Package 'PredTest'

May 7, 2026

Title Preparing Data For, and Calculating the Prediction Test

Version 0.1.0

Description Global hypothesis tests combine information across multiple endpoints to test a single hypothesis. The prediction test is a recently proposed global hypothesis test with good performance for small sample sizes and many endpoints of interest. The test is also flexible in the types and combinations of expected results across the individual endpoints. This package provides functions for data processing and calculation of the prediction test.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests devtools, ggplot2, kableExtra, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5.0)

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Richard Vargas [aut] (ORCID: <https://orcid.org/0009-0008-4977-8776>), Neal Montgomery [aut, cre] (ORCID: <https://orcid.org/0000-0002-1423-0590>)

Maintainer Neal Montgomery <rmontgomery@kumc.edu>

Repository CRAN

Date/Publication 2024-09-16 09:00:06 UTC

Contents

`adjusted_example` *Adjusted Example Dataset*

Description

A simulated dataset demonstrating group differences in three variables. This dataset includes two groups and one covariate, `sex`.

Usage

```
adjusted_example
```

Format

A data frame with 20 rows and 5 variables:

group A binary factor indicating group membership: `0` for control and `1` for treatment.

sex A binary factor indicating sex: `0` for male and `1` for female.

v1 A numeric vector representing the first variable.

v2 A numeric vector representing the second variable.

v3 A numeric vector representing the third variable.

Examples

```
data(adjusted_example)
head(adjusted_example)
```

`create_difference_vector` *Create a Difference Vector*

Description

This function calculates the difference between two groups of data based on a specified location measure (median or mean).

Usage

```
create_difference_vector(grp_1_data, grp_2_data, location = "median")
```

Arguments

`grp_1_data`	A data frame where all columns are numeric, representing the first group of data.
`grp_2_data`	A data frame where all columns are numeric, representing the second group of data.
`location`	A string specifying the location measure to use for calculating differences. Must be either 'median' or 'mean'.

Details

The function checks if the specified location measure is valid ('median' or 'mean'). It also checks if both groups of data are numeric and if they have the same size and column variables. Based on the location measure, it calculates the differences and returns them as a numeric vector.

Value

A numeric vector representing the differences between the second group's location measure and the first group's location measure for each column.

Examples

```
df_1 <- data.frame(v1 = c(1, 2, 100),v2 = c(4, 5, 6))
df_2 <- data.frame(v1 = c(7, 6, 5),v2 = c(4, 3, 2))

create_difference_vector(df_2, df_1, 'median')
create_difference_vector(df_2, df_1, 'mean')
```

`filter_by_group_var` *Filter by Group Variable*

Description

This function takes a data frame with two groups and splits them by a group identifier and specified column variables.

Usage

```
filter_by_group_var(df, grp_var, grp_1, grp_2, vars)
```

Arguments

`df`	A data frame which must have a column to identify two different groups.
`grp_var`	The column with the two groups, e.g., 'Treatment'.
`grp_1`	The first group identifier in the `grp_var` column, e.g., 'control'.
`grp_2`	The second group identifier in the `grp_var` column, e.g., 'drug_a'.
`vars`	The column variables the researcher is interested in. The researcher can subset the columns instead of using all potential column variables.

Details

This function checks if the input data frame, group variable, and column variables are valid. It ensures that the specified groups exist within the group variable column. The function then filters the data for each group and returns a list containing the filtered data frames.

Value

A list of two data frames that are subsets of the original data frame, separated by their group status.

Examples

```
# Load example data
data("group_data_example")

# Use the function to filter by group
result <- filter_by_group_var(df=group_data_example, grp_var="group",
grp_1 ='placebo',grp_2 ='drug',vars=c("v1", "v2"))
print(result$group_1)
print(result$group_2)
```

`filter_by_time_var` *Filter by Time Variable*

Description

This function creates two groups based on their ID and separates them by a time variable. Each ID will be present in each subset exactly once. The subsets will contain the same variables specified by the user.

Usage

```
filter_by_time_var(df, id, time_var, pre, post, vars)
```

Arguments

`df`	A data frame which must have a column to identify two different groups by ID.
`id`	The column variable that will contain all of the IDs that show up twice in the data set.
`time_var`	The column variable that designates the time of the data, e.g., it may be a 0 if data was collected on the first day of treatment and a 6 if data is collected six days later.
`pre`	The value in the `time_var` column that indicates the 'pre' time point.
`post`	The value in the `time_var` column that indicates the 'post' time point.
`vars`	The column variables the researcher is interested in. The researcher can subset the columns instead of using all potential column variables.

Details

This function checks if the input data frame, ID, time variable, and column variablesare valid. It ensures that the specified pre and post values exist within the time variable column. The function then filters the data for each time point, orders the IDs, and returns a list containing the filtered data frames.

Value

A list of two data frames that are subsets of the original data frame, separated by their `time_var` status. The data frames will have the same size in rows.

Examples

```
# Load example data
data("pre_post_data_example")

# Use the function to filter by time variable
result <- filter_by_time_var(pre_post_data_example, id = "ID",
time_var = "time", pre = 0, post = 12, vars = c("v1", "v2"))
print(result$pre)
```

```
print(result$post)
```

`get_results_vector` *Get Results Vector*

Description

This function receives user input on the hypothesis of the experiment results and informs the user if the hypotheses were correct. It can handle hypotheses of 'increase', 'decrease', or 'different', and performs appropriate statistical tests.

Usage

```
get_results_vector(
  hypothesis,
  differences,
  diff_method = "wilcoxon",
  grp_a = NULL,
  grp_b = NULL,
  phi_0 = 0.5
)
```

Arguments

`hypothesis`	A string or a vector of strings where the string or all elements of the vector are in the set of {'decrease', 'increase', 'different'}. If it's a string, it will be converted to a vector of the same length as `differences` with all elements being the valid string.
`differences`	A numeric vector representing the differences between two groups.
`diff_method`	A string specifying the method to use for testing 'different' hypotheses. Valid options are 'wilcoxon' or 't'. Defaults to 'wilcoxon'.
`grp_a`	A data frame representing the first group for testing 'different' hypotheses. This argument is only needed if 'different' is part of the hypothesis.
`grp_b`	A data frame representing the second group for testing 'different' hypotheses. This argument is only needed if 'different' is part of the hypothesis.
`phi_0`	A numeric value on the interval (0, 1) representing the decision rule threshold for the p-value. Defaults to 0.5.

Details

The function checks if the input hypotheses are valid, and performs the necessary statistical tests to determine if the hypotheses were correct. It handles 'increase' and 'decrease' hypotheses by comparing differences, and 'different' hypotheses by performing either a Wilcoxon signed-rank test or a paired t-test.

Value

A numeric vector of 0s and 1s. If the hypothesis was incorrect, a 0 is returned. If the hypothesis was correct, a 1 is returned.

Examples

```
df_1 <- data.frame(v1 = c(1, 2, -100), v2 = c(40, 5, 6))
df_2 <- data.frame(v1 = c(7, 6, 5), v2 = c(4, 3, 2))

differences <- create_difference_vector(df_2, df_1)

# using singular increase
get_results_vector(hypothesis = 'increase', differences = differences)

# using 'different' hypothesis and a pre post scenario
get_results_vector(hypothesis = c('increase', 'different'),
differences = differences, grp_a = df_1, grp_b = df_2, phi_0 = 0.05)
```

---

group_cog_data *Group Cognitive Data*

Description

A dataset representing cognitive scores for control and treatment groups, with various cognitive and demographic variables.

Usage

```
group_cog_data
```

Format

A data frame with 20 rows and 20 variables:

group.factor A factor indicating group membership: `Control` or `ESKD` (End-Stage Kidney Disease).

mean_suv A numeric vector representing the mean SUV (Standard Uptake Value).

blind_moca_uncorrected A numeric vector representing uncorrected MOCA (Montreal Cognitive Assessment) scores.

craft_verbatim A numeric vector representing scores on the Craft Verbatim memory test.

craft_delay_verbatim A numeric vector representing delayed scores on the Craft Verbatim memory test.

number_span_forward A numeric vector representing forward number span scores.

number_span_backward A numeric vector representing backward number span scores.

fluency_f_words_correct A numeric vector representing the number of correct F words in a verbal fluency test.

oral_trail_part_a A numeric vector representing scores on the oral trail making test part A.

oral_trail_part_b A numeric vector representing scores on the oral trail making test part B.

fluency_animals A numeric vector representing the number of animal names listed in a verbal fluency test.

fluency_vegetables A numeric vector representing the number of vegetable names listed in a verbal fluency test.

verbal_naming_no_cue A numeric vector representing scores on a verbal naming test without cues.

age A numeric vector representing the age of each subject.

Examples

```
data(group_cog_data)
head(group_cog_data)
```

`group_data_example` *Group Data Example*

Description

A simulated dataset showing group differences across four variables. The dataset is divided into two groups: `placebo` and `drug`.

Usage

```
group_data_example
```

Format

A data frame with 30 rows and 5 variables:

group A factor indicating group membership: `placebo` or `drug`.

v1 A numeric vector representing the first variable.

v2 A numeric vector representing the second variable.

v3 A numeric vector representing the third variable.

v4 A numeric vector representing the fourth variable.

Examples

```
data(group_data_example)
head(group_data_example)
```

pred_adjusted *Adjusted Predictions Based on Group Comparisons*

Description

This function calculates adjusted predictions for variables of interest, taking into account covariates and group comparisons. It then returns whether the results align with the hypothesized direction of effects.

Usage

```
pred_adjusted(dataset, hypothesis, vars, covariates, group, ref)
```

Arguments

dataset	A data frame containing the data to be analyzed.
hypothesis	A string or vector of strings containing either 'increase' or 'decrease', indicating the expected direction of the effect.
vars	A vector of variable names in the dataset that are the outcomes of interest. These must be numeric columns.
covariates	A vector of covariates to include in the model. These must be numeric columns in the dataset.
group	The name of the grouping variable in the dataset. This must be a column in the dataset and should not overlap with vars or covariates.
ref	The reference category within the group variable. This must be a value present in the group column.

Value

A list with two elements:

results A vector indicating whether each hypothesis was correct (1 for correct, 0 for incorrect).

weights A vector of weights corresponding to each variable in vars, calculated from the correlation matrix.

Examples

```
data("group_cog_data")
data("adjusted_example")

# simple example
pred_adjusted(adjusted_example, c("decrease", "increase"),
c('v1', 'v2'), 'sex', "group", 0)

# simulated example
pred_adjusted(dataset = group_cog_data, hypothesis = "decrease",
vars = c('craft_verbatim', 'fluency_f_words_correct'),
```

```
covariates = c('number_span_forward', 'number_span_backward'),
group = "group.factor", ref = "Control")
```

`pred_results` *Predictive Results Wrapper Function*

Description

This function is a wrapper that conditionally handles filtering by group or time, calculates the difference vector, and evaluates hypotheses to return a list of results.

Usage

```
pred_results(
  dataset,
  id = NULL,
  vars,
  type = "group",
  hypothesis,
  gtvar,
  grp_a,
  grp_b,
  location = "median",
  diff_method = "wilcoxon",
  phi_0 = 0.5
)
```

Arguments

`dataset`	A data frame for research.
`id`	The column that identifies unique subjects. This should be `NULL` if `type` is 'group' and should not be `NULL` if `type` is 'prepost'.
`vars`	The column variables of interest.
`type`	The type of study. Valid values are 'group' for group-based data and 'prepost' for pre-post data. Defaults to 'group'.
`hypothesis`	A vector or string of valid hypotheses: 'increase', 'decrease', or 'different'.
`gtvar`	The column of interest to divide the groups (e.g., time or treatment).
`grp_a`	The first subset of interest within the `gtvar` column (e.g., 'pre' or 'control').
`grp_b`	The second subset of interest within the `gtvar` column (e.g., 'post' or 'treatment').
`location`	The measure of central tendency to use for the difference calculation. Valid options are 'median' or 'mean'. Defaults to 'median'.
`diff_method`	The method to use for testing 'different' hypotheses. Valid options are 'wilcoxon' or 't'. Defaults to 'wilcoxon'.
`phi_0`	The decision rule threshold for the p-value. If p-value < phi_0, then there's sufficient evidence for a success for a difference. Defaults to 0.50.

Details

This function performs error handling to ensure appropriate input values and types. It then filters the data based on the study type, calculates the difference vector, and evaluates the hypotheses using the specified method.

Value

A list containing:

results A vector of 0s and 1s indicating whether each hypothesis was correct.

differences A vector of the differences between groups.

variables The column variables used in the analysis.

Examples

```
data("group_data_example")
data("group_cog_data")
data("pre_post_data_example")
data("pre_post_fit")

# simple group analysis
pred_results(dataset=group_data_example, vars=c('v1', 'v2'),
hypothesis=c("increase", "different"), gtvar="group", grp_a="placebo", grp_b="drug")

# simple prepost analysis
pred_results(dataset=pre_post_data_example, id="ID", vars=c('v1', 'v2', 'v3'),
type="prepost", hypothesis="increase", gtvar="time", grp_a=0, grp_b=12)

# simulated group analysis
pred_results(dataset=group_cog_data, vars=c('blind_moca_uncorrected', 'craft_verbatim'),
type="group", hypothesis="decrease", gtvar="group.factor", grp_a="Control", grp_b="ESKD")

# simulated prepost analysis
pred_results(dataset=pre_post_fit, id="ID", vars=c('Flex_right', 'Flex_left'),
type="prepost", hypothesis="increase", gtvar="Time", grp_a=0, grp_b=1)
```

---

pred_test *Predictive Test Function*

Description

This function performs statistical tests to determine the predictive power of a results set weighted by a corresponding vector of weights. It offers various methods to conduct the test, allowing flexibility depending on the data characteristics and analysis requirements.

Usage

```
pred_test(
  weights_vector,
  results_vector,
  test_type = "exact",
  phi_0 = 0.5,
  sims = 5000
)
```

Arguments

`weights_vector`	A numeric vector where each element represents the weight for a corresponding result in the results vector. Each value must be on the interval $[1/m, 1]$, where $m = 1/length(weights_vector)$.
`results_vector`	A numeric vector of test results where each element is in the set {0, 1}, representing the binary outcome of each prediction.
`test_type`	A character string specifying the type of statistical test to perform. The valid options are 'exact', 'approx', or 'bootstrap'.
`phi_0`	A numeric value on the interval (0, 1) representing the null hypothesis value against which the test results are compared.
`sims`	A natural number that specifies the number of simulations to perform when the bootstrap method is chosen. This parameter allows control over the robustness of the bootstrap approximation.

Details

This function performs error handling to ensure appropriate input values and types. It then calculates the test statistic and evaluates the p-value based on the specified test type.

Value

A list containing:

num_correctly_predicted The number of results correctly predicted as per the specified criteria.

p_value The p-value resulting from the test, indicating the probability of observing the test results under the null hypothesis.

test_stat The test statistic calculated based on the weights and results.

p0 The estimated proportion derived from the weights and results.

ci A confidence interval for the estimated proportion derived from the weights and results using the Wilson score method.

Examples

```
# Example weights and results vectors
weights_vector <- c(1/3, 0.5, 1)
results_vector <- c(0, 1, 1)
```

```
# Exact test
result_exact <- pred_test(weights_vector, results_vector, test_type = 'exact')
result_exact

# Approximate test
result_approx <- pred_test(weights_vector, results_vector, test_type = 'approx')
result_approx

# Bootstrap test
result_bootstrap <- pred_test(weights_vector, results_vector, test_type = 'bootstrap')
result_bootstrap
```

`pred_weights` *Calculate Predictive Weights*

Description

This function calculates predictive weights by computing the inverse square sum of a correlation matrix derived from the specified variables. In 'group' analysis, it directly uses the variables for correlation. In 'prepost' analysis, it calculates the difference between two time points before correlation.

Usage

```
pred_weights(
  dataset,
  vars,
  gtvar,
  type = "group",
  id = NULL,
  pre = NULL,
  post = NULL,
  corr_method = "pearson"
)
```

Arguments

`dataset`	A data frame containing the dataset to be analyzed.
`vars`	A vector of strings specifying the names of the variables to be used in the correlation analysis.
`gtvar`	The name of the categorical variable used to identify groups in 'prepost' type analysis.
`type`	The type of analysis. Valid values are 'group' for group-based correlation analysis or 'prepost' for pre-post analysis. Defaults to 'group'.
`id`	The variable in the dataset that uniquely identifies subjects in a 'prepost' analysis. This should not be `NULL` if `type` is 'prepost'.

pre	Specifies the baseline time point for 'prepost' analysis.
post	Specifies the follow-up time point for 'prepost' analysis.
corr_method	The method of correlation. Valid options are 'pearson', 'kendall', or 'spearman'. Defaults to 'pearson'.

Details

This function performs error handling to ensure appropriate input values and types. It calculates the correlation matrix for the specified variables and then computes the predictive weights as the inverse square sum of the correlation matrix.

Value

A numeric vector of predictive weights for each variable analyzed.

Examples

```
data("group_data_example")
data("group_cog_data")
data("pre_post_data_example")
data("pre_post_fit")

# end points for variables
grp_endpts <- c(
    "mean_suv","blind_moca_uncorrected","craft_verbatim","craft_delay_verbatim",
    "number_span_forward","number_span_backward","fluency_f_words_correct",
    "oral_trail_part_a","oral_trail_part_b","fluency_animals","fluency_vegetables",
    "verbal_naming_no_cue"
    )
prepost_endpts <- c(
    "COPM_p", "COPM_s", "A1_work", "A2_work", "Grip_dom",
    "Grip_ndom",  "Flex_right", "Flex_left"
    )
# simple group
pred_weights(dataset=group_data_example, vars=c('v1', 'v2'), gtvar='group')
# simple prepost
pred_weights(dataset=pre_post_data_example, vars=c('v1','v2','v3'),
gtvar='time', id='ID', pre=0,post=12)
# simulated group
pred_weights(dataset=group_cog_data, vars=grp_endpts, gtvar="group.factor",
type="group",corr_method="pearson")
# simulated prepost
pred_weights(dataset=pre_post_fit, id="ID", vars=prepost_endpts,
gtvar="Time", type="prepost",pre=0, post=1, corr_method="pearson")
```

pre_post_data_example *Pre-Post Data Example*

Description

A simulated dataset showing measurements before and after an intervention. Each subject has multiple measurements over time.

Usage

```
pre_post_data_example
```

Format

A data frame with 30 rows and 6 variables:

ID A unique identifier for each subject.

time A numeric variable indicating time points, `0` for pre-intervention and `12` for post-intervention.

v1 A numeric vector representing the first variable.

v2 A numeric vector representing the second variable.

v3 A numeric vector representing the third variable.

v4 A numeric vector representing the fourth variable.

Examples

```
data(pre_post_data_example)
head(pre_post_data_example)
```

pre_post_fit *Pre-Post Fitness Data*

Description

A dataset showing physical and cognitive performance measures before and after a fitness intervention.

Usage

```
pre_post_fit
```

Format

A data frame with 20 rows and 12 variables:

ID A unique identifier for each subject.

Time A numeric variable indicating time points, `0` for pre-intervention and `1` for post-intervention.

Sex A numeric variable indicating sex: `0` for male and `1` for female.

Age A numeric variable representing the age of each subject.

COPM_p A numeric vector representing performance scores on the Canadian Occupational Performance Measure (COPM).

COPM_s A numeric vector representing satisfaction scores on the COPM.

A1_work A numeric vector representing work capacity scores at time A1.

A2_work A numeric vector representing work capacity scores at time A2.

Grip_dom A numeric vector representing grip strength of the dominant hand.

Grip_ndom A numeric vector representing grip strength of the non-dominant hand.

Flex_right A numeric vector representing right arm flexibility.

Flex_left A numeric vector representing left arm flexibility.

Examples

```
data(pre_post_fit)
head(pre_post_fit)
```

---

solve_p0_score_ci *Calculate the Adjusted Proportion Estimate and Confidence Interval*

Description

This function calculates the adjusted proportion estimate (p0) and the confidence interval for a given proportion estimate (p_hat) and sample size (n) using the score method.

Usage

```
solve_p0_score_ci(p_hat, n, z = 1.96)
```

Arguments

`p_hat`	Numeric value. The proportion estimate. Must be between 0 and 1.
`n`	Numeric value. The sample size. Must be a positive integer.
`z`	Numeric value. The z-score for the desired confidence level. Default is 1.96 (approximately 95% confidence interval).

Value

A list with two elements:

`p0` The adjusted proportion estimate.

`confidence_interval`
A numeric vector of length 2 containing the lower and upper bounds of the confidence interval.

Examples

```
solve_p0_score_ci(p_hat = 9/10, n = 10)
```

Index