Package 'vecmatch'

April 24, 2025

Title Generalized Propensity Score Estimation and Matching for Multiple Groups

Version 1.1.0

Description Implements the Vector Matching algorithm to match multiple treatment groups based on previously estimated generalized propensity scores. The package includes tools for visualizing initial confounder imbalances, estimating treatment assignment probabilities using various methods, defining the common support region, performing matching across multiple groups, and evaluating matching quality. For more details, see Lopez and Gutman (2017) <doi:10.1214/17-STS612>.

License GPL (>= 3)

URL https://github.com/Polymerase3/vecmatch

BugReports https://github.com/Polymerase3/vecmatch/issues

Depends R (>= 3.5)

Imports brglm2, chk, cli, ggplot2, ggpp, ggpubr, grDevices, MASS, Matching, mclogit, nnet, optmatch, productplots, rlang, rstatix, stats, utils, VGAM, withr

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Language en-US

VignetteBuilder knitr

NeedsCompilation no

Author Mateusz Kolek [aut, cre, cph] (<https://orcid.org/0000-0001-6470-4830>)

Maintainer Mateusz Kolek <mati.kolek13@gmail.com>

Repository CRAN

Date/Publication 2025-04-24 12:20:02 UTC

Contents

---

balqual   *Evaluate Matching Quality*

Description

The `balqual()` function evaluates the balance quality of a dataset after matching, comparing it to the original unbalanced dataset. It computes various summary statistics and provides an easy interpretation using user-specified cutoff values.

Usage

```
balqual(
  matched_data = NULL,
  formula = NULL,
  type = c("smd", "r", "var_ratio"),
  statistic = c("mean", "max"),
  cutoffs = NULL,
  round = 3
)
```

Arguments

matched_data   An object of class `matched`, generated by the match_gps() function. This object is essential for the balqual() function as it contains the final data.frame and attributes required to compute the quality coefficients.

formula   A valid R formula used to compute generalized propensity scores during the first step of the vector matching algorithm in estimate_gps(). This formula must match the one used in estimate_gps().

type   A character vector specifying the quality metrics to calculate. Can maximally contain 3 values in a vector created by the `c()`. Possible values include:

- smd - Calculates standardized mean differences (SMD) between groups, defined as the difference in means divided by the standard deviation of the treatment group (Rubin, 2001).

- r - Computes Pearson's r coefficient using the Z statistic from the U-Mann-Whitney test.
- var_ratio - Measures the dispersion differences between groups, calculated as the ratio of the larger variance to the smaller one.

statistic   A character vector specifying the type of statistics used to summarize the quality metrics. Since quality metrics are calculated for all pairwise comparisons between treatment levels, they need to be aggregated for the entire dataset.

- max: Returns the maximum values of the statistics defined in the type argument (as suggested by Lopez and Gutman, 2017).
- mean: Returns the corresponding averages.

To compute both, provide both names using the c() function.

cutoffs   A numeric vector with the same length as the number of coefficients specified in the type argument. Defines the cutoffs for each corresponding metric, below which the dataset is considered balanced. If NULL, the default cutoffs are used: 0.1 for smd and r, and 2 for var_ratio.

round   An integer specifying the number of decimal places to round the output to.

Value

If assigned to a name, returns a list of summary statistics of class quality containing:

- quality_mean - A data frame with the mean values of the statistics specified in the type argument for all balancing variables used in formula.
- quality_max - A data frame with the maximal values of the statistics specified in the type argument for all balancing variables used in formula.
- perc_matched - A single numeric value indicating the percentage of observations in the original dataset that were matched.
- statistic - A single string defining which statistic will be displayed in the console.
- summary_head - A summary of the matching process. If max is included in the statistic, it contains the maximal observed values for each variable; otherwise, it includes the mean values.
- n_before - The number of observations in the dataset before matching.
- n_after - The number of observations in the dataset after matching.
- count_table - A contingency table showing the distribution of the treatment variable before and after matching.

The balqual() function also prints a well-formatted table with the defined summary statistics for each variable in the formula to the console.

References

Rubin, D.B. Using Propensity Scores to Help Design Observational Studies: Application to the Tobacco Litigation. Health Services & Outcomes Research Methodology 2, 169–188 (2001). https://doi.org/10.1023/A:1020363010465

Michael J. Lopez, Roee Gutman "Estimation of Causal Effects with Multiple Treatments: A Review and New Ideas," Statistical Science, Statist. Sci. 32(3), 432-454, (August 2017)

See Also

match_gps() for matching the generalized propensity scores; estimate_gps() for the documentation of the formula argument.

Examples

```
# We try to balance the treatment variable in the cancer dataset based on age
# and sex covariates
data(cancer)

# Firstly, we define the formula
formula_cancer <- formula(status ~ age * sex)

# Then we can estimate the generalized propensity scores
gps_cancer <- estimate_gps(formula_cancer,
  cancer,
  method = "multinom",
  reference = "control",
  verbose_output = TRUE
)

# ... and drop observations based on the common support region...
csr_cancer <- csregion(gps_cancer)

# ... to match the samples using `match_gps()`
matched_cancer <- match_gps(csr_cancer,
  reference = "control",
  caliper = 1,
  kmeans_cluster = 5,
  kmeans_args = list(n.iter = 100),
  verbose_output = TRUE
)

# At the end we can assess the quality of matching using `balqual()`
balqual(
  matched_data = matched_cancer,
  formula = formula_cancer,
  type = "smd",
  statistic = "max",
  round = 3,
  cutoffs = 0.2
)
```

cancer   *Patients with Colorectal Cancer and Adenoma metadata*

Description

This is a synthetically generated dataset containing metadata for healthy individuals and patients diagnosed with colorectal cancer or adenomas. The primary purpose of this dataset in the context of matching is to balance the `status` groups across various covariates and achieve optimal matching quality.

Usage

```
data(cancer)
```

Format

A data frame (`cancer`) with 1,224 rows and 5 columns:

status Patient's health status, which can be one of the following: `healthy`, `adenoma`, `crc_benign` (benign colorectal carcinoma), or `crc_malignant` (malignant colorectal carcinoma).

sex Patient's biological sex, recorded as either `M` (male) or `F` (female).

age Patient's age, represented as a continuous numeric variable.

bmi Patient's Body Mass Index (BMI), represented as a continuous numeric variable.

smoker Smoking status of the patient, recorded as `yes` or `no`.

Source

Data generated artificially

---

csregion   *Filter the data based on common support region*

Description

The `csregion()` function estimates the boundaries of the rectangular common support region, as defined by Lopez and Gutman (2017), and filters the matrix of generalized propensity scores based on these boundaries. The function returns a matrix of observations whose generalized propensity scores lie within the treatment group-specific boundaries.

Usage

```
csregion(gps_matrix, borders = "include")
```

Arguments

gps_matrix   An object of classes gps and data.frame (e.g., created by the estimate_gps()
               function). The first column corresponds to the treatment or grouping variable,
               while the other columns represent the treatment assignment probabilities calcu-
               lated separately for each hypotetical treatment group. The number of columns
               should therefore be equal to the number of unique levels of the treatment vari-
               able plus one (for the treatment variable itself). The number of rows should
               correspond to the number of subjects for which generalized propensity scores
               were estimated.

borders   A character string specifying how to handle observations at the edges of the
               Common Support Region (CSR). Acceptable values are "include" and "exclude".
               If "include" is selected (default), observations with Generalized Propensity
               Scores (GPS) exactly equal to the CSR boundaries are retained for further analy-
               sis. This corresponds to a non-strict inequality: lower_bound <= GPS <= upper_bound.
               If "exclude" is selected, observations lying exactly on the CSR boundaries
               are removed. This corresponds to a strict inequality: lower_bound < GPS <
               upper_bound. Using "exclude" will typically result in a slightly smaller matched
               sample size compared to "include", but may be preferred for more conserva-
               tive matching.

Value

A numeric matrix similar to the one returned by estimate_gps(), but with the number of rows
reduced to exclude those observations that do not fit within the common support region (CSR)
boundaries. The returned object also possesses additional attributes that summarize the calculation
process of the CSR boundaries:

- filter_matrix - A logical matrix with the same dimensions as the gps-part of gps_matrix,
  indicating which treatment assignment probabilities fall within the CSR boundaries,

- filter_vector - A vector indicating whether each observation was kept (TRUE) or removed
  (FALSE), essentially a row-wise sum of filter_matrix,

- csr_summary - A summary of the CSR calculation process, including details of the boundaries
  and the number of observations filtered.

- csr_data - The original dataset used for the estimation of generalized propensity scores
  (original_data attribute of the gps object) filtered by the filter_vector

Examples

```
# We could estimate simples generalized propensity scores for the `iris`
# dataset
gps <- estimate_gps(Species ~ Sepal.Length, data = iris)

# And then define the common support region boundaries using `csregion()`
gps_csr <- csregion(gps)

# The additional information of the CSR-calculation process are
# accessible through the attributes described in the `*Value*` section
attr(gps_csr, "filter_matrix")
```

```
attr(gps_csr, "csr_summary")
attr(gps_csr, "csr_data")
```

---

estimate_gps	*Calculate treatment allocation probabilities*

Description

estimate_gps() computes generalized propensity scores for treatment groups by applying a user-defined formula and method. It returns a matrix of GPS probabilities for each subject and treatment group

Usage

```
estimate_gps(
  formula,
  data = NULL,
  method = "multinom",
  link = NULL,
  reference = NULL,
  by = NULL,
  subset = NULL,
  ordinal_treat = NULL,
  fit_object = FALSE,
  verbose_output = FALSE,
  ...
)
```

Arguments

formula	a valid R formula, which describes the model used to calculating the probabilities of receiving a treatment. The variable to be balanced is on the left side, while the covariates used to predict the treatment variable are on the right side. To define the interactions between covariates, use *. For more details, refer to [stats::formula()]().
data	a data frame with columns specified in the formula argument.
method	a single string describing the model used for the calculation of generalized propensity scores. The default value is set to multinom. For available methods refer to the Details section below.
link	a single string; determines an alternative model for a method used for estimation. For available links, see Details.
reference	a single string describing one class from the treatment variable, referred to as the baseline category in the calculation of generalized propensity scores.

by	a single string with the name of a column, contained in the `data` argument. The dataset will be divided by the groups created by the grouping by variable and the calculation of the propensity scores will be carried out separately for each group. The results will then be merged and presented to the user as a single GPS matrix.
subset	a logical atomic vector of length equal to the number of rows in the `data` arguments. Allows to filter out observations from the further analysis, for which the value of the vector is equal to `FALSE`.
ordinal_treat	an atomic vector of the length equal to the length of unique levels of the treatment variable. Confirms, that the treatment variable is an ordinal variable and adjusts its levels, to the order of levels specified in the argument. Is a call to the function `factor(treat, levels = ordinal_treat, ordered = TRUE`.
fit_object	a logical flag. If `TRUE`, the the fitted object is returned instead of the GPS matrix.
verbose_output	a logical flag. If `TRUE` a more verbose version of the function is run and the output is printed out to the console.
...	additional arguments, that can be passed to the fitting function and are not controlled by the above arguments. For more details and examples refer to the Details section and documentations of corresponding functions.

Details

The main goal of the `estimate_gps()` function is to calculate the generalized propensity scores aka. treatment allocation probabilities. It is the first step in the workflow of the vector matching algorithm and is essential for the further analysis. The returned matrix of class `gps` can then be passed to the `csregion()` function to calculate the rectangular common support region boundaries and drop samples not eligible for the further analysis. The list of available methods operated by the `estimate_gps()` is provided below with a short description and function used for the calculations:

- multinom - multinomial logistic regression model nnet::multinom()
- vglm - vector generalized linear model for multinomial data VGAM::vglm(),
- brglm2 - bias reduction model for multinomial responses using the Poisson trick brglm2::brmultinom(),
- mblogit - baseline-category logit models mclogit::mblogit().
- polr - ordered logistic or probit regression only for ordered factor variables from MASS::polr(). The `method` argument of the underlying MASS::polr() package function can be controlled with the `link` argument. Available options: link = c("logistic", "probit", "loglog", "cloglog", "cauchit")

Value

A numeric matrix of class `gps` with the number of columns equal to the number of unique treatment variable levels plus one (for the treatment variable itself) and the number of row equal to the number of subjects in the initial dataset. The original dataset used for estimation can be accessed as `original_data` attribute.

See Also

csregion() for the calculation of common support region, match_gps() for the matching of generalized propensity scores

Examples

```
library("brglm2")

# Conducting covariate balancing on the `airquality` dataset. Our goal was to
# compare ozone levels by month, but we discovered that ozone levels are
# strongly correlated with wind intensity (measured in mph), and the average
# wind intensity varies across months. Therefore, we need to balance the
# months by wind values to ensure a valid comparison of ozone levels.

# Initial imbalance of means
tapply(airquality$Wind, airquality$Month, mean)

# Formula definition
formula_air <- formula(Month ~ Wind)

# Estimating the generalized propensity scores using brglm2 method using
# maximum penalized likelihood estimators with powers of the Jeffreys
gp_scores <- estimate_gps(formula_air,
  data = airquality, method = "brglm2",
  reference = "5", verbose_output = TRUE,
  control = brglmControl(type = "MPL_Jeffreys")
)

# Filtering the observations outside the csr region
gps_csr <- csregion(gp_scores)

# Calculating imbalance after csr
filter_which <- attr(gps_csr, "filter_vector")
filtered_air <- airquality[filter_which, ]

tapply(filtered_air$Wind, filtered_air$Month, mean)

# We can also investigate the imbalance using the raincloud function
raincloud(filtered_air,
  y = Wind,
  group = Month,
  significance = "t_test"
)
```

---

match_gps *Match the data based on generalized propensity score*

Description

The match_gps() function performs sample matching based on generalized propensity scores
(GPS). It utilizes the k-means clustering algorithm to partition the data into clusters and subsequently matches all treatment groups within these clusters. This approach ensures efficient and
structured comparisons across treatment levels while accounting for the propensity score distribution.

Usage

```
match_gps(
  csmatrix = NULL,
  method = "nnm",
  caliper = 0.2,
  reference = NULL,
  ratio = NULL,
  replace = NULL,
  order = NULL,
  ties = NULL,
  min_controls = NULL,
  max_controls = NULL,
  kmeans_args = NULL,
  kmeans_cluster = 5,
  verbose_output = FALSE,
  ...
)
```

Arguments

csmatrix   An object of class gps and/or csr representing a data frame of generalized
                propensity scores. The first column must be the treatment variable, with ad-
                ditional attributes describing the calculation of the common support region and
                the estimation of generalized propensity scores. It is crucial that the common
                support region was calculated using the csregion() function to ensure compat-
                ibility.

method   A single string specifying the matching method to use. The default is "nnm",
                which applies the k-nearest neighbors matching algorithm. See the Details sec-
                tion for a full list of available methods.

caliper   A numeric value specifying the caliper width, which defines the allowable range
                within which observations can be matched. It is expressed as a percentage of
                the standard deviation of the logit-transformed generalized propensity scores.
                To perform matching without a caliper, set this parameter to a very large value.
                For exact matching, set caliper = 0 and enable the exact option by setting it
                to TRUE.

reference   A single string specifying the exact level of the treatment variable to be used as
                the reference in the matching process. All other treatment levels will be matched
                to this reference level. Ideally, this should be the control level. If no natural
                control is present, avoid selecting a level with extremely low or high covariate
                or propensity score values. Instead, choose a level with covariate or propensity
                score distributions that are centrally positioned among all treatment groups to
                maximize the number of matches.

ratio   A scalar for the number of matches which should be found for each control
                observation. The default is one-to-one matching. Only available for the methods
                "nnm" and "pairopt".

replace   Logical value indicating whether matching should be done with replacement. If
                FALSE, the order of matches generally matters. Matches are found in the same

order as the data is sorted. Specifically, the matches for the first observation will be found first, followed by those for the second observation, and so on. Matching without replacement is generally not recommended as it tends to increase bias. However, in cases where the dataset is large and there are many potential matches, setting replace = FALSE often results in a substantial speedup with negligible or no bias. Only available for the method "nnm"

order   A string specifying the order in which logit-transformed GPS values are sorted before matching. The available options are:

- "desc" – sorts GPS values from highest to lowest (default).
- "asc" – sorts GPS values from lowest to highest.
- "original" – preserves the original order of GPS values.
- "random" – randomly shuffles GPS values. To generate different random orders, set a seed using set.seed().

ties   A logical flag indicating how tied matches should be handled. Available only for the "nnm" method, with a default value of FALSE (all tied matches are included in the final dataset, but only unique observations are retained). For more details, see the ties argument in Matching::Matchby().

min_controls   The minimum number of treatment observations that should be matched to each control observation. Available only for the "fullopt" method. For more details, see the min.controls argument in optmatch::fullmatch().

max_controls   The maximum number of treatment observations that can be matched to each control observation. Available only for the "fullopt" method. For more details, see the max.controls argument in optmatch::fullmatch().

kmeans_args   A list of arguments to pass to stats::kmeans. These arguments must be provided inside a list() in the paired name = value format.

kmeans_cluster An integer specifying the number of clusters to pass to stats::kmeans.

verbose_output a logical flag. If TRUE a more verbose version of the function is run and the output is printed out to the console.

...   Additional arguments to be passed to the matching function.

Details

Propensity score matching can be performed using various matching algorithms. Lopez and Gutman (2017) do not explicitly specify the matching algorithm used, but it is assumed they applied the commonly used k-nearest neighbors matching algorithm, implemented as method = "nnm". However, this algorithm can sometimes be challenging to use, especially when treatment and control groups have unequal sizes. When replace = FALSE, the number of matches is strictly limited by the smaller group, and even with replace = TRUE, the results may not always be satisfactory. To address these limitations, we have implemented an additional matching algorithm to maximize the number of matched observations within a dataset.

The available matching methods are:

- "nnm" – classic k-nearest neighbors matching, implemented using Matching::Matchby(). The tunable parameters in match_gps() are caliper, ratio, replace, order, and ties. Additional arguments can be passed to Matching::Matchby() via the ... argument.

- "fullopt" – optimal full matching algorithm, implemented with optmatch::fullmatch(). This method calculates a discrepancy matrix to identify all possible matches, often optimizing the percentage of matched observations. The available tuning parameters are caliper, min_controls, and max_controls.

- "pairmatch" – optimal 1:1 and 1:k matching algorithm, implemented using optmatch::pairmatch(), which is actually a wrapper around optmatch::fullmatch(). Like "fullopt", this method calculates a discrepancy matrix and finds matches that minimize its sum. The available tuning parameters are caliper and ratio.

Value

A data.frame similar to the one provided as the data argument in the estimate_gps() function, containing the same columns but only the observations for which a match was found. The returned object includes two attributes, accessible with the attr() function:

- original_data: A data.frame with the original data returned by the csregion() or estimate_gps() function, after the estimation of the csr and filtering out observations not within the csr.

- matching_filter: A logical vector indicating which rows from original_data were included in the final matched dataset.

References

Michael J. Lopez, Roee Gutman "Estimation of Causal Effects with Multiple Treatments: A Review and New Ideas," Statistical Science, Statist. Sci. 32(3), 432-454, (August 2017)

See Also

estimate_gps() for the calculation of generalized propensity scores; MatchIt::matchit(), optmatch::fullmatch() and optmatch::pairmatch() for the documentation of the matching functions; stats::kmeans() for the documentation of the k-Means algorithm.

Examples

```
# Defining the formula used for gps estimation
formula_cancer <- formula(status ~ age + sex)

# Step 1.) Estimation of the generalized propensity scores
gp_scores <- estimate_gps(formula_cancer,
  data = cancer,
  method = "multinom",
  reference = "control",
  verbose_output = TRUE
)

# Step 2.) Defining the common support region
gps_csr <- csregion(gp_scores)

# Step 3.) Matching the gps
matched_cancer <- match_gps(gps_csr,
  caliper = 0.25,
  reference = "control",
```

```
    method = "fullopt",
    kmeans_cluster = 2,
    kmeans_args = list(
      iter.max = 200,
      algorithm = "Forgy"
    ),
    verbose_output = TRUE
)
```

---

mosaic *Plot the distribution of categorical covariates*

Description

The `mosaic()` function generates imbalance plots for contingency tables with up to three variables. Frequencies in the contingency table are represented as tiles (rectangles), with each tile's size proportional to the frequency of the corresponding group within the entire dataset. The x-axis scale remains fixed across mosaic plots, enabling straightforward comparisons of imbalance across treatment groups.

Usage

```
mosaic(
  data = NULL,
  y = NULL,
  group = NULL,
  facet = NULL,
  ncol = 1,
  group_counts = FALSE,
  group_counts_size = 4,
  significance = FALSE,
  plot_name = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

data   A non-empty `data.frame` containing at least one numeric column, as specified
                by the y argument. This argument must be provided and does not have a default
                value.

y   A single string or unquoted symbol representing the name of a numeric column
                in the `data`. In the vector matching workflow, it is typically a numeric covariate
                that requires balancing.

group   A single string or unquoted symbol representing the name of a factor or character
                 column in `data`. In `raincloud()` plots, the groups specified by `group` argument
                 will be distinguished by separate `fill` and `color` aesthetics. For clarity, it is
                 recommended to plot fewer than 10 groups, though there is no formal limit.

facet   A single string or unquoted symbol representing the name of a variable in `data`
                 to facet by. This argument is used in a call to `ggplot2::facet_wrap()`, creating
                 separate distribution plots for each unique group in the `facet` variable.

ncol   A single integer. The value should be less than or equal to the number of unique
                 categories in the `facet` variable. This argument is used only when `facet` is not
                 `NULL`, specifying the number of columns in the `ggplot2::facet_wrap()` call.
                 The distribution plots will be arranged into the number of columns defined by
                 `ncol`.

group_counts   A logical flag. If `TRUE`, the sizes of the groups will be displayed inside the
                 rectangles in the plot created by the `mosaic()` function. If `FALSE` (default), the
                 group sizes will not be shown.

group_counts_size
                 A single numeric value that specifies the size of the group count labels in mil-
                 limeters ('mm'). This value is passed to the `size` argument of `ggplot2::geom_text()`.

significance   A logical flag; defaults to `FALSE`. When `TRUE`, a Chi-squared test of indepen-
                 dence is performed on the contingency table of `y` and `group`. Note that `group`
                 must be specified for the test to be calculated. If `facet` is provided, the signifi-
                 cance is assessed separately for each `facet` subgroup. Additionally, the function
                 calculates standardized Pearson residuals (differences between observed and ex-
                 pected counts) and fills mosaic plot cells based on the level of partial significance
                 for each cell.

plot_name   A string specifying a valid file name or path for the plot. If set to `NULL`, the
                 plot is displayed to the current graphical device but not saved locally. If a valid
                 name with `.png` or `.pdf` extension is provided, the plot is saved locally. Users
                 can also include a subdirectory in `plot_name`. Ensure the file path follows the
                 correct syntax for your operating system.

overwrite   A logical flag (default `FALSE`) that is evaluated only if the `save.name` argument
                 is provided. If `TRUE`, the function checks whether a plot with the same name
                 already exists. If it does, the existing plot will be overwritten. If `FALSE` and a
                 plot with the same name exists, an error is thrown. If no such plot exists, the
                 plot is saved normally.

...   Additional arguments to pass to `rstatix::chisq_test` when `significance =`
                 `TRUE`.

Value

A `ggplot` object representing the contingency table of `y` and `group` as a mosaic plot, optionally
grouped by `facet` if specified.

Examples

```
## Example: Creating a Mosaic Plot of the Titanic Dataset
## This plot visualizes survival rates by gender across different passenger
```

```
## classes. By setting `significance = TRUE`, you can highlight statistically
## significant differences within each rectangle of the mosaic plot.
library(ggplot2)

# Load Titanic dataset and convert to data frame
titanic_df <- as.data.frame(Titanic)

# Expand the dataset by repeating rows according to 'Freq'
titanic_long <- titanic_df[rep(
  seq_len(nrow(titanic_df)),
  titanic_df$Freq
), ]

# Remove the 'Freq' column as it is no longer needed
titanic_long$Freq <- NULL

# Plot the data using mosaic() and modify the result using additional ggplot2
# functions
p <- vecmatch::mosaic(
  data = titanic_long,
  y = Survived,
  group = Sex,
  facet = Class,
  ncol = 2,
  significance = TRUE
)

p <- p +
  theme_minimal()

p
```

---

raincloud   *Examine the imbalance of continuous covariates*

Description

The `raincloud()` function allows to generate distribution plots for continuous data in an easy and uncomplicated way. The function is based on the `ggplot2` package, which must already be preinstalled Raincloud plots consist of three main elements:

- Distribution plots, specifically violin plots with the mean values and standard deviations of respective groups,

- Jittered point plots depicting the underlying distribution of the data in the rawest form,

- Boxplots, summarizing the most important statistics of the underlying distribution.

Usage

```
raincloud(
  data = NULL,
  y = NULL,
  group = NULL,
  facet = NULL,
  ncol = 1,
  significance = NULL,
  sig_label_size = 2L,
  sig_label_color = FALSE,
  smd_type = "mean",
  limits = NULL,
  jitter = 0.1,
  alpha = 0.4,
  plot_name = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

data	A non-empty `data.frame` containing at least one numeric column, as specified by the y argument. This argument must be provided and does not have a default value.
y	A single string or unquoted symbol representing the name of a numeric column in the `data`. In the vector matching workflow, it is typically a numeric covariate that requires balancing.
group	A single string or unquoted symbol representing the name of a factor or character column in `data`. In `raincloud()` plots, the groups specified by group argument will be distinguished by separate `fill` and `color` aesthetics. For clarity, it is recommended to plot fewer than 10 groups, though there is no formal limit.
facet	A single string or unquoted symbol representing the name of a variable in `data` to facet by. This argument is used in a call to `ggplot2::facet_wrap()`, creating separate distribution plots for each unique group in the `facet` variable.
ncol	A single integer. The value should be less than or equal to the number of unique categories in the `facet` variable. This argument is used only when `facet` is not NULL, specifying the number of columns in the `ggplot2::facet_wrap()` call. The distribution plots will be arranged into the number of columns defined by `ncol`.
significance	A single string specifying the method for calculating p-values in multiple comparisons between groups defined by the `group` argument. Significant comparisons are represented by bars connecting the compared groups on the left side of the boxplots. Note that if there are many significant tests, the plot size may adjust accordingly. For available methods refer to the *Details* section. If the `significance` argument is not NULL, standardized mean differences (SMDs) are also calculated and displayed on the right side of the jittered point plots.

sig_label_size   An integer specifying the size of the significance and SMD (standardized mean difference) labels displayed on the bars on the right side of the plot.

sig_label_color

       Logical flag. If FALSE (default), significance and SMD bars and text are displayed in the default color (black). If TRUE, colors are applied dynamically based on value: nonsignificant tests and SMD values below 0.10 are displayed in green, while significant tests and SMD values of 0.10 or higher are displayed in red.

smd_type   A single string indicating the type of effect size to calculate and display on the left side of the jittered point plots:

- mean - Cohen's d is calculated,
- median - the Wilcoxon effect size (r) is calculated based on the Z statistic extracted from the Wilcoxon test.

limits   A numeric atomic vector of length two, specifying the y axis limits in the distribution plots. The first element sets the minimum value, and the second sets the maximum. This vector is passed to the [ggplot2::xlim()](#) function to adjust the axis scale.

jitter   A single numeric value between 0 and 1 that controls the amount of jitter applied to points in the [ggplot2::geom_jitter()](#) plots. Higher values of the jitter argument produce more jittered plot. It's recommended to keep this value low, as higher jitter can make the plot difficult to interpret.

alpha   A single numeric value between 0 and 1 that controls the transparency of the density plots, boxplots, and jittered point plots. Lower values result in higher transparency. It is recommended to keep this value relatively high to maintain the interpretability of the plots when using the group argument, as excessive transparency may cause overlap between groups, making it difficult to distinguish them visually.

plot_name   A string specifying a valid file name or path for the plot. If set to NULL, the plot is displayed to the current graphical device but not saved locally. If a valid name with .png or .pdf extension is provided, the plot is saved locally. Users can also include a subdirectory in plot_name. Ensure the file path follows the correct syntax for your operating system.

overwrite   A logical flag (default FALSE) that is evaluated only if the save.name argument is provided. If TRUE, the function checks whether a plot with the same name already exists. If it does, the existing plot will be overwritten. If FALSE and a plot with the same name exists, an error is thrown. If no such plot exists, the plot is saved normally.

...   Additional arguments passed to the function for calculating p-values when the significance argument is specified. For available functions associated with different significance methods, please refer to the *Details* section and consult the documentation for the relevant functions in the rstatix package.

Details

Available methods for the argument significance are:

- "t_test" - Performs a pairwise comparison using the two-sample t-test, with the default Holm adjustment for multiple comparisons. This test assumes normally distributed data and equal variances. The adjustment can be modified via the p.adjust.method argument. The test is implemented via rstatix::pairwise_t_test()

- "dunn_test" - Executes Dunn's test for pairwise comparisons following a Kruskal-Wallis test. It is a non-parametric alternative to the t-test when assumptions of normality or homogeneity of variances are violated. Implemented via rstatix::dunn_test().

- "tukeyHSD_test" - Uses Tukey's Honest Significant Difference (HSD) test for pairwise comparisons between group means. Suitable for comparing all pairs when the overall ANOVA is significant. The method assumes equal variance between groups and is implemented via rstatix::tukey_hsd().

- "games_howell_test" - A post-hoc test used after ANOVA, which does not assume equal variances or equal sample sizes. It's particularly robust for data that violate homogeneity of variance assumptions. Implemented via rstatix::games_howell_test().

- "wilcoxon_test" - Performs the Wilcoxon rank-sum test (also known as the Mann-Whitney U test) for non-parametric pairwise comparisons. Useful when data are not normally distributed. Implemented via rstatix::pairwise_wilcox_test().

Value

A ggplot object representing the distribution of the y variable across the levels of the group and facet variables in data.

See Also

mosaic() which summarizes the distribution of discrete data

Examples

```
## Example: Creating a raincloud plot for the ToothGrowth dataset.
## This plot visualizes the distribution of the `len` variable by
## `dose` (using different colors) and facets by `supp`. Group
## differences by `dose` are calculated using a `t_test`, and standardized
## mean differences (SMDs) are displayed through jittered points.
library(ggplot2)
library(ggpubr)

p <- raincloud(ToothGrowth, len, dose, supp,
  significance = "t_test",
  jitter = 0.15, alpha = 0.4
)

## As `p` is a valid `ggplot` object, we can manipulate its
## characteristics usingthe `ggplot2` or `ggpubr` packages
## to create publication grade plot:
p <- p +
  theme_classic2() +
  theme(
    axis.line.y = element_blank(),
    axis.ticks.y = element_blank()
```

```
    ) +
    guides(fill = guide_legend("Dose [mg]")) +
    ylab("Length [cm]")

  p
```

---

vecmatch	*vecmatch*

Description

vecmatch

Index