Package ‘lifelogr’

October 13, 2022

Title Life Logging

Version 0.1.0

Description Provides a framework for combining self-data (exercise, sleep, etc.) from multiple sources (fitbit, Apple Health), creating visualizations, and experimenting on onself.

Depends R (>= 3.0.2)

License GPL (>= 2)

Encoding UTF-8

LazyData true

Collate 'EX.R' 'global_var.R' 'Person.R' 'experiments.R' 'lifelogr.R' 'shinyApp.R' 'viz_daily.R' 'viz_intraday.R' 'viz_sleep.R'

RoxygenNote 6.0.1

Imports ggplot2, shiny, dplyr, lubridate, modelr, stringr, tidyr, grDevices, lazyeval, stats, R6, fitbitScraper, tibble, plyr

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Rohisha Adke [aut],
Lisa Ann Yu [aut, cre]

Maintainer Lisa Ann Yu <lisaann.yu@gmail.com>

Repository CRAN

Date/Publication 2017-05-12 23:23:16 UTC

R topics documented:

`agg_sleep_weekday` *A function to preprocess sleep data for the Person object.*

Description

Preprocesses data to be used by the plot_sleep_weekday() function. Specifically, it calculates the sleep duration and time asleep for each day of the week (in hours).

Usage

```
agg_sleep_weekday(person)
```

Arguments

`person`	An instance of the Person class

Value

A tidy data frame with the columns weekday, measure, and hours

Examples

```
data(EX)
agg_sleep_weekday(person = EX)
```

`compare_groups` *Prints statistics on dataset, grouped by group assignments*

Description

Groups the dataset by each group assignment named in names_of_groupings (must be found in person$groupings, or passed in as a dataframe in the list of addl_grouping_assignments). Prints statistics by group.

Usage

```
compare_groups(dataset, person, names_of_groupings = NA,
  addl_grouping_assignments = NA, variables_to_compare)
```

Arguments

`dataset`	dataset from create_dataset that contains all variables and measures of interest
`person`	an instantiated Person object
`names_of_groupings`	names of groupings to test (default is groupings in person$groupings)
`addl_grouping_assignments`	list of named dataframes, where each data frame provides a mapping from a value of a specified variable to group on to the group assignment for observations with that value for that variable
`variables_to_compare`	variables to print grouped statistics on

Value

NULL - prints statistics

Examples

```
data(EX)

dataset <- create_dataset(person = EX, all_variables = list("util" = c("month"),
                                  "fitbit_daily" = c("sleepDuration", "steps",
                                  "restingHeartRate")), time_var = c("date"))

indiv_months <- data.frame("month"= c("Jan", "Feb", "Mar", "Apr", "May",
                                      "Jun", "Jul", "Aug", "Sep", "Oct",
                                      "Nov", "Dec"),
                           "group" = c(1:12))
compare_groups(dataset, person = EX,
            addl_grouping_assignments = list("indiv_months" = indiv_months),
            names_of_groupings = c("indiv_months"),
            variables_to_compare = c("steps", "restingHeartRate"))
```

`correlation` *Correlation between each variable vs each measure*

Description

Prints and returns Pearson's correlation between each variable and each measure listed. Can pass in a dataset from create_dataset, or function calls create_dataset itself.

Usage

```
correlation(dataset = NA, person, variables, measures, time_var = NA)
```

Arguments

`dataset`	dataset from create_dataset that contains all variables and measures of interest
`person`	an instantiated Person object
`variables`	list of variables in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`measures`	list of measures in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`time_var`	the time variable that variables and measures are observed in (time, date, or datetime) - only needed if dataset is not passed in

Value

Pearson's correlation between each variable and each measure

Note

‘correlation‘ uses "pairwise.complete.obs", which only computes the correlation between all complete pairs of observations.

Examples

```
data(EX)

dataset <- create_dataset(person = EX,
            all_variables = list("fitbit_daily" = c("sleepDuration",
                                                     "steps")),
            time_var = c("date"))

correlation_df <- correlation(dataset, person = EX,
                            variables = list("fitbit_daily" =
                                                  c("sleepDuration")),
                            measures = list("fitbit_daily" = c("steps")),
                           time_var = "date")
```

---

create_dataset — *Creates a dataset across data sources in a Person object*

Description

Joins all variables (across sources) by time_var into one dataframe, which is returned

Usage

```
create_dataset(person, all_variables, time_var)
```

Arguments

`person`	an instantiated Person object
`all_variables`	list of variables in person to join, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`time_var`	the time variable to join the datasets across (time, date, or datetime) as a character

Value

one dataframe with all variables in all_variables, joined by time_var

Examples

```
data(EX)
dataset <- create_dataset(person = EX,
                          all_variables = list("util" = c("month"),
                                               "fitbit_daily" = c("steps")),
                          time_var = c("date"))
```

EX — *A subset of the data for one user for about one month, from 2017-01-19 to 2017-02-17, containing* `fitbit_daily`*,* `fitbit_intraday`*, and* `util` *data frames.*

Description

A subset of the data for one user for about one month, from 2017-01-19 to 2017-02-17, containing `fitbit_daily`, `fitbit_intraday`, and `util` data frames.

Usage

```
EX
```

Format

An object of class `Person` (inherits from `R6`) of length 14.

---

experiment — *Do the specified analysis of the impact of the variables on the measure*

Description

Performs the analysis specified on the variables (X) and measures (Y).

Usage

```
experiment(person, variables, measures, analysis = c("plot", "correlation",
  "anova", "compare_groups", "regression"), time_var)
```

Arguments

`person`	an instantiated Person object
`variables`	list of variables in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`measures`	list of measures in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`analysis`	list of ways in which to analyze the relationship between each variable and each measure - options are "plot", "correlation", "anova", "compare_groups", "regression"
`time_var`	the time variable that variables and measures are observed in (time, date, or datetime)

Value

NULL - results of analysis chosen are printed

Examples

```
data(EX)
experiment(person = EX, variables = list("fitbit_daily" = c("sleepDuration"),
                                         "util" = c("day_of_week")),
                        measures = list("fitbit_daily" =
                                         c("restingHeartRate")),
                        analysis = c("plot"), time_var = c("date"))
```

`get_hr_zones` *Calculate Heart Rate Zones.*

Description

Heart Rate Zones are calculated on the basis of age. The estimated maximum heart rate is calculated as 220 - the age of the user. The peak heart rate zone is 85 cardio heart rate zone is between 70 and 84 heart rate zone is between 50 and 69

Usage

```
get_hr_zones(person)
```

Arguments

`person`	An instance of the Person class

Value

Returns a list with 3 vectors of length 2: peak, cardio, and fat_burn

See Also

https://help.fitbit.com/articles/en_US/Help_article/1565#zones

Examples

```
data(EX)
get_hr_zones(EX)
```

---

lifelogr *lifelogr: A package for life logging in R.*

Description

The lifelogr package provides a framework for creating visualizations and experimenting on onself using self-tracking health data from multiple sources. It provides an example of what a user's combined dataset might look like: `EX`.

Details

To learn more about lifelogr, start with the vignette: `browseVignettes(package = "lifelogr")`

---

lifelogrApp *Run the Shiny app.*

Description

Meant to be used as an example showcasing the visualizations. Uses the EX Person instance used throughout this package.

Usage

```
lifelogrApp()
```

Examples

```
lifelogrApp
```

l_anova — *ANOVA test to assess impact of all variables (together) upon each measure*

Description

Prints and returns ANOVA test on all variables and interactions for each measure. Can pass in a dataset from create_dataset, or function calls create_dataset itself.

Usage

```
l_anova(dataset = NA, person, variables, measures, time_var = NA)
```

Arguments

`dataset`	dataset from create_dataset that contains all variables and measures of interest
`person`	an instantiated Person object
`variables`	list of variables in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`measures`	list of measures in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`time_var`	the time variable that variables and measures are observed in (time, date, or datetime) - only needed if dataset is not passed in

Value

list of ANOVAs for each measure

Examples

```
data(EX)

dataset <- create_dataset(person = EX,
                          all_variables = list("util" = c("day_of_week"),
                     "fitbit_daily" = c("sleepDuration",
                                        "steps",
                                        "restingHeartRate")),
                                         time_var = c("date"))
all_anovas <- l_anova(dataset, person = EX, variables = list("util" = c("day_of_week"),
                                              "fitbit_daily" = c("sleepDuration",
                                              "steps")),
                      measures = list("fitbit_daily" = c("restingHeartRate")))
```

l_plot *Plot each variable vs each measure of interest*

Description

Plots each variable vs each measure listed. Can pass in a dataset from create_dataset, or function calls create_dataset itself.

Usage

```
l_plot(dataset = NA, person, variables, measures, time_var = NA)
```

Arguments

`dataset`	dataset from create_dataset that contains all variables and measures of interest
`person`	an instantiated Person object
`variables`	list of variables in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`measures`	list of measures in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`time_var`	the time variable that variables and measures are observed in (time, date, or datetime) - only needed if dataset is not passed in

Value

NULL - plots for each variable vs each measure are printed

Examples

```
data(EX)

l_plot(person = EX, variables = list("fitbit_daily" = c("sleepDuration",
                                                  "steps", "distance"),
                                     "util" = c("day_of_week", "day_type")),
       measures = list("fitbit_daily" = c("restingHeartRate")),
       time_var = c("date"))

dataset <- create_dataset(person = EX, all_variables = list(
                                           "util" = c("month"),
                                           "fitbit_daily" = c("steps")),
                                       time_var = c("date"))

l_plot(dataset, person = EX, variables = list("util" = c("month")),
        measures = list("fitbit_daily" = c("steps")))
```

`l_regression` *Performs linear regression with all variables and interactions upon each measure*

Description

Prints and returns linear regression on all variables and interactions for each measure. Can pass in a dataset from create_dataset, or function calls create_dataset itself.

Usage

```
l_regression(dataset = NA, person, variables, measures, time_var = NA)
```

Arguments

`dataset`	dataset from create_dataset that contains all variables and measures of interest
`person`	an instantiated Person object
`variables`	list of variables in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`measures`	list of measures in person of interest, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2
`time_var`	the time variable that variables and measures are observed in (time, date, or datetime) - only needed if dataset is not passed in

Value

list of linear models for each measure

Examples

```
data(EX)

dataset <- create_dataset(person = EX,
                          all_variables = list("util" = c("day_of_week"),
                      "fitbit_daily" = c("sleepDuration",
                                         "steps",
                                         "restingHeartRate")),
                                          time_var = c("date"))
all_models <- l_regression(dataset, person = EX, variables = list("util" = c("day_of_week"),
                                     "fitbit_daily" = c("sleepDuration",
                                     "steps")),
                         measures = list("fitbit_daily" = c("restingHeartRate")))
```

merge_lists *Merge a list of lists into one list*

Description

merges list of lists specifying source and variables from each source into one list

Usage

```
merge_lists(list_of_lists)
```

Arguments

`list_of_lists`	list of lists, each with structure list(source1 = c(var1, var2), source2 = c(var3, var4)) where source is a source of data as defined in a Person object, and var1 and var2 are variables from source1, while var3 and var4 are variables from source2

Value

one list, with structure list(source1 = c(var1, var2), source2 = c(var3, var4)), where variables from the same source have been grouped in that source's sublist

Examples

```
variables = list("fitbit_intraday" = c("steps"),
                 "fitbit_daily" = c("sleepDuration"),
                 "util" = c("day_of_week", "day_type", "month"))
measures = list("fitbit_daily" = c("distance", "restingHeartRate"))
all_variables <- merge_lists(list(variables, measures))
```

`Person` *A `Person` object is a complete view of an individual over a certain time period, as seen through data from multiple sources*

Description

`Person` is an object that encapsulates an individual's data over a specified date range (start and end date stored as `Date` objects. An individual consists of basic information, such as name, age, and gender (a `list` with named elements), data from their self-tracking devices such as Fitbit, Apple health, etc. (data from each source is a tibble dataframe), individual goals such as target steps (`numeric`), additional data from self-tracking apps or one's own collection system (stored as a tibble dataframe), and ways of grouping the data a user may be interested in, such as grouping by seasons, or comparing weekend to weekday behavior and health (stored as a list of named dataframes, which each contain group assignments).

Usage

```
Person
```

Format

An `R6Class` generator object

Fields

`fitbit_daily` tibble dataframe of fitbit variables (for user account info provided) observed daily. Columns include:

- date: unique for each row (date)
- datetime: includes date and time, time is an arbitrary time, which is consistent for each day (date)
- dateInForJavascriptLocalFormatting: chr
- steps: total number of steps for that day (dbl)
- distance: total distance for that day, in miles (dbl)
- distanceKm: total distance for that day, in kilometers (dbl)
- floors: total number of floors for that day (dbl)
- minutesVery: minutes 'very active' that day (dbl)
- caloriesBurned: calories (kcal) burned that day (dbl)
- caloriesIntake: calories (kcal) consumed that day, user must input this, either into this data frame or into the fitbit (dbl)
- restingHeartRate: resting heart rate in beats per minute (bpm) (dbl)
- startTime: sleeping start time for that day (chr)
- endTime: sleeping end time for that day (chr)
- startDateTime: sleeping start date and time for that day (chr)
- endDateTime: sleeping end date and time for that day (chr)

- sleepDuration: sleep duration for that day, in minutes (int)
- sleepDurationHrs: sleep duration for that day, in hours (dbl)
- minAsleep: time asleep that day, in minutes (int)
- minAsleepHrs: time asleep that day, in hours, derived from minAsleep (dbl)
- minRestlessAwake: $sleepDuration - awakeCount$ (int)
- awakeCount: int
- restlessCount: int
- awakeDuration: int
- restlessDuraton: int
- restlessProp: proportion of sleep spent restless, calculated as $restlessProp = (sleepDurationHrs - minAsleepHrs)/sleepDurationHrs * 100$ (dbl)
- sleepQualityScoreB: dbl
- sleepQualityScoreA: int
- sleepQualityGraphicPercentA: dbl
- sleepQualityGraphicPercentB: dbl
- sleepBucketTextB: one of "ok", "good", "great" (chr)
- sleepBucketTextA: one of "ok", "good", "great" (chr)
- clusters: list of chr
- breaks: list of chr

`fitbit_intraday` tibble dataframe of fitbit variables (for user account info provided) observed multiple times a day. Columns include:

- date: unique for each row (date)
- time: a combination of an arbitrary date ("1970-01-01") and the time of the observation, generally in 5 minute intervals (dttm)
- datetime: includes date from ‘date‘ and time from ‘time‘ (dttm)
- steps: number of steps in 15 minute interval (dbl)
- distance: distance traveled in 15 minute interval, in miles (dbl)
- distanceKm: distance traveled in 15 minute interval, in kilometers (dbl)
- floors: number of floors went up and down in 15 minute interval (dbl)
- activeMin: number of active minutes in 15 minute interval (dbl)
- activityLevel: hypothesized activity level, one of: "SEDENTARY", "LIGHTLY_ACTIVE", "MODERATELY_ACTIVE", or "VERY_ACTIVE" (chr)
- bpm: average heart rate in 5 minute interval (int)
- confidence: one of -1, 1, 2, or 3 (int)
- caloriesBurned: calories (kcal) burned in 5 minute interval (dbl)
- defaultZone: chr
- customZone: lgl
- weight: weight, in lbs (dbl)
- weightKg: weight, in kg (dbl)

`util` tibble dataframe that maps each date in the date range to utility information about that date Columns include:

- date: unique for each row (date)

- datetime: date from ‘date‘ and an arbitrary time (16:00:00) (dttm)
- day_of_week: day of the week, with Sun as first (ord)
- day_type: weekend or weekday (fctr)
- month: month, with Jan as first (ord)

`target_steps` the person’s target number of steps (numeric) for each day (default 10,000)

`start_date` start of user’s date range of interest (Date object)

`end_date` end of user’s date range of interest (Date object)

`user_info` provided user info, such as "age", "gender", "name" (list)

`groupings` named list of dataframes, each with two columns - a known variable, and group, with the group assignment for observations where that variable has appropriate value

`apple` tibble dataframe of user’s provided Apple Health data. These columns depend on which columns are passed in by the user. However, these columns match fitbit columns:

- datetime: dttm
- steps: Original steps data for total number of steps in 60 minutes, but divided by 4 to match fitbit steps data, which is the total number of steps in 15 minutes (dbl)
- distance: Average distance in 15 minutes in miles. Original distance data for total distance in 60 minutes, but divided by 4 to match fitbit distance data, which is the total distance in 15 minutes (dbl)
- distanceKm: Average distance in 15 minutes in km. Original distance data for total distance in 60 minutes, but divided by 4 to match fitbit distance data, which is the total distance in 15 minutes (dbl)
- floors: Average number of floors in 15 minutes. Original floors data for total number of floors in 60 minutes, but divided by 4 to match fitbit floors data, which is the total number of floors in 15 minutes(dbl)
- bpm: average heart rate for the given hour, most users will not have this data (dbl)

`addl_data` dataframe of data from another source provided by user

`addl_data2` dataframe of data from another source provided by user

Methods

`Person$new(fitbit_user_email, fitbit_user_pw, user_info = NA, apple_data_file, target_steps, addl_data,`
Creates a new `Person` with specified data, and data from provided Fitbit account. If provided, start_date and end_date must be characters with " All defaults are `NA` - user can provide sources of data of interest.

Examples

```
library("lifelogr")

group_months <- data.frame("month" = c("Jan", "Feb", "Mar", "Apr", "May",
                                       "Jun", "Jul", "Aug",
                                       "Sep", "Oct", "Nov", "Dec"),
                                       "group" = c(0, 0, 0, 1, 1, 1, 1, 1,
                                                   1, 0, 0, 0))
ash <- Person$new(user_info = list("name" = "Ash", "age" = 26,
                  "gender" = "female"),
                         target_steps = 20000,
                         group_assignments = list("group_months" = group_months),
                         start_date = "2017-03-11",
                         end_date = "2017-03-12")

## Not run:
bailey <- Person$new(fitbit_user_email = "bailey@gmail.com",
                 fitbit_user_pw = "baileypw",
                 #apple_data_file = "apple.csv",
                 start_date = "2017-03-11",
                 end_date = "2017-03-12")
## End(Not run)
```

`plot_cal` *Plot calories over time.*

Description

Prints a line plot plotting calories burned over time. If calories consumed are in the dataset, it also plots calories consumed.

Usage

```
plot_cal(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plot printed to screen

Examples

```
data(EX)
plot_cal(EX)
```

plot_d *Line graph for continuous variable(s).*

Description

A "quick-and-dirty" approach to plotting a generic line graph with default axis labels. Can plot one or more variables.

Usage

```
plot_d(person, measures)
```

Arguments

`person`	An instance of the Person class
`measures`	A character vector of length one or more indicating the variable(s) of interest. Options include: "steps", "floors", "distance", "calories", "mins_very", "rest_hr".

Value

NULL, but plot printed to screen

Examples

```
data(EX)
plot_d(EX, "steps")
plot_d(EX, c("steps", "distance"))
```

plot_daily *Plot daily health totals.*

Description

Prints one of six plots, each showing daily totals over time.

Usage

```
plot_daily(person, measure_var = "all", ...)
```

Arguments

`person`	An instance of the Person class
`measure_var`	Default is to print all six plots. Options include: "steps", "floors", "distance", "calories", "mins_very", "rest_hr", "all".
`...`	Extra arguments used to specify unit for the distance plot.

Value

NULL, but plots printed to screen

Examples

```
data(EX)
plot_daily(EX, "steps")
plot_daily(EX, "distance", "km")
```

`plot_daily_all` *Plot a series of six graphs.*

Description

Prints six plots, each showing daily totals over time: 1. Steps 2. Floors 3. Distance: in the default unit, miles 4. Calories 5. Minutes 'very active' 6. Resting heart rate

Usage

```
plot_daily_all(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plots printed to screen

Examples

```
data(EX)
plot_daily_all(EX)
```

`plot_distance` *Plot distance per day over time.*

Description

Prints a line plot plotting distance in miles or kilometers per day over time.

Usage

```
plot_distance(person, unit = "mi")
```

Arguments

`person`	An instance of the Person class
`unit`	a unit of distance, 'mi' or 'km'. The default value is 'mi'

Value

NULL, but plot printed to screen

Examples

```
data(EX)
plot_distance(EX)
plot_distance(EX, "mi")
plot_distance(EX, "km")
```

`plot_floors` *Plot number of floors per day over time.*

Description

Prints a line plot plotting number of floors per day over time.

Usage

```
plot_floors(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plot printed to screen

Examples

```
data(EX)
plot_floors(EX)
```

`plot_i` *Line graph for a single continuous variable.*

Description

Provides a "quick-and-dirty" approach to plotting a line graph for a single continuous variable using defaults for axis and title labels. Users can specify if they want to look at an aggregate of a variable over the course of a day (avg_to_get_typical_day = TRUE) or look at that variable at every interval (i.e. every 15 minutes for the entire date range).

Usage

```
plot_i(person, measure_var, avg_to_get_typical_day = TRUE)

plot_i_steps(person, avg_to_get_typical_day = TRUE)

plot_i_floors(person, avg_to_get_typical_day = TRUE)

plot_i_cal(person, avg_to_get_typical_day = TRUE)

plot_i_active_min(person, avg_to_get_typical_day = TRUE)

plot_i_hr(person, avg_to_get_typical_day = TRUE)

plot_i_hr_datetime(person)

plot_i_weight(person, avg_to_get_typical_day = TRUE, unit = "lb")
```

Arguments

`person`	An instance of the Person class
`measure_var`	character vector denoting the variables of interest. Options are one or more of: "steps", "floors", "distance", "caloriesBurned","bpm" (heart rate), "weight".
`avg_to_get_typical_day`	Logical variable "daily" for an aggregate of the variable over the course of a day, or "intraday" for the variable at every interval over the range. Default is TRUE.
`unit`	Unit of measurement for plot_i_weight(). Default is "lb", but "kb" can also be specified

Value

ggplot object

Functions

- `plot_i_steps`: Line graph for steps taken per 15 minute interval over date-time.
- `plot_i_floors`: Line graph for floors gone up per 15 minute interval over date-time.
- `plot_i_cal`: Line graph for calories burned per 15 minute interval over date-time.
- `plot_i_active_min`: Line graph for active minutes per 15 minute interval over date-time.
- `plot_i_hr`: Line graph for heart rate per 5 minute interval across a typical day or over date-time.
- `plot_i_hr_datetime`: Line graph for heart rate per 5 minute interval across a typical day.
- `plot_i_weight`: Line graph for weight over time.

See Also

`get_hr_zones`

Examples

```
data(EX)
plot_i(EX, "steps")
plot_i(EX, "distance", FALSE)
```

`plot_intraday` *Switch table to plot intraday variables.*

Description

Plot one continuous intraday variable across time. Users can specify if they want to look at an aggregate of a variable over the course of a day (avg_to_get_typical_day = TRUE) or look at that variable at every interval (i.e. every 15 minutes for the entire date range).

Usage

```
plot_intraday(person, measure_var = "all", avg_to_get_typical_day = TRUE,
  ...)
```

Arguments

`person`	An instance of the Person class
`measure_var`	Character vector of length 1 denoting the variable of interest. Options include: "steps", "floors", "distance", "caloriesBurned", "activeMin", "bpm" (heart rate), "weight". By default, all are plotted.
`avg_to_get_typical_day`	Logical vector of length 1. If TRUE, plot gives an aggregate of the variable over the course of a typical day. If FALSE, plot gives the variable at every interval over the range specified when the Person object was instantiated.
`...`	Extra arguments used to specify unit for the distance and weight plots.

Value

NULL, but plots print to screen

See Also

plot_intraday

Examples

```
data(EX)
plot_intraday(EX, "steps")
plot_intraday(EX, "distance", unit = "km")
plot_intraday(EX, "caloriesBurned", FALSE)
plot_intraday(EX, "steps", FALSE)
plot_intraday(EX, "bpm")
```

---

plot_intraday_all *Plot all intraday variables.*

Description

Plots all seven intraday variables using default settings.

Usage

```
plot_intraday_all(person, avg_to_get_typical_day = TRUE)
```

Arguments

person	An instance of the Person class.
avg_to_get_typical_day	Logical vector of length 1. If TRUE, plot gives an aggregate of the variable over the course of a typical day. If FALSE, plot gives the variable at every interval over the range specified when the Person object was instantiated.

Value

NULL, plots print to screen

See Also

plot_i

Examples

```
data(EX)
plot_intraday_all(EX)
```

`plot_i_distance` *Plot distance over time.*

Description

Plot distance over time in units of either miles or kilometers.

Usage

```
plot_i_distance(person, avg_to_get_typical_day = TRUE, unit = "mi")
```

Arguments

`person`	An instance of the Person class.
`avg_to_get_typical_day`	Logical vector of length 1. If TRUE, plot gives an aggregate of the variable over the course of a typical day. If FALSE, plot gives the variable at every interval over the range specified when the Person object was instantiated.
`unit`	The unit of distance, 'mi' by default, but can also specify 'km'

Value

NULL, but plot prints to screen.

Examples

```
data(EX)
plot_i_distance(EX, FALSE)
plot_i_distance(EX, unit = "km")
```

`plot_mins_very` *Plot minutes 'very active' over time.*

Description

Prints a line plot plotting minutes 'very active' per day over time. 'Very active' is a subjective term defined by fitbit.

Usage

```
plot_mins_very(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plot printed to screen

Examples

```
data(EX)
plot_mins_very(EX)
```

plot_rest_hr *Plot resting heart rate over time.*

Description

Prints a line plot plotting heart rate (in beats per minute) over time. According to the National Institute of Health, the average resting heart rate for persons 10 and older (including seniors) is 60 - 100. However, well-trained athletes can have resting heart rates between 40 and 60.

Usage

```
plot_rest_hr(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plot printed to screen

See Also

http://www.heart.org/HEARTORG/HealthyLiving/PhysicalActivity/FitnessBasics/Target-Heart-Rates_UCM_434341_Article.jsp#.WM3bCxiZMdU

Examples

```
data(EX)
plot_rest_hr(EX)
```

plot_sleep *Plot sleep.*

Description

Prints one of six plots: two are related to quantity of sleep, and four are related to quality of sleep 1. Sleep by day of week (bar graph) 2. Start and end of sleep period for each day in the range 3. Duration of sleep and time asleep over time 4. Proportion of time spent restless out of total sleep duration over time 5. Time spent restless over time (in minutes) 6. Sleep quality over time (subjective score, out of 100)

Usage

```
plot_sleep(person, plot_type = "all", ...)
```

Arguments

`person`	An instance of the Person class
`plot_type`	The type of plot. Options include: "by_weekday", "by_start_end_time", "by_datetime", "by_restless_prop", "by_restless_min", "by_quality". Default is to plot all six.
`...`	Extra arguments used to specify the ‘color_var‘ for the ‘by_start_end_time‘ plot

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep(person = EX)
```

plot_sleep_all *Plot a series of six sleep graphs.*

Description

Prints six plots: two are related to quantity of sleep, and four are related to quality of sleep 1. Sleep by day of week (bar graph) 2. Start and end of sleep period for each day in the range 3. Duration of sleep and time asleep over time 4. Proportion of time spent restless out of total sleep duration over time 5. Time spent restless over time (in minutes) 6. Sleep quality over time (subjective score, out of 100)

Usage

```
plot_sleep_all(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plot prints to screen

Examples

```
data(EX)
plot_sleep_all(person = EX)
```

`plot_sleep_over_time` *A function to plot sleep over time.*

Description

Returns a line plot plotting sleep over time. Includes sleep duration and time asleep (in hours).

Usage

```
plot_sleep_over_time(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep_over_time(person = EX)
```

`plot_sleep_quality` *A function to plot sleep quality over time.*

Description

Returns a line plot plotting sleep quality over time. Sleep quality is a subjective score given by Fitbit

Usage

```
plot_sleep_quality(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep_quality(person = EX)
```

`plot_sleep_restless_min` *A function to plot the minutes of restless sleep over time.*

Description

Returns a line plot plotting the length of restless sleep over time (in minutes).

Usage

```
plot_sleep_restless_min(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep_restless_min(person = EX)
```

---

plot_sleep_restless_prop

A function to plot the proportion of restless sleep over time.

Description

Returns a line plot plotting the proportion of restless sleep over time. The proportion is calculated as the difference between sleep duration and time spent asleep over sleep duration.

Usage

```
plot_sleep_restless_prop(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep_restless_prop(person = EX)
```

---

plot_sleep_start_end

A function to plot sleep each night by start time and end time.

Description

Returns a plot with start time of sleep and end time of sleep each night, colored by weekday vs. weekend.

Usage

```
plot_sleep_start_end(person, color_var = "day_type")
```

Arguments

`person`	An instance of the Person class
`color_var`	"day_type" by default for weekend/weekday, or "day_of_week" for day of week. Determines color of the lines.

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep_start_end(person = EX)
plot_sleep_start_end(person = EX, "day_of_week")
```

`plot_sleep_weekday` *A function to plot sleep by day of week.*

Description

Returns a bar graph plotting sleep by day of week (Sunday, Monday, ...).

Usage

```
plot_sleep_weekday(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plots print to screen

Examples

```
data(EX)
plot_sleep_weekday(person = EX)
```

`plot_steps` *Plot steps per day over time.*

Description

Prints a line plot plotting steps per day over time. The reference line refers to the user's target number of steps.

Usage

```
plot_steps(person)
```

Arguments

`person`	An instance of the Person class

Value

NULL, but plot printed to screen

Examples

```
data(EX)
plot_steps(EX)
```

`tidy_multi_meas_data` *Tidy daily data.*

Description

Tidy daily data with multiple measures.

Usage

```
tidy_multi_meas_data(data)
```

Arguments

`data`	Data frame or tibble with a column named 'date' and other columns of interest.

Value

Tidy tibble with the columns date, measures, and value.

Examples

```
a <- tibble::tibble(date =
         lubridate::ymd("1970-01-01", "1970-01-02", "1970-01-03"),
         sleepDurationHrs = c(7.5, 8.0, 7.9),
         minAsleepHrs = c(7.4, 7.0, 7.7))
tidy_multi_meas_data(a)
```

Index