Package 'ActisoftR'

October 16, 2018

Type Package

Title A Toolbox for Parsing Scored Actigraphy Data

Version 0.0.2

Date 2018-10-01

Depends R (>= 3.1.0)

Description 'Actigraphy' is a cost-effective and convenient tool for activity-based monitoring. 'ActisoftR' was designed for parsing 'actigraphy' outputs and to summarise scored data across user-defined intervals. It consists of several functions for importing, generating reports and statistics, and for data visualization.

URL https://github.com/SWRC/ActisoftR

Imports ggplot2, dplyr, scales, tibble, lubridate, data.table, magrittr, methods, RColorBrewer, tidyr

Suggests DT, Rcpp, rmarkdown, roxygen2, knitr, testthat, readr, plotly, devtools

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Edgar Santos-Fernandez [aut], Lora Wu [aut, cre], Margo van den Berg [aut]

Maintainer Lora Wu <L.Wu@massey.ac.nz>

Repository CRAN

Date/Publication 2018-10-16 10:00:09 UTC

R topics documented:

`act` *Actigraphy data*

Description

Contains data from three participants.

Usage

```
act
```

Format

A data frame

actigraph_brand actigraph_brand
subject_ID subject_ID
interval_type interval_type
interval_number interval_number
duration duration
efficiency efficiency
sleep_time sleep_time
bad bad
datime_start datime_start
datime_end datime_end
comments comments

Source

www.sleepwake.ac.nz

Examples

```
data("act")
```

---

csd *Cumulative sleep debt.*

Description

Cumulative sleep debt.

Usage

```
csd(x, baseline_sleep, reset = 2, plot = TRUE, ylabel, ...)
```

Arguments

x	a report period. See report_period().
baseline_sleep	a data frame containing the normal sleep duration or baseline by participant.
reset	consecutive number of sleep periods that will set as zero the cumulative sleep debt when the total sleep in 24h is greater than the baseline_sleep. It does not depend on the exceeding duration or credit.
plot	a logical variable. By default is TRUE.
ylabel	label of the y-axis
...	optional parameters

Details

if no reset is desired just set a large value e.g. 1E3. Three cumsum values are computed: cumsum, cumsum_reset and cumsum_actual. The last one allows the deficiency to be in credit (positive values).

Value

a data frame and optionally a graph. def is the daily sleep deficiency.

Examples

```
# Example 1
library(lubridate)
data(act)
act$datime_start <- ymd_hms(act$datime_start)
act$datime_end <- ymd_hms(act$datime_end)

par <- data.frame(subject_ID = 1,
summary_duration_h = 24,
summary_type = "sequential",
summary_start_datime = ymd_hms("2017-12-05 00:00:00 UTC"),
summary_end_datime = ymd_hms("2017-12-15 00:00:00 UTC"))

rep <- report_period(period = par , acti_data = act)

start_date <- act[act$subject_ID==1 & act$interval_type == "FLIGHT" &
act$comments == "SIN-AMS",]$datime_end
sel <- act[act$datime_start >= start_date & act$datime_end <=
start_date + days(10),]

par_afterflight <- data.frame(subject_ID = 1,
summary_duration_h = 24,
summary_type = "sequential",
summary_start_datime = round.POSIXt(start_date, "day"),
summary_end_datime = round.POSIXt(start_date, "day") + days(10))
rep_afterflight <- report_period(period = par_afterflight , acti_data = sel)
baseline <- data.frame(subject_ID = 1, baseline_sleep = mean(rep$total_sleep))
reset <- 2
z <- csd(x = rep_afterflight, baseline_sleep = baseline, reset = 2)
z[[2]]
plot(z[[1]])
# Example 2 with no reset by adding large reset value
csd(x = rep_afterflight, baseline_sleep = baseline, reset = 1E5)
```

`deagg` *Deaggregation of reports for summary_type sequential or daily.*

Description

Deaggregation of reports for summary_type sequential or daily.

Usage

```
deagg(period)
```

Arguments

`period`	a dataframe containing participants, start and end date/time period

Value

a dataframe

Examples

```
library("lubridate")
data("act")
par <- data.frame(subject_ID = 1,
                  summary_duration_h = 24,
                  summary_type = "sequential",
                  summary_start_datime = ymd_hms("2017-12-05 00:00:00 UTC"),
                  summary_end_datime = ymd_hms("2017-12-15 00:00:00 UTC"))
deagg(period = par)
```

---

home.night.shade *Generates home night time periods*

Description

This function is used by the Darwent plot

Usage

```
home.night.shade(x, shadow.start = "20:00:00", shadow.end = "06:00:00",
  homeTZ, tz = "UTC", ...)
```

Arguments

x	a dataframe
shadow.start	starting time of the home night, 22:00:00 by defaults
shadow.end	ending time of the home night, 08:00:00 by defaults
homeTZ	participant's home time zone. A data frame containing for each participant corresponding home time zone.
tz	is the time zone
...	Optional parameters

Value

a dataframe

Examples

```
library("dplyr")
library("lubridate")
data(act)
dat <- act %>% group_by(subject_ID) %>%
mutate(start = min(datime_start)) %>%
ungroup %>% filter(datime_start <= start + lubridate::days(5),
subject_ID == 1, interval_type != "ACTIVE")
homeTZ = data.frame(subject_ID = "1",
TZ = "Pacific/Auckland", stringsAsFactors = FALSE)
home.night.shade(dat, shadow.start = "20:00:00", shadow.end = "06:00:00",
homeTZ = homeTZ)
```

---

int_startend *Returns the starting/ending point of a report*

Description

Returns the starting/ending point of a report

Usage

```
int_startend(x, tz = "UTC", ...)
```

Arguments

x	a dataframe
tz	is the time zone
...	Optional parameters

Value

a dataframe

---

local.night.shade *Generates local night time periods*

Description

This function is used by the Darwent plot

Usage

```
local.night.shade(x, shadow.start = "20:00:00", shadow.end = "06:00:00",
  localTZ, tz = "UTC", ...)
```

Arguments

x	a dataframe
shadow.start	starting time of the home night, 22:00:00 by defaults
shadow.end	ending time of the home night, 08:00:00 by defaults
localTZ	a dataframe containing the local time zone by participant during the days worked.
tz	the time zone
...	Optional parameters

Value

a dataframe

Examples

```
library("dplyr")
library("lubridate")
data(act)
# Selecting the date/time arrival to Amsterdam for subject_ID 1
start_date <- act[act$subject_ID==1 & act$interval_type == "FLIGHT" &
act$comments == "SIN-AMS",]$datime_end
sel <- act[act$datime_start >= start_date & act$datime_end <= start_date + lubridate::days(10),]
localTZ <- data.frame(subject_ID = "1",
                      TZ = "Europe/Amsterdam", stringsAsFactors = FALSE)
local.night.shade(sel, shadow.start = "20:00:00",
                  shadow.end = "06:00:00", localTZ = localTZ)
```

---

plot_Darwent *Plots participants' SLEEP/WAKE intervals.*

Description

Plots participants' SLEEP/WAKE intervals.

Usage

```
plot_Darwent(x, datebreaks = "12 hour", base = "TRUE", acolor, decal,
  export = FALSE, show_plot = TRUE, si, tz = "UTC", tz2, homeTZ,
  shade = FALSE, local.shade = FALSE, ...)
```

Arguments

x	a dataframe.
datebreaks	is the distance between breaks in the x-axis. "12 hour" by default.
base	matches the participants at the same start date.
acolor	specifies the colors for the interval_type.
decal	is a parameter for shifting the start date.
export	if TRUE, it will export the data as CSV.
show_plot	logical to produce the plot or not.
si	defines the size of the geom_segment, 8 by default.
tz	is the times zone. tz = "UTC" by default.
tz2	an additional time zone used for a secondary x-axis.
homeTZ	a data frame containing the all subject_IDs in x and the time zone. It is used for the home night shading.
shade	if TRUE, it will draw in light green the home night period. FALSE by default.
local.shade	(deprecated) if TRUE, it will draw in gray the local night period. FALSE by default.
...	optional parameters.

Value

a plot.

Examples

```
library("dplyr")
library("lubridate")
data(act)
dat <- act %>% group_by(subject_ID) %>%
  mutate(start = min(datime_start)) %>%
  ungroup %>% filter(datime_start <= start + days(10), interval_type != "ACTIVE")
plot_Darwent(dat,  acolor = c("#D55E00", "black", "#56B4E9"),
                  tz2 = "Pacific/Auckland",
                  si = c(4, 3, 2.5))

# Adding the 'home' night shade
# creating a data frame with the home time zone of the individuals
homeTZ = data.frame(subject_ID = unique(dat$subject_ID),
                    TZ = rep("Pacific/Auckland", 5),
                    stringsAsFactors = FALSE)
plot_Darwent(dat, acolor = c("#D55E00", "black", "#56B4E9"),
             si = c(4, 3, 2.5),
             tz = "UTC", tz2 = "Pacific/Auckland", shade = TRUE,
             homeTZ = homeTZ)
```

plot_long *Plots with-in participant activity intervals (SLEEP, REST, ACTIVE, etc) in 24 hour sections.*

Description

Plots with-in participant activity intervals (SLEEP, REST, ACTIVE, etc) in 24 hour sections.

Usage

```
plot_long(dat, acolor, si, tz = "UTC", tz2, sp = "00:00:00",
  with_date = FALSE, alphas = 1, hourbreaks = 5, ...)
```

Arguments

dat	a dataframe.
acolor	the color of the lines.
si	defines the size of the geom_segment, 1.25 by default.
tz	the time zone.
tz2	an additional time zone.
sp	the starting point in the x-axis. Set as 00:00:00 by default.
with_date	allows adding the cutpoint date to the y-axis. with_date = FALSE by default.
alphas	is the transparency.
hourbreaks	is the number of labels on the x-axis.
...	Optional parameters.

Details

The date-time data in the data frame dat will keep time zone. The argument tz and tz2 are used to specify the time in the x-axis.

Value

a plot.

Examples

```
library("dplyr")
data(act)
act[act$subject_ID == 1 & act$interval_type != "ACTIVE",] %>%
plot_long(., si = c(3, 2.5, 2, 1.5),
             tz2 = "Pacific/Auckland",
             acolor = c("#D55E00", "black", "#56B4E9"))
```

`portion` *#Slices a data.frame containing date-time type columns.*

Description

#Slices a data.frame containing date-time type columns.

Usage

```
portion(x, from, to, ...)
```

Arguments

`x`	at dataframe
`from`	a vector of lenght = participants
`to`	a vector of lenght = participants
`...`	Optional parameters

Details

it does not include overlaps periods: periods that started (ended) before (after) start.period (end.period)

Value

a dataframe

Examples

```
library("lubridate")
start.period <- ymd_hms("2017-12-10 12:00:00")
end.period <- ymd_hms("2017-12-12 12:00:00")
portion(act[act$subject_ID==1,], from = start.period,  to = end.period)
```

`portion_withoverlaps` *Slices a data.frame containing date-time type columns including overlaping periods.*

Description

Slices a data.frame containing date-time type columns including overlaping periods.

Usage

```
portion_withoverlaps(x, from, to, tz = "UTC", ...)
```

Arguments

x	a dataframe
from	a vector of lenght = participants
to	a vector of lenght = participants
tz	is the time zone
...	Optional parameters

Details

it does include overlaps periods: periods that started (ended) before (after) start.period (end.period)

Value

a dataframe

Examples

```
library("lubridate")
start.period <- ymd_hms("2017-12-10 12:00:00")
end.period <- ymd_hms("2017-12-12 12:00:00")
portion_withoverlaps(act[act$subject_ID==1,], from = start.period,  to = end.period)
```

---

read_actigraph_csv *Imports scored actigraphy data from different actigraph brand output files.*

Description

Specify the path to the files. So far it supports CSV files only. The imported files are merged creating an organic single file. If the CSV files have different number of columns the function will fill missing columns with NAs.

Usage

```
read_actigraph_csv(x = "PATH_TO_DATA", tz = "UTC", ...)
```

Arguments

x	the file path
tz	the time zone
...	Optional parameters

Value

a dataframe

`read_csv_filename` *Imports a CSV file and attaches the file path as a column.*

Description

Imports a CSV file and attaches the file path as a column.

Usage

```
read_csv_filename(filename)
```

Arguments

`filename`	file path

Value

a dataframe

`report_period` *Generates reports based on input periods.*

Description

Generates reports based on input periods.

Usage

```
report_period(period, acti_data, remove_bad = TRUE, tz = "UTC", ...)
```

Arguments

`period`	a dataframe containing participants and the start and end date/time periods.
`acti_data`	a dataframe of the form 'acti_data'.
`remove_bad`	a logical value used to indicate if 'bad' intervals or 'EXCLUDED' are removed. Set as TRUE.
`tz`	is the time zone. Optional argument. tz = "UTC" by default.
`...`	Optional parameters

Details

Periods partially scored as 'bad' or 'EXCLUDED' are due to off-wrist or other reasons. In that case all summary estimates except 'time in bed' will be set to NA, and the column 'with_excluded_bad' will show a TRUE. See examples in the Vignettes. This can be avoided using remove_bad = FALSE. Sleep efficiency will be zero when there is an attempt to sleep (no sleep achieved in the period) within a rest. However, the sleep efficiency will be NA when there is not REST or SLEEP with in an interval of time.

Value

a dataframe

Examples

```
# Example 1
library("lubridate")
data("act")
par <- data.frame(subject_ID = 1,
summary_duration_h = 24,
summary_type = "sequential",
summary_start_datime = ymd_hms("2017-12-05 00:00:00 UTC"),
summary_end_datime = ymd_hms("2017-12-15 00:00:00 UTC"))
report_period(period = par , acti_data = act)
```

---

report_point *Generates reports based on input points.*

Description

Generates reports based on input points.

Usage

```
report_point(period, acti_data, tz = "UTC", ...)
```

Arguments

period	a dataframe containing participants, start and end date/time interval
acti_data	a dataframe of the form "acti_data".
tz	is the time zone. Optional argument. tz = "UTC" by default.
...	Optional parameters

Details

Periods partially scored as 'bad' or 'EXCLUDED' are due to off-wrist or other reasons. In that case the column 'with_excluded_bad' will show TRUE. See examples in the Vignettes.

Value

a dataframe

Examples

```
# Example 1
library(lubridate)
par_point <- data.frame(subject_ID = c(1,1),
                        time_point_datime = ymd_hms(c("2017-12-06 00:00:00 UTC",
                        "2017-12-07 00:00:00 UTC")))
report_point(period = par_point, acti_data = act)
```

`sheep_counter` *Sheep counter*

Description

Computes the empirical probability that a group or an individual are sleeping at a given time of the day.

Usage

```
sheep_counter(dat, tz = "UTC", interv = "10 mins", datebreaks = "2 hour",
  work_data)
```

Arguments

`dat`	a data frame.
`tz`	the time zone.
`interv`	interval of time, set as 10min.
`datebreaks`	is the distance between breaks in the x-axis. "2 hour" by default.
`work_data`	optional parameter used to import work data.

Value

a plot.

Examples

```
# Example 1:
data(act)
# Prob of being sleeping for ID 1 during the baseline sleep
b <- act[act$subject_ID==1 & act$datime_end < as.POSIXct("2017-12-15 00:00:00", tz = "UTC"),]
sheep_counter(b)
```

`write_actigraph_csv` *Exports data as CSV.*

Description

Exports data as CSV.

Usage

```
write_actigraph_csv(x, path = "~", ...)
```

Arguments

`x`	the file path
`path`	is the path for saving the file
`...`	Optional parameters

Value

a dataframe

Index