Package 'card'

October 12, 2022

Type Package

Title Cardiovascular and Autonomic Research Design

Version 0.1.0

Description Tools that can aid in the assessment of the autonomic regulation of cardiovascular physiology. The aims of this package are to: 1) study electrocardiography (both intervals and morphology) as extensions of signal processing, 2) study circadian rhythms and how it effects autonomic physiology, 3) assess clinical risk of autonomic dysfunction on cardiovascular health through the perspective of epidemiology and causality. The analysis of circadian rhythms through cosinor analysis are built upon the methods by Cornelissen (2014) <doi:10.1186/1742-4682-11-16> and Refinetti, Cornelissen, Halberg (2014) <doi:10.1080/09291010600903692>.

License MIT + file LICENSE

URL https://github.com/asshah4/card

BugReports https://github.com/asshah4/card/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Depends R (>= 3.5.0)

Imports utils, stats, sf, lutz, magrittr, ggplot2, tidyr, dplyr, purrr, tibble, readr, lubridate, stringr, rlang, survival, broom, hardhat, recipes, data.table, Hmisc, generics, methods, ggrepel

Suggests kableExtra, plyr, stargazer, knitr, rmarkdown, covr, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Anish S. Shah [aut, cre, cph] (<https://orcid.org/0000-0002-9729-1558>)

Maintainer Anish S. Shah <asshah4@emory.edu>

Repository CRAN

Date/Publication 2020-09-03 07:52:10 UTC

R topics documented:

`augment.cosinor` *Augment data with information from a* `cosinor` *object*

Description

Augment accepts a `cosinor` model object and adds information about each observation in the dataset. This includes the predicted values in the `.fitted` column and the residuals in the `.resid` column. New columns always begin with a `.` prefix to avoid overwriting columns in original dataset.

Usage

```
## S3 method for class 'cosinor'
augment(x, ...)
```

Arguments

x	A cosinor object created by card::cosinor()
...	For extensibility

Value

a tibble object

See Also

Other cosinor: cosinor(), ggcosinor()

---

build_sequential_models

Model Building

Description

Simplify the process of building multiple models in a sequential order. This is particularly helpful in epidemiological cases of testing effect of additional parameters. Every parameter should be theoretically a part of the causal model for the exposure-outcome relationship.

Usage

```
build_sequential_models(formula, data, exposure = NULL, engine = "lm")
```

Arguments

formula	an object of class formula that shows the names of the outcomes (can be more than 1) and the names of the predictors (which should contain the exposure variable).
data	data frame or data table (or tibble) that contains the named variables
exposure	Variable that is forced to be maintained in every model as a predictor.
engine	Set the "engine" or the regression tool that will be used

Details

This is considering what is available with the modelr package and the tidymodels approach, and finding an in-between for the causality / epidemiology approach of building intentional, sequential models. Expect changes in the process, and potential future dependencies on the tidymodels appraoches.

Value

A tidy tibble of models. Each one will likely be grouped by its outcome, and then with sequential columns using increased/additive models. Each model, in a tidy format, will have two additional columns.

- `outcomes` identifies which outcome was used for the specific regression
- `covar` number of covariates used in sequence of predictors given, with exposure always being placed in position 1

Examples

```
data(geh)
f <- svg_mag + qrs_tang ~ lab_hba1c + bmi
build_sequential_models(f, data = geh)
```

`circ_center` *Center Time Around a Zeitgeiber*

Description

Based on a centering time point, shifts a vector to a "before" and "after" system to help align multiple individuals to a universal time, like the sunrise or any other appropriate zeitgeiber. Originally intended to expand upon the card::circ_sun function.

Usage

```
circ_center(times, zeitgeiber)
```

Arguments

`times`	Vector of time series. The earliest time point is presumed to be the time series onset. Built with the assumption that the duration would be approximately 24 hours (or less) to remove issues with circadian rhythms and repeat zeitgeibers (e.g. sunrise). Most importantly, the time series should be roughly equally spaced, such as 1 hour apart.
`zeitgeiber`	A single timestamp that should exist within the proposed `times`. It can be a POSIX* variable or it can just be a character of a time stamp in an HMS format. Its used to create a centering point.

Value

Vector of centered times around zeitgeiber. Function guesses units of time based on time series that is input (e.g. duration / number of events). It returns a vector of relative time in guessed units as `double`, which allows centering around the zeitgeiber (Z=0).

Examples

```
data("twins")
df <- subset(twins, patid == 7) # Single patient
times <- df$dyxtime
zeitgeiber <- as.POSIXct("2002-03-22 06:40:18", tz = "UTC")
df$zeit <- circ_center(times, zeitgeiber)
```

---

circ_compare_groups *Compare Repeated Measurements by Group*

Description

Takes data and returns a summary table of continuous variable based on a categorical variable. This summary is repeat by time groups to help describe a circadian pattern.

Usage

```
circ_compare_groups(data, x, y, time)
```

Arguments

`data`	Dataframe containing all the following variables
`x`	Continuous variable of interest (x ~ y)
`y`	Grouping variable to apply to the `cvar` (x ~ y). Must be binary for t-test, otherwise will return data set without pvalues
`time`	Name of the time-dependent variable, usually hours

Details

Applies a simple data transformation to identify the summary statistics of the data frame by the stated variables. Results in a mean, standard deviation, and standard error term. This data is also used for making a t-test based table, which can then also be graphed in card::ggcircadian.

Value

Returns a dataframe that has the time variable, the categorical variable, and the statistics (including p-value) of the continuous variable

Examples

```
data("twins")
circ_compare_groups(data = twins, x = "rDYX", y = "sad_cat", time = "hour")
```

`circ_odds` *Odds Ratio Table by Time Point*

Description

Creates an OR table for each time point of data given, initially applied to any grouping variable (particularly hour/time of day).

Usage

```
circ_odds(data, time, outcome, covar)
```

Arguments

`data`	Dataframe containing subsequent columns
`time`	Column name that contains the grouping variable of time
`outcome`	Column name that identifies the per-row outcome, binary
`covar`	Vector of independent variables names. First variable needs to be exposure.

Details

This function creates an OR table based on the covariate names supplied. It requires that there is an appropriate outcome variable selected. It performs a logistic regression. This model does not allow for conditioning variables (yet).

Value

A data frame of odds ratios

Examples

```
# Data
data(twins)

# Create odds ratio tables by hour of day for covariate of interest
ot <- circ_odds(twins, "hour", "sad_bin", "rDYX")
```

circ_rad *Convert Time to Radians*

Description

Converts a time unit to radians given a known period. Allows for vectorization.

Usage

```
circ_rad(t, period)
```

Arguments

t	vector of time units
period	period of the time units (e.g. 24 hours)

Details

This function supports other centering functions to study time series.

Value

Vector converted into radians

circ_sun *Sunrise and Sunset Times*

Description

Gets sunrise and sunset times based on date and location using the algorithm by the United States Naval Observatory. Uses the sensible default of official zenith for calculations. Requires geographic position in latitude and longitude to calculate sunrise or sunset. Uses the function lutz::tz_lookup_coords.

Usage

```
circ_sun(date, lat, lon, zenith = "official", sunset = FALSE)
```

Arguments

date	Vector of dates to calculate sun times for
lat	Latitude vector in degrees (e.g. Atlanta is 33.749), with negative values representing south. Each date must have a corresponding latitude.
lon	Longitude vector in degrees (e.g. Atlanta is -84.388), with negative values representing west

zenith	Zenith is the sun's zenith. There are several types, with different values. "official" = 90.8333 degrees, "civil" = 96 degrees, "nautical" = 102 degrees, "astronomical" = 108 degrees. They refer to the angle at which light allows for visibility, which can be affected by atmosphere and refraction.
sunset	Logical value for if sunset is wanted instead of sunrise. Default is FALSE (thus returning sunrises).

Value

Returns vector of sunrise/sunset times based on the date and location given. The time zone offset is included for the time zone represented by the latitude/longitude. The vector always returns UTC, but it has actually been corrected to the appropriate time-zone. Can use `as.character()` to strip the time zone away.

Examples

```
data("twins")
twins$lat <- 33.749
twins$lon <- -84.388

# Using latitude/longitude from Atlanta, GA, USA
twins$sunrise <- circ_sun(twins$date, twins$lat, twins$lon)
```

`cosinor` *Fit a* `cosinor`

Description

`cosinor()` fits a regression model of a time variable to a continuous outcome use trigonometric features. This approaches uses the linearization of the parameters to assess their statistics and distribution.

Usage

```
cosinor(t, ...)

## Default S3 method:
cosinor(t, ...)

## S3 method for class 'data.frame'
cosinor(t, y, tau, population = NULL, ...)

## S3 method for class 'matrix'
cosinor(t, y, tau, population = NULL, ...)

## S3 method for class 'formula'
cosinor(formula, data, tau, population = NULL, ...)
```

```
## S3 method for class 'recipe'
cosinor(t, data, tau, population = NULL, ...)
```

Arguments

`t`	Represents the *ordered* time indices that provide the positions for the cosine wave. Depending on the context: • A **data frame** of a time-based predictor/index. • A **matrix** of time-based predictor/index. • A **recipe** specifying a set of preprocessing steps created from `recipes::recipe()`.
`...`	Not currently used, but required for extensibility.
`y`	When `t` is a **data frame** or **matrix**, `y` is the outcome specified as: • A **data frame** with 1 numeric column. • A **matrix** with 1 numeric column. • A numeric **vector**.
`tau`	A vector that determines the periodicity of the time index. The number of elements in the vector determine the number of components (e.g. single versus multiple cosinor). • A **vector** with a single element = single-component cosinor, e.g. period = c(24) • A **vector** with multiple elements = multiple-component cosinor, e.g. period = c(24, 12)
`population`	Represents the population to be analyzed with a population-mean cosinor. Defaults to NULL, assuming individual cosinors are being generated. When a **recipe** or **formula** is used, `population` is specified as: • A **character** name of the column contained in `data` that contains identifiers for each subject. Every row will have a subject name which should be duplicated for each time index given. When a **data frame** or **matrix** is used, `population` is specified as: • A **vector** of the same length as `t`, with values representing each subject at the correct indices.
`formula`	A formula specifying the outcome terms on the left-hand side, and the predictor terms on the right-hand side.
`data`	When a **recipe** or **formula** is used, `data` is specified as: • A **data frame** containing both the predictors and the outcome.

Value

A `cosinor` object.

See Also

Other cosinor: `augment.cosinor()`, `ggcosinor()`

Examples

```
# Data setup
data("twins")

# Formula interface
model <- cosinor(rDYX ~ hour, twins, tau = 24)
```

`cosinor_area` *Area of Ellipse*

Description

Formulas for creating the area of the ellipse to identify confidence intervals, direction, and graphing purposes.

Usage

```
cosinor_area(object, level = 0.95, ...)
```

Arguments

`object`	Model of class `cosinor`
`level`	Confidence level requested
`...`	Not currently used, but required for extensibility.

Value

Area of potential cosinor for graphical analysis as matrix stored in a list.

`cosinor_features` *Multiple Component Cosinor Features*

Description

Extract the special/global features of a multiple component cosinor. In a multiple component model, there are specific parameters that are not within the model itself, but must be extracted from the model fit. When extracted, can be used to improve the plot of a multiple component cosinor. However, this is only possible if the cosinor is harmonic (see `details`). For single-component models, the orthophase is the same as the acrophase and the global amplitude

- Global Amplitude (Ag) = the overall amplitude is defined as half the difference between the peak and trough values
- Orthophase (Po) = the lag until the peak time
- Bathyphase (Pb) = the lag until the trough time

Usage

```
cosinor_features(object, population = TRUE, ...)
```

Arguments

`object`	Model of class `cosinor` with multiple periods
`population`	If the object is a population cosinor, should the features be calculated for the individual cosinors or for the population-cosinors. Default is TRUE. This has no effect on "Individual" cosinor objects. • If TRUE, then will calculate features for entire population. • If FALSE, then will calculate features for every individual cosinor in the population.
`...`	For extensibility

Details

These calculations can only occur if the periods of the cosinor are harmonic - as in, the longest period is a integer multiple of the smallest period (known as the fundamental frequency). Otherwise, these statistics are not accurate or interpretable.

Value

When returning the cosinor features for a single model, will return an object of class `list`. When returning the cosinor features for every individual in a population cosinor, will return an object of class `tibble`.

Examples

```
data(twins)
model <- cosinor(rDYX ~ hour, twins, c(24, 8), "patid")
results <- cosinor_features(model, population = FALSE)
head(results)
```

---

cosinor_goodness_of_fit

Goodness of Fit of Cosinor

Description

Goodness of fit of a cosinor from data that has multiple collections at different timepoints or from multiple cycles. The RSS is partitioned into pure error (SSPE) and lack of fit (SSLOF). An F-test compares the SSPE and SSLOF to detect appropriateness of model.

$$SSLOF = RSS - SSPE$$

$$SSPE = \sum_i \sum_l (Y_{il} - \overline{Y}_i)^2$$

The fitted values for each time point are:

$$\overline{Y}_i = \frac{\sum_l Y_{il}}{n_i}$$

Usage

```
cosinor_goodness_of_fit(object, level = 0.95, ...)
```

Arguments

`object`	requires cosinor model generated with card::cosinor to calculate statistics.
`level`	confidence level desired
`...`	additional parameters may be needed for extensibility

Value

f-statistic as result of goodness of fit

`cosinor_zero_amplitude` *Zero Amplitude Test*

Description

Zero amplitude test assesses how well the circadian pattern fits the data, essentially detecting the present of a rhythm to the data.

Usage

```
cosinor_zero_amplitude(object, level = 0.95)
```

Arguments

`object`	model of class `cosinor`
`level`	confidence level

Value

Returns a list of test statistics, as well prints out a report of analysis.

geh *GEH parameters in a large clinical cohort*

Description

Used in the model-building examples for repeat testing.

Usage

```
geh
```

Format

A tibble

geom_residuals *Plotting Residual of a Model*

Description

`geom_residuals` makes a diagnostic plot of residuals versus fitted data for linear models. Does not yet accept logistic models

Usage

```
geom_residuals(model)
```

Arguments

`model`	Model to be analyzed, currently only accepts linear models.

Details

Generate residuals versus fitted plot. Functions as an additional geom layer on ggplot. Models must be linear/gaussian in nature. Covariates can be included in the model.

Value

Returns a ggplot object of geom type, other layers can be added on as seen in example.

`ggcircadian` *Circadian Plot by Group*

Description

Converts the output of card::circ_compare_groups into a complex geom that is broken down by time/hour and HRV (or any other continuous variable). Each hour is then separated by the grouping variable.

Usage

```
ggcircadian(
  data,
  outcome,
  time = "hour",
  mean = "mean",
  n = "n",
  sd = "sd",
  se = "se",
  pval = "pval"
)
```

Arguments

`data`	Table generated which has time variable, categorical outcome variable, and summary statistics, including a possible column called "pval" which, if present, will document statistical significance in plot.
`outcome`	Name of categorical variable to stratify the y-axis
`time`	Name of time group variable, such as hours of day, which ends up being the x-axis
`mean, n, sd, se, pval`	Summary statistics to be included in graphics.

Details

Currently creates a ggplot that shows a error bar and point estimate of values by group (e.g. clinical status). If t-test values are available in the data frame, shows points of significance.

Value

Returns a ggplot object of geom type, other layers can be added on as seen in example.

Examples

```
# Data
data(twins)
tbl <- circ_compare_groups(twins, "rDYX", "sad_cat", "hour")

# Plot
library(ggplot2)
ggcircadian(tbl, outcome = "sad_cat") +
  labs(title = "Example") +
  scale_color_viridis_d(option = "A", begin = 0.0, end = 0.75)
```

`ggcosinor` *ggplot of cosinor model*

Description

ggplot of cosinor model that can visualize a variety of cosinor model subtypes, including single-component, multiple-component, individual, and population cosinor models, built using card::cosinor. For single component cosinor, the following values are plotted:

- M = midline estimating statistic of rhythm
- A = amplitude
- P = phi or acrophase (shift from 0 to peak)

If using a multiple-component cosinor, the terms are different. If the periods or frequencies resonate or are harmonic, then the following are calculated. If the periods are not harmonic, the values are just descriptors of the curve.

- M = midline estimating statistic of rhythm
- Ag = global amplitude, which is the distance between peak and trough (this is the same value as the amplitude from single component)
- Po = orthophase (the equivalent of the acrophase in a single component), the lag time to peak value
- Pb = bathyphase, the lag time to trough value

Usage

```
ggcosinor(object, labels = TRUE, ...)
```

Arguments

`object`	Model of class `cosinor`. If instead of a single cosinor model, multiple objects are to be plotted, can provide a list of cosinor models. Plotting multiple models simultaneously is preferred if the outcome variable is similar in scale.

labels	Logical value if annotations should be placed on plot, default = TRUE. The labels depend on the type of plot. The labels are attempted to be placed "smartly" using the ggrepel::geom_label_repel() function.
...	For extensibility. This function will use different implementations based on the type of model (single or multiple component). Attributes of the object will be passed down, or calculated on the fly.

Value

Object of class ggplot that can be layered

See Also

Other cosinor: augment.cosinor(), cosinor()

Examples

```
data(triplets)
m1 <- cosinor(rDYX ~ hour, twins, tau = 24)
m2 <- cosinor(rDYX ~ hour, twins, tau = c(24, 12))
ggcosinor(m1, labels = FALSE)
ggcosinor(m2)
ggcosinor(list(single = m1, multiple = m2))
```

---

ggellipse *Graphical Assessment of Amplitude and Acrophase*

Description

This is a ggplot-styled graphical representation of the ellipse region generated by the cosinor analysis. It requires the same data used by cosinor model to be fit with the model card::cosinor. This includes the amplitude, acrophase,

Usage

```
ggellipse(object, level = 0.95, ...)
```

Arguments

object	Requires a cosinor model to extract the correct statistics to generate the plot.
level	Confidence level for ellipse
...	Additional parameters may be needed for extensibility

Value

Object of class ggplot to help identify confidence intervals

Examples

```
data("twins")
m <- cosinor(rDYX ~ hour, twins, tau = 24)
ggellipse(m)
```

---

ggerror *Plotting Error of Models*

Description

Creates a ggplot geom that can be extended and accept other ggplot layers. Shows residual error from the regression mean for different types of regression models.

Usage

```
ggerror(model)
```

Arguments

model	Model to be analyzed. The function will detect what type of family the model is (e.g. linear = "gaussian", logistic = "binomial") and plot the appropriate type of model.

Details

Generate residuals for models. Currently accepts only linear models. Does not account for covariates yet, although may be able to do this in the future.

Value

Returns a ggplot object of geom type, other layers can be added on as seen in example.

Examples

```
data("twins")
model <- lm(beck_total ~ HR, data = subset(twins, hour == 7))
ggerror(model)
```

ggforest *Forest Plot of Hourly Odds*

Description

Creates an OR plot for each hour of data given. Its a ggplot format so additional variables, like titles, can be added in.

Usage

```
ggforest(ot, time = "time", or = "OR", lower = "Lower", upper = "Upper")
```

Arguments

ot	Odd Ratio table with the following columns.
time	Name of time variable (or "grouping" variable)
or	Name of column containing odds ratio
lower	Name of column of lower boundary of 95 percent CI
upper	Name of column of upper boundary of 95 percent CI

Details

This function creates a forest plot using the OR developed by the card::circ_odds function in this package. By default, it takes the output, which is a tibble named "ot", and will generate a forest plot based on the grouping variable (default is time of day). Original data can be restricted or the hours can be reduced).

Value

A ggplot of forest plot that can be extended. Default theme is minimal and default color scheme is viridis.

Examples

```
# Data
data(twins)
ot <- circ_odds(twins, "hour", "sad_bin", "rDYX")

# Plot
library(ggplot2)
ggforest(ot) +
  labs(title = "Example") +
  scale_color_viridis_c(option = "A")
```

heart-class heart *class*

Description

heart is an S4 class that serves as an object that holds together different components that represent the anatomical heart. A heart object allows for storing clinical information about a patient together, which may more readily allow for analytical approaches to be developed. The heart has plumbing (the coronary arteries), electricity (the nerves), and walls (cardiac chambers). The slots represent these objects.

Slots

pipes the epicardial and resistance vessels

electric the conduction system of nerves

structural the cardiac chambers and valves

hrv *Output from MATLAB HRV Toolbox*

Description

Data is a single patient data output from HRV Toolbox. It contains granular data of calculated HRV in 5-second sliding windows.

Usage

```
hrv
```

Format

An tibble data frame

hrv_linear_model *HRV Linear Modeling*

Description

hrv_linear_model Linear models for each HRV measure.

Usage

```
hrv_linear_model(data, covar, hrv, prop.weight = FALSE)
```

Arguments

data	Data frame that contains all covariates and outcomes. First column should be ID
covar	Vector names of the covariates, with first covariate being the primary exposure variable for linear regression
hrv	Vector names of the HRV measures, contained in data, that should be used. Can be generalized to any dependent variable set.
prop.weight	This is a logical value if propensity weighting should be done instead of traditional covariate adjustment. This calls for the propensity weighting function defined by card::recurrent_propensity that will generate both a PROP_SCORE column and PROP_WEIGHT column. Defaults to FALSE

Details

Linear models built with dependent variable being the HRV measures (e.g. HF, LF, SDNN, etc). Allows for covariates to be included as available.

Value

List of models with names

mims *Recurrent event sample data*

Description

Data is from a outcomes study on cardiovascular outcomes. It contains the first visit date, the last known date, and times of various events that have happened. They document death at right censoring as well. These events are non-ordered.

Usage

```
mims
```

Format

An tibble data frame

`predict.cosinor` *Predict from a* `cosinor`

Description

Predict from a `cosinor`

Usage

```
## S3 method for class 'cosinor'
predict(object, new_data, type = "numeric", ...)
```

Arguments

`object`	A `cosinor` object.
`new_data`	A data frame or matrix of new predictors.
`type`	A single character. The type of predictions to generate. Valid options are: • `"numeric"` for numeric predictions.
`...`	Not used, but required for extensibility.

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in `new_data`.

`proc_hrv_matlab` *Process Toolbox HRV*

Description

Takes the output from HRV Toolbox and converts it for analysis. Uses the package `data.table::fread()` for reading in data due to size/speed.

Usage

```
proc_hrv_matlab(loc, name, time = 3600)
```

Arguments

`loc`	Location of the folder that contains all of the patients that were analyzed by the Main_HRV_Analysis.m function from the Toolbox.
`name`	Name of the patient/ID. There should exist a folder with the name inside the `loc` folder. Inside this folder are all the Toolbox parameters and HRV results in CSV format.
`time`	Number of seconds to group the HRV data by. Defaults to 3600 seconds (which is 1 hour)

Details

The data is taken sequentially (sliding windows), and summarized over the course of certain time lengths. The data comes in a standardized pattern from the toolbox. It requires processing due to its large file sizes (e.g. 24 hours of data for a single patient can be up to 2 MB in size).

Value

Data frame of HRV summarized by the grouping variable (e.g. 3600 seconds = 1 hour). Also returns an additional column of percent missing (e.g. 20.0% missing data) by time group.

`read_hrv_matlab` *Read in Toolbox HRV*

Description

Takes the output from HRV Toolbox and reads it in for an individual patient. Unlike card::proc_hrv_matlab, this does not process or summarize the data, it just reads it the raw analysis. Uses the package `data.table::fread()` for reading in data due to size/speed.

Usage

```
read_hrv_matlab(loc, name)
```

Arguments

`loc`	Location of the folder that contains all of the patients that were analyzed by the Main_HRV_Analysis.m function from the Toolbox
`name`	Name of the patient/ID. There should exist a folder with the name inside the `loc` folder. Inside this folder are all the Toolbox parameters and HRV results in CSV format. 3600 seconds (which is 1 hour)

Details

The data is taken sequentially (sliding windows), and is not processed. It is reported and is likely a large file.

Value

Data frame of HRV summarized by the grouping variable (e.g. 3600 seconds = 1 hour). Also returns an additional column of percent missing (e.g. 20.0% missing data) by time group.

`recurrent_model_building`

Recurrent Event Sequential Model Building

Description

Takes a different covariate groups to generate several models for recurrent event survival analyses.

Usage

```
recurrent_model_building(data, covar.builds, model.type, prop.scores = NULL)
```

Arguments

`data`	Data frame that is the survival format, potentially made by the card::recur_survival_table. Has to be merged with the superset of covariates that are being tested.
`covar.builds`	This is a vector that names the individual vectors for each model, likely sequential and additive. The individual vectors contain the names of the columns in the data frame that will generate regressions.
`model.type`	Type of recurrent event data, selected from c("marginal", "pwptt", "pwpgt")
`prop.scores`	This is a vector of the names of which `covar.builds` should be performed with propensity weighting. This will call a separate function card::recurrent_propensity that will generate both a PROP_SCORE column and PROP_WEIGHT column. Optional parameter, defaults to NULL.

Details

Using the survival models in different types (e.g. marginal, PWP, etc), to create Cox regressions that are in a sequential order. Using the covariates given, will create the models on the fly. Need to specify model type and provide data in a certain format.

Value

List of models in sequential order.

recurrent_propensity *Propensity Score Weighting*

Description

recurrent_propensity Adds propensity score to any data set that is being regressed upon.

Usage

```
recurrent_propensity(data, vars)
```

Arguments

data	Data frame that contains all covariates and outcomes. First column should be ID
vars	Variables used for regression. Outcome variable must be first.

Details

Using a logistic regression, will take covariates and create propensity scores, and adds the weights. Uses the standard logistic regression to evaluate the propensity score.

Value

Returns a modified table from what was originally given with the new columns propensity scores. Essentially original df + 2 columns.

recur_followup_table *Initial and Final Visit Table*

Description

Makes a before/after dataset using a unique ID that follows patients between studies, to allow for comparison over time.

Usage

```
recur_followup_table(data, studyid, keyid, date)
```

Arguments

data	Data frame containing all clinical covariates of interest
studyid	Should be one ID for every study date/visit. Can have multiples ONLY if there were several data points gathered on a single visit (e.g. heart rate measured multiple times on the same day).
keyid	Should be the ID that corresponds to each studyid throughout each visit
date	Name of column containing the date of each visit

Details

Currently functions by taking two input IDs, one being a ID that is the same between studies (a true key ID) and an ID that is unique to that study itself. It will arrange by dates, and and slice data into an initial visit and the most recent visit. Each row should have a KEY ID and a STUDY ID. The data is in a long format, such that the STUDY IDs are unique / not duplicated.

Value

Returns list of initial and most recent data sets. These can easily be merged after with any naming nomenclature as chosen, or with any merging keys as chosen (in case there are several merging variables, like keyid + hour of day for circadian data).

---

recur_summary *Recurrent Event Summary Table by Group*

Description

`recur_summary` Creates a table with summary of recurrent events

Usage

```
recur_summary(data, covar)
```

Arguments

`data`	Recurrent event data in marginal format. There must be an ID column. Must merge in the covariate of interest into this data set.
`covar`	Name of covariate of interest to serve as grouping variable.

Details

This function allows for taking the output of card::recur_survival_table marginal format repeat event data, and creates a summary table that describes the number of events by strata/event.

Value

Summary table by grouping variable, can be placed into a latex environment with kableExtra. Assumes that death events may be present when most recent non-EVENT has status 1.

`recur_survival_table` *Recurrent Survival Data Format*

Description

Reformats recurrent event data (wide) into different models for survival analysis, but can also be used for simple survival analysis tables as well. Importantly, for large datasets, this function will show significant slow-down since it uses an intuitive approach on defining the datasets. Future iterations will create a vectorized approach that should provide performance speed-ups.

- For recurrent events, the final censoring event can include death, or can be ignored if its not considered a failure event.
- For simple survival analysis, death censoring should be left as NULL, and the event (e.g. "date_of_death"), should be used as a single `event.dates` parameter. The function will do the rest.

Usage

```
recur_survival_table(
  data,
  id,
  first,
  last,
  event.dates,
  model.type,
  death = NULL
)
```

Arguments

`data`	A dataframe containing the subsequent parameters
`id`	Column in dataframe that contains unique IDs for each row
`first`	Column with left/enrollment dates
`last`	Column with right/censoring time point, or last contact
`event.dates`	Vector of columns that contain event dates
`model.type`	Character/string = c("marginal", "pwptt", "pwpgt")
`death`	Column created for if death is known (0 or 1), original dataframe (e.g. can add column of zeroes PRN). Death defaults to null for intermediate calculations otherwise.

Details

This function takes every data event date, and creates several types of recurrent event tables. It orders the data chronologically for repeat events. Currently does marginal and conditional A and B models. The large

Value

A data frame organized into a survival table format

Examples

```
# Data
data("mims")

# Parameters
id <- "patid"
first <- "first_visit_date_bl"
last <- "ldka"
event.dates <- c("mi_date_1", "mi_date_2", "mi_date_3")
model.type <- "marginal"
death <- "DEATH_CV_YN"

# Run analysis
tbl <- recur_survival_table(
  mims, id, first, last, event.dates, model.type, death
)
```

---

tidy.cosinor *Tidy a(n) cosinor object*

Description

Tidy summarizes information about the components of a `cosinor` model.

Usage

```
## S3 method for class 'cosinor'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `cosinor` object created by `card::cosinor()`
`conf.int`	Logical indicating whether or not to include confidence interval in tidied output
`conf.level`	The confidence level to use if `conf.int = TRUE`. Must be between 0 and 1, with default to 0.95 (the 95% confidence interval).
`...`	For extensibility

Details

`cosinor` objects do not necessarily have a T-statistic as the standard error is not based on a mean value, but form a joint-confidence interval. The standard error is generated using Taylor series expansion as the object is a subspecies of harmonic regressions.

Value

a `tibble` object

`triplets` *Hourly time series data with clinical covariates*

Description

Clinical data is also available for visualization and comparison. Other HRV measures are used here for comparison and testing out functions.

Usage

```
triplets
```

Format

An tibble data frame

`twins` *Hourly time series data with clinical covariates*

Description

Data is from an algorithm that generates a summary HRV measure using the Poincare phase-space plot, generated from kurtoses of the x and y axis. Clinical data is also available for visualization and comparison. There are repeat rows for each hour that Dyx was taken.

Usage

```
twins
```

Format

An tibble data frame

zipcode *Zipcodes with Associated Latitude and Longitude*

Description

This is a dataset from the archived/orphaned zipcode package.

Usage

```
zipcode
```

Format

A data frame with character vector zipcodes and latitude/longitude

Index