Package 'dropR'

July 22, 2025

Type Package

Version 1.0.3

Title Dropout Analysis by Condition

Description Analysis and visualization of dropout between conditions in surveys and (online) experiments. Features include computation of dropout statistics, comparing dropout between conditions (e.g. Chi square), analyzing survival (e.g. Kaplan-Meier estimation), comparing conditions with the most different rates of dropout (Kolmogorov-Smirnov) and visualizing the result of each in designated plotting functions. Sources: Andrea Frick, Marie-Terese Baechtiger & Ulf-Dietrich Reips (2001) <https://www.researchgate.net/publication/223956222_Financial_incentives_personal_information_and_drop-out_in_online_studies>; Ulf-Dietrich Reips (2002) ``Standards for Internet-Based Experimenting" <doi:10.1027//1618-3169.49.4.243>.

Depends R (>= 3.0.0)

Imports shiny, ggplot2, data.table, survival, lifecycle

Suggests DT, shinydashboard, knitr, rmarkdown, kableExtra

Date 2024-06-24

License GPL (>= 3)

LazyData true

RoxygenNote 7.3.1

URL https://iscience-kn.github.io/dropR/, https://github.com/iscience-kn/dropR

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Annika Tave Overlander [aut, cre], Matthias Bannert [aut], Ulf-Dietrich Reips [ctb]

Maintainer Annika Tave Overlander <annika-tave.overlander@uni.kn>

Repository CRAN

Date/Publication 2024-07-03 16:20:02 UTC

Contents

`add_dropout_idx` *Add Dropout Index to a Data.Frame*

Description

Find drop out positions in a data.frame that contains multiple questions that had been asked sequentially. This function adds the Dropout Index variable `do_idx` to the data.frame which is necessary for further analyses of dropout.

Use this function *first* to prepare your dropout analysis. Then, keep going by creating the dropout statistics using `compute_stats()`.

Usage

```
add_dropout_idx(df, q_pos)
```

Arguments

`df`	data.frame containing `NA`s
`q_pos`	numeric range of columns that contain question items

Details

Importantly, this function will start counting missing data at the end of the data frame. Any missing data which is somewhere in between, i.e. a single item that was skipped or forgotten will not be counted as dropout. The function will identify sequences of missing data that go until the end of the data frame and add the number of the last answered question in `do_idx`.

Therefore, the variables must be in the order that they were asked, otherwise analyses will not be valid.

Value

Returns original data frame with column do_idx added.

Source

R/add_dropout_idx.R

See Also

compute_stats() which is usually the next step for dropout analysis.

Examples

```
dropout <- add_dropout_idx(dropRdemo, 3:54)
```

---

compute_stats *Compute Dropout Statistics*

Description

This is the *second step* in conducting dropout analysis with dropR. Outputs all necessary statistics to analyze and visualize dropout, such as the sample size N of the data (and in each condition if selected), cumulative dropout and remaining participants in absolute numbers and percent. If no experimental condition is added, the stats are only calculated for the whole data in total.

Usage

```
compute_stats(df, by_cond = "None", no_of_vars)
```

Arguments

df	data.frame containing variable do_idx from add_dropout_idx()
by_cond	character name of condition variable in the data, defaults to 'None' to output total statistics.
no_of_vars	numeric number of variables that contain questions

Value

A data frame with 6 columns (q_idx, condition, cs, N, remain, pct_remain) and as many rows as questions in original data (for overall data and if conditions selected again for each condition).

Examples

```
do_stats <- compute_stats(df = add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)
```

do_chisq *Compute Chisq-Test Given a Question Position*

Description

This function performs a chi-squared contingency table test on dropout for a given question in the data. Note that the input data should be in the format as computed by `compute_stats()`. The test can be performed on either all conditions (excluding total) or on select conditions.

Usage

```
do_chisq(do_stats, chisq_question, sel_cond_chisq, p_sim = TRUE)
```

Arguments

`do_stats`	data.frame of dropout statistics as computed by `compute_stats()`.
`chisq_question`	numeric Which question to compare dropout at.
`sel_cond_chisq`	vector (same class as in conditions variable in original data set) selected conditions.
`p_sim`	boolean Simulate p value parameter (by Monte Carlo simulation)? Defaults to `TRUE`.

Value

Returns test results from chisq.test between experimental conditions at defined question.

See Also

`add_dropout_idx()` and `compute_stats()` which are necessary for the proper data structure.

Examples

```
do_stats <- compute_stats(add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)

do_chisq(do_stats, 47, c(12, 22), TRUE)
```

do_kpm *Kaplan-Meier Survival Estimation*

Description

This function needs a data set with a dropout index added by `add_dropout_idx()`. The `do_kpm` function performs survival analysis with Kaplan-Meier Estimation and returns a list containing survival steps, the original data frame, and the model fit type. The function can fit the survival model either for the entire data set or separately by a specified condition column.

Usage

```
do_kpm(df, condition_col = "experimental_condition", model_fit = "total")
```

Arguments

`df`	data set with `do_idx` added by `add_dropout_idx()`
`condition_col`	character denoting the experimental conditions to model
`model_fit`	character Should be either "total" for a total model or "conditions"

Value

Returns a list containing `steps` (survival steps extracted from the fitted models), `d` (the original data frame), and `model_fit` (the model fit type).

See Also

`survival::Surv()` used to fit survival object.

Examples

```
demo_kpm <- do_kpm(df = add_dropout_idx(dropRdemo, 3:54),
condition_col = "experimental_condition",
model_fit = "total")

head(demo_kpm$steps)
```

do_ks *Compute Kolmogorov-Smirnov Test for most extreme conditions*

Description

This test is used for survival analysis between the most extreme conditions, so the ones with the most different rates of dropout. This function automatically prepares your data and runs `stats::ks.test()` on it.

Usage

```
do_ks(do_stats, question)
```

Arguments

`do_stats`	A data frame made from `compute_stats()`, containing information on the percent remaining per question per condition
`question`	Index of question to be included in analysis, commonly the last question of the survey.

Value

Returns result of Kolmogorov-Smirnoff test including which conditions have the most different dropout rates.

Examples

```
do_stats <- compute_stats(df = add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)

do_ks(do_stats, 52)
```

do_or_table *Dropout Odds Ratio Table*

Description

This function calculates an Odds Ratio table at a given question for selected experimental conditions. It needs data in the format as created by `compute_stats()` as input.

Usage

```
do_or_table(do_stats, chisq_question, sel_cond_chisq)
```

Arguments

do_stats	data.frame statistics table as computed by compute_stats().
chisq_question	numeric Which question to calculate the OR table for
sel_cond_chisq	character vector naming the experimental conditions to compare

Value

Returns a Matrix containing the Odds Ratios of dropout between all selected conditions.

See Also

compute_stats()

Examples

```
do_stats <- compute_stats(df = add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)

do_or_table(do_stats, chisq_question = 51, sel_cond_chisq = c("11", "12", "21", "22"))
```

---

do_steps *Calculate Steps for Uneven Data Points*

Description

The do_steps function calculates steps for data points represented by numbers of questions from the original experimental or survey data in x and remaining percent of participants in y.

Usage

```
do_steps(x, y, return_df = TRUE)
```

Arguments

x	Numeric vector representing the question numbers
y	Numeric vector representing the remaining percent of participants
return_df	Logical. If TRUE, the function returns a data frame; otherwise, it returns a list.

Details

Due to the nature of dropout/ survival data, step functions are necessary to accurately depict participants remaining. Dropout data includes the time until the event (a.k.a. dropout at a certain question or time), so that changes in remaining participants are discrete rather than continuous. This means that changes in survival probability occur at specific points and are better represented as steps than as a continuum.

Value

Returns a data frame or a list containing the modified x and y values.

Examples

```
x <- c(1, 2, 3, 4, 5)
y <- c(100, 100, 95, 90, 85)
do_steps(x, y)

# Using the example dataset dropRdemo

do_stats <- compute_stats(df = add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)

tot_stats <- do_stats[do_stats$condition == "total", ]
do_steps(tot_stats$q_idx, tot_stats$pct_remain)
```

---

dropRdemo *Demo Dataset for Dropout in an Online Survey*

Description

Simulated demo data set for dropout in a survey.

Format

A data frame with 246 rows and 54 variables (in the order they were presented in the fictional survey).

obs_id Observation ID

experimental_condition experimental condition

vi_1 item 1

vi_2 item 2

vi_3 item 3

vi_4 item 4

vi_5 item 5

vi_6 item 6

vi_7 item 7

vi_8 item 8

vi_9 item 9

vi_10 item 10

vi_11 item 11

vi_12 item 12
vi_13 item 13
vi_14 item 14
vi_15 item 15
vi_16 item 16
vi_17 item 17
vi_18 item 18
vi_19 item 19
vi_20 item 20
vi_21 item 21
vi_22 item 22
vi_23 item 23
vi_24 item 24
vi_25 item 25
vi_26 item 26
vi_27 item 27
vi_28 item 28
vi_29 item 29
vi_30 item 30
vi_31 item 31
vi_32 item 32
vi_33 item 33
vi_34 item 34
vi_35 item 35
vi_36 item 36
vi_37 item 37
vi_38 item 38
vi_39 item 39
vi_40 item 40
vi_41 item 41
vi_42 item 42
vi_43 item 43
vi_44 item 44
vi_45 item 45
vi_46 item 46
vi_47 item 47
vi_48 item 48
vi_49 item 49
vi_50 item 50
vi_51 item 51
vi_52 item 52

Source

dropRdemo Demo data for dropout.

`get_odds` *Compute Odds From Probabilities*

Description

Compute odds from probabilities. The function is vectorized and can handle a vector of probabilities, e.g. remaining percent of participants as calculated by `compute_stats()`.

Usage

```
get_odds(p)
```

Arguments

p	vector of probabilities. May not be larger than 1 or smaller than zero.

Value

Returns numerical vector of the same length as original input reflecting the odds.

Examples

```
get_odds(0.7)
get_odds(c(0.7, 0.2))
```

`get_odds_ratio` *Compute Odds Ratio*

Description

Computes odds ratio given two probabilities. In this package, the function can be used to compare the percentages of remaining participants between two conditions at a time.

Usage

```
get_odds_ratio(a, b)
```

Arguments

a	numeric probability value between 0 and 1.
b	numeric probability value between 0 and 1.

Value

Returns numerical vector of the same length as original input reflecting the Odds Ratio (OR).

See Also

`get_odds()`, as this is the basis for calculation.

Examples

```
get_odds_ratio(0.7, 0.6)
```

`get_steps_by_cond` *Get Steps Data by Condition*

Description

The `get_steps_by_cond` function calculates steps data based on survival model results. This utility function is used inside the `do_kpm()` function of `dropR`.

Usage

```
get_steps_by_cond(sfit, condition = NULL)
```

Arguments

`sfit`	An object representing survival model results (e.g., from a Kaplan-Meier model).
`condition`	Optional. An experimental condition to include in the output data frame, defaults to `NULL`.

Value

Returns a data frame containing the steps data, including time, survival estimates, upper confidence bounds, and lower confidence bounds.

See Also

`do_kpm()`

get_survdiff *Test Survival Curve Differences*

Description

This function compares survival curves as modeled with do_kpm(). It outputs a contingency table and a Chisq measure of difference.

Usage

```
get_survdiff(kds, cond, test_type)
```

Arguments

kds	data set of a survival model such as do_kpm()
cond	character of experimental condition variable in the data
test_type	numeric (0 or 1) parameter that controls the type of test (0 means rho = 0; log-rank, 1 means rho = 1; Peto & Peto Wilcox)

Value

Returns survival test results as called from survival::survdiff().

Examples

```
kpm_est <- do_kpm(add_dropout_idx(dropRdemo, 3:54))
get_survdiff(kpm_est$d, "experimental_condition", 0)
get_survdiff(kpm_est$d, "experimental_condition", 1)
```

plot_do_curve *Plot Dropout Curves*

Description

This functions uses ggplot2to create drop out curves. Please note that you should use add_dropout_idx() and compute_stats() on your data before running this function as it needs a certain data structure and variables to work properly.

Usage

```
plot_do_curve(
  do_stats,
  linetypes = TRUE,
  stroke_width = 1,
  full_scale = TRUE,
  show_points = FALSE,
  show_confbands = FALSE,
  color_palette = "color_blind"
)
```

Arguments

do_stats	data.frame containing dropout statistics table computed by compute_stats(). Make sure your do_stats table contains a q_idx column indexing all question-items sequentially.
linetypes	boolean Should different line types be used? Defaults to TRUE.
stroke_width	numeric stroke width, defaults to 1.
full_scale	boolean Should y axis range from 0 to 100? Defaults to TRUE, FALSE cuts off at min percent remaining (>0).
show_points	boolean Should dropout curves show individual data points? Defaults to FALSE.
show_confbands	boolean Should there be confidence bands added to the plot? Defaults to FALSE.
color_palette	character indicating which color palette to use. Defaults to 'color_blind', alternatively choose 'gray' or 'default' for the ggplot2 default colors.

Value

Returns a ggplot object containing the dropout curve plot. Using the Shiny App version of dropR, this plot can easily be downloaded in different formats.

See Also

add_dropout_idx() and compute_stats() which are necessary for the proper data structure.

Examples

```
do_stats <- compute_stats(add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)

plot_do_curve(do_stats)
```

plot_do_kpm *Plot a Kaplan Meier Survival Estimation*

Description

The plot_do_kpm function generates a Kaplan-Meier survival plot based on the output from the do_kpm() function. It allows for customization of conditions to display, confidence intervals, color palettes, and y-axis scaling.

Usage

```
plot_do_kpm(
  kds,
  sel_conds = c("11", "12", "21", "22"),
  kpm_ci = TRUE,
  full_scale_kpm = FALSE,
  color_palette_kp = "color_blind"
)
```

Arguments

kds	list object as modeled by do_kpm()
sel_conds	character Which experimental conditions to plot.
kpm_ci	boolean Should there be confidence bands in the plot? Defaults to TRUE.
full_scale_kpm	boolean Should the Y axis show the full range from 0 to 100? Defaults to FALSE.
color_palette_kp	character indicating which color palette to use. Defaults to 'color_blind', alternatively choose 'gray' for gray scale values or 'default' for the ggplot2 default colors.

Value

Returns a ggplot object containing the Kaplan-Meier survival plot. Using the Shiny App version of dropR, this plot can easily be downloaded in different formats.

Examples

```
plot_do_kpm(do_kpm(d = add_dropout_idx(dropRdemo, 3:54),
condition_col = "experimental_condition",
model_fit = "total"))

plot_do_kpm(do_kpm(d = add_dropout_idx(dropRdemo, 3:54),
condition_col = "experimental_condition",
model_fit = "conditions"), sel_conds = c("11", "12", "21", "22"))
```

plot_do_ks *Plot Most Extreme Conditions to Visualize Kolmogorov-Smirnov Test Results*

Description

With this function, you can easily plot the most extreme conditions, a.k.a. those with the most different dropout rates at a certain question. You need to define that question in the function call of `do_ks()` already, or just call that function directly inside the plot function.

Usage

```
plot_do_ks(
  do_stats,
  ks,
  linetypes = FALSE,
  show_confbands = FALSE,
  color_palette = c("#E69F00", "#CC79A7")
)
```

Arguments

do_stats	data.frame containing dropout statistics table computed by `compute_stats()`. Make sure your do_stats table contains a q_idx column indexing all question-items sequentially.
ks	List of results from the `do_ks()` function coding most extreme dropout conditions
linetypes	boolean Should different line types be used? Defaults to FALSE.
show_confbands	boolean Should there be confidence bands added to the plot? Defaults to FALSE.
color_palette	character indicating which color palette to use. Defaults to color blind friendly values, alternatively choose 'gray' or create your own palette with two colors, e.g. using R `colors()` or HEX-values

Value

Returns a `ggplot` object containing the survival curve plot of the most extreme dropout conditions. Using the Shiny App version of dropR, this plot can easily be downloaded in different formats.

See Also

`compute_stats()`, `do_ks()`

Examples

```
do_stats <- compute_stats(add_dropout_idx(dropRdemo, 3:54),
by_cond = "experimental_condition",
no_of_vars = 52)

ks <- do_ks(do_stats, 52)

plot_do_ks(do_stats, ks, color_palette = "gray")

# ... or call the do_ks() function directly inside the plotting function
plot_do_ks(do_stats, do_ks(do_stats, 30))

plot_do_ks(do_stats, ks, linetypes = TRUE,
show_confbands = TRUE, color_palette = c("red", "violet"))
```

`start_app` *Start the dropR Shiny App*

Description

Starts the interactive web application to use dropR in your web browser. Make sure to use Google Chrome or Firefox for best experience.

Usage

```
start_app()
```

Details

The app will give less experienced R users or statisticians a good overview of how to conduct dropout analysis. For more experienced analysts, it can still be very helpful in guiding how to use the package as there are some steps that should be taken in order, which is outlined in the app (as well as function documentation).

Value

No return value; starts the shiny app as a helper to get started with dropout analysis. All app procedures are available as functions.

Index