Package ‘digiRhythm’

December 19, 2024

Type Package

Title Analyzing Animal's Rhythmicity

Version 2.4

Author Hassan-Roland Nasser [aut, cre],
Marie Schneider [aut, ctb],
Joanna Stachowicz [aut, rev],
Christina Umstaetter [aut, ths]

Maintainer Hassan-Roland Nasser <hassan.nasser@me.com>

Description Analyze and visualize the rhythmic behavior of animals using the degree of functional coupling (See Scheibe (1999) <doi:10.1076/brhm.30.2.216.1420>), compute and visualize harmonic power, actograms, average activity and diurnality index.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 4.0.0)

Imports tidyr, readr (>= 2.0.1), magrittr, dplyr, xts, pracma, ggplot2, lubridate, stringr, zoo, crayon, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0), usethis

VignetteBuilder knitr

URL https://nasserdr.github.io/digiRhythm/

NeedsCompilation no

Repository CRAN

Date/Publication 2024-12-19 07:40:02 UTC

Contents

`actogram` *Plot a an single actogram over a period of time for a specific variable*

Description

Takes an activity dataset as input and plot and save an actogram of the specified activity column

Usage

```
actogram(df, activity, activity_alias, start, end, save = "actogram")
```

Arguments

`df`	The dataframe containing the activity data
`activity`	the name of activity
`activity_alias`	A string containing the name of the activity to be shown on the graph.
`start`	The start day (in "%Y-%m-%d" format).
`end`	The end day (in "%Y-%m-%d" format).
`save`	if NULL, the image is not saved. Otherwise, this parameter will be the name of the saved image. it should contain the path and name without the extension.

Value

A ggplot2 object that contains the actogram plot

Examples

```
data("df516b_2")
df <- df516b_2
activity <- names(df)[2]
start <- "2020-05-01" # year-month-day
end <- "2020-08-13" # year-month-day
activity_alias <- "Motion Index"
my_actogram <- actogram(df, activity, activity_alias, start, end,
  save = NULL
)
print(my_actogram)
```

`daily_activity_wrap_plot`
Plot daily average over a period of time for a specific variable.

Description

Takes an activity dataset as input and plot and save the daily average of the specified activity column

Usage

```
daily_activity_wrap_plot(
  df,
  activity,
  activity_alias,
  start,
  end,
  sampling_rate,
  ncols,
  save = "daily_wrap_plot"
)
```

Arguments

`df`	The dataframe containing the activity data
`activity`	the name of activity
`activity_alias`	A string containing the name of the activity to be shown on the graph.
`start`	The start day (in "%Y-%m-%d" format).
`end`	The end day (in "%Y-%m-%d" format).
`sampling_rate`	the sampling rate of the data.
`ncols`	the number of columns to spread the graphs on. be the name of the saved image. it should contain the path and name without the extension.
`save`	if NULL, the image is not saved. Otherwise, this parameter will

Value

A ggplot2 object that contains the daily average activity plot

Examples

```
data("df516b_2")
df <- df516b_2
activity <- names(df)[2]
activity_alias <- "Motion Index"
start <- "2020-05-01" # year-month-day
end <- "2020-05-07" # year-month-day
ncols <- 3
sampling_rate <- 30
my_dwp <- daily_activity_wrap_plot(
  df, activity, activity_alias, start, end, sampling_rate,
  ncols
)
```

`daily_average_activity` *Plot daily average over a period of time for a specific variable.*

Description

Takes an activity dataset as input and plot and save the daily average of the specified activity column

Usage

```
daily_average_activity(df, activity, activity_alias, start, end, save)
```

Arguments

`df`	The dataframe containing the activity data
`activity`	the name of activity
`activity_alias`	A string containing the name of the activity to be shown on the graph.
`start`	The start day (in "%Y-%m-%d" format).
`end`	The end day (in "%Y-%m-%d" format).
`save`	if NULL, the image is not saved. Otherwise, this parameter will be the name of the saved image. it should contain the path and name without the extension.

Value

None

Examples

```
data("df516b_2")
df <- df516b_2
activity <- names(df)[2]
start <- "2020-05-01" # year-month-day
end <- "2020-08-13" # year-month-day
activity_alias <- "Motion Index"
my_daa <- daily_average_activity(df, activity, activity_alias, start, end,
  save = NULL
)
print(my_daa)
```

`df516b_2` *df516b_2 Activity Data Sets*

Description

A dataset containing the Motion index and steps count of a cow. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df516b_2
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

Motion.Index The motion index of the cow during the time sample

Steps The number of steps during the time sample

Source

Agroscope Tanikon

`df603` *df603 Activity Data Sets*

Description

A dataset containing the x and y acceleration from an accelerometer installed on a cattle. There are missing days in this dataset. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df603
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

move_x The acceleration along the x axis

move_y The acceleration along the y axis

Source

Agroscope Posieux

`df625` *df625 Activity Data Sets*

Description

A dataset containing the x and y acceleration from an accelerometer installed on a cattle. There are missing days in this dataset. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df625
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

move_x The acceleration along the x axis

move_y The acceleration along the y axis

Source

Agroscope Posieux

`df678_2` *df678_2 Activity Data Sets*

Description

A dataset containing the Motion index and steps count of a cow. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df678_2
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

Motion.Index The motion index of the cow during the time sample

Steps The number of steps during the time sample

Source

Agroscope Tanikon

`df689b_3` *df689b_3 Activity Data Sets*

Description

A dataset containing the Motion index and steps count of a cow. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df689b_3
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

Motion.Index The motion index of the cow during the time sample

Steps The number of steps during the time sample

Source

Agroscope Tanikon

`df691b_1` *df691b_1 Activity Data Sets*

Description

A dataset containing the Motion index and steps count of a cow. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df691b_1
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

Motion.Index The motion index of the cow during the time sample

Steps The number of steps during the time sample

Source

Agroscope Tanikon

`df759a_3` *df759a_3 Activity Data Sets*

Description

A dataset containing the Motion index and steps count of a cow. The data set is sampled with 15 minutes samples. The data is as follows:

Usage

```
df759a_3
```

Format

A data frame of 3 columns

datetime a POSIX formatted datetime

Motion.Index The motion index of the cow during the time sample

Steps The number of steps during the time sample

Source

Agroscope Tanikon

`dfc` *Computes the Degree of Function coupling (DFC), Harmonic Part (HP) and Weekly Lomb-Scargle Spectrum (LSP Spec) for one variable in an activity dataset. The dataset should be digiRhythm friendly.*

Description

The computation of DFC/HP/LSP parameters is done using a rolling window. If the rolling window is 7 days, first, we compute the parameters of Days 1-7 then, of days 2-8 and so on). For each window of the 7 days, the function will compute the LSP spectrum to determine the power of each frequency. Using Baluev (2008), we will compute the significance of the amplitude of each frequency component and determine whether it is significant or not. Then, we will have all the significant frequencies, whose amplitudes' summation will be denominated as SUMSIG. Among all the available frequencies, some are harmonic (those that correspond to waves of period 24h, 12h, 24h/3, 24h/4, ...). As a result, we will have frequency components that are significant and harmonic, whose powers' summation is called SSH (sum significant and harmonic). The summation of all frequency components up to a frequency reflecting a 24h period is called SUMALL. Therefore, DFC and HP are computed as follows:

Usage

```
dfc(
  data,
  activity,
  sampling = 15,
  alpha = 0.05,
  harm_cutoff = 12,
  rolling_window = 7,
  plot = TRUE,
  plot_harmonic_part = TRUE,
  verbose = TRUE,
  plot_lsp = TRUE
)
```

Arguments

`data`	The activity data set.
`activity`	The name of the activity.
`sampling`	The sampling period of the data set in minutes. the Lomb Scargle Periodogram is computed.
`alpha`	The significance level that should be used to determine the significant frequency component.

`harm_cutoff`	the order of the highest harmonic needed to be considered. An integer equal to 1, 2, 3, ... Default is 12.
`rolling_window`	The rolling window used to compute the LSP. Default is 7 days.
`plot`	if TRUE, the DFC/HP plot will be shown.
`plot_harmonic_part`	if TRUE, it shows the harmonic part in the DFC plot
`verbose`	if TRUE, print weekly progress.
`plot_lsp`	if TRUE, the LSP of each sliding week will be plotted

Details

DFC <- SSH / SUMSIG HP <- SSH / SUMALL

Value

A list containing 2 dataframe. DFC dataframe that contain the results of a DFC computation and SPEC Dataframe that contains the result of spectrum computation. The DFC contains 3 columns: ** The date in format YYYY-MM-DD. ** The DFC computed using a @rolling_window days. ** The Harmonic Part (ratio). Data are supposed to sampled with a specific smpling rate. It should be the same sampling rate as in the given argument @sampling Missing days are not permitted. If you have data with half day, it should be removed.

Examples

```
sampling_period <- 15 * 60 # seconds
two_weeks <- 2 * 7 * 24 * 60 * 60 # seconds
amplitude_24h <- 5
amplitude_12h <- 3
noise_sd <- 2
time_seq <- seq(0, two_weeks, by = sampling_period)
time_posix <- as.POSIXct(time_seq, origin = "1970-01-01")
sine_24h <- amplitude_24h * sin(2 * pi * time_seq / (24 * 60 * 60))
sine_12h <- amplitude_12h * sin(2 * pi * time_seq / (12 * 60 * 60))
noise <- rnorm(length(time_seq), mean = 0, sd = noise_sd)
data <- sine_24h + sine_12h + noise
df <- data.frame(time = time_posix, value = data)
names(df) <- c("datetime", "activity")
print(str(df))
my_lsp <- dfc(df, "activity", alpha = 0.05, harm_cutoff = 12, plot = TRUE)
```

`df_act_info` *Outputs some information about the activity dataframe*

Description

Outputs some information about the activity dataframe

Usage

```
df_act_info(df)
```

Arguments

`df`	The dataframe containing the activity data

Value

No return value. Prints the head and tail as well as the starting and end date of a digiRhythm friendly dataframe.

`dgm_periodicity` *Returns the periodicity of a digiRhythm dataframe*

Description

Returns the periodicity of a digiRhythm dataframe

Usage

```
dgm_periodicity(data)
```

Arguments

`data`	a digiRhythm friendly dataframe

Value

returns a periodicity object of type xts.

Examples

```
data("df516b_2", package = "digiRhythm")
df <- df516b_2
dgm_periodicity(df)
```

diurnality — *Computes the diurnality index based on an activity dataframe*

Description

Computes the diurnality index based on an activity dataframe

Usage

```
diurnality(
  data,
  activity,
  day_time = c("06:30:00", "16:30:00"),
  night_time = c("18:00:00", "T05:00:00"),
  save = NULL
)
```

Arguments

`data`	a digiRhythm-friendly dataset
`activity`	The number of non-useful lines to skip (lines to header)
`day_time`	an array containing the start and end of the day period. Default: c("06:30:00", "16:30:00").
`night_time`	an array containing the start and end of the night period. Default: c("18:00:00", "T05:00:00").
`save`	if NULL, the image is not saved. Otherwise, this parameter will be the name of the saved image. it should contain the path and name without the extension.

Value

A ggplot2 object that contains the diurnality plot in addition to a dataframe with 2 col: date and diurnality index

Examples

```
data("df516b_2", package = "digiRhythm")
data <- df516b_2
data <- remove_activity_outliers(data)
activity <- names(data)[2]
d_index <- diurnality(data, activity)
```

diurnality_customTimes *Computes the diurnality index, using different start and end definitions for each day and night, based on an activity dataframe*

Description

Computes the diurnality index, using different start and end definitions for each day and night, based on an activity dataframe

Usage

```
diurnality_customTimes(data, activity, timedata, save = NULL)
```

Arguments

`data`	a digiRhythm-friendly dataset
`activity`	The number of non-useful lines to skip (lines to header)
`timedata`	a dataset, including 4 columns of POSIXct format, including date and time "day_start", "day_end", "night_start", "night_end"
`save`	if NULL, the image is not saved. Otherwise, this parameter will be the name of the saved image. it should contain the path and name without the extension.

Value

A ggplot2 object that contains the Sliding diurnality plot in addition to a dataframe with 2 col: date and sliding diurnality index

Examples

```
data("df516b_2", package = "digiRhythm")
data <- df516b_2
data <- remove_activity_outliers(data)
activity <- names(data)[2]
data("timedata", package = "digiRhythm")
timedata <- timedata
d_index <- diurnality_customTimes(data, activity, timedata)
```

`highest_possible_harm_cutoff`

*Function to calculate the smallest possible harmonic to consider given a sampling frequency. The minimum possible harmonic = 2 x the period of the maximum frequency according to the Shanon theorem. Example: if the sampling period is 15 min, the minimum possible treatable period is 30 minutes and that corresponds to the 48th harmonic (24 hours * 60 minutes / 48 = 30 minutes)*

Description

Function to calculate the smallest possible harmonic to consider given a sampling frequency. The minimum possible harmonic = 2 x the period of the maximum frequency according to the Shanon theorem. Example: if the sampling period is 15 min, the minimum possible treatable period is 30 minutes and that corresponds to the 48th harmonic (24 hours * 60 minutes / 48 = 30 minutes)

Usage

```
highest_possible_harm_cutoff(sampling_period_in_minutes)
```

Arguments

`sampling_period_in_minutes`	The sampling period of the acquired data in minutes

Value

Returns the smallest possible harmonic (of 24 hours) to consider given a sampling frequency.

`import_raw_activity_data`

Reads Raw Activity Data from csv files

Description

Reads Activity Data (data, time, activity(ies)) from a CSV file where we can skip some lines (usually representing the metadata) and select specific activities.

Usage

```
import_raw_activity_data(
  filename,
  skipLines = 0,
  act.cols.names = c("Date", "Time", "Motion Index", "Steps"),
  date_format = "%d.%m.%Y",
  time_format = "%H:%M:%S",
  sep = ",",
  original_tz = "CET",
  target_tz = "CET",
  sampling = 15,
  trim_first_day = TRUE,
  trim_middle_days = TRUE,
  trim_last_day = TRUE,
  verbose = FALSE
)
```

Arguments

`filename`	The file name (full or relative path with extension)
`skipLines`	The number of non-useful lines to skip (lines to header)
`act.cols.names`	A vector containing the names of columns to read (specific to the activity columns)
`date_format`	The POSIX format of the Date column (or first column)
`time_format`	The POSIX format of the Time column (or second column)
`sep`	The delimiter/separator between the columns
`original_tz`	The time zone with which the datetime are encoded
`target_tz`	The time zone with which you want to process the data. Setting this argument to 'GMT' will help you coping with daylight saving time where changes occur two time a year.
`sampling`	The sampling frequency in minutes (default 15 min)
`trim_first_day`	if True, removes the data from the first day if it contains less than 80% of the expected data points.
`trim_middle_days`	if True, removes the data from the MIDDLE days if they contain less than 80% of the expected data points.
`trim_last_day`	if True, removes the data from the last day if it contains less than 80% of the expected data points.
`verbose`	print out some useful information during the execution of the function

Details

This function prepare the data stored in a csv to be compatible with the digiRhythm package. You have the possibility to skip the first lines and choose which columns to read. You also have the possibility to sample the data. You can also choose whether to remove partial days (where no data over a full day is present) by trimming last, middle or last days. This function expects that the first and second columns are respectively date and time where the format should be mentioned.

file <- file.path('data', 'sample_data') colstoread <- c("Date", "Time", "Motion Index", 'Steps') #The colums that we are interested in data <- improt_raw_icetag_data(filename = file, skipLines = 7, act.cols.names = colstoread, sampling = 15, verbose = TRUE)

Value

A dataframe with datetime column and other activity columns, ready to be used with other functions in digirhythm

Examples

```
filename <- system.file("extdata", "sample_data.csv", package = "digiRhythm")
data <- import_raw_activity_data(
  filename,
  skipLines = 7,
  act.cols.names = c("Date", "Time", "Motion Index", "Steps"),
  sep = ",",
  original_tz = "CET",
  target_tz = "CET",
  date_format = "%d.%m.%Y",
  time_format = "%H:%M:%S",
  sampling = 15,
  trim_first_day = TRUE,
  trim_middle_days = TRUE,
  trim_last_day = TRUE,
  verbose = TRUE
)
print(head(data))
```

`is_dgm_friendly` *Informs if a dataset is digiRhythm Friendly*

Description

Takes an activity dataset as input and gives information about 1) If a dataset is digiRhythm friendly, i.e., the functions used can work with this dataset and 2) Tells what's wrong, if any.

Usage

```
is_dgm_friendly(data, verbose = FALSE)
```

Arguments

`data`	The dataframe containing the activity data
`verbose`	if TRUE, prints info about the dataset

Value

Boolean. If True, the dataframe is digirhythm friendly. If False, the dataframe is not digirhythm friendly.

Examples

```
data("df516b_2", package = "digiRhythm")
d <- df516b_2
is_dgm_friendly(data = d, verbose = TRUE)
```

`levopt` *Returns the level given the p-value computed with pbaluev (2008). Copied from the LOMB library.*

Description

Returns the level given the p-value computed with pbaluev (2008). Copied from the LOMB library.

Usage

```
levopt(Z, alpha, fmax, tm)
```

Arguments

`Z`	the power of the frequency
`alpha`	the significance level
`fmax`	the maximum frequency in the spectrum
`tm`	the time grid of the original time series

Value

Returns the level given the p-value computed with pbaluev (2008).

`lomb_scargle_periodogram` *Computes the Lomb Scargle Periodogram and returns the information needed for computing the DFC and HP. A plot visualizing the Harmonic Frequencies presence in the spectrum is possible. The function is inspired from the Lomb library in a great part, with modifications to fit the requirements of harmonic powers and computation of the DFC. This function is inspired by the lsp function from the lomb package and adapted to add different colors for harmonic and non harmonic frequencies in the signal. For more information about lomb::lsp, please refer to: https://cran.r-project.org/web/packages/lomb/*

Description

Computes the Lomb Scargle Periodogram and returns the information needed for computing the DFC and HP. A plot visualizing the Harmonic Frequencies presence in the spectrum is possible. The function is inspired from the Lomb library in a great part, with modifications to fit the requirements of harmonic powers and computation of the DFC. This function is inspired by the lsp function from the lomb package and adapted to add different colors for harmonic and non harmonic frequencies in the signal. For more information about lomb::lsp, please refer to: https://cran.r-project.org/web/packages/lomb/

Usage

```
lomb_scargle_periodogram(
  data,
  alpha = 0.01,
  harm_cutoff = 12,
  sampling = 15,
  plot = TRUE,
  extra_info_plot = TRUE
)
```

Arguments

`data`	a digiRhythm friendly dataframe of only two columns
`alpha`	the statistical significance for the false alarm
`harm_cutoff`	the order of the highest harmonic needed to be considered. An integer equal to 1, 2, 3, ... Default is 12.
`sampling`	the sampling period in minutes. default = 15 min.
`plot`	if TRUE, the LSP will be plotted
`extra_info_plot`	if True, extra information will be shown on the plot

Value

a list that contains a dataframe (detailed below), the significance level and significance (for the record). The dataframe contains the power the frequency, the frequency in HZ, the p values according to Baluev 2008, the period that corresponds to the frequency in seconds and in hours and finally, a boolean to tell whether the frequency is harmonic or not.

Examples

```
data("df516b_2", package = "digiRhythm")
data <- df516b_2[1:672, c(1, 2)]
lomb_scargle_periodogram(data, alpha = 0.01, harm_cutof = 12, plot = TRUE)
```

`pbaluev` *Returns p-value of a frequency peak according to pbaluev (2008) given Z, fmax and tm. Reused from the LOMB library (https://rdrr.io/cran/lomb/)*

Description

Returns p-value of a frequency peak according to pbaluev (2008) given Z, fmax and tm. Reused from the LOMB library (https://rdrr.io/cran/lomb/)

Usage

```
pbaluev(Z, fmax, tm)
```

Arguments

`Z`	the power of the frequency
`fmax`	the maximum frequency in the spectrum
`tm`	the time grid of the original time series

Value

an intermediate calculation step needed to compute the p-value according to pbaluev (2008).

`print_v` *Print if Verbose is true*

Description

Print if Verbose is true

Usage

```
print_v(string, verbose)
```

Arguments

`string`	The string to print
`verbose`	if TRUE, print the string

Value

No return value. Prints the string concatenated with a verbose if the latter is not NULL.

remove_activity_outliers *Remove outliers from the data*

Description

Remove outliers from the data

Usage

```
remove_activity_outliers(df)
```

Arguments

df	The dataframe containing the activity data

Value

return a dataframe where columns start the second one have undergone an outlier removal.

resample_dgm *Change the sampling of a digiRhythm friendly dataset*

Description

This function upsamples the data but does not downsample them. The new sampling should be a multiple of the current sampling period, and should be given in minutes.

Usage

```
resample_dgm(data, new_sampling)
```

Arguments

data	The dataframe containing the activity data
new_sampling	The new sampling (multiple of current sampling) in minutes

Value

A digiRhythm friendly dataset with the new sampling

Examples

```
data("df516b_2", package = "digiRhythm")
df <- df516b_2
df <- remove_activity_outliers(df)
new_sampling <- 30
new_dgm <- resample_dgm(df, new_sampling)
```

`timedata` *timedata Dataset of start and end of day and night*

Description

A dataset of start and endtime of the morning milking and evening milking on a dairy farm.

Usage

```
timedata
```

Format

A data frame of 4 columns

day_start a POSIX formatted datetime

day_end a POSIX formatted datetime

night_start a POSIX formatted datetime

night_end a POSIX formatted datetime

Source

Johann Heinrich von Thünen- Institute of Organic Farming

Index