Package 'PAutilities'

March 6, 2019

Type Package

Title Streamline Physical Activity Research

Version 0.1.2

Depends R (>= 2.10)

Description A collection of utilities that are useful for a broad range of tasks that are common in physical activity research, including the following: creation of Bland-Altman plots, formatted descriptive statistics, metabolic calculations (e.g. basal metabolic rate predictions) and conversions, demographic calculations (age and age-for-body-mass-index percentile), bout analysis of moderate-to-vigorous intensity physical activity, and analysis of bout detection algorithm performance.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/paulhibbing/PAutilities

BugReports https://github.com/paulhibbing/PAutilities/issues

RoxygenNote 6.1.1

Imports AGread (>= 0.1.2), dplyr (>= 0.7), ggplot2 (>= 2.2), graphics, lazyeval (>= 0.2), magrittr (>= 1.5), matchingMarkets (>= 1.0.1), rlang (>= 0.3.1), stats

Suggests testthat

NeedsCompilation no

Author Paul R. Hibbing [aut, cre], Centers for Disease Control and Prevention [ctb]

Maintainer Paul R. Hibbing <paulhibbing@gmail.com>

Repository CRAN

Date/Publication 2019-03-06 17:14:31 UTC

R topics documented:

`ba_plot` *Create a Bland-Altman plot*

Description

Create a Bland-Altman plot

Usage

```
ba_plot(plotdata, x_var, y_var, x_name, y_name, ...)
```

Arguments

`plotdata`	dataframe from which to build the plot
`x_var`	character expression to evaluate for the x-axis
`y_var`	character expression to evaluate for the y-axis
`x_name`	axis label for the x-axis
`y_name`	axis label for the y-axis
`...`	further arguments passed to `theme`

Value

a Bland-Altman plot

References

Bland, J. M., & Altman, D. G. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. lancet, 1(8476), 307-310.

Examples

```
data(ex_data, package = "PAutilities")

# Reduce the number of data points (for illustration purposes) by isolating
# the 150 largest cases

illustration_threshold <-
    quantile(ex_data$Axis1, probs = 1 - (150 / nrow(ex_data)))
ex_data <- ex_data[ex_data$Axis1 > illustration_threshold, ]

# Generate the plot
my_ba <- ba_plot(
    ex_data,
    "(Axis1 + Axis3) / 2",
    "Axis1 - Axis3",
    "mean(Axis1, Axis3)",
    "Axis1 - Axis3"
)

my_ba


# You can add to the plot as you would a normal ggplot object
    my_ba +
      ggplot2::geom_text(
      x = 2000, y = 9000, label = "A",
      size = 8, fontface = "bold", colour = "blue"
      )

# With caution, you can change some automatic options (e.g. color of
# regression line) by overwriting in a new layer

    my_ba + ggplot2::geom_smooth(method = "lm", se = FALSE, colour = "blue")
```

---

bout_mvpa *Classify moderate-to-vigorous physical activity in bouts of a specific minimum length*

Description

Classify moderate-to-vigorous physical activity in bouts of a specific minimum length

Usage

```
bout_mvpa(intensity, var_type = c("METs", "Intensity"), min_duration = 10,
  termination = 3, MoreArgs = list(breaks = c(-Inf, 1.51, 3, Inf), labels =
  c("SB", "LPA", "MVPA"), right = FALSE), ..., timestamps = NULL,
  output_var = c("is_MVPA", "bout_tracker"))
```

Arguments

intensity	a vector of intensity classifications to be re-classified according to the bout definition
var_type	character scalar indicating whether the intensity variable is a numeric vector of metabolic equivalents, or a factor variable giving activity intensity classification
min_duration	numeric scalar: minimum duration of a qualifying bout, in minutes
termination	numeric scalar: consecutive minutes of non-MVPA required to terminate the bout
MoreArgs	required arguments passed to cut
...	optional arguments passed to cut for converting METs to intensity classification
timestamps	optional vector of POSIX-formatted timestamps. Must have same length as intensity
output_var	the output variable(s) to give

Note

output_var gives one or both of is_MVPA and bout_tracker, the former being a vector of indicators (1 or 0) specifying whether a minute is part of a valid MVPA bout, and the latter being a collapsed data frame giving only the valid bouts of MVPA and the relevant information (i.e., duration of the bout, minutes of MVPA, and percentage of time spent in MVPA within the bout). If both are selected, they are returned in a list.

Examples

```
data(ex_data, package = "PAutilities")
ex_data$DateTime <- as.POSIXct(ex_data$DateTime, "UTC")


# Runs with a warning

bout_mvpa(ex_data$METs, "METs")

bout_mvpa(ex_data$METs, "METs", timestamps = ex_data$DateTime)


# Recommended usage
lapply(split(ex_data, strftime(ex_data$DateTime, "%Y-%m-%d", "UTC")),
function(x) {
bout_mvpa(x$METs, "METs", timestamps = x$DateTime)
})

lapply(split(ex_data, strftime(ex_data$DateTime, "%Y-%m-%d", "UTC")),
function(x) {
bout_mvpa(x$METs,
"METs",
timestamps = x$DateTime,
output_var = "is_MVPA")
```

```
})

lapply(split(ex_data, strftime(ex_data$DateTime, "%Y-%m-%d", "UTC")),
function(x) {
bout_mvpa(x$METs,
"METs",
timestamps = x$DateTime,
output_var = "bout_tracker")
})
```

`descriptives` *Compute descriptive statistics for a variable in the metabolic data set*

Description

Compute descriptive statistics for a variable in the metabolic data set

Usage

```
descriptives(dataset, variable, group = NULL)
```

Arguments

`dataset`	the dataset to analyze
`variable`	character scalar giving the variable name to summarize
`group`	character scalar giving an optional grouping variable for the summary

Examples

```
data(ex_data, package = "PAutilities")
ex_data$group_var <- rep(
 c("One", "Two", "Three"),
 each = ceiling(nrow(ex_data)/3)
)[seq(nrow(ex_data))]
descriptives(ex_data, "Axis1", "group_var")
```

ex_data — *Example data for calculating bouts of moderate-to-vigorous physical activity*

Description

A dataset containing accelerometer data and predicted energy expenditure in metabolic equivalents (METs) that can be used to classify moderate-to-vigorous physical activity in continuous bouts.

Usage

```
ex_data
```

Format

A data frame with 10080 rows and 12 variables:

FileID character. Name of the file originating the data

Date character giving the date ("%m/%d/%Y")

Time character giving the time ("%H:%M:%S")

DateTime full timestamp (%Y-%m-%d %H:%M:%S) given as character

dayofyear numeric day of the year (i.e., julian date)

minofday numeric minute of the day (i.e., 0 for midnight and 1439 for 11:59)

Axis1 activity counts for the device's first axis

Axis2 activity counts for the device's second axis

Axis3 activity counts for the device's third axis

Steps number of steps taken

Vector.Magnitude vector magnitude (Euclidian norm) of the activity counts from the three axes

METs predicted energy expenditure, in metabolic equivalents

get_age — *Calculate age*

Description

Takes two Date objects and calculates age based on `difftime` (in days) divided by 365.2425 days per year (for age in years) or 30.4375 days per month (for age in months).

Usage

```
get_age(birthdate, current_date, units = c("years", "months"))
```

Arguments

birthdate	Date object giving the date of birth
current_date	Date object giving the date from which age is to be calculated
units	The units in which age should be reported

Value

Numeric value giving age in the specified units.

Examples

```
get_age(as.Date("2000-01-01"), Sys.Date(), "years")
```

---

get_BMI_percentile *Calculate youth BMI percentile from CDC standards*

Description

Calculate youth BMI percentile from CDC standards

Usage

```
get_BMI_percentile(weight_kg, height_cm, age_yrs, age_mos = NULL,
  sex = c("M", "F"), output = c("percentile", "classification", "both"))
```

Arguments

weight_kg	Weight in kilograms
height_cm	height in centimeters
age_yrs	age in years
age_mos	age in months (optional)
sex	Character scalar indicating participant's sex
output	What should be returned: raw percentile, weight status classification, or both?

Details

If age_mos is *not* provided, it will be calculated based on age_yrs, assuming 365.2425 days per year and 30.4375 days per month. Depending on how the initial age calculation was made, rounding error will occur. Thus, use of the get_age function is recommended. If age_mos *is* provided, age_yrs can be passed as NULL.

Value

One of: A numeric scalar giving the BMI percentile (for output = "percentile"); a factor scalar giving the weight status (for output = "classification"); or a list with the percentile and classification (for output = "both").

References

This function was developed with reference to public domain resources provided by the Centers for Disease Control and Prevention. For more information, see:

https://www.cdc.gov/obesity/childhood/defining.html

https://www.cdc.gov/healthyweight/assessing/bmi/childrens_bmi/tool_for_schools.html

Examples

```
get_BMI_percentile(39.4, 144.5, 12.35, sex = "M")
```

`get_bmr` *Retrieve estimated basal metabolic rate for an individual*

Description

Retrieve estimated basal metabolic rate for an individual

Usage

```
get_bmr(Sex = c("M", "F"), Ht = NULL, Wt, Age, verbose = FALSE,
  RER = NULL, equation = c("ht_wt", "wt", "both"), kcal_table = c("Lusk",
  "Peronnet", "both"), method = c("Schofield", "FAO", "both"),
  MJ_conversion = c("thermochemical", "dry", "convenience", "all"),
  kcal_conversion = 5)
```

Arguments

`Sex`	The individual's sex
`Ht`	The individual's height, in meters
`Wt`	The individual's weight, in kilograms
`Age`	The individual's age, in years
`verbose`	Logical. Should processing updates be printed?
`RER`	numeric. The respiratory exchange ratio
`equation`	The equation to apply
`kcal_table`	The table to reference for converting kilocalories to oxygen consumption. See `get_kcal_vo2_conversion`
`method`	The calculation method to use
`MJ_conversion`	The value to use for converting megajoules to kilocalories. Defaults to thermochemical.
`kcal_conversion`	numeric. If RER is NULL (default), the factor to use for converting kilocalories to oxygen consumption

References

Schofield, W. N. (1985). Predicting basal metabolic rate, new standards and review of previous work. *Human nutrition. Clinical nutrition*, 39, 5-41.

Examples

```
# Get BMR for an imaginary 900-year-old person (Age is only
# used to determine which equations to use, in this case the
# equations for someone older than 60)

get_bmr(
  Sex = "M", Ht = 1.5, Wt = 80, Age = 900, equation = "both",
  method = "both", RER = 0.865, kcal_table = "both",
  MJ_conversion = c("all")
)

get_bmr(
  Sex = "M", Ht = 1.5, Wt = 80, Age = 900, MJ_conversion = "all",
  kcal_conversion = 4.86
)

get_bmr(
  Sex = "M", Ht = 1.5, Wt = 80, Age = 900, method = "FAO",
  kcal_conversion = 4.86
)
```

---

get_kcal_vo2_conversion

Retrieve conversion factors from kilocalories to oxygen consumption

---

Description

Retrieve conversion factors from kilocalories to oxygen consumption

Usage

```
get_kcal_vo2_conversion(RER, kcal_table = c("Lusk", "Peronnet", "both"))
```

Arguments

RER	numeric. The respiratory exchange ratio
kcal_table	The table to reference for converting kilocalories to oxygen consumption. See get_kcal_vo2_conversion

Details

RER values are matched to the table entries based on the minimum absolute difference. If there is a tie, the lower RER is taken.

Value

numeric vector giving the conversion factor from the specified table(s)

References

Peronnet, F., & Massicotte, D. (1991). Table of nonprotein respiratory quotient: an update. *Can J Sport Sci*, 16(1), 23-29.

Lusk, G. (1924). Analysis of the oxidation of mixtures of carbohydrate and fat: a correction. *Journal of Biological Chemistry*, 59, 41-42.

Examples

```
get_kcal_vo2_conversion(0.85, "both")
```

---

get_transition_info *Convert a set of predicted and actual activity transitions to an object that can be analyzed*

Description

Convert a set of predicted and actual activity transitions to an object that can be analyzed

Usage

```
get_transition_info(predictions, references, window_size = 1, ...)
```

Arguments

`predictions`	A dummy-coded vector of predicted transitions (1) interspersed with non-transitions (0)
`references`	A dummy-coded vector of actual (i.e., reference) transitions (1) interspersed with non-transitions (0)
`window_size`	The maximum number of indices that are allowed to separate a predicted and reference transition, before the two are considered incompatible
`...`	additional arguments passed to or from methods, not currently used

Value

an object of class `transition` that contains necessary information for evaluating the effectiveness of the predictions.

Examples

```
predictions <- sample(c(0,1), 100, TRUE, c(3, 1))
references  <- sample(c(0,1), 100, TRUE, c(4,1))
get_transition_info(predictions, references, 10)
```

`mean_sd` *Compute the mean and standard deviation of a vector, returning a formatted string containing the values as 'M +/- SD'*

Description

Compute the mean and standard deviation of a vector, returning a formatted string containing the values as 'M +/- SD'

Usage

```
mean_sd(x = NULL, MoreArgs = NULL, give_df = TRUE, ..., mean_x = NULL,
  sd_x = NULL)
```

Arguments

`x`	numeric vector of values to summarize
`MoreArgs`	named list of arguments to pass to `mean` and `sd`
`give_df`	logical. Should mean, sd, and summary string be returned in a data frame?
`...`	additional arguments passed to `format`
`mean_x`	an already-calculated mean value for `x`
`sd_x`	an already-calculated sd value for `x`

Examples

```
mean_sd(rnorm(1:100, 50))
```

`PAutilities` *PAutilities: Streamline Physical Activity Research*

Description

A collection of utilities that are useful for a broad range of tasks that are common in physical activity research. The main features (with associated functions in parentheses) are:

Details

* Bland-Altman plots (ba_plot) * Bout analysis for moderate-to-vigorous physical activity (bout_mvpa) * Formatted descriptive statistics descriptives * Demographic calculations (get_age and get_BMI_percentile) * Metabolic calculations (get_bmr, weir_equation, and get_kcal_vo2_conversion) * Analysis of bout detection algorithm performance (get_transition_info and associated methods, e.g. summary and plot)

---

plot.transition *Plot the transitions and matchings from a* transition *object*

Description

Plot the transitions and matchings from a transition object

Usage

```
## S3 method for class 'transition'
plot(x, ...)
```

Arguments

x	the object to print
...	further methods passed to or from methods, currently unused

Value

A plot of the predicted and actual transitions in a transition object, as well as the matchings between them

Examples

```
predictions <- sample(c(0,1), 100, TRUE, c(3, 1))
references  <- sample(c(0,1), 100, TRUE, c(4,1))
transitions <- get_transition_info(predictions, references, 10)
plot(transitions)
```

`weir_equation` *Calculate energy expenditure using the Weir equation*

Description

Calculate energy expenditure using the Weir equation

Usage

```
weir_equation(VO2, VCO2, epochSecs)
```

Arguments

`VO2`	Oxygen consumption
`VCO2`	Carbon dioxide production
`epochSecs`	The averaging window of the metabolic data, in seconds

Examples

```
weir_equation(3.5, 3.1, 60)
```

Index