station distances
D <- eseis::spatial_distance(stations = sta[,1:2],
                             dem = dem)

model_amplitude(data = s,
                source = c(500, 600),
                d_map = D$maps,
                v = 500,
```

```
                q = 50,
                f = 10)

model_amplitude(data = s,
                distance = c(254, 8254, 9280, 11667),
                model = "SurfBodySpreadAtten",
                v = 500,
                q = 50,
                f = 10,
                k = 0.5)


## End(Not run)
```

`model_bedload` *Model the seismic spectrum due to bedload transport in rivers*

Description

The function calculates a seismic spectrum as predicted by the model of Tsai et al. (2012) for river bedload transport. The code was written to R by Sophie Lagarde and integrated to the R package 'eseis' by Michael Dietze.

Usage

```
model_bedload(
  gsd,
  d_s,
  s_s,
  r_s,
  q_s,
  h_w,
  w_w,
  a_w,
  f = c(1, 100),
  r_0,
  f_0,
  q_0,
  e_0,
  v_0,
  x_0,
  n_0,
  n_c,
  res = 100,
  adjust = TRUE,
  eseis = FALSE,
  ...
)
```

Arguments

gsd	data frame grain-size distribution function. Must be provided as data frame with two variables: grain-size class (in m, first column) and wgt/vol percentage per class (second column). See examples for details.
d_s	Numeric value, mean sediment grain diameter (m). Alternative to gsd.
s_s	Numeric value, standard deviation of sediment grain diameter (m). Alternative to gsd.
r_s	Numeric value, specific sediment density (kg / m^3)
q_s	Numeric value, unit sediment flux (m^2 / s)
h_w	Numeric value, fluid flow depth (m)
w_w	Numeric value, fluid flow width (m)
a_w	Numeric value, fluid flow inclination angle (radians)
f	Numeric vector, frequency range to be modelled. If of length two the argument is interpreted as representing the lower and upper limit and the final length of the frequency vector is set by the argument res. If f contains more than two values it is interpreted as the actual frequency vector and the value of res is ignored.
r_0	Numeric value, distance of seismic station to source
f_0	Numeric value, reference frequency (Hz)
q_0	Numeric value, ground quality factor at f_0. "Reasonable value may be 20" (Tsai et al. 2012).
e_0	Numeric value, exponent characterizing quality factor increase with frequency (dimensionless). "Reasonable value may be 0" (Tsai et al. 2012).
v_0	Numeric value, phase speed of the Rayleigh wave at f_0 (m/s). Assuming a shear wave velocity of about 2200 m/s, Tsai et al. (2012) yield a value of 1295 m/s for this parameter.
x_0	Numeric value, exponent of the power law variation of Rayleigh wave velocities with frequency
n_0	Numeric vector of length two, Greens function displacement amplitude coefficients. Cf. N_ij in eq. 36 in Gimbert et al. (2014)
n_c	Numeric value, option to include single particle hops coherent in time, causing spectrum modulation due to secondary effects. Omitted is no value is specified, here. Usual values may be between 2 and 4.
res	Numeric value, output resolution, i.e. length of the spectrum vector. Default is 1000.
adjust	Logical value, option to adjust PSD for wide grain-size distributions, according to implementation by Tsai et al. (2012).
eseis	Character value, option to return an eseis object instead of a data frame. Default is FALSE.
...	Further arguments passed to the function.

Details

The model uses a set of predefined constants. These can also be changed by the user, using the ... argument:

- g = 9.81, gravitational acceleration (m/s^2)
- r_w = 1000, fluid specific density (kg/m^3)
- k_s = 3 * d_50, roughness length (m)
- log_lim = c(0.0001, 100), limits of grain-size distribution function template (m)
- log_length = 10000, length of grain-size distribution function template
- nu = 10^(-6), specific density of the fluid (kg/m^3)
- power_d = 3, grain-size power exponent
- gamma = 0.9, gamma parameter, after Parker (1990)
- s_c = 0.8, drag coefficient parameter
- s_p = 3.5, drag coefficient parameter
- c_1 = 2 / 3, inter-impact time scaling, after Sklar & Dietrich (2004)

When no user defined grain-size distribution function is provided,the function calculates the raised cosine distribution function as defined in Tsai et al. (2012) using the default range and resolution as specified by log_lim and log_length (see additional arguments list above). These default values are appropriate for mean sediment sizes between 0.001 and 10 m and log standard deivations between 0.05 and 1. When more extreme distributions are to be used, it is necessary to either adjust the arguments log_lim and log_length or use a user defined distribution function.

The adjustment option (implemented with package version 0.6.0) is only relevant for wide grain-size distributions, i.e., s_s > 0.2. In such cases, the unadjusted version tends to underestimate seismic power.

Value

eseis object containing the modelled spectrum.

Author(s)

Sophie Lagarde, Michael Dietze

References

Tsai, V. C., B. Minchew, M. P. Lamb, and J.-P. Ampuero (2012), A physical model for seismic noise generation from sediment transport in rivers, Geophys. Res. Lett., 39, L02404, doi:10.1029/2011GL050255.

Examples

```
## calculate spectrum (i.e., fig. 1b in Tsai et al., 2012)
p_bedload <- model_bedload(d_s = 0.7,
                           s_s = 0.1,
                           r_s = 2650,
                           q_s = 0.001,
                           h_w = 4,
```

```
                            w_w = 50,
                            a_w = 0.005,
                            f = c(0.1, 20),
                            r_0 = 600,
                            f_0 = 1,
                            q_0 = 20,
                            e_0 = 0,
                            v_0 = 1295,
                            x_0 = 0.374,
                            n_0 = 1,
                            res = 100,
                            eseis = TRUE)

## plot spectrum
plot_spectrum(data = p_bedload,
              ylim = c(-170, -110))

## define empiric grain-size distribution
gsd_empiric <- data.frame(d = c(0.70, 0.82, 0.94, 1.06, 1.18, 1.30),
                          p = c(0.02, 0.25, 0.45, 0.23, 0.04, 0.00))

## calculate spectrum
p_bedload <- model_bedload(gsd = gsd_empiric,
                            r_s = 2650,
                            q_s = 0.001,
                            h_w = 4,
                            w_w = 50,
                            a_w = 0.005,
                            f = c(0.1, 20),
                            r_0 = 600,
                            f_0 = 1,
                            q_0 = 20,
                            e_0 = 0,
                            v_0 = 1295,
                            x_0 = 0.374,
                            n_0 = 1,
                            res = 100,
                            eseis = TRUE)

## plot spectrum
plot_spectrum(data = p_bedload,
              ylim = c(-170, -110))

## define mean and sigma for parametric distribution function
d_50 <- 1
sigma <- 0.1

## define raised cosine distribution function following Tsai et al. (2012)
d_1 <- 10^seq(log10(d_50 - 5 * sigma),
              log10(d_50 + 5 * sigma),
              length.out = 20)

sigma_star <- sigma / sqrt(1 / 3 - 2 / pi^2)
```

```
p_1 <- (1 / (2 * sigma_star) *
          (1 + cos(pi * (log(d_1) - log(d_50)) / sigma_star))) / d_1
p_1[log(d_1) - log(d_50) > sigma_star] <- 0
p_1[log(d_1) - log(d_50) < -sigma_star] <- 0
p_1 <- p_1 / sum(p_1)

gsd_raised_cos <- data.frame(d = d_1,
                             p = p_1)
```

`model_turbulence` *Model the seismic spectrum due to hydraulic turbulence*

Description

The function calculates the seismic spectrum as predicted by the model of Gimbert et al. (2014) for hydraulic turbulence. The code was written to R by Sophie Lagarde and integrated to the R package 'eseis' by Michael Dietze.

Usage

```
model_turbulence(
  d_s,
  s_s,
  r_s = 2650,
  h_w,
  w_w,
  a_w,
  f = c(1, 100),
  r_0,
  f_0,
  q_0,
  v_0,
  p_0,
  n_0,
  res = 1000,
  eseis = FALSE,
  ...
)
```

Arguments

`d_s`	`Numeric` value, mean sediment grain diameter (m)
`s_s`	`Numeric` value, standard deviation of sediment grain diameter (m)
`r_s`	`Numeric` value, specific sediment density (kg / m^3)
`h_w`	`Numeric` value, fluid flow depth (m)

w_w	Numeric value, fluid flow width (m)
a_w	Numeric value, fluid flow inclination angle (radians)
f	Numeric vector, frequency range to be modelled. If of length two the argument is interpreted as representing the lower and upper limit and the final length of the frequency vector is set by the argument res. If f contains more than two values it is interpreted as the actual frequency vector and the value of res is ignored.
r_0	Numeric value, distance of seismic station to source
f_0	Numeric value, reference frequency (Hz)
q_0	Numeric value, ground quality factor at f_0
v_0	Numeric value, phase velocity of the Rayleigh wave at f_0 (m/s)
p_0	Numeric value, variation exponent of Rayleigh wave velocities with frequency (dimensionless)
n_0	Numeric vector of length two, Greens function displacement amplitude coefficients. Cf. N_ij in eq. 36 in Gimbert et al. (2014)
res	Numeric value, output resolution, i.e. length of the spectrum vector. Default is 1000.
eseis	Character value, option to return an eseis object instead of a data frame. Default is FALSE.
...	Further arguments passed to the function.

Details

The model uses a set of predefined constants. These can also be changed by the user, using the ... argument:

- c = 0.5, instantaneous fluid-grain friction coefficient (dimensionless)
- g = 9.81, gravitational acceleration (m/s^2)
- k = 0.5, Kolmogrov constant (dimensionless)
- k_s = 3 * d_s, roughness length (m)
- h = k_s / 2, reference height of the measurement (m)
- e_0 = 0, exponent of Q increase with frequency (dimensionless)
- r_w = 1000, specific density of the fluid (kg/m^3)
- c_w = 0.5, instantaneous fluid-grain friction coefficient (dimensionless)

Value

eseis object containing the modelled spectrum.

Author(s)

Sophie Lagarde, Michael Dietze

Examples

```
## model the turbulence-related power spectrum
P <- model_turbulence(d_s = 0.03, # 3 cm mean grain-size
                      s_s = 1.35, # 1.35 log standard deviation
                      r_s = 2650, # 2.65 g/cm^3 sediment density
                      h_w = 0.8, # 80 cm water level
                      w_w = 40, # 40 m river width
                      a_w = 0.0075, # 0.0075 rad river inclination
                      f = c(1, 200), # 1-200 Hz frequency range
                      r_0 = 10, # 10 m distance to the river
                      f_0 = 1, # 1 Hz Null frequency
                      q_0 = 10, # 10 quality factor at f = 1 Hz
                      v_0 = 2175, # 2175 m/s phase velocity
                      p_0 = 0.48, # 0.48 power law variation coefficient
                      n_0 = c(0.6, 0.8), # Greens function estimates
                      res = 1000) # 1000 values build the output resolution

## plot the power spectrum
plot_spectrum(data = P)
```

---

ncc_correlate *Noise Cross Correlation routine*

Description

The function creates a cross correlation time series of two input data sets. The input data is cut into overlapping snippets and, optionally, further smaller sub snippets for averaging the results per time snippet. The data can be made subject to a series of optional preprocessing steps, i.e. be aggregated, deconvolved, filtered, cut by standard deviation, and sign-cut. the cross correlation function is calculated and returned for a defined lag time window. The output of the function is supposed to be used as input for the function `ncc_process()`.

Usage

```
ncc_correlate(
  start,
  stop,
  ID,
  component,
  dir,
  window,
  overlap = 0,
  window_sub,
  lag,
  dt,
  deconvolve = FALSE,
  f,
```

```
  pick = FALSE,
  whiten = FALSE,
  sd,
  sign = FALSE,
  cpu,
  buffer = 0.05,
  eseis = TRUE,
  ...
)
```

Arguments

start	POSIXct value, start time for data set to be analysed. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
stop	POSIXct value, stop time for calculation. If a character string (or vector) is provided, the function will try to convert it to POSIXct.
ID	Character vector of length two, IDs of the station pair to be processed. This can either be two separate stations (assuming the components of those two stations are identical) or the same station (assuming the components of that one stations differ).
component	Character vector of length two, seismic components to be used. If omitted, the function will use c("Z", "Z") by default.
dir	Character value, path to directory that contains the seismic files. See read_data for details and constraints on the data structure.
window	Numeric value, window size for the correlation calculation, in seconds.
overlap	Numeric value, fraction of window overlap. Set to zero by default, i.e. no overlap.
window_sub	Numeric value, size of the sub window used for averaging (stacking) the correlation function within a window. If omitted, no averaging will be performed.
lag	Numeric value, window size of the cross correlation function, in seconds.
dt	Numeric mandatory value, sampling interval to which the input data will be aggregated. Note that the aggregation dt has to be a multiple of the data sets' dt values. If unsure, set value to original sampling interval of input data. See details for further information.
deconvolve	Logical value, option to deconvolve the input data sets. If set to TRUE, further parameters need to be provided, see signal_deconvolve() for details.
f	Numeric value or vector of length two, lower and/or upper cutoff frequencies (Hz). If two values are provided, the function assumes a bandpass filter. See signal_filter for details.
pick	Logical value, option to remove signal snippets where erratic seismic events can be picked in any of the signals. The supported method is pick_stalta and requires the provision of all mandatory arguments (sta, lta, on, off, freeze). Attention, the sta and lta lengths must be provided in seconds, not as the number of samples!

whiten	Logical value, option to perform spectral whitening. Default is FALSE (no whitening).
sd	Numeric value, option to perform standard deviation based signal amplitude cutting. If omitted, no cutting will be performed. The value can be any natural number, describing a multiplier of the standard deviation magnitude of the data.
sign	Logical value, option to perform sign based cutting of the signal amplitude. Default is FALSE (no cutting).
cpu	Numeric value, fraction of CPUs to use. If omitted, only one CPU will be used.
buffer	Numeric value, size of the two sided buffer that will be added to the windowed time series to account for preprocessing artifacts. The buffer is defined as the fraction of the window size. By default it is 0.05.
eseis	Logical value, option to return data as eseis object, default is TRUE.
...	Further arguments passed to the functions signal_deconvolve, signal_filter

Details

The sampling interval (dt must be defined). It is wise to set it to more than twice the filter's higher corner frequency (f[2]). Aggregation is recommended to improve computational efficiency, but is mandatory if data sets of different sampling intervals are to be analysed. In that case, it must be possible to aggregate the data sets to the provided aggregation sampling interval (dt), otherwise an error will arise. As an example, if the two data sets have sampling intervals of 1/200 and 1/500, the highest possible aggregated sampling interval is 1/100. See aux_commondt() for further information.

The function supports parallel processing. However, keep in mind that calculating the cross correlation functions for large data sets and large windows will draw serious amounts of memory. For example, a 24 h window of two seismic signals recorded at 200 Hz will easily use 15 GB of RAM. Combining this with parallel processing will multiply that memory size. Therefore, it is better think before going for too high ambitions, and check how the computer system statistics evolve with increasing windows and parallel operations.

Deconvolution is recommended if different station hardware and setup is used for the stations to analyse (i.e., different sensors, loggers or gain factors).

To account for biases due to brief events in the signals, the data sets can be truncated (cut) in their amplitude. This cutting can either be done based on the data sets' standard deviations (or a multiplicator of those standard deviations, see signal_cut() for further details), using the argument sd = 1, for one standard deviation. Alternatively, the data can also be cut by their sign (i.e., positive and negative values will be converted to 1 and -1, respectively).

Value

List with spectrogram matrix, time and frequency vectors.

Author(s)

Michael Dietze

Examples

```
## Not run:

## calculate correlogram
cc <- ncc_correlate(start = "2017-04-09 00:30:00",
                    stop = "2017-04-09 01:30:00",
                    ID = c("RUEG1", "RUEG2"),
                    dt = 1/10,
                    component = c("Z", "Z"),
                    dir = paste0(system.file("extdata",
                                             package = "eseis"), "/"),
                    window = 600,
                    overlap = 0,
                    lag = 20,
                    deconvolve = TRUE,
                    sensor = "TC120s",
                    logger = "Cube3extBOB",
                    gain = 1,
                    f = c(0.05, 0.1),
                    sd = 1)

## plot output
plot_correlogram(cc)


## End(Not run)
```

---

ncc_stretch *Estimate relativ wave velocity change (dv/v) by correlation stretching*

Description

The function estimates the relative seismic wave velocity changes over time based on matching iteratively stretched master correlations to previously calculated correlograms (cf. `ncc_correlate`).

Usage

```
ncc_stretch(
  data,
  lag,
  master = "mean",
  normalise = TRUE,
  sides = "both",
  range = 0.01,
  steps = 100,
  method = "r",
  reject = 0,
  ...
)
```

Arguments

data	eseis object of type correlation, output of aux_correlate.
lag	Numeric vector of length two, range of the time lage to analyse. If omitted, the time lag of the input data (xCClag) is used.
master	Character vector or value, either a user defined master correlation function or a keyword denoting the method used to calculate master the correlation function. One out of "mean", "median" and "quantile". Default is "mean". if "quantile" is used, the quantile probability must be specified as well, e.g., "probs = 0.5".
normalise	Logical value, option to normalise the data set before calculating the master trace. Default is TRUE.
sides	Character value. One out of "both" (both sides of the input data), "left" (only negative time lags), "right" (only positive time lags) and single (only right side is used, expecting data from a single source and direction). Default is "both".
range	Numeric value, relative range of the stretch. Default is 0.01 (1 percent).
steps	Numeric value, number of stretch steps (step resolution). Default is 100.
method	Charcter value, method used to identify best match of cross correlation time slices with stretched master data set. One out of "rmse" (minimum root mean square error) and "r" (maximum R^2). Default is "r".
reject	Numeric value, rejection threshold for stretch values. This value defines up to which quantile matching stretch solutions will be treated as valid solutions. Default is 0 (Only the minimum RMSE value or the maximum R^2 value is returned, and the returned standard deviation will be NA). A change to 0.05 will return mean and standard deviation of the five best percent of the solutions.
...	Further arguments passed to the function.

Value

A data.frame, object with the time and relative wave velocity change estimate.

Author(s)

Michael Dietze

Examples

```
## Not run:

cc <- ncc_preprocess(start = "2017-04-09 00:30:00",
                     stop = "2017-04-09 01:30:00",
                     ID = c("RUEG1", "RUEG2"),
                     component = c("Z", "Z"),
                     dir = paste0(system.file("extdata",
                                              package = "eseis"), "/"),
                     window = 600,
                     overlap = 0,
```

```
                    lag = 20,
                    deconvolve = TRUE,
                    sensor = "TC120s",
                    logger = "Cube3extBOB",
                    gain = 1,
                    f = c(0.05, 0.1),
                    sd = 1)

  ## estimate dv/v
  dv <- ncc_stretch(data = cc, range = 0.05)

  ## plot result
  plot(dv$time, dv$dvv, type = "l")


## End(Not run)
```

`pick_correlation` *Signal correlation based event picking*

Description

The function picks (identifies) events from continuous data by comparing the data patterns against a template signal using Pearson's correlation coefficient, defining an event when that coefficient is above a threshold value.

Usage

```
pick_correlation(data, on, template, dur_min, time, dt)
```

Arguments

`data`	`eseis` object, `numeric` vector or list of objects, data set to be processed.
`on`	`Numeric` value, minimum correlation coefficient to define event occurence.
`template`	`eseis` object or signal vector, template event with which the data set is correlated.
`dur_min`	`Numeric` value, minimum duration of the event. This is required as the routine tends to identify multipe picks with similarly high correlation coefficients due to autocorrelation effects. If omitted, `dur_min` is set to `0`, i.e., all picks are returned.
`time`	`POSIXct` vector, time vector of the signal(s). If not provided, a synthetic time vector will be created.
`dt`	`Numeric` value, sampling period. If omitted, either estimated from `time` or set to 0.01 s (i.e., f = 100 Hz).

Value

`data.frame`, picked events.

Author(s)

Michael Dietze

Examples

```
## create synthetic event signal
p <- sin(seq(0, 10 * pi, by = 0.35)) * 0.2 *
  (1 + sin(seq(0, pi, length.out = 90)))^5

## show event signal
plot(p, type = "l")

## create synthetic noise signal
x <- runif(n = 1000, min = -1, max = 1)
t <- seq(from = Sys.time(), length.out = length(x), by = 1/200)
ii <- floor(runif(n = 3, min = 100, max = 900))

## add events to noise
for(k in 1:length(ii)) {

  nn <- ii[k]:(ii[k] + 89)
  x[nn] <- x[nn] + p
}

## show resulting time series
plot(x = t, y = x, type = "l")

## pick events based on template
picks <- eseis::pick_correlation(data = x,
                                 on = 0.8,
                                 template = p,
                                 time = t,
                                 dt = 1/200)

## show result
print(picks)
```

`pick_kurtosis` *Kutosis based event picking*

Description

The function picks (identifies) events from continuous data using the kurtosis of the signal, and when it reaches beyond a defined threshold value. The end of an event is determined by the signal-to-noise ratio (SNR)

Usage

```
pick_kurtosis(
  data,
  on,
  off = 1,
  dur_min = 0,
  dur_max,
  window_kurt,
  window_amp,
  time,
  dt
)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
on	Numeric value, kurtosis threshold that defines the onset of an event.
off	Numeric value, ratio of average post and pre event signal amplitude inside a running window. Default is 1.
dur_min	Numeric value, minimum duration of the event. This is required as the kurtosis routine tends to identify multipe picks in the beginning of an event.
dur_max	Numeric value, maximum duration of the event. This value can be omitted but would increase computational speed as it determines the length of samples to look for the amplitude ratio that signals the end of an event
window_kurt	Numeric value, size of the running window (in number of samples) in which the kurtosis is calculated.
window_amp	Numeric value, size of the running window (in number of samples) in which the running mean is calculated.
time	POSIXct vector, time vector of the signal(s). If not provided, a synthetic time vector will be created.
dt	Numeric value, sampling period. If omitted, either estimated from time or set to 0.01 s (i.e., f = 100 Hz).

Details

Further reading:

Baillard, C., Crawford, W.C., Ballu, V., Hibert, C., Mangeney, A., 2014. An automatic kurtosis-based p- and s-phase picker designed for local seismic networks. Bull. Seismol. Soc. Am. 104 (1), 394–409.

Hibert, C., Mangeney, A., Grandjean, G., Baillard, C., Rivet, D., Shapiro, N.M., Satriano, C., Maggi, A., Boissier, P., Ferrazzini, V., Crawford, W., 2014. Automated identification, location, and volume estimation of rockfalls at Piton de la Fournaise Volcano. J. Geophys. Res. Earth Surf. 119 (5), 1082–1105. http://dx.doi.org/10.1002/2013JF002970.

Value

`data.frame`, picked events.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## preprocess signal (aggregate to increase speed, filter, envelope)
s <- signal_aggregate(data = rockfall_eseis, n = 4)
s <- signal_filter(data = s, f = c(5, 20), lazy = TRUE)
e <- signal_envelope(data = s)

## pick events based on signal kurtosis
p <- eseis::pick_kurtosis(data = e,
                          window_kurt = 200,
                          on = 15,
                          off = 5,
                          dur_min = 10,
                          dur_max = 90,
                          window_amp = 300)

p$picks
```

---

pick_stalta *Calculate stal-lta-ratio.*

Description

The function calculates the ratio of the short-term-average and long-term-average of the input signal.

Usage

```
pick_stalta(data, time, dt, sta, lta, freeze = FALSE, on, off)
```

Arguments

`data`	`eseis` object, `numeric` vector or list of objects, data set to be processed.
`time`	`POSIXct` vector, time vector of the signal(s). If not provided, a synthetic time vector will be created.
`dt`	`Numeric` value, sampling period. If omitted, either estimated from `time` or set to 0.01 s (i.e., f = 100 Hz).

sta	Numeric value, number of samples for short-term window.
lta	Numeric value, number of samples for long-term window.
freeze	Logical value, option to freeze lta value at start of an event. Useful to avoid self-adjustment of lta for long-duration events.
on	Numeric value, threshold value for event onset.
off	Numeric value, threshold value for event end.

Value

data frame, detected events (ID, start time, duration in seconds, STA-LTA vaue).

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## filter signal
rockfall_f <- signal_filter(data = rockfall_eseis,
                            f = c(1, 90),
                            p = 0.05)

## calculate signal envelope
rockfall_e <- signal_envelope(data = rockfall_f)

## pick earthquake and rockfall event
p <- pick_stalta(data = rockfall_e,
                 sta = 100,
                 lta = 18000,
                 freeze = TRUE,
                 on = 5,
                 off = 3)

p$picks
```

---

plot_components *Plot three seismic components against each other*

Description

The function visualises the time evolution of three seismic components of the same signal against each other as line graphs. There are three different visualisation types available: 2D (a panel of three 2D plots), 3D (a perspective threedimensional plot) and scene (an interactive threedimensional plot, mainly for exploratory purpose).

Usage

```
plot_components(data, type = "2D", order = "xyz", ...)
```

Arguments

data	List, data frame or matrix, seismic componenents to be processed. If data is a matrix, the components must be organised as columns. Also, data can be a list of eseis objects.
type	Character value, plot type. One out of "2D" (panel of three 2-dimensional plots), "3D" (perspective 3D plot) and "scene" (interactive 3D plot). Default is "2D".
order	Caracter value, order of the seismic components. Describtion must contain the letters "x","y" and "z" in the order according to the input data set. Default is "xyz" (NW-SE-vertical).
...	Further arguments passed to the plot function.

Details

The plot type type = "3D" requires the package plot3D being installed. The plot type type = "scene" requires the package rgl being installed.

Value

A plot

Author(s)

Michael Dietze

Examples

```
## load example data set
data(earthquake)

## filter seismic signals
s <- eseis::signal_filter(data = s,
                          dt = 1/200,
                          f = c(0.05, 0.1))

## integrate signals to get displacement
s_d <- eseis::signal_integrate(data = s, dt = 1/200)

## plot components in 2D
plot_components(data = s_d,
                type = "2D")

## plot components with time colour-coded
plot_components(data = s_d,
                type = "2D",
                col = rainbow(n = length(s$BHE)))
```

```
## plot components with used defined coulour ramp
col_user <- colorRampPalette(colors = c("grey20", "darkblue", "blue",
                                        "green", "red", "orange"))

plot_components(data = s_d,
                type = "2D",
                col = col_user(n = length(s$BHE)))

## plot components as 3D plot, uncomment to use
#plot_components(data = s_d,
#                 type = "3D",
#                 col = rainbow(n = length(s$BHE)))
```

`plot_correlogram` *Plot a correlogram from noise cross correlation analysis*

Description

The function uses the output of `ncc_correlate()` to show an image plot of a noise cross correlation analysis.

Usage

```
plot_correlogram(data, agg = c(1, 1), legend = TRUE, keep_par = FALSE, ...)
```

Arguments

`data`	`List` object, spectrogram to be plotted. Must be output of `ncc_correlate()` or of equivalent structure.
`agg`	`Integer` vector of length two, factors of image aggregation, i.e. in time and lag dimension. Useful to decrease image size. Default is `c(1, 1)` (no aggregation).
`legend`	`Logical` value, option to add colour bar legend. Legend label can be changed by `zlab`.
`keep_par`	`Logical` value, option to omit resetting plot parameters after function execution. Useful for adding further data to the plot. Default is `FALSE` (parameters are reset to original values).
`...`	Additional arguments passed to the plot function.

Value

Graphic output of a correlogram.

Author(s)

Michael Dietze

See Also

ncc_correlate

Examples

```
## Not run:

  ## calculate correlogram
  cc <- ncc_correlate(start = "2017-04-09 00:30:00",
                      stop = "2017-04-09 01:30:00",
                      ID = c("RUEG1", "RUEG2"),
                      component = c("Z", "Z"),
                      dir = paste0(system.file("extdata",
                                   package = "eseis"), "/"),
                      window = 600,
                      overlap = 0,
                      lag = 20,
                      f = c(0.05, 0.1),
                      sd = 1)

   ## explicit plot function call with adjusted resolution
   plot_correlogram(data = cc, agg = c(2, 5))

   ## define plot colour scale
   cls <- colorRampPalette(colors = c("brown", "white", "green"))

   ## simple function call with user-defined colour scale
   plot(cc, col = cls(100))

## End(Not run)
```

---

plot_event *Create a comprehensive multi panel plot of a seismic waveform*

Description

The function creates from an input waveform a multi panel plot, including a seismogram, spectrum and spectrogram, and additional frequency statistics information.

Usage

```
plot_event(data, ratio = c(0.3, 0.3), ...)
```

Arguments

data	eseis object, or numeric vector, data set to be plotted.
ratio	Numeric vector of length two, ratios of the plot panels, i.e. in horizonal and vertical direction. Default is c(0.3, 0.3).

...	Additional arguments passed to the plot function. See details for further information

Details

Note that plot generation time can get long when other than short events are passed to the function. The axes limits can only be changed for the spectrum and spectrogram plots, `ylim` affects the frequency range, `zlim` affects the spectral power range.

The function uses the native plot function `plot_signal()`, `plot_spectrum()` and `plot_spectrogram()` along with `signal_stats()` to build a four panel plot.

Value

Graphic output of an event waveform.

Author(s)

Michael Dietze

Examples

```
## Not run:

## load and deconvolve example event
data(rockfall)
rockfall_eseis <- signal_deconvolve(rockfall_eseis)

## plot event straight away
plot_event(data = rockfall_eseis)

## plot event with adjusted parameters
plot_event(data = rockfall_eseis,
           ratio = c(0.4, 0.3),
           method = "periodogram",
           n = 100,
           window = 6,
           overlap = 0.8,
           window_sub = 4,
           overlap_sub = 0.8,
           format ="%M:%S",
           col = "jet",
           ylim = c(5, 80),
           zlim = c(-170, -100))

## End(Not run)
```

plot_ppsd *Plot a probabilistic power spectral density estimate (PPSD)*

Description

The function uses the output of `signal_spectrogram()` to plot a probabilistic power spectral density estimate.

Usage

```
plot_ppsd(data, res = c(500, 500), n, ...)
```

Arguments

`data`	`List` object, spectrogram to be plotted. Must be output of `signal_spectrogram()` or of equivalent structure.
`res`	`Integer` vector of length two, factors of image resolution in pixels, i.e. in time and frequency dimension. Default is `c(100, 100)`.
`n`	`Integer` vector of length two, factors by which the image will be smoothend by a running average. `n` sets the filter window size, in x and y direction, respectively. By default, the window sizes are set to one percent of the input data set dimension.
`...`	Additional arguments passed to the plot function.

Value

Graphic output of a spectrogram.

Author(s)

Michael Dietze

See Also

`signal_spectrogram`

Examples

```
## load example data set
data(rockfall)

## deconvolve data set
r <- signal_deconvolve(data = rockfall_eseis)

## calculate PSD
p <- signal_spectrogram(data = r)

## plot PPSD
```

```
plot_ppsd(data = p$PSD)

## plot PPSD with lower resolution, more smoothing and other colour
ppsd_color <- colorRampPalette(c("white", "black", "red"))

plot_ppsd(data = p$PSD,
          res = c(200, 200),
          n = c(15, 20),
          col = ppsd_color(200))
```

`plot_signal` *Plot a seismic signal*

Description

This function plots a line graph of a seismic signal. To avoid long plot preparation times the signal is reduced to a given number of points.

Usage

```
plot_signal(data, time, n = 10000, ...)
```

Arguments

`data`	`eseis` object or `numeric` vector, data set to be plotted.
`time`	`POSIXct` vector, corresponding time vector.
`n`	`Numeric` value, number of values to which the dataset is reduced. Default is `10000`.
`...`	Further arguments passed to the plot function.

Details

The `format` argument is based on hints provided by Sebastian Kreutzer and Christoph Burow. It allows plotting time axis units in user defined formats. The time format must be provided as character string using the POSIX standard (see documentation of `strptime` for a list of available keywords), e.g., " "

Value

A line plot of a seismic wave form.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## plot data set straightforward
plot_signal(data = rockfall_eseis)

## plot data set with lower resolution
plot_signal(data = rockfall_eseis, n = 100)

## plot data set but not as an eseis object
plot_signal(data = rockfall_z, time = rockfall_t)

## load earthquake data set
data(earthquake)

## plot all three components (after changing plot options)
pars <- par(no.readonly = TRUE)
par(mfcol = c(3, 1))

plt <- lapply(s, plot_signal, t = t)

par(pars)
```

---

plot_spectrogram *Plot spectrograms (power spectral density estimates)*

Description

This function plots spectrograms of seismic signals. It uses the output of `signal_spectrogram`.

Usage

```
plot_spectrogram(data, legend = TRUE, keep_par = FALSE, agg = c(1, 1), ...)
```

Arguments

`data`	`List` object, spectrogram to be plotted. Must be output of `signal_spectrogram` or of equivalent structure.
`legend`	`Logical` value, option to add colour bar legend. Legend label can be changed by `zlab`.
`keep_par`	`Logical` value, option to omit resetting plot parameters after function execution. Useful for adding further data to the PSD plot. Default is `FALSE` (parameters are reset to original values).
`agg`	`Integer` vector of length two, factors of image aggregation, i.e. in time and frequency dimension. Useful to decrease image size. Default is `c(1, 1)` (no aggregation).
`...`	Additional arguments passed to the plot function.

Details

As of version 0.7.2, the value range (`zlim`) is no longer set to the full data range but to the range between quantiles 0.01 and 0.99. For the full value range to be plotted, use `zlim = range(data$PSD$S)`.

As of version 0.7.2, the default plot colour has changed from the "jet" colour palette to the "Inferno" palette. This due to perception issues with the "jet" palette. If one wants to decisively use the "jet" colours, this can be done by adding the keyword `col = "jet"`. To use other colour schemes, such as sequential HCL schemes from the colorspace package, specify them as additional argument, e.g. `col = colorspace::sequential_hcl(200, palette = "Plasma")`, `col = colorspace::sequential_hcl(200, palette = "Inferno")`, `col = colorspace::sequential_hcl(200, palette = "Viridis")`.

Value

Graphic output of a spectrogram.

Author(s)

Michael Dietze

See Also

signal_spectrogram

Examples

```
## load example data set
data(rockfall)

## deconvolve signal
rockfall <- signal_deconvolve(data = rockfall_eseis)

## calculate spectrogram
PSD <- signal_spectrogram(data = rockfall)

## plot spectrogram
plot_spectrogram(data = PSD)

## plot spectrogram with legend and labels in rainbow colours
plot_spectrogram(data = PSD,
                 xlab = "Time (min)",
                 ylab = "f (Hz)",
                 main = "Power spectral density estimate",
                 legend = TRUE,
                 zlim = c(-220, -70),
                 col = rainbow(100))

## plot spectrogram with frequencies in log scale
plot_spectrogram(data = PSD, log = "y")

## plot spectrogram with formatted time axis (minutes and seconds)
plot_spectrogram(data = PSD, format = "%M:%S")
```

plot_spectrum *Plot a spectrum of a seismic signal*

Description

This function plots a line graph of the spectrum of a seismic signal.

Usage

```
plot_spectrum(data, unit = "dB", n = 10000, ...)
```

Arguments

data	eseis object or data frame with two elements, frequency vector and spectrum vector.
unit	Character value. One out of "linear", "log", "dB". Default is "dB".
n	Numeric value, number of values to which the dataset is reduced. Default is 10000.
...	Further arguments passed to the plot function.

Value

A line plot.

Author(s)

Michael Dietze

See Also

signal_spectrum

Examples

```
## load example data set
data(rockfall)

## calculate spectrum
spectrum_rockfall <- signal_spectrum(data = rockfall_eseis)

## plot data set with lower resolution
plot_spectrum(data = spectrum_rockfall)
```

read_data *Load seismic data from an archive*

Description

The function loads seismic data from a data directory structure (see aux_organisecubefiles) based on the event start time, duration, component and station ID. The data to be read needs to be adequately structured. The data directory must contain mseed or SAC files. These files will either be identified automatically or can be defined explicitly by the parameter format.

Usage

```
read_data(
  start,
  duration,
  station,
  component = "BHZ",
  format,
  dir,
  pattern = "eseis",
  simplify = TRUE,
  interpolate = FALSE,
  eseis = TRUE,
  try = TRUE,
  ...
)
```

Arguments

start	POSIXct value, start time of the data to import. If lazy users only submit a text string instead of a POSIXct object, the function will try to convert that text string, assuming UTC as time zone.
duration	Numeric value, duration of the data to import, in seconds.
station	Character value, seismic station ID, which must correspond to the ID in the file name of the data directory structure (cf. aux_organisecubefiles).
component	Character value, seismic component, which must correspond to the component name in the file name of the data directory structure (cf. aux_organisecubefiles). Information can be restricted to a single character, denoting the last character of the station component code, i.e. "Z" would use the first and last component of c("BHZ", "HHE", "HHZ"). Default is "BHZ" (vertical component of a seismic file).
format	Character value, seismic data format. One out of "sac" and "mseed". If omitted, the function will try to identify the right format automatically.
dir	Character value, path to the seismic data directory. See details for further info on data structure.

pattern	Character value, either keyword or pattern string with wildcards, describing the data organisation. Supported keywords are "eseis" and "seiscomp". See details for keyword definition and format of pattern strings. Default option is eseis.
simplify	Logical value, option to simplify output when possible. This basically means that if only data from one station is loaded, the list object will have one level less. Default is TRUE.
interpolate	Logical value, option to interpolate possible gaps in the resulting data stream. If enabled, NA values will be identified and linearly interpolated using the function signal_fill. Default is FALSE, i.e. NA gaps will remain in the imported data set.
eseis	Logical value, option to read data to an eseis object (recommended, see documentation of aux_initiateeseis), default is TRUE
try	Logical value, option to run the function in try-mode, i.e., to let it return NA in case an error occurs during data import. Default is FALSE.
...	Further arguments to describe data structure, only needed for pattern type seiscomp. These arguments can be one or more of the following: "network", "type", "location". If omitted, the function will identify all files in the SeisComP data archive that fulfill the criteria. If other than data files (type = "D") or files from another network are in the archive, these may lead to crashes of the function.

Details

Data organisation must follow a consistent scheme. The default scheme, eseis (Dietze, 2018 ESurf) requires hourly files organised in a directory for each Julian Day, and in each calendar year. The file name must be entirely composed of station ID, 2-digit year, Julian Day, hour, minute, second and channel name. Each item must be separated by a full stop, e.g. "2013/203/IGB01.13.203.16.00.00.BHZ" for a file from 2013, Julian Day 203, from station IGB01, covering one hour from "16:00:00 UTC", and containing the BHZ component. Each Julian Day directory can contain files from different components and stations. The respective pattern string to describe that file organisation is "%Y/%j/%STA.%y.%j.%H.%M.%S.%CMP". The percent sign indicates a wild card, where %Y is the 4-digit year, %j the 3-digit Julian Julian Day, %STA the station ID, %y the 2-digit year, %H the 2-digit hour, %M the 2-digit minute, %S the 2-digit second and %CMP the component ID. The files can have a further file extension which does not need to be explicitly defined in the pattern string. The slashes in the above pattern string define subdirectories.

An alternative organisation scheme is the one used by SeisComP, indicated by the keyword "seiscomp" or the pattern string "%Y/%NET/%STA/%CMP/%NET.%STA.%LOC.%CMP.%TYP.%Y.%j". The wild card "NET" means the network ID, "LOC" the location abbreviation and "TYP" the data type. The other wild cards are as defined above. Hence, the SeisComP scheme consists of directories of the calendar year, the network to which the data belongs, the station it has been recorded by, and the component it belongs to. The files in that latter directory must be daily files.

Value

A list object containing either a set of eseis objects or a data set with the time vector ($time) and a list of seismic stations ($station_ID) with their seismic signals as data frame ($signal). If simplify = TRUE (the default option) and only one seismic station is provided, the output object containseither just one eseis object or the vectors for $time and $signal.

Author(s)

Michael Dietze

Examples

```
## set seismic data directory
dir_data <- paste0(system.file("extdata", package="eseis"), "/")

## load the z component data from a station
data <- read_data(start = as.POSIXct(x = "2017-04-09 01:20:00",
                                     tz = "UTC"),
                      duration = 120,
                      station = "RUEG1",
                      component = "BHZ",
                      dir = dir_data)
## plot signal
plot_signal(data = data)

## load data from two stations
data <- read_data(start = as.POSIXct(x = "2017-04-09 01:20:00",
                                     tz = "UTC"),
                  duration = 120,
                  station = c("RUEG1", "RUEG2"),
                  component = "BHZ",
                  dir = dir_data)

## plot both signals
par(mfcol = c(2, 1))
lapply(X = data, FUN = plot_signal)
```

---

read_fdsn *Download and import seismic data from an FDSN service provider*

Description

The function implements download and import of seismic data from FDSN data providers via the fdsnws-dataselect service (see https://www.fdsn.org/webservices/). It is basically a wrapper for the query approach.

Usage

```
read_fdsn(
  start,
  duration,
  station,
  network,
  component = "BHZ",
  url,
```

```
  eseis = TRUE,
  ...
)
```

Arguments

start	POSIXct value, start time of the data to import. If lazy users only submit a text string instead of a POSIXct object, the function will try to convert that text string, assuming UTC as time zone.
duration	Numeric value, duration of the data to import, in seconds.
station	Character vector, optional 3-4-digit FDSN station ID.
network	Character vector, optional 2-digit FDSN network ID.
component	Character vector, seismic component to search for. If omitted, the function will look for "BHZ" by default.
url	Chracter vector, URL of the FDSN data provider. Should be of the form "http://service.iris.edu", i.e., without further URL parts. URLs can be submitted as a vector. If omitted, the function will look in the two most comprehensive providers, i.e. url = c("http://service.iris.edu", "http://eida-federator.ethz.ch"). See details for further information.
eseis	Logical value, option to read data to an eseis object (recommended, see documentation of aux_initiateeseis), default is TRUE
...	Additional query arguments sent to the FDSN data provider. See details for available argument names and conventions.

Details

The FDSN (International Federation of Digital Seismograph Networks) provides access to a large number of seismic stations worldwide. The data are organised by network, station, component and further arguments. In order to use the eseis function read_fdsn, one must know at least the former three criteria for the data of interest. A list of networks is available here: https://www.fdsn.org/networks/. The function expects the 2-digit network code, the 3- or 4-digit station code, a single seismic component ID, and the URL to the data archive. Additional query arguments can be added (and must be added to point at a single seismic trace to download and import). A complete list of query arguments is available here: https://www.fdsn.org/webservices/fdsnws-dataselect-1.1.pdf.

For each network listed there, one can find the URL that gives access to the data (if existing) under "Data Access". Note that the function only requires the first URL part, e.g., https://geofon.gfz-potsdam.de.

Value

An eseis object or a list with the time ($time) and $signal vectors as well as meta information.

Author(s)

Michael Dietze

Examples

```
## Not run:

## read and plot 10 min of data from Ecuador, specifying the component
s <- read_fdsn(start = "2020-05-16 22:42:00",
               duration = 360,
               station = "IMBA",
               network = "EC",
               component = "HHZ")
plot(s)

## read and plot 10 min of data from Germany, specifying the URL
s <- read_fdsn(start = "2017-03-21 04:38:00",
               duration = 360,
               station = "RGN",
               network = "GE",
               url = "http://geofon.gfz-potsdam.de")
plot(s)


## End(Not run)
```

`read_mseed` *Read mseed files.*

Description

This function reads mseed files. If `append = TRUE`, all files will be appended to the first one in the order as they are provided. In the append-case the function returns a either a list with the elements `signal`, `time`, `meta` and `header` or a list of the class `eseis` (see documentation of `aux_initiateeseis()`). If `append = FALSE` and more than one file is provided, the function returns a list of the length of the input files, each containing the above elements.

Usage

```
read_mseed(
  file,
  append = TRUE,
  signal = TRUE,
  time = TRUE,
  meta = TRUE,
  header = TRUE,
  eseis = TRUE,
  type = "waveform"
)
```

Arguments

file	Character vector, input file name(s), with extension.
append	Logical value, option to append single files to one continuous file, keeping only the header information of the first file, default is TRUE.
signal	Logical value, option to import the signal vector, default is TRUE.
time	Logical value, option to create the time vector. The timezone is automatically set to "UTC", default is TRUE.
meta	Logical value, option to append the meta data part, default is TRUE.
header	Logical value, option to append the header part, default is TRUE.
eseis	Logical value, option to read data to an eseis object (recommended, see documentation of aux_initiateeseis), default is TRUE
type	Character value, type keyword of the data. One out of "waveform", "envelope", "fft", "spectrum", "spectrogram", "other", hilbert, hvratio. Default is "waveform".

Details

The mseed data format is read based on C code that was part of the now CRAN-archived package IRISSeismic v. 1.6.6 (https://cran.r-project.org/src/contrib/Archive/IRISSeismic/). The C code and wrapper are a simplified version of the material from IRISSeismic written by Jonathan Callahan. A future version of read_mseed may use a further simplified version, restricting the header information to the pure information, required by eseis to build its meta information.

Value

List object, optionally of class eseis

Author(s)

Michael Dietze

Examples

```
## Not run:

## read mseed file with default options
x <- read_mseed(file = "input.miniseed")

## read mseed file, only signal trace, not as eseis object
x <- read_mseed(file = "input.miniseed",
                time = FALSE,
                meta = FALSE,
                header = FALSE,
                eseis = FALSE)

## read more than one mseed files and append traces
x <- read_mseed(file = c("input_1.miniseed", "input_2.miniseed"))
```

```
## End(Not run)
```

---

read_sac *Read sac files.*

Description

This function reads sac files.

Usage

```
read_sac(
  file,
  append = TRUE,
  signal = TRUE,
  time = TRUE,
  meta = TRUE,
  header = TRUE,
  eseis = TRUE,
  get_instrumentdata = FALSE,
  endianness = "little",
  biglong = FALSE,
  type = "waveform"
)
```

Arguments

file	Character vector, input file name(s), with extension.
append	Logical value, option append single files to one continuous file, keeping only the header information of the first file, default is TRUE.
signal	Logical value, option to import the signal vector, default is TRUE.
time	Logical value, option to create the time vector. The timezone is automatically set to "UTC", default is TRUE.
meta	Logical value, option to append the meta data part, default is TRUE.
header	Logical value, option to append the header part, default is TRUE.
eseis	Logical value, option to read data to an eseis object (recommended, see documentation of aux_initiateeseis), default is TRUE
get_instrumentdata	Logical value, option to fill meta information (sensor name, logger name, logger gain) from SAC user fields (field 127-129, KUSER0-KUSER2). Default is FALSE.
endianness	Logical value, endianness of the sac file. One out of "little", "big" and "swap". Default is "little".
biglong	Logical value, number coding format. Default is FALSE.

type — Character value, type keyword of the data. One out of "waveform", "envelope", "fft", "spectrum", "spectrogram", "other", hilbert, hvratio. Default is "waveform".

Details

The function reads one or more sac-files. If append = TRUE, all files will be appended to the first one in the order as they are provided. In the append-case the function returns a either a list with the elements signal, time, meta and header or a list of the class eseis (see documentation of aux_initiateeseis). If append = FALSE and more than one file is provided, the function returns a list of the length of the input files, each containing the above elements.

The sac data format is implemented as descibed on the IRIS website (https://ds.iris.edu/files/sac-manual/manual/file_format.html).

Value

List object, optionally of class eseis.

Author(s)

Michael Dietze

Examples

```
## Not run:
## read one file
file1 <- "~/Data/sac/EXMP01.14.213.01.00.00.BHE.SAC"

sac1 <- read_sac(file = file1)

## read two (or more files) without meta and header parts
file2 <- c("~/Data/sac/EXMP01.14.213.01.00.00.BHE.SAC",
           "~/Data/sac/EXMP01.14.213.02.00.00.BHE.SAC")

sac2 <- read_sac(file = file2,
                 meta = FALSE,
                 header = FALSE,
                 eseis = FALSE)

## End(Not run)
```

---

rockfall — *Seismic trace of a rockfall event.*

Description

The dataset comprises the seismic signal (vertical component) of a rockfall event, preceded by an earthquake. The data have been recorded at 200 Hz sampling frequency with an Omnirecs Cube ext 3 data logger.

The dataset comprises the time vector corresponding the to seismic signal of the rockfall event from the example data set "rockfall".

The dataset comprises the seismic signal (vertical component) of a rockfall event, preceeded by an earthquake. The data have been recorded at 200 Hz sampling frequency with an Omnirecs Cube ext 3 data logger.

Usage

```
rockfall_z

rockfall_t

rockfall_eseis
```

Format

The format is: num [1:98400] 65158 65176 65206 65194 65155 ...

The format is: POSIXct[1:98400], format: "2015-04-06 13:16:54" ...

List of 4 $ signal : num [1:98399] 65211 65192 65158 65176 65206 ... $ meta :List of 12 ..$ station : chr "789 " ..$ network : chr "XX " ..$ component: chr "p0 " ..$ n : int 98399

Examples

```
## load example data set
data(rockfall)

## plot signal vector using base functionality
plot(x = rockfall_t, y = rockfall_z, type = "l")

## plot signal vector using the package plot function
plot_signal(data = rockfall_z, time = rockfall_t)


## load example data set
data(rockfall)


## load example data set
data(rockfall)
```

signal_aggregate *Aggregate a signal vector*

Description

The signal vector data is aggregated by an integer factor n. If an eseis object is provided, the meta data is updated. The function is a wrapper for the funcion decimate of the package signal.

Usage

```
signal_aggregate(data, n = 2, type = "iir")
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
n	Numeric value, number of samples to be aggregated to one new data value. Must be an integer value greater than 1. Default is 2.
type	Character value, filter type used for aggregation. For details see documentation of signal::decimate. Default is "iir".

Value

Aggregated data set.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## aggregate signal by factor 4 (i.e., dt goes from 1/200 to 1/50)
rockfall_agg <- signal_aggregate(data = rockfall_z,
                                 n = 4)

## create example data set
s <- 1:10

## aggregate x by factor 2
s_agg_2 <- signal_aggregate(data = s,
                            n = 2)

## aggregate x by factor 3
s_agg_3 <- signal_aggregate(data = s,
                            n = 3)
```

```
## plot results
plot(x = s,
     y = rep(x = 1, times = length(s)),
     ylim = c(1, 3))

points(x = s_agg_2,
       y = rep(x = 2, times = length(s_agg_2)),
       col = 2)

points(x = s_agg_3,
       y = rep(x = 3, times = length(s_agg_3)),
       col = 3)

abline(v = s_agg_2,
       col = 2)

abline(v = s_agg_3,
       col = 3)

## create signal matrix
X <- rbind(1:100, 1001:1100, 10001:10100)

## aggregate signal matrix by factor 4
X_agg <- signal_aggregate(data = X,
n = 4)
```

`signal_clip` *Clip signal based on time vector.*

Description

The function clips a seismic signal based on the corresponding time vector.

Usage

```
signal_clip(data, limits, time)
```

Arguments

`data`	`eseis` object, `numeric` vector or list of objects, data set to be processed.
`limits`	`POSIXct` vector of length two, time limits for clipping. Can also be a text string that will be converted to POSIXct format, with UTC assigned automatically.
`time`	`POSIXct` vector, corresponding time vector. Only needed if `data` is no `eseis` object.

Value

`Numeric` data set clipped to provided time interval.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## define limits (second 10 to 20 of the signal)
limits <- c(rockfall_t[1] + 10, rockfall_t[1] + 20)

## clip signal
rockfall_clip <- signal_clip(data = rockfall_z,
                             time = rockfall_t,
                             limits = limits)

## clip signal using the eseis object
rockfall_clip <- signal_clip(data = rockfall_eseis,
                             limits = limits)
```

---

signal_correlation *Calculate signal cross-correlation values*

Description

This function calculates the running cross-correlation of two or more seismic signas and returns that characteristic function.

Usage

```
signal_correlation(data, window = 200)
```

Arguments

data	`eseis` object, `list` object with data sets to be processed. The list can contain numeric vectors or eseis objects. It is assumed that all vectors have the same length.
window	`Numeric` value, size of the running window, in number of samples

Value

`Numeric` running cross-correlation of the input signals.

Author(s)

Michael Dietze

Examples

```
## calculate cross-correlation
s_cc <- signal_correlation(data = list(a = runif(1000),
                                       b = runif(1000),
                                       c = runif(1000)),
                           window = 200)
```

---

signal_cut *Cut signal amplitude at standard deviation-defined level.*

Description

This function cuts the amplitude of signal parts that exceede a user defined threshold set by k times the standard deviation of the signal.

Usage

```
signal_cut(data, k = 1)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
k	Numeric value, multiplier of the standard deviation threshold used to cut the signal amplitude. Default is 1 (1 sd).

Value

Numeric vector or list of vectors, cut signal.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## cut signal
rockfall_cut <- signal_cut(data = rockfall_eseis)
```

signal_deconvolve *Deconvolve a signal vector.*

Description

The function removes the instrument response from a signal vector.

Usage

```
signal_deconvolve(
  data,
  xml,
  sensor = "TC120s",
  logger = "Cube3BOB",
  gain = 1,
  use_metadata = FALSE,
  dt,
  url,
  xml_use,
  p = 10^-6,
  waterlevel = 10^-6,
  na.replace = FALSE,
  verbose = FALSE
)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
xml	station XML file to use, either an obspy object or path to a XML file. If provided, all other station parameters will be ignored and the station XML file will be used. Currently, the station XML approach is only supported through Obspy and when eseis objects are provided. In that case, lists of eseis cannot be supported.
sensor	Character value or list object, seismic sensor name. Must be present in the sensor library (list_sensor) or parameters must be added manually (see examples). Default is "TC120s".
logger	Character value, seismic logger name. Must be present in the logger library (list_logger) or parameters must be added manually. Default is "Cube3extBOB".
gain	Numeric value, signal gain level of the logger. Default is 1.
use_metadata	Logical value, option to take keywords for sensor, logger and gain from eseis object meta data element instead of using explicitly provided arguments. Default is FALSE.
dt	Numeric value, sampling rate. Only needed if data is not an eseis object

url	Character value, URL of the FDSN data provider. Should be of the form "http://service.iris.edu", i.e., without further URL parts. URLs can be submitted as a vector. See aux_getxml for further details. Only needed when XML file is to be downloaded.
xml_use	Logical value, only needed internally, will be set automatically.
p	Numeric value, proportion of signal to be tapered. Default is10^-6.
waterlevel	Numeric value, waterlevel value for frequency division, default is 10^-6.
na.replace	Logical value, option to replace NA values in the data set by zeros. Default is FALSE. Attention, the zeros will create artifacts in the deconvolved data set. However, NA values will result in no deconvolution at all.
verbose	Logical value, option to allow messages and warnings to be shown. Default is FALSE.

Details

The function requires a set of input parameters, apart from the signal vector. These parameters are contained in and read from the function list_sensor() and list_logger(). Poles and zeros are used to build the transfer function. The value s is the generator constant in Vs/m. The value k is the normalisation factor. AD is the analogue-digital conversion factor n Volts per count. If the signal was recorded with a gain value other than 1, the resulting signal needs to be corrected for this, as well. As of v. 0.8.0, the function also supports deconvolution using the station XML scheme. However, that feature is implemented through the python toolbox Obspy, which needs to be installed separately.

Value

Numeric vector or list of vectors, deconvolved signal.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## deconvolve signal with minimum effort
rockfall_decon <- signal_deconvolve(data = rockfall_eseis)

## plot time series
plot_signal(data = rockfall_decon,
     main = "Rockfall, deconvolved signal",
     ylab = "m/s")

## add new logger manually
logger_new <- list_logger()[[1]]

## add logger data
```

```
logger_new$ID <- "logger_new"
logger_new$name <- "logger_new"
logger_new$AD <- 2.4414e-07

## deconvolve signal with new logger
rockfall_decon <- signal_deconvolve(data = rockfall_eseis,
                                    sensor = "TC120s",
                                    logger = logger_new)

## Change the setup of a logger, here: Centaur AD is changed due to
## other than default Vpp value, according to AD = V / (2^24).

## extract default Centaur logger
Centaur_10V <- list_logger()[[2]]

## replace AD value
Centaur_10V$AD <- 20/(2^24)

## Not run:

## working with station XML files:

## download and import example data set
s <- read_fdsn(start = "2023-06-10",
               duration = 600,
               station = "DAVA",
               network = "OE",
               component = "BHZ")

## download and save station XML file
xml <- aux_getxml(xml = "OE.DAVA.XML",
                  start = "2023-06-10",
                  duration = 600,
                  network = "OE",
                  station = "DAVA",
                  component = "BHZ",
                  url = "http://service.iris.edu")

## deconvolve data set with downloaded XML file
s_d <- signal_deconvolve(data = s,
                         xml = "OE.DAVA.XML")

## alternatively, deconvolve data set by online XML file (no saving)
s_d <- signal_deconvolve(data = s,
                         xml = TRUE,
                         url = "http://service.iris.edu")


## End(Not run)
```

signal_demean *Remove mean of signal vector.*

Description

The function removes the mean from a signal vector.

Usage

```
signal_demean(data)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.

Value

Numeric vector or list of vectors, data set with mean subtracted.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## remove mean from data set
rockfall_demean <- signal_demean(data = rockfall_eseis)

## compare data ranges
range(rockfall_eseis$signal)
range(rockfall_demean$signal)

## show mean of initial signal
mean(rockfall_eseis$signal)
```

signal_detrend *Detrend a signal vector.*

Description

The function removes a trend from a signal vector.

Usage

```
signal_detrend(data, method = "linear")
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
method	Character value, method used for detrending. One out of "simple" and "linear". Default is "linear".

Details

The method "simple" subtracts a linear trend built from the first and last sample of the data set. The method "linear" uses the linear function as implemented in pracma::detrend.

Value

Numeric vector or list of vectors, detrended data set.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## remove linear trend from data set
rockfall_detrend <- signal_detrend(data = rockfall_eseis)

## compare data ranges
range(rockfall_eseis$signal)
range(rockfall_detrend$signal)
```

signal_differentiate *Differentiate a signal vector*

Description

The function integrates a signal vector to, for example convert values from displacement to velocity.

Usage

```
signal_differentiate(data)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.

Value

Derivative of the input data set.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## detrend and demean data set
s <- signal_demean(signal_detrend(data = rockfall_eseis))

## differentiate
s_d = signal_differentiate(data = s)

## plot result
plot(s_d)
```

signal_envelope *Calculate signal envelope.*

Description

The function calculates envelopes of the input signals as cosine-tapered envelope of the Hilbert-transformed signal. The signal should be detrended and/or the mean should be removed before processing.

Usage

```
signal_envelope(data, p = 0)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
p	Numeric value, proportion of the signal to be tapered, default is 0.

Value

Numeric vector or list of vectors, signal envelope.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## detrend data set
rockfall_detrend <- signal_detrend(data = rockfall_eseis)

## calculate envelope
rockfall_envelope <- signal_envelope(data = rockfall_detrend)

## plot envelope
plot_signal(data = rockfall_envelope)
```

---

signal_fill *Fill NA-gaps of a signal*

Description

This function performs linear interpolation of NA values or pads them with zeros.

Usage

```
signal_fill(data, method = "linear")
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
method	Character value, method to use for filling the data gap. One out of "linear" (linear interpolation) and "zeros" (padding with zeros). Default is "linear".

Details

Note that the procedure will contaminate the signal by artefacts as increasingly larger data gaps are filled with interpolated or zero values.

Value

`eseis` object, numeric vector or list of objects, gap-filled data set(s).

Author(s)

Michael Dietze

Examples

```
## create synthetic data set and add NA-gaps
data(rockfall)
x <- rockfall_z[25000:26000]
x_gap <- x
x_gap[100:102] <- NA
x_gap[500:530] <- NA

## fill gaps
y <- signal_fill(data = x_gap)

## plot filled data set
plot(y, type = "l")

## filter both data sets
x <- signal_filter(data = x, f = c(1, 3), dt = 1/200, lazy = TRUE)
y <- signal_filter(data = y, f = c(1, 3), dt = 1/200, lazy = TRUE)

## plot both data sets
plot(y, type = "l", col = "grey", lwd = 3)
lines(x, col = "red")
```

`signal_filter` *Filter a seismic signal in the time or frequency domain*

Description

The function filters the input signal vector in the time or frequency domain.

Usage

```
signal_filter(
  data,
  f,
  fft = FALSE,
```

```
  dt,
  type,
  shape = "butter",
  order = 2,
  p,
  zero = FALSE,
  lazy = FALSE
)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
f	Numeric value or vector of length two, lower and/or upper cutoff frequencies (Hz).
fft	Logical value, option to filter in the time domain (fft = FALSE) or the frequency domain (fft = TRUE). Default is (fft = FALSE).
dt	Numeric value, sampling rate If omitted, dt is set to 1/200.
type	Character value, type of filter, one out of "LP" (low pass), "HP" (high pass), "BP" (band pass) and "BR" (band rejection). If omitted, the type is interpreted from f. If f is of length two, type is set to "BP". If f is of length one, type is set to "HP".
shape	Character value, one out of "butter" (Butterworth), default is "butter".
order	Numeric value, order of the filter, default is 2. Only needed if data is no eseis object.
p	Numeric value, fraction of the signal to be tapered. If omitted, no tapering will be done.
zero	Logical value, option to run filter in zero phase shift mode. Note that this will triple the length of the signal vector during calculation. Default is FALSE.
lazy	Logical value, option to pre- and post-process data, including detrending, demeaning and tapering (p = 0.02). Default if FALSE.

Value

Numeric vector or list of vectors, filtered signal vector.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## filter data set by bandpass filter between 1 and 90 Hz
rockfall_bp <- signal_filter(data = rockfall_eseis,
                             f = c(1, 90))
```

```
## taper signal to account for edge effects
rockfall_bp <- signal_taper(data = rockfall_bp,
                            n = 2000)

## plot filtered signal
plot_signal(data = rockfall_bp)

## compare time domain versus frequency domain filtering
rockfall_td <- signal_filter(data = rockfall_eseis,
                             f = c(10, 40),
                             fft = FALSE)

rockfall_td_sp <- signal_spectrum(data = rockfall_td)

rockfall_fd <- signal_filter(data = rockfall_eseis,
                             f = c(10, 40),
                             fft = TRUE)

rockfall_fd_sp <- signal_spectrum(data = rockfall_fd)

plot_spectrum(data = rockfall_td_sp)
plot_spectrum(data = rockfall_fd_sp)
```

`signal_hilbert` *Calculate Hilbert transform.*

Description

The function calculates the Hilbert transform of the input signal vector.

Usage

```
signal_hilbert(data)
```

Arguments

`data`	`eseis` object, `numeric` vector or list of objects, data set to be processed.

Value

`Numeric` vector or list of vectors, Hilbert transform.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## calculate hilbert transform
rockfall_h <- signal_hilbert(data = rockfall_eseis)
```

---

signal_hvratio *Calculate h-v-ratio of seismic components*

Description

This function uses three components of a seismic signal, evaluates their spectra and builds the ratio of horizontal to vertical power. For details see http://www.geopsy.org/documentation/geopsy/hv.html.

Usage

```
signal_hvratio(
  data,
  dt,
  log = FALSE,
  method = "periodogram",
  kernel,
  order = "xyz"
)
```

Arguments

data	List, data frame or matrix, seismic componenents to be processed. If data is a matrix, the components must be organised as columns. Also, data can be a list of eseis objects.
dt	Numeric value, sampling period.
log	Logical value, unit of spectral power. If set to TRUE power will be used in dB, if set to FALSE, power is used in amplitude squared. Default is FALSE.
method	Character value, method for calculating the spectra. One out of "periodogram" , "autoregressive" and "multitaper", default is "periodogram".
kernel	Numeric value, window size (number of samples) of the moving window used for smoothing the spectra. By default no smoothing is performed.
order	Character value, order of the seismic components. Describtion must contain the letters "x","y" and "z" in the order according to the input data set. Default is "xyz" (EW-SN-vertical).

Details

The spectra should be smoothed. This can either be done directly during their calculation or before the calculation of the ratio. For the former case set `method = "autoregressive"`. For the latter case provide a value for `"kernel"`, which is the smoothing window size. Smoothing is performed with the logarithms of the spectral power data, using `caTools::runmean()` with the `endrule = "NA"`. After smoothing the data is re-linearised.

Value

A `data frame` with the h-v-frequency ratio.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(earthquake)

## ATTENTION, THIS EXAMPLE DATA SET IS FAR FROM IDEAL FOR THIS PURPOSE

## detrend data
s <- signal_detrend(data = s)

## calculate h-v-ratio, will be very rugged
hv <- signal_hvratio(data = s,
                     dt = 1 / 200)
plot(hv$ratio,
     type = "l")

## calculate h-v-ratio using the autogressive spectrum method
hv <- signal_hvratio(data = s,
                     dt = 1 / 200,
                     method = "autoregressive")
plot(hv, type = "l")

## calculate h-v-ratio with a smoothing window equivalent to dt
hv <- signal_hvratio(data = s,
                     dt = 1 / 200,
                     kernel = 200)
plot(hv, type = "l")
```

`signal_integrate` *Integrate a seismic signal*

Description

The function integrates a signal vector to convert values from velocity to displacement. Two methods are available

Usage

```
signal_integrate(data, dt, method = "fft", waterlevel = 10^-6)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
dt	Numeric scalar, sampling rate.
method	Character scalar, method used for integration. One out of "fft" (convert in the frequency domain) and "trapezoid" (integrate using the trapezoidal rule). Default is "fft".
waterlevel	Numeric scalar, waterlevel value for frequency division, default is 10^-6. Only used when method = "fft".

Value

Numeric vector or list of vectors, integrated signal.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## deconvolve signal
rockfall_decon <- signal_deconvolve(data = rockfall_eseis)

## integrate signal
rockfall_int <- signal_integrate(data = rockfall_decon)

## Note that usually the signal should be filtered prior to integration.
```

signal_interpolate *Interpolate a signal vector*

Description

The signal vector data is interpolated by an integer factor n. If an eseis object is provided, the meta data is updated. The function is a wrapper for the function interp of the package signal. Note that interpolation does not create new meaningful information but rather artefacts above the initial frequency range.

Usage

```
signal_interpolate(data, n, l = 4, ...)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
n	Numeric value, number of samples to be interpolated by. Must be an integer value greater than 1. Default is 2.
l	Character value, FIR filter length. For details see documentation of signal::decimate. Default is 4.
...	further arguments passed to signal::interp. See details.

Details

The function calls the function signal::interp and wraps the output into the eseis object structure. The ...-argument may contain the passed argument Wc (FIR filter cutoff frequency). Note that by convention the argument n of eseis::signal_interpolate does not equal the argument n of signal::interp. Rather this is the argument l that is passed as n.

Value

Interpolated data set.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## detrend data set
s <- signal_detrend(data = rockfall_eseis)

## interpolate by factor 2
```

```
s_int = signal_interpolate(data = s, n = 2)

## calculate and plot spectrogram
p_int = signal_spectrogram(data = s_int)
plot(p_int)
```

---

signal_kurtosis *Calculate signal kurtosis*

Description

This function calculates the running kurtosis of a seismic signal and returns that characteristic function.

Usage

```
signal_kurtosis(data, window = 200)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
window	Numeric value, size of the running window, in number of samples

Value

Numeric running kurtosis of the input signal.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## calculate kurtosis
rockfall_kurtosis <- signal_kurtosis(data = rockfall_eseis,
                                     window = 200)
```

signal_merge *Merge several signal streams into one*

Description

This function merges two or more single signals into one common. Only `eseis` objects are supported. Gaps will be filled with `NA` values unless the argument `fill = TRUE`. The resulting data length will correspond to the combined length of the input data. The sampling frequency must be the same for all input data sets. The meta data of the first stream will be used for the output data.

Usage

```
signal_merge(..., fill = FALSE)
```

Arguments

`...`	`eseis` objects that will be merged into one output `eseis` object. Can also be one list that contains `eseis` objects.
`fill`	`Logical` value, option to fill data gaps between merged streams. Default is `FALSE`.

Value

`eseis` object with merged input data sets

Author(s)

Michael Dietze

Examples

```
## load rockfall data set
data("rockfall")
s_1 <- rockfall_eseis

## duplicate data set and shift start time (incl. gap)
s_2 <- s_1
s_2$meta$starttime <- s_2$meta$starttime + 500

## merge data sets
s_merged <- signal_merge(s_1, s_2)

## plot merged data set
plot(s_merged)

## merge and fill gap
s_merged_filled <- signal_merge(s_1, s_2, fill = TRUE)

## plot merged data set
```

```
plot(s_merged_filled)
```

`signal_motion` *Calculate particle motion parameters*

Description

The function calculates from a data set of three seismic components of the same signal the following particle motion paramters using a moving window approach: horizontal-vertical eigenvalue ratio, azimuth and inclination.

Usage

```
signal_motion(data, time, dt, window, step, order = "xyz")
```

Arguments

`data`	`List` of eseis objects or `matrix`, seismic components to be processed. If `data` is a matrix, the components must be organised as columns.
`time`	`POSIXct` vector, time vector corresponding to the seismic signal components. If omitted, a synthetic time vector will be generated. If omitted, the sampling period (`dt`) must be provided or is taken from the first eseis object in the `data` list.
`dt`	`Numeric` value, sampling period. Only needed if `time` is omitted or if `data` is no `eseis` object.
`window`	`Numeric` value, time window length (given as number of samples) used to calculate the particle motion parameters. If value is even, it will be set to the next smaller odd value. If omitted, the window size is set to 1 percent of the time series length by default.
`step`	`Numeric` value, step size (given as number of samples), by which the window is shifted over the data set. If omitted, the step size is set to 50 percent of the window size by default.
`order`	`Character` value, order of the seismic components. Description must contain the letters `"x"`,`"y"` and `"z"` in the order according to the input data set. Default is `"xyz"` (EW-NS-vertical).

Details

The function code is loosely based on the function GAZI() from the package RSEIS with removal of unnecessary content and updated or rewritten loop functionality. In its current form, it also uses additional workflows from obspy.signal.polarisation, specifically following the Flinn (1965) approach. It windows the input signals, calculates the covariance matrix and performs a singular values decomposition to use the eigen values and vectors to determine the ratios corresponding to the output values rectilinearity, angularity, azimuth and incidence.

Note that the names of the output objects as well as the calculation routine have changed from the earlier version (V. 0.6.0), to increase computational efficiency and fix a bug in the windowing implementation.

Value

A List object with rectilinearity (`rectilinearity`), angularity (`polarity`), azimuth (`azimuth`) and incidence (`incidence`), as well as the corresponding time vector for these values.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(earthquake)

## filter seismic signals
s <- eseis::signal_filter(data = s,
                          dt = 1/200,
                          f = c(1, 3))

## convert list of signal vectors to column-wise matrix
s <- do.call(cbind, s)

## calculate particle motion parameters
pm <- signal_motion(data = s,
                    dt = 1 / 200,
                    window = 500,
                    step = 250)

## plot function output
par_original <- par(no.readonly = TRUE)
par(mfcol = c(2, 2))

plot(pm$time, pm$rect, type = "b")
plot(pm$time, pm$plan, type = "b")
plot(pm$time, pm$azimuth, type = "b")
plot(pm$time, pm$incidence, type = "b")

par(par_original)
```

---

signal_pad *Pad signal with zeros.*

Description

The function adds zeros to the input vector to reach a length, corresponding to the next higher power of two.

Usage

```
signal_pad(data)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.

Value

Numeric vector or list of vectors, signal vector with added zeros.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## pad with zeros
rockfall_pad <- signal_pad(data = rockfall_eseis)

## compare lengths
rockfall_eseis$meta$n
rockfall_pad$meta$n
```

---

signal_rotate *Rotate signal vectors using a 3-D rotation matrix.*

Description

The function rotates the horizontal components of the input data according to the specified angle.

Usage

```
signal_rotate(data, angle)
```

Arguments

data	List, data frame or matrix, seismic componenents to be processed. If data is a matrix, the components must be organised as rows. Also, data can be a list of eseis objects. If a matrix, this matrix must contain either two columns (x- and y-component) or three columns (x-, y-, and z-component), in exactly that order of the components.
angle	Numeric value, rotation angle in degrees.

Value

Numeric matrix, the 3-dimensional rotation matrix.

Author(s)

Michael Dietze

Examples

```
## create synthetic data set
data <- rbind(x = sin(seq(0, pi, length.out = 10)),
y = sin(seq(0, pi, length.out = 10)),
z = rep(0, 10))

## rotate the data set
x_rot <- signal_rotate(data = data,
                       angle = 15)

## plot the rotated data set
plot(x_rot[1,], col = 1, ylim = c(-2, 2))
points(x_rot[2,], col = 2)
points(x_rot[3,], col = 3)
```

---

signal_sign *Convert amplitude signal to one bit signed signal*

Description

This function assigns 1 for positive values and -1 for negative input values of a signal.

Usage

```
signal_sign(data)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.

Value

Numeric vector or list of vectors, sign-cut signal.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## sign-cut signal
rockfall_sign <- signal_sign(data = rockfall_eseis)
```

---

signal_snr *Calculate signal-to-noise-ratio.*

Description

The function calculates the signal-to-noise ratio of an input signal vector as the ratio between mean and max.

Usage

```
signal_snr(data, detrend = FALSE)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
detrend	Logical value, optionally detrend data set before calcualting snr.

Value

Numeric value, signal-to-noise ratio.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## calculate snr with detrend option off and on
snr <- signal_snr(data = rockfall_eseis)
print(snr$snr)

snr <- signal_snr(data = rockfall_eseis,
                  detrend = TRUE)
print(snr$snr)
```

signal_spectrogram *Calculate spectrograms (power spectral density estimates) from time series.*

Description

This function creates spectrograms from seismic signals. It supports the standard spectrogram approach and the Welch method.

Usage

```
signal_spectrogram(
  data,
  time,
  dt,
  Welch = FALSE,
  window,
  overlap = 0.5,
  window_sub,
  overlap_sub = 0.5,
  method = "periodogram",
  cpu = NULL,
  plot = FALSE,
  ...
)
```

Arguments

data	Numeric vector or list of vectors, seismic signal to be processed.
time	POSIX.ct vector with time values. If omitted, an artificial time vector will be created, based on dt. Only needed if data is no eseis object.
dt	Numeric value, sampling period. If omitted, either estimated from time or set to 0.01 s (i.e., f = 100 Hz). Only needed if data is no eseis object.
Welch	Logical value, option to use the Welch method for calculations.
window	Numeric value, time window length in seconds used to calculate individual spectra. Set to 1 percent of the time series length by default.
overlap	Numeric value, fraction of window overlap.
window_sub	Numeric value, length of the sub-window in seconds used to calculate spectra. Only relevant if Welch = TRUE. If omitted, the sub-window length is set to 10 percent of the main window length.
overlap_sub	Numeric value, fraction of sub-window overlap.
method	Character value, method to calculate the spectra. One out of "periodogram" and "autoregressive". Default is "periodogram".
cpu	Numeric value between 0 and 1, fraction of CPU cores to use. If omitted, only one CPU is used.

plot	Logical value, toggle plot output. Default is FALSE. For more customised plotting see plot_spectrogram.
...	Additional arguments passed to the function.

Details

Data containing NA values is replaced by zeros and set to NA in the output data set.

Value

List with spectrogram matrix, time and frequency vectors.

Author(s)

Michael Dietze

Examples

```
## load example data set
data("earthquake")

## calculate and plot PSD straight away
P <- signal_spectrogram(data = s$BHZ,
                                time = t,
                                dt = 1 / 200,
                                plot = TRUE)

## calculate and plot PSD with defined window sizes and the Welch method
P <- signal_spectrogram(data = s$BHZ,
                                time = t,
                                dt = 1 / 200,
                                window = 5,
                                overlap = 0.9,
                                window_sub = 3,
                                overlap_sub = 0.9,
                                Welch = TRUE,
                                plot = TRUE)
```

---

signal_spectrum *Calculate the spectrum of a time series*

Description

The power spectral density estimate of the time series is calculated using different approaches.

Usage

```
signal_spectrum(data, dt, method = "periodogram", n, res, log = FALSE, ...)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
dt	Numeric value, sampling period. If omitted, dt is set to 1/200. Only needed if data is no eseis object.
method	Character value, calculation method. One out of "periodogram" and "autoregressive". default is "periodogram".
n	Numeric value, optional number of samples in running window used for smoothing the spectrogram. Only applied if a number is provided. Smoothing is performed as running mean.
res	Numeric value, optional resolution of the spectrum, i.e. the number of power and frequency values. If omitted, the full resolution is returned. If used, a spline interpolation is performed.
log	Logical value, option to interpolate the spectrum with log spaced frequency values. Default is FALSE.
...	Additional arguments passed to the function.

Details

If the res option is used, the frequency and power vectors will be interpolated using a spline interpolator, using equally spaced frequency values. If desired, the additional option log = TRUE can be used to interpolate with log spaced frequency values.

Value

Data frame with frequency and power vector

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## calculate spectrum with standard setup
s <- signal_spectrum(data = rockfall_eseis)

## plot spectrum
plot_spectrum(data = s)
```

signal_stalta *Calculate the short-time-average to long time average ratio*

Description

This function calculates the short-time-average to long time average (STA-LTA) ratio of a seismic signal and returns that ratio time series.

Usage

```
signal_stalta(data, sta, lta, lazy = FALSE)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
sta	Numeric value, length of the short time average window size, in number of samples.
lta	Numeric value, length of the long time average window size, in number of samples.
lazy	Logical value, option to preprocess data, including demeaning, detrending and envelope calculation. Default is FALSE.

Value

Numeric vector or list of vectors, STA-LTA ratio signal.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## calculate STA-LTA ratio
rockfall_stalta <- signal_stalta(data = rockfall_eseis,
                                 sta = 0.5 * 200,
                                 lta = 10 * 200,
                                 lazy = TRUE)
```

signal_stats *Calculate signal statistics*

Description

This function calculates a set of statistics for the seismic signal submitted.

Usage

```
signal_stats(data, stats, range_f, res_psd = 1, dt, cut = TRUE)
```

Arguments

data	eseis object, data set to be processed.
stats	Character vector, keywords of statistics to be calculated. If omitted, all statistics will be calculated. Wrongly spelled keywords will be omitted without warning.
range_f	Numerical vector of length two, range of the frequency spectra used to calculate spectral properties. This is recommended to account for spurious or unwanted frequency pars, for example caused by ocean micro seism or high frequency effects.
res_psd	Numerical value, resolution of the spectrogram used to calculate statistics, in seconds. Default is 1 sec. The spectrogram will be calculated with 90 running window of 5 sec.
dt	Numeric value, sampling period. If omitted, dt is set to 1/200.
cut	Logical value, option to cut output vector to the required statistics, instead of returning the full length of statistics, filled with NA values where no statistic was calculated. Default is TRUE.

Details

Available statistics keywords are: 1. ‘"t_duration"‘ (Duration of the signal) 1. ‘"t_rise"‘ (Signal rise time, time from start to maximum amplitude) 1. ‘"t_fall"‘ (Signal fall time, tme from maximum amplitude to end) 1. ‘"t_risefall"‘ (Ratio of rise to fall time) 1. ‘"a_skewness"‘ (Skewness of the signal amplitude, see seewave::specprop) 1. ‘"a_kurtosis"‘ (Kurtosis of the signal amplitude, see seewave::specprop) 1. ‘"a1_kurtosis"‘ (Kurtosis of the filtered (0.1-1 Hz) signal amplitude, see seewave::specprop) 1. ‘"a2_kurtosis"‘ (Kurtosis of the filtered (1-3 Hz) signal amplitude, see seewave::specprop) 1. ‘"a3_kurtosis"‘ (Kurtosis of the filtered (3-10 Hz) signal amplitude, see seewave::specprop) 1. ‘"a4_kurtosis"‘ (Kurtosis of the filtered (10-20 Hz) signal amplitude, see seewave::specprop) 1. ‘"a5_kurtosis"‘ (Kurtosis of the filtered (20-50 Hz) signal amplitude, see seewave::specprop) 1. ‘"e_maxmean"‘ (Ratio of maximum and mean envelope value, see Hibert et al. (2017)) 1. ‘"e_maxmedian"‘ (Ratio of maximum and median envelope value, see Hibert et al. (2017)) 1. ‘"e_skewness"‘ (Skewness of the signal envelope, see seewave::specprop) 1. ‘"e_kurtosis"‘ (Kurtosis of the signal envelope, see seewave::specprop) 1. ‘"e1_logsum"‘ (Logarithm of the filtered (0.1-1 Hz) envelope sum, see Hibert et al. (2017)) 1. ‘"e2_logsum"‘ (Logarithm of the filtered (1-3 Hz) envelope sum, see Hibert et al. (2017))

1. ‘"e3_logsum"‘ (Logarithm of the filtered (3-10 Hz) envelope sum, see Hibert et al. (2017)) 1. ‘"e4_logsum"‘ (Logarithm of the filtered (10-20 Hz) envelope sum, see Hibert et al. (2017)) 1. ‘"e5_logsum"‘ (Logarithm of the filtered (20-50 Hz) envelope sum, see Hibert et al. (2017)) 1. ‘"e_rmsdecphaseline"‘ (RMS of envelope from linear decrease, see Hibert et al. (2017)) 1. ‘"c_peaks"‘ (Number of peaks (excursions above 75 1. ‘"c_energy1"‘ (Sum of the first third of the signal cross correlation function, see Hibert et al. (2017)) 1. ‘"c_energy2"‘ (Sum of the last two thirds of the signal cross correlation function, see Hibert et al. (2017)) 1. ‘"c_energy3"‘ (Ratio of c_energy1 and c_energy2, see Hibert et al. (2017)) 1. ‘"s_peaks"‘ (Number of peaks (excursions above 75 1. ‘"s_peakpower"‘ (Mean power of spectral peaks, see Hibert et al. (2017)) 1. ‘"s_mean"‘ (Mean spectral power, see Hibert et al. (2017)) 1. ‘"s_median"‘ (Median spectral power, see Hibert et al. (2017)) 1. ‘"s_max"‘ (Maximum spectral power, see Hibert et al. (2017)) 1. ‘"s_var"‘ (Variance of the spectral power, see Hibert et al. (2017)) 1. ‘"s_sd"‘ (Standard deviation of the spectral power, see `seewave::specprop`) 1. ‘"s_sem"‘ (Standard error of the mean of the spectral power, see `seewave::specprop`) 1. ‘"s_flatness"‘ (Spectral flatness, see `seewave::specprop`) 1. ‘"s_entropy"‘ (Spectral entropy, see `seewave::specprop`) 1. ‘"s_precision"‘ (Spectral precision, see `seewave::specprop`) 1. ‘"s1_energy"‘ (Energy of the filtered (0.1-1 Hz) spectrum, see Hibert et al. (2017)) 1. ‘"s2_energy"‘ (Energy of the filtered (1-3 Hz) spectrum, see Hibert et al. (2017)) 1. ‘"s3_energy"‘ (Energy of the filtered (3-10 Hz) spectrum, see Hibert et al. (2017)) 1. ‘"s4_energy"‘ (Energy of the filtered (10-20 Hz) spectrum, see Hibert et al. (2017)) 1. ‘"s5_energy"‘ (Energy of the filtered (20-30 Hz) spectrum, see Hibert et al. (2017)) 1. ‘"s_gamma1"‘ (Gamma 1, spectral centroid, see Hibert et al. (2017)) 1. ‘"s_gamma2"‘ (Gamma 2, spectral gyration radius, see Hibert et al. (2017)) 1. ‘"s_gamma3"‘ (Gamma 3, spectral centroid width, see Hibert et al. (2017)) 1. ‘"f_modal"‘ (Modal frequency, see `seewave::specprop`) 1. ‘"f_mean"‘ (Mean frequency (aka central frequency), see `seewave::specprop`) 1. ‘"f_median"‘ (Median frequency, see `seewave::specprop`) 1. ‘"f_q05"‘ (Quantile 0.05 of the spectrum, see `seewave::specprop`) 1. ‘"f_q25"‘ (Quantile 0.25 of the spectrum, see `seewave::specprop`) 1. ‘"f_q75"‘ (Quantile 0.75 of the spectrum, see `seewave::specprop`) 1. ‘"f_q95"‘ (Quantile 0.95 of the spectrum, see `seewave::specprop`) 1. ‘"f_iqr"‘ (Inter quartile range of the spectrum, see `seewave::specprop`) 1. ‘"f_centroid"‘ (Spectral centroid, see `seewave::specprop`) 1. ‘"p_kurtosismax"‘ (Kurtosis of the maximum spectral power over time, see Hibert et al. (2017)) 1. ‘"p_kurtosismedian"‘ (Kurtosis of the median spectral power over time, see Hibert et al. (2017)) 1. ‘"p_maxmean"‘ (Mean of the ratio of max to mean spectral power over time, see Hibert et al. (2017)) 1. ‘"p_maxmedian"‘ (Mean of the ratio of max to median spectral power over time, see Hibert et al. (2017)) 1. ‘"p_peaksmean"‘ (Number of peaks in normalised mean spectral power over time, see Hibert et al. (2017)) 1. ‘"p_peaksmedian"‘ (Number of peaks in normalised median spectral power over time, see Hibert et al. (2017)) 1. ‘"p_peaksmax"‘ (Number of peaks in normalised max spectral power over time, see Hibert et al. (2017)) 1. ‘"p_peaksmaxmean"‘ (Ratio of number of peaks in normalised max and mean spectral power over time, see Hibert et al. (2017)) 1. ‘"p_peaksmaxmedian"‘ (Ratio of number of peaks in normalised max and median spectral power over time, see Hibert et al. (2017)) 1. ‘"p_peaksfcentral"‘ (Number of peaks in spectral power at central frequency over time, see Hibert et al. (2017)) 1. ‘"p_diffmaxmean"‘ (Mean difference between max and mean power, see Hibert et al. (2017)) 1. ‘"p_diffmaxmedian"‘ (Mean difference between max and median power, see Hibert et al. (2017)) 1. ‘"p_diffquantile21"‘ (Mean difference between power quantiles 2 and 1, see Hibert et al. (2017)) 1. ‘"p_diffquantile32"‘ (Mean difference between power quantiles 3 and 2, see Hibert et al. (2017)) 1. ‘"p_diffquantile31"‘ (Mean difference between power quantiles 3 and 1, see Hibert et al. (2017))

References: - Hibert C, Provost F, Malet J-P, Maggi A, Stumpf A, Ferrazzini V. 2017. Automatic identification of rockfalls and volcano-tectonic earthquakes at the Piton de la Fournaise volcano

using a Random Forest algorithm. Journal of Volcanology and Geothermal Research 340, 130-142.

Value

`data frame` with calculated statsitics

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## clip data to event of interest
eq <- signal_clip(data = rockfall_eseis,
                  limits = as.POSIXct(c("2015-04-06 13:18:50",
                                        "2015-04-06 13:20:10"),
                                      tz = "UTC"))

## calculate full statistics
eq_stats <- signal_stats(data = eq)

## show names of statistics
names(eq_stats)

## calculate and show selected statistics, with truncated frequency range
eq_stats_sub <- signal_stats(data = eq,
                             stats = c("t_rise",
                                       "c_peaks",
                                       "f_centroid"),
                             range_f = c(1, 90))
print(eq_stats_sub)
```

`signal_sum` *Calculate signal vector sum.*

Description

The function calculates the vector sum of the input signals.

Usage

```
signal_sum(...)
```

Arguments

`...`	`Numeric` vectors or `eseis` objects, input signal, that must be of the same length.

Value

Numeric vector, signal vector sum.

Author(s)

Michael Dietze

Examples

```
## create random vectors
x <- runif(n = 1000, min = -1, max = 1)
y <- runif(n = 1000, min = -1, max = 1)
z <- runif(n = 1000, min = -1, max = 1)

## calculate vector sums
xyz <- signal_sum(x, y, z)
```

---

signal_taper *Taper a signal vector.*

Description

The function tapers a signal vector with a cosine bell taper, either of a given proportion or a discrete number of samples.

Usage

```
signal_taper(data, p = 0, n)
```

Arguments

data	eseis object, numeric vector or list of objects, data set to be processed.
p	Numeric value, proportion of the signal vector to be tapered. Alternative to n.
n	Numeric value, number of samples to be tapered at each end of the signal vector.

Value

Data frame, tapered signal vector.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## remove mean from data set
rockfall <- signal_demean(data = rockfall_eseis)

## create artefact at the beginning
rockfall_eseis$signal[1:100] <- runif(n = 100, min = -5000, max = 5000)

## taper signal
rockfall_taper <- signal_taper(data = rockfall, n = 1000)

## plot both data sets
plot_signal(data = rockfall_eseis)
plot_signal(rockfall_taper)
```

`signal_whiten` *Perform spectral whitening of a signal vector*

Description

The function normalises the input signal within a given frequency window. If a time series is provided, it is converted to the spectral domain, whitening is performed, and it is transformed back to the time domain.

Usage

```
signal_whiten(data, f, dt)
```

Arguments

`data`	`eseis` object, or `complex` vector, data set to be processed.
`f`	`Numeric` vector of length two, frequency window within which to normalise. If omitted, the entire bandwidth is normalised.
`dt`	`Numeric` value, sampling period. Only needed if the input object is not an `eseis` object

Value

`Numeric` vector or eseis object, whitened signal vector.

Author(s)

Michael Dietze

Examples

```
## load example data set
data("rockfall")

## whiten data set between 10 and 30 Hz
rockfall_2 <- signal_whiten(data = rockfall_eseis,
                            f = c(10, 30))

## plot whitened data set
plot(rockfall_2)
```

`spatial_amplitude` *Locate the source of a seismic event by modelling amplutide attenuation*

Description

The function fits a model of signal amplitude attenuation for all grid cells of the distance data sets and returns the residual sum as measure of the most likely source location of an event.

Usage

```
spatial_amplitude(
  data,
  coupling,
  d_map,
  aoi,
  v,
  q,
  f,
  a_0,
  normalise = TRUE,
  output = "variance",
  cpu
)
```

Arguments

`data`	`Numeric` matrix or `eseis` object, seismic signals to work with. Since the function will calculate the maxima of the data it is usually the envelopes of the data that should be used here.
`coupling`	`Numeric` vector, coupling efficiency factors for each seismic station. The best coupled station (or the one with the highest amplification) must receive `1`, the others must be scaled relatively to this one.
`d_map`	`List` object, distance maps for each station. Output of `spatial_distance`.

aoi	raster object that defines which pixels are used to locate the source. If omitted, the entire distance map extent is used. aoi and d_map objects must have the same extents, projections and pixel sizes. The aoi map must be of logical values.
v	Numeric value, mean velocity of seismic waves (m/s).
q	Numeric value, quality factor of the ground.
f	Numeric value, frequency for which to model the attenuation.
a_0	Logical value, start parameter of the source amplitude, if not provided, a best guess is made as 100 times the maximum amplitude value of the data set.
normalise	Logical value, option to normalise sum of residuals between 0 and 1. Default is TRUE.
output	Character value, type of metric the function returns. One out of "residuals" (sums of the squared model residuals) or "variance" (variance reduction, cf. Walter et al. (2017)). Default is "variance".
cpu	Numeric value, fraction of CPUs to use. If omitted, only one CPU will be used.

Value

A raster object with the location output metrics for each grid cell.

Author(s)

Michael Dietze

Examples

```
## Not run:

## create synthetic DEM
dem <- terra::rast(xmin = 0, xmax = 10000,
                   ymin= 0, ymax = 10000,
                   res = c(500, 500),
                   vals = rep(0, 400))

## define station coordinates
sta <- data.frame(x = c(1000, 9000, 5000),
                  y = c(1000, 1000, 9000),
                  ID = c("A", "B", "C"))

## create synthetic signal (source in towards lower left corner of the DEM)
s <- rbind(dnorm(x = 1:1000, mean = 500, sd = 50) * 100,
           dnorm(x = 1:1000, mean = 500, sd = 50) * 2,
           dnorm(x = 1:1000, mean = 500, sd = 50) * 1)

## plot DEM and stations
terra::plot(dem)
text(x = sta$x,
    y = sta$y,
    labels = sta$ID)
```

```
## calculate spatial distance maps and inter-station distances
D <- eseis::spatial_distance(stations = sta[,1:2],
                             dem = dem)

## locate signal
e <- eseis::spatial_amplitude(data = s,
                              d_map = D$maps,
                              v = 500,
                              q = 50,
                              f = 10)

## get most likely location coordinates (example contains two equal points)
e_max <- spatial_pmax(data = e)

## plot output
terra::plot(e)
points(e_max[1],
       e_max[2],
       pch = 20)
points(sta[,1:2])


## End(Not run)
```

`spatial_clip` *Clip values of spatial data.*

Description

The function replaces raster values based on different thresholds.

Usage

```
spatial_clip(data, quantile, replace = NA, normalise = TRUE)
```

Arguments

`data`	`SpatRaster` object, spatial data set to be processed.
`quantile`	`Numeric` value, quantile value below which raster values are clipped.
`replace`	`Numeric` value, replacement value, default is `NA`.
`normalise`	`Logical` value, optionally normalise values above threshold quantile between 0 and 1. Default is `TRUE`.

Value

`SpatRaster` object, data set with clipped values.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(volcano)

## convert matrix to raster object
volcano <- terra::rast(volcano)

## clip values to those > quantile 0.5
volcano_clip <- spatial_clip(data = volcano,
                             quantile = 0.5)

## plot clipped data set
terra::plot(volcano_clip)
```

---

spatial_convert *Convert coordinates between reference systems*

Description

Coordinates are converted between reference systems.

Usage

```
spatial_convert(data, from, to)
```

Arguments

data	Numeric vector of length two or data frame, x-, y-coordinates to be converted.
from	Character value, proj4 string of the input reference system.
to	Character value, proj4 string of the output reference system.

Value

Numeric data frame with converted coordinates.

Author(s)

Michael Dietze

Examples

```
## create lat lon coordinates
xy <- c(13, 55)

## define output coordinate systems
proj_in <- "+proj=longlat +datum=WGS84"
proj_out <- "+proj=utm +zone=32 +datum=WGS84"

## convert coordinate pair
spatial_convert(data = xy,
                from = proj_in,
                to = proj_out)

## define set of coordinates
xy <- data.frame(x = c(10, 11),
                 y = c(54, 55))

## convert set of coordinates
spatial_convert(data = xy,
                from = proj_in,
                to = proj_out)
```

---

spatial_crop *Crop extent of spatial data.*

Description

The function crops the spatial extent of raster objects or other spatial objects based on bounding box coordinates.

Usage

```
spatial_crop(data, bbox)
```

Arguments

data	raster object, spatial data set to be processed.
bbox	Numeric vector of length four, bounding box coordinates in the form c(xmin, xmax, ymin, ymax)

Value

spatial object, cropped to bounding box

Author(s)

Michael Dietze

Examples

```
## create example data set
x <- terra::rast(nrows = 100, ncols = 100,
                 xmin = 0, xmax = 10,
                 ymin = 0, ymax = 10)
terra::values(x) <- 1:10000

## create crop extent vector
bbox <- c(3, 7, 3, 7)

## crop spatial object
y <- spatial_crop(data = x,
                  bbox = bbox)

## plot both objects
terra::plot(x)
terra::plot(y, add = TRUE)
```

`spatial_distance` *Calculate topography-corrected distances for seismic waves.*

Description

The function calculates topography-corrected distances either between seismic stations or from seismic stations to pixels of an input raster.

Usage

```
spatial_distance(
  stations,
  dem,
  topography = TRUE,
  maps = TRUE,
  matrix = TRUE,
  aoi,
  verbose = FALSE
)
```

Arguments

`stations`	`Numeric` matrix of length two, x- and y-coordinates of the seismic stations to be processed (column-wise orgnaised).The coordinates must be in metric units, such as the UTM system and match with the reference system of the `dem`.
`dem`	`SpatRaster` object, the digital elevation model (DEM) to be processed. The DEM must be in metric units, such as the UTM system and match with the reference system of the coordinates of `stations`. See `terra` for supported types and how to read these to R.

topography	Logical scalar, option to enable topography correction, default is TRUE.
maps	Logical scalar, option to enable/disable calculation of distance maps. Default is TRUE.
matrix	Logical scalar, option to enable/disable calculation of interstation distances. Default is TRUE.
aoi	Numeric vector of length four, bounding coordinates of the area of interest to process, in the form c(x0, x1, y0, y1).
verbose	Logical value, option to show extended function information as the function is running. Default is FALSE.

Details

Topography correction is necessary because seismic waves can only travel on the direct path as long as they are within solid matter. When the direct path is through air, the wave can only travel along the surface of the landscape. The function accounts for this effect and returns the corrected travel distance data set.

Value

List object with distance maps (list of SpatRaster objects from terra package) and station distance matrix (data.frame).

Author(s)

Michael Dietze

Examples

```
## Not run:

data("volcano")
dem <- terra::rast(volcano)
dem <- dem * 10
terra::ext(dem) <- terra::ext(dem) * 10
terra::ext(dem) <-terra::ext(dem) + c(510, 510, 510, 510)

## define example stations
stations <- cbind(c(200, 700), c(220, 700))

## plot example data
terra::plot(dem)
points(stations[,1], stations[,2])

## calculate distance matrices and stations distances
D <- spatial_distance(stations = stations,
                      dem = dem)

D_map_1 <- terra::rast(crs = D$maps[[1]]$crs,
                       ext = D$maps[[1]]$ext,
                       res = D$maps[[1]]$res,
```

```
                      val = D$maps[[1]]$val)

## plot distance map
terra::plot(D_map_1)

## show station distance matrix
print(D$matrix)

## calculate with AOI and in verbose mode
D <- spatial_distance(stations = stations,
                      dem = dem,
                      verbose = TRUE,
                      aoi = c(0, 200, 0, 200))

## plot distance map for station 2
terra::plot(D$maps[[1]])


## End(Not run)
```

`spatial_migrate` *Migrate signals of a seismic event through a grid of locations.*

Description

The function performs signal migration in space in order to determine the location of a seismic signal.

Usage

```
spatial_migrate(
  data,
  d_stations,
  d_map,
  snr,
  v,
  dt,
  normalise = TRUE,
  verbose = FALSE
)
```

Arguments

`data`	`Numeric` matrix or `eseis` object, seismic signals to cross-correlate.
`d_stations`	`Numeric` matrix, inter-station distances. Output of `spatial_distance`.
`d_map`	`List` object, distance maps for each station. Output of `spatial_distance`.

snr	Numeric vector, optional signal-to-noise-ratios for each signal trace, used for normalisation. If omitted it is calculated from input signals.
v	Numeric value, mean velocity of seismic waves (m/s).
dt	Numeric value, sampling period.
normalise	Logical value, option to normalise stations correlations by signal-to-noise-ratios.
verbose	Logical value, option to show extended function information as the function is running. Default is FALSE.

Details

With the switch from the package raster to the package terra, the resulting distance maps can no longer be saved in lists as distance maps. Thus, the function re-defines the distance SpatRaster objects by a list of data on crs, extent, resolution and raster values. As a consequence, plotting the data requires turning them into a SpatRaster object, first (see examples).

Value

A SpatialGridDataFrame-object with Gaussian probability density function values for each grid cell.

Author(s)

Michael Dietze

Examples

```
## Not run:

## create synthetic DEM
dem <- terra::rast(nrows = 20, ncols = 20,
                   xmin = 0, xmax = 10000,
                   ymin= 0, ymax = 10000,
                   vals = rep(0, 400))

## define station coordinates
sta <- data.frame(x = c(1000, 9000, 5000),
                  y = c(1000, 1000, 9000),
                  ID = c("A", "B", "C"))

## create synthetic signal (source in the center of the DEM)
s <- rbind(dnorm(x = 1:1000, mean = 500, sd = 50),
           dnorm(x = 1:1000, mean = 500, sd = 50),
           dnorm(x = 1:1000, mean = 500, sd = 50))

## plot DEM and stations
terra::plot(dem)

text(x = sta$x,
     y = sta$y,
     labels = sta$ID)
```

```
## calculate spatial distance maps and inter-station distances
D <- spatial_distance(stations = sta[,1:2],
                              dem = dem)

## restore SpatRaster object for plotting purpose
D_map_1 <- terra::rast(crs = D$maps[[1]]$crs,
                       ext = D$maps[[1]]$ext,
                       res = D$maps[[1]]$res,
                       val = D$maps[[1]]$val)

## plot distance map
terra::plot(D_map_1)

## locate signal
e <- spatial_migrate(data = s,
                     d_stations = D$matrix,
                     d_map = D$maps,
                     v = 1000,
                     dt = 1/100)

## get most likely location coordinates
e_max <- spatial_pmax(data = e)

## plot location estimate, most likely location estimate and stations
terra::plot(e)
points(e_max[1],
       e_max[2],
       pch = 20)
points(sta[,1:2])


## End(Not run)
```

`spatial_parabola` *Locate signals of a seismic event by time difference parabola overlay*

Description

The function performs event location in space by finding the grid cell with minimum average travel time difference using the parabola approach. For further information see also Hibert et al. (2014) DOI: 10.1002/2013JF002970.

Usage

```
spatial_parabola(data, d_map, v, dt, plot, ...)
```

Arguments

data	Numeric matrix or eseis object, seismic signals to cross-correlate.
d_map	List object, distance maps for each station. Output of spatial_distance.
v	Numeric value or vector, apparent seismic wave velocity (m/s).
dt	Numeric value, sampling period.
plot	Character value, keyword defining if or which output is to be plotted. If omitted, no plot output is generated. If set to "parabola", a plot with all overlaid station pair parabolas is created. If set to "offset", a plot of the average time offset of each grid cell is created.
...	Additional arguments passed to the plot function.

Value

A terra raster with average travel time offsets for each grid cell, implying the most likely source location coinciding with the smallest offset value.

Author(s)

Michael Dietze, Clement Hibert (ITES Strasbourg)

Examples

```
## Not run:

## create synthetic DEM
dem <- terra::rast(nrows = 20, ncols = 20,
                   xmin = 0, xmax = 10000,
                   ymin= 0, ymax = 10000,
                   vals = rep(0, 400))

## define station coordinates
sta <- data.frame(x = c(1000, 9000, 5000),
                  y = c(1000, 1000, 9000),
                  ID = c("A", "B", "C"))

## create synthetic signal (source in the center of the DEM)
s <- rbind(dnorm(x = 1:1000, mean = 400, sd = 50),
           dnorm(x = 1:1000, mean = 400, sd = 50),
           dnorm(x = 1:1000, mean = 800, sd = 50))

## plot DEM and stations
terra::plot(dem)

text(x = sta$x,
     y = sta$y,
     labels = sta$ID)

## calculate spatial distance maps and inter-station distances
D <- spatial_distance(stations = sta[,1:2],
```

```
                              dem = dem)

## restore SpatRaster object for plotting purpose
D_map_1 <- terra::rast(crs = D$maps[[1]]$crs,
                       ext = D$maps[[1]]$ext,
                       res = D$maps[[1]]$res,
                       val = D$maps[[1]]$val)

## plot distance map
terra::plot(D_map_1)

## locate signal
e <- spatial_parabola(data = s,
                      d_map = D$maps,
                      v = 1000,
                      dt = 1/100,
                      plot = "parabola",
                      zlim = c(0, 2))

## End(Not run)
```

`spatial_pmax` *Get most likely source location*

Description

The function identifies the location of a seismic source with the heighest likelihood (P_max).

Usage

```
spatial_pmax(data)
```

Arguments

`data`	`SpatRaster` object, spatial data set with source location estimates.

Value

`data.frame`, coordinates (x and y) of the most likely s ource location(s).

Author(s)

Michael Dietze

Examples

```
## create example source location likelihood raster
x <- terra::rast(nrows = 100, ncols = 100,
                 xmin = 0, xmax = 10,
                 ymin = 0, ymax = 10)
terra::values(x) <- runif(n = 100)

## identify location of highest likelihood
p_max <- spatial_pmax(data = x)

## show result
print(p_max)
```

---

spatial_track *Track a spatially mobile seismic source*

Description

This function allows tracking a spatially mobile seismic source and thereby estimating the source amplitude and the model's variance reduction as a measure of quality or robustness of the time-resolved estimates.

Usage

```
spatial_track(
  data,
  coupling,
  window,
  overlap = 0,
  d_map,
  aoi,
  v,
  q,
  f,
  k,
  qt = 1,
  dt,
  model = "SurfSpreadAtten",
  cpu,
  verbose = FALSE,
  plot = FALSE
)
```

Arguments

data	Numeric matrix or eseis object, seismic signals used for source tracking. Note that the function will start tracking within a smaller time window, narrows be

	the maximum signal arrival time differences as defined by the maximum inter station distance and the seismic velocity. The signals should be the envelopes of waveforms.
coupling	Numeric vector, coupling efficiency factors for each seismic station. The best coupled station (or the one with the highest amplification) must receive 1, the others must be scaled relatively to this one.Numeric vector, coupling efficiency factors for each seismic station. The best coupled station (or the one with the highest amplification) must receive 1, the others must be scaled relatively to this one.
window	Numeric value, time window for which the source is tracked. If omitted, ten time steps are generated.
overlap	Numeric value between 0 and 1, fraction of overlap of time windows used for source tracking. Default is 0.
d_map	List object, distance maps for each station (i.e., SpatialGridDataFrame objects). Output of spatial_distance.
aoi	Raster object (optional) that defines which pixels are used to locate the source. If omitted, the entire distance map extent is used. aoi and d_map objects must have the same extents, projections and pixel sizes. The aoi map must be of logical values.
v	Numeric value, mean velocity of seismic waves (m/s).
q	Numeric value, quality factor of the ground.
f	Numeric value, frequency for which to model the attenuation.
k	Numeric value, fraction of surface wave contribution to signals. Only relevant for models that include mixture of surface and body waves (see model_amplitude).
qt	Numeric value, quantile threshold that defines acceptable location estimates. Default is 1 (only single best estimate is kept).
dt	Numeric value, sampling frequency. Only required if input signals are no eseis objects.
model	Character value,
cpu	Numeric value, fraction of CPUs to use for parallel processing. If omitted, one CPU is used
verbose	Logical value, optional screen output of processing progress. Default is FALSE.
plot	Logical value, enable graphical output of key results. Default is FALSE.

Details

The method is based on ideas published by Burtin et al. (2016), Walter et al. 82017) and Perez-Guillen et al. (2019) and implemented in the R package eseis by Dietze (2018). It is related to the function spatial_amplitude, which can be used to locate spatially stable seismic sources by the same technique, and it resuires prepared input data as delivered by the function spatial_distance.

The input data (data) should ideally be a list of eseis objects (alternatively a matrix with seismic signal traces) containing the envelopes of the seismic event to track (i.e., describe by its location and amplitude as a function of propagation time). The temporal resolution of the track is defined by the arguments window and overlap (as a fraction between 0 and 1). The approach is based on fitting

known amplitude-distance functions (for an overview of available functions see `model_amplitude`) to the envelope time snippets for each pixel of a grid, which provides the distance from a pixel to each seismic station, i.e., the distance map set `d_map`. To avoid fitting each of the pixels of the distance map, one can provide an area of interest, AOI (`aoi`), which has the same extent and resolution as the distance map set and pixel values are either `TRUE` or `FALSE`. Depending on which amplitude-distance function is chosen, further arguments need to be provided (ground quality factor `q`, center frequency of the signal `f`). The apparent seismic wave velocity `v` is required regardless, either as fit model parameter or to correct the amplitude time snippets for the travel time delay from the source to the respective pixel of the distance map set. The output of the function can be provided with uncertainty estimates on all output values. The uncertainty is based on the size of accepted location estimates per time step, as defined by the variance reduction quantile threshold `qt` (i.e., all locations above this quantile will be assumed to be valid location estimates, whose parameters will be used to estimate the uncertainty). Note that usually, `qt` should be set to around 0.99, a value that depends on the number of pixels in the distance map set and that affects the location uncertainty, which in many cases is about 10 Note however, that this value is purely arbitrary and should be based on field-based control data. It is possible to run the function in a multi-CPU mode, to speed up computational time, using the argument `cpu`. Also, the function can generate generic plot output of the results, a panel of three plots: source trajectory, source amplitude and variance reduction.

Note that depending on the resolution of the distance map set, number of included seismic stations, and number of time windows, the function can take significant processing time. 50 time steps for 5 stations and 5000 pixels per distance map requires about 10 minutes time on a normal grade computer using a single CPU.

Value

A `List` object with summarising statistics of the fits.

References

Burtin, A., Hovius, N., and Turowski, J. M.: Seismic monitoring of torrential and fluvial processes, Earth Surf. Dynam., 4, 285–307, https://doi.org/10.5194/esurf-4-285-2016, 2016.

Dietze, M.: The R package 'eseis' – a software toolbox for environmental seismology, Earth Surf. Dynam., 6, 669–686, https://doi.org/10.5194/esurf-6-669-2018, 2018.

Perez-Guillen, C., Tsunematsu, K., Nishimura, K., and Issler, D.: Seismic location and tracking of snow avalanches and slush flows on Mt. Fuji, Japan, Earth Surf. Dynam., 7, 989–1007, https://doi.org/10.5194/esurf-7-989-2019, 2019.

Walter, F., Burtin, A., McArdell, B. W., Hovius, N., Weder, B., and Turowski, J. M.: Testing seismic amplitude source location for fast debris-flow detection at Illgraben, Switzerland, Nat. Hazards Earth Syst. Sci., 17, 939–955, https://doi.org/10.5194/nhess-17-939-2017, 2017.

Examples

```
## Not run:

x <- spatial_track(data = data,
                   window = 5,
                   overlap = 0.5,
                   d_map = D$maps,
```

```
                    aoi = aoi,
                    v = 800,
                    q = 40,
                    f = 12,
                    qt = 0.99)


## End(Not run)
```

`time_aggregate` *Aggregate a time series*

Description

The time series `x` is aggregated by an integer factor `n`.

Usage

```
time_aggregate(data, n = 2)
```

Arguments

`data`	`POSIXct` vector, time to be processed.
`n`	`Numeric` value, number of samples to be aggregated to one new data value. Must be an integer value greater than 1. Default is 2.

Value

`POSIXct` vector, aggregated data.

Author(s)

Michael Dietze

Examples

```
## load example data set
data(rockfall)

## aggregate time series
rockfall_t_agg <- time_aggregate(data = rockfall_t,
                           n = 2)

## compare results
range(rockfall_t)
diff(rockfall_t)

range(rockfall_t_agg)
diff(rockfall_t_agg)
```

time_clip *Clip time vector.*

Description

The function clips a time vector based on provided limits.

Usage

```
time_clip(time, limits)
```

Arguments

time	POSIXct vector, time vector.
limits	POSIXct vector of length two, time limits for clipping.

Value

POSIXct vector, clipped time vector.

Author(s)

Michael Dietze

Examples

```
## load example data
data(rockfall)

## define limits to clip to
limits <- c(min(rockfall_t) + 10,
            max(rockfall_t) - 10)

## clip data set
rockfall_t_clip <- time_clip(time = rockfall_t,
                             limits = limits)

## compare time ranges
range(rockfall_t)
range(rockfall_t_clip)
```

time_convert *Convert Julian Day to Date and vice versa*

Description

The function converts a Julian Day value to a date, to POSIXct if a year is provided, otherwise to POSIXlt.

Usage

```
time_convert(input, output, timezone = "UTC", year)
```

Arguments

input	Numeric vector, input time Supported formats are YYYY-MM-DD, JD and POSIXct.
output	Numeric vector, output time. Supported formats are YYYY-MM-DD, JD and POSIXct.
timezone	Character vector, time zone of the output date. Default is "UTC".
year	Character vector, year of the date. Only used when input is JD. If omitted, the current year is used.

Value

Numeric vector,

Author(s)

Michael Dietze

Examples

```
## convert Julian Day 18 to POSIXct
time_convert(input = 18, output = "POSIXct")

## convert Julian Day 18 to yyyy-mm-dd
time_convert(input = 18, output = "yyyy-mm-dd")

## convert yyyy-mm-dd to Julian Day
time_convert(input = "2016-01-18", output = "JD")

## convert a vector of Julian Days to yyyy-mm-dd
time_convert(input = 18:21, output = "yyyy-mm-dd")
```

time_jd *Convert time string to Julian Day*

Description

This function converts a POSIXct-like date into the corresponding Julian Day number and returns it as string format.

Usage

```
time_jd(time)
```

Arguments

time	Character value, date to convert to Julian Day. If no POSIXct format is provided, the function will try to convert the input data to POSIXct, assuming the time zone is UTC.

Details

There is also a more powerful function to convert between different time formats, see time_convert for details.

Value

Character value, Julian Day corresponding to the input date.

Author(s)

Michael Dietze

Examples

```
time_jd(time = "2020-05-01")
```

write_mseed *Write seismic traces as mseed file to disk.*

Description

This function converts seismic traces to mseed files and writes them to disk. It makes use of the Python library 'ObsPy'. Thus, this software must be installed, to make use of this function.

Usage

```
write_mseed(data, file, time, component, station, location, network, dt)
```

Arguments

data	eseis object or numeric vector, data set to be processed. Most other arguments can be omitted if data is an eseis object.
file	Character scalar, mseed file name with extension.
time	POSIXct vector, time vector corresponding to the seismic trace. Alternatively, the start time stamp can be provided as POSIXct value and a value for dt must be given.
component	Character value, component ID, optional.
station	Character value, station ID, optional.
location	Character vector of length four, station location data (latitude, longitude, elevation, depth), optional.
network	Character value, network ID, optional.
dt	Numeric value, sampling period. Only needed if no time vector is provided.

Details

The ObsPy Python library can be installed following the information provided here: "https://github.com/obspy/obspy/w

Since the ObsPy functionality through R is not able to interpret path definitions using the tilde symbol, e.g. "~/Downloads", this Linux type definition must be avoided.

Value

A binary file written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:
## load example data
data("rockfall")

## write as mseed file
write_mseed(data = rockfall_eseis, file = "rockfall.mseed")


## End(Not run)
```

write_report *Create a HTML report for (RLum) objects*

Description

This function creates a HTML report for a given eseis object, listing its complete processing history. The report serves both as a convenient way of browsing through objects and as a proper approach to documenting and saving scientific data and workflows.

Usage

```
write_report(object, file, title = "eseis report", browser = FALSE, css)
```

Arguments

object	eseis object to be reported on
file	Character value, name of the output file (without extension)
title	Character value, title of the report
browser	Logical value, optionally open the HTML file in the default web browser after it has been rendered.
css	Character value, path to a CSS file to change the default styling of the HTML document.

Details

The function heavily lends ideas from the function report_RLum() written by Christoph Burow, which is contained in the package Luminescence. This function here is a truncated, tailored version with minimised availabilities.

Value

HTML and .Rds file.

Author(s)

Michael Dietze

Examples

```
## Not run:
## load example data set
data(rockfall)

## make report for rockfall object
write_report(object = rockfall_eseis,
             browser = TRUE)
```

```
## End(Not run)
```

`write_sac` *Write seismic traces as sac file to disk.*

Description

This function converts seismic traces to sac files and writes them to disk.

Usage

```
write_sac(
  data,
  file,
  time,
  component,
  unit,
  station,
  location,
  network,
  dt,
  autoname = FALSE,
  parameters,
  biglong = FALSE
)
```

Arguments

`data`	`eseis` object or `numeric` vector, data set to be processed. Most other arguments can be omitted if `data` is an `eseis` object.
`file`	`Character` scalar, sac file name with extension.
`time`	`POSIXct` vector, time vector corresponding to the seismic trace. Alternatively, the start time stamp can be provided as `POSIXct` value and a value for `dt` must be given.
`component`	`Character` value, component ID, optional.
`unit`	`Character` value, unit of the signal, optional. One out of `"unknown"`, `"displacement"`, `"velocity"`, `"volts"`, `"acceleration"`. Default is `"unknown"`.
`station`	`Character` value, station ID, optinal.
`location`	`Character` vector of length four, station location data (latitude, longitude, elevation, depth), optional.
`network`	`Character` value, network ID, optional.
`dt`	`Numeric` value, sampling period. Only needed if no time vector is provided.
`autoname`	`Logical` value, option to let the function generate the file name automatically. Default is `FALSE`.

parameters	Data frame sac parameter list, as obtained from list_sacparameters. Allows user-specific modifications. If this data frame is provided, it overrides all other arguments.
biglong	Logical value, biglong option, default is FALSE

Details

For description of the sac file format see https://ds.iris.edu/files/sac-manual/manual/file_format.html. Currently the following parameters are not supported when writing the sac file: LAT, LON, ELEVATION, NETWORK.

Value

A binary file written to disk.

Author(s)

Michael Dietze

Examples

```
## Not run:
## load example data
data("rockfall")

## write as sac file
write_sac(data = rockfall_eseis)


## End(Not run)
```

Index