Package 'phase'

July 23, 2025

Title Analyse Biological Time-Series Data

Version 1.2.9

Author Lakshman Abhilash [aut, cre] (ORCID:
<https://orcid.org/0000-0002-9933-8989>)

Maintainer Lakshman Abhilash <labhilash@gc.cuny.edu>

Description Compiles functions to trim, bin, visualise, and analyse activity/sleep time-series data collected from the Drosophila Activity Monitor (DAM) system (Trikinetics, USA). The following methods were used to compute periodograms - Chi-square periodogram: Sokolove and Bushell (1978) <doi:10.1016/0022-5193(78)90022-X>, Lomb-Scargle periodogram: Lomb (1976) <doi:10.1007/BF00648343>, Scargle (1982) <doi:10.1086/160554> and Ruf (1999) <doi:10.1076/brhm.30.2.178.1422>, and Autocorrelation: Eijzenbach et al. (1986) <doi:10.1111/j.1440-1681.1986.tb00943.x>. Identification of activity peaks is done after using a Savitzky-Golay filter (Savitzky and Golay (1964) <doi:10.1021/ac60214a047>) to smooth raw activity data. Three methods to estimate anticipation of activity are used based on the following papers - Slope method: Fernandez et al. (2020) <doi:10.1016/j.cub.2020.04.025>, Harrisingh method: Harrisingh et al. (2007) <doi:10.1523/JNEUROSCI.3680-07.2007>, and Stoleru method: Stoleru et al. (2004) <doi:10.1038/nature02926>. Rose plots and circular analysis are based on methods from - Batschelet (1981) <ISBN:0120810506> and Zar (2010) <ISBN:0321656865>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Imports circular, lubridate, plotly, pracma, signal, zeitgebr, zoo, behavr, wesanderson, shiny, shinythemes, shinydashboard, shinycssloaders, shinyFiles

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2023-03-09 00:20:02 UTC

Contents

`allActograms` *Generate actograms*

Description

This function generates a composite figure with actograms for all flies in a DAM scanned monitor file. Input for this function must be an output from the binData() function. The output of this function is a large plotly object. This function requires the packages "plotly" and "zoo".

Usage

```
allActograms(data, bin = 30, t.cycle = 24, color = rgb(0, 0, 0, 1))
```

Arguments

`data`	Input data file. The input for this function must be the output of the function binData(). See ??binData().
`bin`	Intervals in which data are saved (in minutes). This defaults to 30.
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.

color — Color of actograms in rgb format. The input for this must be a vector with 4 values, i.e., r,g,b,transparency. The values for r,g,b can only be between 0 and 1. 0,0,0 would be black and 1,1,1 would be white. Transparency values can also go from 0 to 1, 0 being fully transparent and 1 being fully opaque.

Value

A plotly htmlwidget with 32 actograms in a 4-by-8 array.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 3, bin = 1, t.cycle = 24)
bd <- binData(data = td)
actograms <- allActograms(data = bd)
```

---

allPeriodogramsAct *Periodogram analysis for activity data*

Description

This function generates a composite figure with periodogram plots for all flies in a DAM scanned monitor file. Input for this function must be an output from the trimData() function. The output of this function is a list with two components - (a) large plotly object with periodogram plots for all flies, and (b) a table which has channel wise information of significant period and adjusted power values, from the chosen time-series analysis method. This function requires the packages "plotly" and "zeitgebr".

Usage

```
allPeriodogramsAct(
  data,
  bin = 1,
  method = "ChiSquare",
  low.per = 16,
  high.per = 32,
  alpha = 0.05,
  time.res = 20
)
```

Arguments

data — Input data file. If the method for analysis is "ChiSquare", then the input for this function must be the output of the function trimData(). See ??trimData(). Otherwise, the input for this function must be the output of the function binData(). See ??binData().

bin	Intervals in which input data are sampled (in minutes). This defaults to 1. This must be changed appropriately depending on method of analysis and the input data set.
method	Choose the method for performing time-series analysis. Currently, three methods are implemented for analysis - "ChiSquare", "Autocorrelation", and "Lomb-Scargle". This defaults to "ChiSquare".
low.per	Choose the lowest period (in hours) for analysis. This defaults to 16.
high.per	Choose the highest period (in hours) for analysis. This defaults to 32.
alpha	Choose the significance level for periodogram analysis. This defaults to 0.05.
time.res	Resolution of periods (in minutes) to analyse while using the "ChiSquare" periodogram. For instance, if users wish to scan periods from low.per to high.per in the following manner: 16, 16.5, 17, 17.5, and so on, then time.res must be 30. This defaults to 20.

Value

A `list` with two items:

Plots A `plotly` `htmlwidget` with all periodograms in a 4-by-8 array.

Data A `matrix` `array` with 32 rows (one for each fly) and 2 columns (Period and Adjusted Power).

Examples

```
## Not run:
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 5, bin = 1, t.cycle = 24)
all.periodograms.act <- allPeriodogramsAct(data = td[,1:15])

## End(Not run)
```

---

allPeriodogramsSleep *Periodogram analysis for sleep data*

Description

This function generates a composite figure with periodogram plots for all flies in a DAM scanned monitor file. Input for this function must be an output from the sleepData() function. The output of this function is a list with two components - (a) large plotly object with periodogram plots for all flies, and (b) a table which has channel wise information of significant period and adjusted power values, from the chosen time-series analysis method. This function requires the packages "plotly" and "zeitgebr".

Usage

```
allPeriodogramsSleep(
  data,
  bin = 30,
  method = "ChiSquare",
  low.per = 16,
  high.per = 32,
  alpha = 0.05,
  time.res = 20
)
```

Arguments

data	Input data file. The input for this function must be an output from one of either sleepData(). See ??sleepData().
bin	Intervals in which input data is saved (in minutes). This defaults to 30.
method	Choose the method for performing time-series analysis. Currently, three methods are implemented for analysis - "ChiSquare", "Autocorrelation", and "Lomb-Scargle". This defaults to "ChiSquare".
low.per	Choose the lowest period (in hours) for analysis. This defaults to 16.
high.per	Choose the highest period (in hours) for analysis. This defaults to 32.
alpha	Choose the significance level for periodogram analysis. This defaults to 0.05.
time.res	Resolution of periods (in minutes) to analyse while using the ChiSquare periodogram. For instance, if users wish to scan periods from low.per to high.per in the following manner: 16, 16.5, 17, 17.5, and so on, then time.res must be 30. This defaults to 20.

Value

A `list` with two items:

Plots A `plotly` `htmlwidget` with all periodograms in a 4-by-8 array.

Data A `matrix` `array` with 32 rows (one for each fly) and 2 columns (Period and Adjusted Power).

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 5, bin = 1, t.cycle = 24)
sd <- sleepData(td)
all.periodograms.sleep <- allPeriodogramsSleep(data = sd[,1:6])
```

`allSomnograms` *Generate actograms for sleep data (Somnograms)*

Description

This function generates a composite figure with actograms (referred to as somnograms, here) for all flies in a DAM scanned monitor file. Input for this function must be an output from the trimData() function. The output of this function is a large plotly object. This function requires the packages "plotly" and "zoo". In a particular bin, sleep is calculated as the total minutes of inactivity equal to or greater than the defined threshold (sleep.def; typically, 5-minutes).

Usage

```
allSomnograms(
  data,
  sleep.def = c(5),
  bin = 30,
  t.cycle = 24,
  color = rgb(0, 0, 0, 1)
)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
`sleep.def`	Definition of sleep. Traditionally, a single bout of sleep is defined as any duration of inactivity that is equal to or greater than 5-minutes. However, sometimes it may be of interest to examine longer bouts of sleep or specific bout durations; sleep.def allows users to change the definition of sleep. The default input is a single value vector of value 5. If users wish to analyse sleep only between 5 to 20 mins, the input must be c(5,20).
`bin`	Intervals in which data are saved (in minutes). This defaults to 30. The value of bin cannot be lower than that of sleep.def.
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
`color`	Color of somnograms in rgb format. The input for this must be a vector with 4 values, i.e., r,g,b,transparency. The values for r,g,b can only be between 0 and 1. 0,0,0 would be black and 1,1,1 would be white. Transparency values can also go from 0 to 1, 0 being fully transparent and 1 being fully opaque.

Value

A `plotly` `htmlwidget` with 32 somnograms in a 4-by-8 array.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 3, bin = 1, t.cycle = 24)
somnograms <- allSomnograms(data = td[,1:15])
```

`anticipationAct` *Compute anticipation for activity data*

Description

This function calculates anticipation of activity in user-defined windows. The input for this function must be the output of the trimData() function. There are three choices for the estimation of anticipation. The "Slope" method simply finds best-fitting lines to activity data in the defined windows (by using linear regressions). The slope of this line, for each fly, is used as a measure of anticipation. The "Stoleru" method is based on the estimates used in the paper by Stoleru et al., 2004 (https://doi.org/10.1038/nature02926). The authors estimate progressive increase in activity before environmental transitions, scaled by the startle response (immediately post-transition). The "Harrisingh" method is based on the method used by Harrisingh et al., 2007 (10.1523/JNEUROSCI.3680-07.2007). In this method, the authors simply examine activity in a 3-hour interval before environmental transitions, scaled by activity in the 6-hour interval before the transitions. Only if method = "Slope", the output of this function is a list with three elements, i.e., plots of the regression line along with the activity in the morning window, plots of the regression line along with the activity in the evening window, and a table with the morning and evening anticipation values. In the other two cases, the output of this function is only a table with the morning and evening anticipation values. No plots will be generated.

Usage

```
anticipationAct(
  data,
  method = "Slope",
  t.cycle = 24,
  morn.win.start,
  eve.win.start,
  eve.win.end,
  rm.channels = c(),
  max.y.morn = "auto",
  max.y.eve = "auto"
)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
`method`	Choose the method for estimating anticipation. The three possible choices are, (i) "Slope", (ii) "Stoleru" and (iii) "Harrisingh". This defaults to "Slope". See Description for more details on these methods.

t.cycle	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
morn.win.start	Define the start of morning window in ZT. ZT00 is the onset of zeitgeber; in case of light/dark cycles, ZT00 is the time at which lights turn ON. The end of the morning window is considered as ZT00.
eve.win.start	Define the start of evening window in ZT.
eve.win.end	Define the end of evening window.
rm.channels	A vector of channels from a DAM monitor that must be discarded from analysis. If channels 1 to 5 must be removed, type in c(1:5). If channels 1 to 5 and 10 to 13 and 15 and 17 must be removed, type in c(1:5,10:13,15,17). Default is to include all individuals.
max.y.morn	Set the upper limit of the y-axis in the plot for morning window.
max.y.eve	Set the upper limit of the y-axis in the plot for evening window.

Value

If method = "Slope", a `list` with two items, else a `matrix` `array` with 32 rows (one for each fly) and 3 columns (Channel/Fly identity, Morning anticipation index and Evening anticipation index).

If method = "Slope":

Plot.morn A `plotly` `htmlwidget` with the anticipation estimates for the morning window.

Plot.eve A `plotly` `htmlwidget` with the anticipation estimates for the evening window.

Data A `matrix` `array` with 32 rows (one for each fly) and 3 columns (Channel/Fly identity, Morning anticipation index and Evening anticipation index).

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
anticip <- anticipationAct(data = td, method = "Stoleru", t.cycle = 24,
                           morn.win.start = 21,
                           eve.win.start = 9, eve.win.end = 12,
                           rm.channels = c())
```

---

binData *Bin activity data to desired intervals*

Description

Allows users to bin data sets into intervals of time different from the data collection interval. The input for this function must be the output of the trimData() function. The output of this function is a data frame. The first column of which stores Zeitgeber Time values (assuming that the start.time in the trimData() function was set at Zeitgeber Time 00). All subsequent columns have binned activity data for each fly.

Usage

```
binData(data, input.bin = 1, output.bin = 30, t.cycle = 24)
```

Arguments

data	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
input.bin	Define the bin size (in minutes) in the input file. This defaults to 1.
output.bin	Define the desired output bin size (in minutes). This defaults to 30.
t.cycle	Define the period of the environmental cycle or a single day in hours. This defaults to 24.

Value

A matrix array with 33 columns (number of rows depends on number of days, and the input parameters of this function):

ZT ZT values starting at ZT00 (time at which light turns ON).

I1:I32 Columns of binned locomotor activity data (each column represents a single fly).

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
bd <- binData(data = td)
bd <- binData(data = td, input.bin = 1, output.bin = 15, t.cycle = 24)
```

CoM *Objectively quantify and visualise phases and calculate consolidation in pre-defined time-windows*

Description

This function calculates phase of center of mass of either activity or sleep data, and generates polar plots. The function also computes consolidation of activity or sleep in the defined windows of time. The output of this function is a list, the elements of which are a polar plot with the phases in each window for each fly, depicted, and a table which has the window specific times in ZT units and the consolidation values. For this function to work as expected, users must start the analysis at ZT00.

Usage

```
CoM(
  input,
  data = "Activity",
  bin = 30,
  t.cycle = 24,
  window = list(c(21, 3), c(9, 15)),
  rm.channels = c()
)
```

Arguments

input	Input data file. The input for this function must be the output of the function binData(), sleepData() or ??wakeData(). See ??binData(), ??sleepData() and ??wakeData().
data	If the input for this function is the output of binData() or sleepData(), then this must be "Activity" or "Sleep", respectively.
bin	Intervals in which input data are saved (in minutes). This defaults to 30.
t.cycle	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
window	A list of vectors, each defining the start and end ZT values of the window of interest. For instance, if users want to define the morning window as ZT21 to ZT03 and evening window as ZT09 to ZT15, then this should be list(c(21,3), c(9,15)). This defaults to list(c(18,6), c(6,18)). For the purposes of the polar plot, data from only the first two windows are used. But, for the data table, users can define as many windows as they like, and the function should work as expected.
rm.channels	A vector of channels from a DAM monitor that must be discarded from analysis. If channels 1 to 5 must be removed, type in c(1:5). If channels 1 to 5 and 10 to 13 and 15 and 17 must be removed, type in c(1:5,10:13,15,17). Default is to include all individuals.

Value

A `list` with two items:

Plot A `plotly htmlwidget` with the center of mass plotted and averages plotted for each window.

Data A `data.frame` with 32 rows (one for each fly) and 1 + (number of user defined windows * 2) columns (First column contains channel/fly identity and the remaining columns contain the fly-wise consolidation and center of mass values.)

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
bd <- binData(td)
phase <- CoM(input = bd)
```

---

df *Example DAM data*

Description

A sample `data.frame` which has beam crossing data was collected every minute to facilitate activity and sleep analysis. Data is collected for wildtype flies under LD 12:12 and DD at 25 degree Celsius using the Drosophila Activity Monitoring (DAM) system, Trikinetics.

Usage

```
df
```

Format

A `data.frame` containing 28234 rows and 42 variables:

V1:V10 Several columns of information generated by DAMScan. Only two columns here are of interest to us. Column V2 which has date information and column V3 which has time of day information.

V11:V42 Activity counts for each of the 32 flies recorded in a single activity monitor.

`indActogram` *Plot actogram of individual fly*

Description

Allows users to generate individual actograms. The input for this function must be the output of the binData() function. The output of this function is a plotly object.

Usage

```
indActogram(
  data,
  bin = 30,
  t.cycle = 24,
  ind = 1,
  key.acto = 1,
  color = rgb(0, 0, 0, 1)
)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function binData(). See ??binData().
`bin`	Define the bin size (in minutes) in which input data is saved.
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
`ind`	The channel number (or individual) whose actogram must be plotted.
`key.acto`	Key for reactive input tables in the shiny app.
`color`	Color of actograms in rgb format. The input for this must be a vector with 4 values, i.e., r,g,b,transparency. The values for r,g,b can only be between 0 and 1. 0,0,0 would be black and 1,1,1 would be white. Transparency values can also go from 0 to 1, 0 being fully transparent and 1 being fully opaque.

Value

A plotly htmlwidget with the actogram of a user defined fly.

Examples

```
## Not run:
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
bd <- binData(td)
ind.actogram <- indActogram(data = bd, bin = 30, ind = 2)

## End(Not run)
```

---

indPeriodogramAct *Periodogram plot for activity data of individual flies*

Description

This function generates a periodogram for the activity data of a single fly. Input for this function must be an output from the trimData() function. The output of this function is a plotly object.

Usage

```
indPeriodogramAct(
  data,
  bin = 1,
  method = "ChiSquare",
  low.per = 16,
  high.per = 32,
  alpha = 0.05,
  time.res = 20,
  ind = 1
)
```

Arguments

data	Input data file. If the method for analysis is "ChiSquare", then the input for this function must be the output of the function trimData(). See ??trimData(). Otherwise, the input for this function must be the output of the function binData(). See ??binData().
bin	Intervals in which data are sampled (in minutes). This defaults to 1. This must be changed appropriately depending on method of analysis and the input data set.
method	Choose the method for performing time-series analysis. Currently, three methods are implemented for analysis - "ChiSquare", "Autocorrelation", and "Lomb-Scargle". This defaults to "ChiSquare".
low.per	Choose the lowest period (in hours) for analysis. This defaults to 16.

high.per	Choose the highest period (in hours) for analysis. This defaults to 32.
alpha	Choose the significance level for periodogram analysis. This defaults to 0.05.
time.res	Resolution of periods (in minutes) to analyse while using the ChiSquare periodogram. For instance, if users wish to scan periods from low.per to high.per in the following manner: 16, 16.5, 17, 17.5, and so on, then time.res must be 30. This defaults to 20.
ind	The channel number (or individual) whose periodogram must be plotted.

Value

A `plotly` `htmlwidget` with the individual periodogram of a user defined fly.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
ind.periodogram.act <- indPeriodogramAct(data = td, ind = 13)
```

---

indPeriodogramSleep *Periodogram plot for sleep data of individual flies*

Description

This function generates a periodogram for the sleep data of a single fly. The output of this function is a plotly object.

Usage

```
indPeriodogramSleep(
  data,
  bin = 30,
  method = "ChiSquare",
  low.per = 16,
  high.per = 32,
  alpha = 0.05,
  time.res = 20,
  ind = 1
)
```

Arguments

data	Input data file. The input for this function must be an output from one of either sleepData(), deepSleepData() or lightSleepData() functions. See ??sleepData(), ??deepSleepData() and/or ??lightSleepData().
bin	Intervals in which input data is saved (in minutes). This defaults to 30.

method	Choose the method for performing time-series analysis. Currently, three methods are implemented for analysis - "ChiSquare", "Autocorrelation", and "Lomb-Scargle". This defaults to "ChiSquare".
low.per	Choose the lowest period (in hours) for analysis. This defaults to 16.
high.per	Choose the highest period (in hours) for analysis. This defaults to 32.
alpha	Choose the significance level for periodogram analysis. This defaults to 0.05.
time.res	Resolution of periods (in minutes) to analyse while using the ChiSquare periodogram. For instance, if users wish to scan periods from low.per to high.per in the following manner: 16, 16.5, 17, 17.5, and so on, then time.res must be 30. This defaults to 20.
ind	The channel number (or individual) whose periodogram must be plotted.

Value

A plotly htmlwidget with the individual periodogram of a user defined fly.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 3, bin = 1, t.cycle = 24)
sd <- sleepData(td)
ind.periodogram.sleep <- indPeriodogramSleep(data = sd, ind = 10)
```

---

indSomnogram *Generate actograms for sleep data (Somnograms) for individual flies*

Description

This function generates a somnogram for a single fly. Input for this function must be an output from the trimData() function. The output of this function is a plotly object. In a particular bin, sleep is calculated as the total minutes of inactivity equal to or greater than the defined threshold (sleep.def; typically, 5-minutes).

Usage

```
indSomnogram(
  data,
  sleep.def = c(5),
  bin = 30,
  t.cycle = 24,
  ind = 1,
  key.somno = 1,
  color = rgb(0, 0, 0, 1)
)
```

Arguments

data	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
sleep.def	Definition of sleep. Traditionally, a single bout of sleep is defined as any duration of inactivity that is equal to or greater than 5-minutes. However, sometimes it may be of interest to examine longer bouts of sleep; sleep.def allows users to change the definition of sleep. This defaults to 5.
bin	Intervals in which data are saved (in minutes). This defaults to 30. The value of bin cannot be lower than that of sleep.def.
t.cycle	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
ind	The channel number (or individual) whose periodogram must be plotted.
key.somno	Key for reactive input tables in the shiny app.
color	Color of somnograms in rgb format. The input for this must be a vector with 4 values, i.e., r,g,b,transparency. The values for r,g,b can only be between 0 and 1. 0,0,0 would be black and 1,1,1 would be white. Transparency values can also go from 0 to 1, 0 being fully transparent and 1 being fully opaque.

Value

A plotly htmlwidget with the somnogram of a user defined fly.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
somnogram <- indSomnogram(data = td, ind = 21)
```

---

peakIdentifier *Phase identifier for activity data*

Description

This function generates a list of outputs. The first element is a plot of raw activity along with the smoothed data and objectively estimated phases of peak of activity data. Smoothing of activity is done using a Savitzky-Golay filter. The input of this function must be the output of the trimData function. This function requires the packages "plotly", "pracma" and "signal". As of now, this function works only for 24-hour T-cycles.

Usage

```
peakIdentifier(
  data,
  filt.order = 3,
  filt.length = 51,
```

```
  min.peak.dist = 100,
  peak.ht.scal = 0.5,
  ZT = c(0, 12),
  rm.channels = c()
)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData(). This assumes that the output from trimData has data binned in 1-min intervals.
`filt.order`	The filter order. This defaults to 3.
`filt.length`	The length of filter in indices. This defaults to 51.
`min.peak.dist`	The minimum distance between peaks to be picked (in indices). This defaults to 100.
`peak.ht.scal`	A scaling factor to identify peaks. This defaults to 0.5. A value of 0.5 will only find peaks that are at least half as tall as that of the tallest peak.
`ZT`	A vector defining the regions around which the algorithm must looks for peaks. This defaults to c(0,12). The first value in this vector is always assumed to be the morning peak.
`rm.channels`	All the channels that users want to remove from their averaging. This must be a vector, i.e., channels must be separated by commas. For instance, if users choose to remove channels 1 to 5, 25 and 32, then the input should be either c(1,2,3,4,5,25,32) or c(1:5,25,32). This defaults to an empty vector, meaning no individuals are removed from analysis.

Value

A `list` with two items:

Plots A `plotly` `htmlwidget` with all the averaged activity overlayed with the smoothed data and markers to point out identified peaks in a 4-by-8 array.

Data A `matrix` `array` with 32 rows (one for each fly) and 9 columns (Channel/Fly identity, Phases of morning peak onset, maxima and offset, Morning peak height, Phases of evening peak onset, maxima and offset, and Evening peak height (all phases are measured in ZT)).

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 3, bin = 1, t.cycle = 24)
pks <- peakIdentifier(data = td)
```

`profilesAct` *Compute and plot activity profiles*

Description

Users can compute average profiles and visualise the same. Averages can be performed either over Flies, Days, Both or None. Except in the case of "None" the output of this function will be a list with two elements. One is the generated plot and the other is a table with the activity values and corresponding standard errors of the mean.

Usage

```
profilesAct(
  data,
  bin = 30,
  t.cycle = 24,
  average.type = "Both",
  rm.channels = c()
)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function binData(). See ??binData().
`bin`	Intervals in which data are saved (in minutes). This defaults to 30. This value must be the same as that for binData().
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24. This value must be the same as that for binData().
`average.type`	Define how the averaging must be done for computing profiles. There are 4 choices; i) "Flies": This will average over all the flies for the number of days of data that there are, and produce an averaged time-series, ii) "Days": This will average activity over all the days but for each fly and provide 32 averaged plots, iii) "Both": This will average over both flies and days and provide one composite average profile, iv) "None": This will not average and produce any plots; the output will be the same as the input file. This defaults to "Both".
`rm.channels`	All the channels that users want to remove from their averaging. This must be a vector, i.e., channels must be separated by commas. For instance, if users choose to remove channels 1 to 5, 25 and 32, then the input should be either c(1,2,3,4,5,25,32) or c(1:5,25,32). This defaults to an empty vector, meaning no individuals are removed from analysis.

Value

Except when average.type = "None", a `list` with two items. When average.type = "None", input file is returned.

If average.type = "Days":

Profiles **ZT** Column with ZT values.

I1:I32 Data averaged over days for each of 32 flies.

ZT Column with ZT values.

I1:I32 SEM (across days) for each of 32 flies.

Plot A `plotly` `htmlwidget` with the activity profiles in a 4-by-8 array.

If average.type = "Flies":

Profiles **ZT** Column with ZT values.

Mean Data averaged over all 32 flies for the entire duration of chosen days.

SEM SEM (across flies).

Plot A `plotly` `htmlwidget` with the activity time-series.

If average.type = "Both":

Profiles **ZT** Column with ZT values.

Mean Data averaged over all days and all 32 flies.

SEM SEM (across flies).

Plot A `plotly` `htmlwidget` with the activity profile.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
bd <- binData(td)
pro <- profilesAct(data = bd)
pro <- profilesAct(data = bd, bin = 60, t.cycle = 32,
average.type = "Days", rm.channels = c(1:5,25,32))
```

`profilesSleep` *Compute and plot sleep profiles*

Description

Users can compute average profiles and visualise the same. Averages can be performed either over Flies, Days, Both or None. Except in the case of "None" the output of this function will be a list with two elements. One is the generated plot and the other is a table with the activity values and corresponding standard errors of the mean.

Usage

```
profilesSleep(
  data,
  bin = 30,
  t.cycle = 24,
  average.type = "Both",
  rm.channels = c()
)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function sleepData(). See ??sleepData().
`bin`	Intervals in which data are saved (in minutes). This defaults to 30. This value must be the same as that for sleepData().
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24. This value must be the same as that for sleepData().
`average.type`	Define how the averaging must be done for computing profiles. There are 4 choices; i) "Flies": This will average over all the flies for the number of days of data that there are, and produce an averaged time-series, ii) "Days": This will average activity over all the days but for each fly and provide 32 averaged plots, iii) "Both": This will average over both flies and days and provide one composite average profile, iv) "None": This will not average and produce any plots; the output will be the same as the input file. This defaults to "Both".
`rm.channels`	All the channels that users want to remove from their averaging. This must be a vector, i.e., channels must be separated by commas. For instance, if users choose to remove channels 1 to 5, 25 and 32, then the input should be either c(1,2,3,4,5,25,32) or c(1:5,25,32). This defaults to an empty vector, meaning no individuals are removed from analysis.

Value

Except when average.type = "None", a `list` with two items. When average.type = "None", input file is returned.

If average.type = "Days":

Profiles **ZT** Column with ZT values.

I1:I32 Data averaged over days for each of 32 flies.

ZT Column with ZT values.

I1:I32 SEM (across days) for each of 32 flies.

Plot A `plotly` `htmlwidget` with the sleep profiles in a 4-by-8 array.

If average.type = "Flies":

Profiles **ZT** Column with ZT values.

Mean Data averaged over all 32 flies for the entire duration of chosen days.

SEM SEM (across flies).

Plot A `plotly` `htmlwidget` with the sleep time-series.

If average.type = "Both":

Profiles **ZT** Column with ZT values.

Mean Data averaged over all days and all 32 flies.

SEM SEM (across flies).

Plot A `plotly` `htmlwidget` with the sleep profile.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 3, bin = 1, t.cycle = 24)
sd <- sleepData(td)
pro <- profilesSleep(data = sd)
```

`rosePlotsAct` *Generate rose plots for averaged activity data*

Description

Users can generate rose plots for the averaged activity data. The input for this function must be output from binData(). The output of this function is a plotly object. By default the averaging of profiles is done over all flies and days.

Usage

```
rosePlotsAct(data, bin = 30, t.cycle = 24, rm.channels = c())
```

Arguments

`data`	Input data file. The input for this function must be the output of the function binData(). See ??binData().
`bin`	Intervals in which data are saved (in minutes). This defaults to 30. This value must be the same as that for binData().
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24. This value must be the same as that for binData().
`rm.channels`	All the channels that users want to remove from their averaging. This must be a vector, i.e., channels must be separated by commas. For instance, if users choose to remove channels 1 to 5, 25 and 32, then the input should be either c(1,2,3,4,5,25,32) or c(1:5,25,32). This defaults to an empty vector, meaning no individuals are removed from analysis.

Value

A `plotly` htmlwidget with rose plots for locomotor activity.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
bd <- binData(td)
r.plot <- rosePlotsAct(data = bd)
```

`rosePlotsSleep` *Generate rose plots for averaged sleep data*

Description

Users can generate rose plots for the averaged sleep data. The input for this function must be output from sleepData(). The output of this function is a plotly object. By default the averaging of profiles is done over all flies and days.

Usage

```
rosePlotsSleep(data, bin = 30, t.cycle = 24, rm.channels = c())
```

Arguments

`data`	Input data file. The input for this function must be the output of the function sleepData(). See ??sleepData().
`bin`	Intervals in which data are saved (in minutes). This defaults to 30. This value must be the same as that for sleepData().
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24. This value must be the same as that for sleepData().
`rm.channels`	All the channels that users want to remove from their averaging. This must be a vector, i.e., channels must be separated by commas. For instance, if users choose to remove channels 1 to 5, 25 and 32, then the input should be either c(1,2,3,4,5,25,32) or c(1:5,25,32). This defaults to an empty vector, meaning no individuals are removed from analysis.

Value

#' @return A `plotly` htmlwidget with rose plots for sleep data.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 1, bin = 1, t.cycle = 24)
sd <- sleepData(td)
r.plot <- rosePlotsSleep(data = sd)
```

`sleepData` *Equivalent of binData() but for sleep data*

Description

Allows users to bin data sets into intervals of time different from the data collection interval. The input for this function must be the output of the trimData() function. The output of this function is a data frame. The first column of which stores Zeitgeber Time values (assuming that the start.time in the trimData() function was set at Zeitgeber Time 00). All subsequent columns have binned sleep data for each fly. In a particular bin, sleep is calculated as the total minutes of inactivity equal to or greater than the defined threshold (sleep.def; typically, 5-minutes).

Usage

```
sleepData(data, sleep.def = c(5), bin = 30, t.cycle = 24)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
`sleep.def`	Definition of sleep. Traditionally, a single bout of sleep is defined as any duration of inactivity that is equal to or greater than 5-minutes. However, sometimes it may be of interest to examine longer bouts of sleep or specific bout durations; sleep.def allows users to change the definition of sleep. The default input is a single value vector of value 5. If users wish to analyse sleep only between 5 to 20 mins, the input must be c(5,20).
`bin`	Intervals in which data are saved (in minutes). This defaults to 30. The value of bin cannot be lower than that of sleep.def.
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.

Value

A `data.frame` with 33 columns (number of rows depends on number of days, and the input parameters of this function):

ZT ZT values starting at ZT00 (time at which light turns ON).

I1:I32 Columns of binned sleep data (each column represents a single fly).

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 4, bin = 1, t.cycle = 24)
sd <- sleepData(data = td[,1:15])
```

`sleepOnsetBoutLength` *Computes, tabulates and plots day-time and night-time onset of sleep bouts and their respective lengths*

Description

This function allows users to compute day-time and night-time onset of sleep bouts and their respective bout lengths. Also generates a plot to visualise and compare within and across day and night-time windows. The input for this function must be the output from the trimData() function. Number of days to analyse must be at least 2 days. If the number of days is more than 2 days, the function will compute statistics for the first day only. The output of this function is a list with three elements, i.e., "Daytime.Data" is a data frame which has the onset of sleep bout (minutes since the start of the day window) and the length of that bout (in minutes) for each fly, "Nighttime.Data" which is the same as "Daytime.Data" but for the night window, and "Plots" which allows the visualisation of the two data sets.

Usage

```
sleepOnsetBoutLength(data, sleep.def = c(5), t.cycle = 24, photoperiod = 12)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
`sleep.def`	Definition of sleep. Traditionally, a single bout of sleep is defined as any duration of inactivity that is equal to or greater than 5-minutes. However, sometimes it may be of interest to examine longer bouts of sleep or specific bout durations; sleep.def allows users to change the definition of sleep. The default input is a single value vector of value 5. If users wish to analyse sleep only between 5 to 20 mins, the input must be c(5,20).
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
`photoperiod`	Duration (in hours) of what can be considered day-phase. This defaults to 12.

Value

A `list` with three items:

Daytime.Data **Channel** Fly identity.

Start Onset of sleep bout (in minutes since the start of day window)

BoutLength Length of the sleep bout (in minutes)

Nighttim.Data **Channel** Fly identity.

Start Onset of sleep bout (in minutes since the start of night window)

BoutLength Length of the sleep bout (in minutes)

Examples

```
td <- trimData(data = df, start.date = "28 Dec 20", start.time = "21:00",
n.days = 2, bin = 1, t.cycle = 24)
bout.onset.vs.length <- sleepOnsetBoutLength(data = td)
```

---

sleepStages *Plot sleep stages in individual flies*

Description

Allows users to generate individual plots of sleep stages in flies. Sleep stages are defined as follows: 5 to 30-min as short sleep (light blue), 30 to 60-min as intermediate sleep (medium blue) and 60 to 720-min as deep sleep (dark blue). Activity is plotted in red. The input for this function must be the output of the trimData() function. The output of this function is a plot. Note: At this moment, this works accurately only for 24-h days.

Usage

```
sleepStages(data, n.days, channel = 1, photoperiod = 12)
```

Arguments

data	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
n.days	The number of cycles for which sleep stages must be visualized.
channel	The channel number of the fly to be visualized.
photoperiod	This value determines the duration of photo-phase and scoto-phase of the 24-h day. Defaults to 12.

Value

A plot with the day-time and night-time sleep stages of a user defined fly for the number of cycles provided by the user.

Examples

```
## Not run:
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 2, bin = 1, t.cycle = 24)
sleepStages(data = td, n.days = 1)

## End(Not run)
```

`sleepStat` *Computes and tabulates day-time and night-time sleep statistics*

Description

This function allows users to estimate day-time and night-time average sleep bout duration, number and latency. Sleep bout latency is defined as the time taken (in minutes) for the occurrence of the first sleep bout since respective transitions. The input for this function must be the output from the trimData() function. The output of this function is a matrix which contains fly-wise (each row) data.

Usage

```
sleepStat(data, sleep.def = c(5), t.cycle = 24, photoperiod = 12)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
`sleep.def`	Definition of sleep. Traditionally, a single bout of sleep is defined as any duration of inactivity that is equal to or greater than 5-minutes. However, sometimes it may be of interest to examine longer bouts of sleep or specific bout durations; sleep.def allows users to change the definition of sleep. The default input is a single value vector of value 5. If users wish to analyse sleep only between 5 to 20 mins, the input must be c(5,20).
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.
`photoperiod`	Duration (in hours) of what can be considered day-phase. This defaults to 12.

Value

A `matrix` `array` matrix with 32 rows (one for each fly) and 9 columns:

Channel Fly identity.

Day.BoutNumber Number of sleep bouts in the user defined day time.

Day.BoutDuration.Mean Mean sleep duration in the user defined day time.

Day.BoutDuration.Median Median sleep duration in the user defined day time.

Day.Latency Time taken for the first sleep bout to occur in the user defined day time.

Day.Total Total minutes of daytime sleep.

Night.BoutNumber Number of sleep bouts in the user defined night time.

Night.BoutDuration.Mean Mean sleep duration in the user defined night time.

Night.BoutDuration.Median Median sleep duration in the user defined night time.

Night.Latency Time taken for the first sleep bout to occur in the user defined night time.

Night.Total Total minutes of nighttime sleep.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
slp.stat <- sleepStat(data = td)
```

`transitionProbs` *Plot and generate data for transition frequencies between sleep stages*

Description

This function generates a sankey plot enabling the visualization of percentage of times transtions from various states of activity and sleep occur. This function also generates a data table to download individual values of these transition percentages. Sleep stages are defined as follows: 5 to 30-min as short sleep (light blue), 30 to 60-min as intermediate sleep (medium blue) and 60 to 720-min as deep sleep (dark blue). Activity is defined as the standard number of beam crosses. The input for this function must be the output of the trimData() function. The output of this function is a list. Note: At this moment, this works accurately only for 24-h days.

Usage

```
transitionProbs(data, n.days, photoperiod = 12)
```

Arguments

`data`	Input data file. The input for this function must be the output of the function trimData(). See ??trimData().
`n.days`	The number of cycles for which sleep stages must be visualized.
`photoperiod`	This value determines the duration of photo-phase and scoto-phase of the 24-h day. Defaults to 12.

Value

A `list` with two items:

Plots A `plotly` `htmlwidget` with the Sankey diagram (representing the average transition percentages).

Data A `matrix` `array` with individual wise transition percentages.

Examples

```
## Not run:
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 2, bin = 1, t.cycle = 24)
tp <- transitionProbs(data = td, n.days = 1)

## End(Not run)
```

`trimData` *Trim data for downstream analyses*

Description

trimData() allows users to define start date, start time and number of days for analysing data sets.

Usage

```
trimData(data, start.date, start.time, n.days, bin = 1, t.cycle = 24)
```

Arguments

`data`	Input data file. This must be DAM scanned monitor files, saved in 1-min intervals starting at time 00:00.
`start.date`	Define starting date for analysis. Date inputs must be in this form: "DD Mon YY"; August 6, 2020 must be input as "6 Aug 20". Any other form will result in computation errors.
`start.time`	Define start time for analysis. Times must be input in the 24-h format and in the following form: "HH:MM"; 10 AM must be "10:00" and 10 PM must be "22:00".
`n.days`	Number of days to analyse.
`bin`	Intervals in which data are sampled (in minutes). This defaults to 1.
`t.cycle`	Define the period of the environmental cycle or a single day in hours. This defaults to 24.

Value

A `data.frame` containing trimmed data.

Examples

```
td <- trimData(data = df, start.date = "19 Dec 20", start.time = "21:00",
n.days = 10, bin = 1, t.cycle = 24)
```

Index