Package 'esvis'

April 9, 2018

Type Package

Title Visualization and Estimation of Effect Sizes

Version 0.2.0

Description A variety of methods are provided to estimate and visualize distributional differences in terms of effect sizes. Particular emphasis is upon evaluating differences between two or more distributions across the entire scale, rather than at a single point (e.g., differences in means). For example, Probability-Probability (PP) plots display the difference between two or more distributions, matched by their empirical CDFs (see Ho and Reardon, 2012; <doi:10.3102/1076998611411918>), allowing for examinations of where on the scale distributional differences are largest or smallest. The area under the PP curve (AUC) is an effect-size metric, corresponding to the probability that a randomly selected observation from the x-axis distribution will have a higher value than a randomly selected observation from the y-axis distribution. Binned effect size plots are also available, in which the distributions are split into bins (set by the user) and separate effect sizes (Cohen's d) are produced for each bin - again providing a means to evaluate the consistency (or lack thereof) of the difference between two or more distributions at different points on the scale. Evaluation of empirical CDFs is also provided, with built-in arguments for providing annotations to help evaluate distributional differences at specific points (e.g., semi-transparent shading). All function take a consistent argument structure. Calculation of specific effect sizes is also possible. The following effect sizes are estimable: (a) Cohen's d, (b) Hedges' g, (c) percentage above a cut, (d) transformed (normalized) percentage above a cut, (e) area under the PP curve, and (f) the V statistic (see Ho, 2009; <doi:10.3102/1076998609332755>), which essentially transforms the area under the curve to standard deviation units. By default, effect sizes are calculated for all possible pairwise comparisons, but a reference group (distribution) can be specified.

Depends R (>= 3.1)

Imports sfsmisc

URL https://github.com/DJAnderson07/esvis

BugReports https://github.com/DJAnderson07/esvis/issues

License MIT + file LICENSE

LazyData true

RoxygenNote 6.0.1

Suggests testthat, viridisLite

NeedsCompilation no

Author Daniel Anderson [aut, cre]

Maintainer Daniel Anderson <daniela@uoregon.edu>

Repository CRAN

Date/Publication 2018-04-09 18:02:00 UTC

R topics documented:

auc *Calculate the area under the curve*

Description

This function is used within pp_plot to calculate the area under the pp curve. The area under the curve is also a useful effect-size like statistic, representing the probability that a randomly selected individual from distribution a will have a higher value than a randomly selected individual from distribution b.

Usage

```
auc(formula, data, ref_group = NULL, tidy = TRUE)
```

Arguments

formula	A formula of the type out ~ group where out is the outcome variable and group is the grouping variable. Note this variable can include any arbitrary number of groups.
data	The data frame that the data in the formula come from.
ref_group	Optional. If the name of the reference group is provided (must be character and match the grouping level exactly), only the estimates corresponding to the given reference group will be returned.
tidy	Logical. Should the data be returned in a tidy data frame? (see Wickham, 2014). If false, effect sizes returned as a vector.

Value

By default the area under the curve for all possible pairings of the grouping factor are returned as a tidy data frame. Alternatively, a vector can be returned, and/or only the auc corresponding to a specific reference group can be returned.

Examples

```
free_reduced <- rnorm(800, 80, 20)
pay <- rnorm(500, 100, 10)
d <- data.frame(score = c(free_reduced, pay),
frl = c(rep("free_reduced", 800),
rep("pay", 500)))

auc(score ~ frl, d)
# Compute AUC for all pairwise comparisons
auc(reading ~ condition, star)

# Specify regular-sized classrooms as the reference group
auc(reading ~ condition,
star,
ref_group = "reg")
```

```
# Return a vector instead of a data frame
auc(reading ~ condition,
star,
ref_group = "reg",
tidy = FALSE)
```

`benchmarks` *Synthetic benchmark screening data*

Description

Across the country many schools engage in seasonal benchmark screenings to monitor to progress of their students. These are relatively brief assessments administered to "check-in" on students' progress throughout the year. This dataset was simulated from a real dataset from one large school district using the terrific synthpop R package. Overall characteristics of the synthetic data are remarkably similar to the real data.

Usage

```
benchmarks
```

Format

A data frame with 10240 rows and 9 columns.

sid Integer. Student identifier.

cohort Integer. Identifies the cohort from which the student was sampled (1-3).

sped Character. Special Education status: "Non-Sped" or "Sped"

ethnicity Character. The race/ethnicity to which the student identified. Takes on one of seven values: "Am. Indian", "Asian", "Black", "Hispanic", "Native Am.", "Two or More", and "White"

frl Character. Student's eligibility for free or reduced price lunch. Takes on the values "FRL" and "Non-FRL".

ell Character. Students' English language learner status. Takes on one of values: "Active", "Monitor", and "Non-ELL". Students coded "Active" were actively receiving English language services at the time of testing. Students coded "Monitor" had previously received services, but not at the time of testing. Students coded "Non-ELL" did not receive services at any time.

season Character. The season during which the assessment was administered: "Fall", "Winter", or "Spring"

reading Integer. Reading scale score.

math Integer. Mathematics scale score.

binned_plot *Quantile-binned effect size plot*

Description

Plots the effect size between two groups by matched (binned) quantiles (i.e., the results from qtile_es), with the matched quantiles plotted along the x-axis and the effect size plotted along the y-axis. The intent is to examine how (if) the magnitude of the effect size varies at different points of the distributions.

Usage

```
binned_plot(formula, data, ref_group = NULL, qtiles = seq(0, 1, 0.3333),
  scheme = "ggplot2", se = TRUE, shade_col = NULL, shade_alpha = 0.3,
  annotate = FALSE, refline = TRUE, refline_lty = 2, refline_lwd = 2,
  rects = TRUE, rect_colors = c(rgb(0.2, 0.2, 0.2, 0.1), rgb(0.2, 0.2, 0.2,
  0)), lines = TRUE, points = TRUE, legend = NULL, theme = "standard",
  ...)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
ref_group	Optional character vector (of length 1) naming the reference group to be plotted on the x-axis. Defaults to the highest scoring group.
qtiles	The quantile bins to split the data by and calculate effect sizes. This argument is passed directly to qtile_es. Essentially, this is the binning argument. Defaults to `seq(0, 1, .33)` which splits the distribution into thirds (lower, middle, upper). Any sequence is valid, but it is recommended the bins be even. For example `seq(0, 1, .1)` would split the distributions into deciles.
scheme	What color scheme should the lines follow? Defaults to mimic the ggplot2 color scheme. Other options come from the viridisLite package, and must be installed first. These are the same options available in the package: "viridis", "magma", "inferno", and "plasma". These color schemes work well for color blindness and print well in black and white. Alternatively, colors can be supplied manually through a call to `col` (through `...`).
se	Logical. Should the standard errors around the effect size point estimates be displayed? Defaults to `TRUE`, with the uncertainty displayed with shading.
shade_col	Color of the standard error shading, if `se == TRUE`. Defaults to the same color as the lines.
shade_alpha	Transparency level of the standard error shading. Defaults to 0.3.

annotate	Logical. Defaults to FALSE. When TRUE and legend == "side" the plot is rendered such that additional annotations can be made on the plot using low level base plotting functions (e.g., arrows). However, if set to TRUE, dev.off must be called before a new plot is rendered (i.e., close the current plotting window). Otherwise the plot will be attempted to be rendered in the region designated for the legend). Argument is ignored when legend != "side".
refline	Logical. Defaults to TRUE. Should a diagonal reference line, representing the point of equal probabilities, be plotted?
refline_lty	Line type of the reference line. Defaults to 2.
refline_lwd	Line width of the reference line. Defaults to 2.
rects	Logical. Should semi-transparent rectangles be plotted in the background to show the binning? Defaults to TRUE.
rect_colors	Color of rectangles to be plotted in the background, if rects == TRUE. Defaults to alternating gray and transparent. Currently not alterable when theme == "dark", in which case the rects alternate a semi-transparent white and transparent.
lines	Logical. Should the points between effect sizes across qtiles be connected via a line? Defaults to TRUE.
points	Logical. Should points be plotted for each qtiles be plotted? Defaults to TRUE.
legend	The type of legend to be displayed, with possible values "base", "side", or "none". Defaults to "side", when there are more than two groups and "none" when only comparing two groups. If the option "side" is used the plot is split into two plots, via layout, with the legend displayed in the second plot. This scales better than the base legend (i.e., manually manipulating the size of the plot after it is rendered), but is not compatible with multi-panel plotting (e.g., par(mfrow = c(2, 2)) for a 2 by 2 plot). When producing multi-panel plots, use "none" or "base", the latter of which produces the legend with the base legend function.
theme	Visual properties of the plot. There are currently only two themes implemented - a standard plot and a dark theme. If NULL (default), the theme will be produced with a standard white background. If "dark", a dark gray background will be used with white text and axes.
...	Additional arguments passed to plot. Note that it is best to use the full argument rather than partial matching, given the method used to call the plot. While some partial matching is supported (e.g., m for main, it is generally safest to supply the full argument).

Examples

```
# Default binned effect size plot
binned_plot(math ~ condition, star)

# Change the reference group to regular sized classrooms
binned_plot(math ~ condition,
star,
ref_group = "reg")
```

```
# Change binning to deciles
binned_plot(math ~ condition,
star,
ref_group = "reg",
qtiles = seq(0, 1, .1))

# Suppress the standard error shading
binned_plot(math ~ condition,
star,
se = FALSE)

# Change to dark theme
binned_plot(math ~ condition,
star,
theme = "dark")
```

`cdfs` *Compute the empirical distribution functions for each of several groups.*

Description

This function is a simple wrapper that splits the data frame by the grouping variable, then loops ecdf through the split data to return a CDF function for each group.

Usage

```
cdfs(formula, data, center = FALSE)
```

Arguments

`formula`	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
`data`	The data frame that the data in the formula come from.
`center`	Logical. Should the functions be centered prior to plotting?

Value

A list with one function per group (level in the grouping factor).

Examples

```
cdfs(math ~ condition, star)
```

coh_d *Compute Cohen's* d

Description

This function calculates effect sizes in terms of Cohen's *d*, also called the uncorrected effect size. See hedg_g for the sample size corrected version. Also see Lakens (2013) for a discussion on different types of effect sizes and their interpretation. Note that missing data are removed from the calculations of the means and standard deviations.

Usage

```
coh_d(formula, data, ref_group = NULL, tidy = TRUE)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
data	The data frame that the data in the formula come from.
ref_group	Optional. If the name of the reference group is provided (must be character and match the grouping level exactly), only the estimates corresponding to the given reference group will be returned.
tidy	Logical. Should the data be returned in a tidy data frame? (see Wickham, 2014). If false, effect sizes returned as a vector.

Value

By default the Cohen's *d* for all possible pairings of the grouping factor are returned as a tidy data frame.

Examples

```
# Calculate Cohen's d for all pairwise comparisons
coh_d(reading ~ condition, star)

# Report only relative to regular-sized classrooms
coh_d(reading ~ condition,
star,
ref_group = "reg")

# Return a vector instead of a data frame
coh_d(reading ~ condition,
star,
ref_group = "reg",
tidy = FALSE)
```

col_hue *Color hues*

Description

Emulates ggplot's default colors. Evenly spaced hues around the color wheel.

Usage

```
col_hue(n, ...)
```

Arguments

n	The number of colors to be produced
...	Additional arguments passed to hcl, such as alpha.

Examples

```
col_hue(1)
col_hue(5)
col_hue(20)
```

col_scheme *Determine the color scheme to be used for the plotting*

Description

Determine the color scheme to be used for the plotting

Usage

```
col_scheme(scheme, n, ...)
```

Arguments

scheme	The chosen color scheme. Options are "ggplot2", "viridis", "magma", "inferno", or "plasma". Note all but "ggplot2" depend upon the viridisLite package
n	The number of colors to be produced for the given scheme.
...	Additional arguments passed, typically being alpha.

create_base_legend *Create a base legend for a plot*

Description

This function creates a legend using the base legend function, but with more abbreviated syntax.

Usage

```
create_base_legend(labels, position = "bottomright", ...)
```

Arguments

labels	Labels for the legend (line labels).
position	Where the legend should be positioned. Defaults to "bottomright".
...	Additional arguments passed to legend (typically colors, line width, etc).

create_cut_refs *Create a set of reference lines according to a cut score*

Description

Create a set of reference lines according to a cut score

Usage

```
create_cut_refs(cut, calcs, p, scheme)
```

Arguments

cut	The cut scores on the raw scale
calcs	object from pp_calcs
p	output from empty_plot
scheme	What color scheme should the lines follow? Defaults to mimic the ggplot2 color scheme. Other options come from the viridisLite package, and must be installed first. These are the same options available in the package: "viridis", "magma", "inferno", and "plasma". These color schemes work well for color blindness and print well in black and white. Alternatively, colors can be supplied manually through a call to col (through ...).

create_legend *Create a legend for a plot*

Description

This is an alternative legend for plots which uses the actual plotting environment to create the legend, rather than overlaying it. I prefer this legend because it scales better than the base legend. It is currently only implemented to support lines.

Usage

```
create_legend(n, leg_labels, left_mar = 0, height = NULL,
  main_cols = NULL, cut = NULL, cut_cols = NULL, n_1 = FALSE, ...)
```

Arguments

n	Number of lines to produce on the legend.
leg_labels	Labels for the lines in the legend.
left_mar	Left margin argument. Defaults to 0. Larger numbers push the legend more to the right.
height	The height of the legend. Counter-intuitively, larger numbers result in a smaller legend (more squished to the bottom).
main_cols	Primary colors (of the lines, rather than the cut scores)
cut	Cut scores (see pp_plot).
cut_cols	The color of the lines/points for the cut scores.
n_1	Should the lines on the legend be displayed when there is only one value? Defaults to FALSE, and is relevant when values to cut are provided.
...	Additional arguments passed to lines.

create_vec *Create a named vector of all possible combinations*

Description

Alternative to tidied data frame return.

Usage

```
create_vec(levs, fun)
```

Arguments

levs	The levels of the grouping factor from which to create the matrix
fun	The function to apply.

Value

Matrix of values according to the function supplied.

`ecdf_plot` *Empirical Cumulative Distribution Plot*

Description

This function dresses up the plot.ecdf function and provides some additional functionality to directly compare distributions at specific locations along the scale. Specifically, multiple empirical CDFs can be plotted with a single call, and the differences between any pair, or all, CDFs can optionally be plotted in terms of both raw percentage differences and/or in terms of standard deviation units through inverse normal transformations. See Ho & Reardon, 2012. (Note, not all features implemented yet)

Usage

```
ecdf_plot(formula, data, ref_cut = NULL, center = FALSE, max_line = FALSE,
  ref_hor = FALSE, ref_rect = TRUE, scheme = "ggplot2", legend = "side",
  annotate = FALSE, theme = "standard", ...)
```

Arguments

`formula`	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
`data`	The data frame that the data in the formula come from.
`ref_cut`	Optional numeric vector stating the location of reference line(s) and/or rectangle(s).
`center`	Logical. Should the functions be centered prior to plotting? Defaults to `FALSE`.
`max_line`	Logical. Should the maximum distance between any two curves be plotted? This distance is equivalent to the value tested by the Kolmogorov-Smirnov test. Defaults to `FALSE`.
`ref_hor`	Logical, defaults to `FALSE`. Should horizontal reference lines be plotted at the location of `ref_cut`?
`ref_rect`	Logical, defaults to `TRUE`. Should semi-transparent rectangle(s) be plotted at the locations of `ref_cut`?
`scheme`	What color scheme should the lines follow? Defaults to mimic the ggplot2 color scheme. Other options come from the viridisLite package, and must be installed first. These are the same options available in the package: "viridis", "magma", "inferno", and "plasma". These color schemes work well for color blindness and print well in black and white. Alternatively, colors can be supplied manually through a call to `col` (through `...`).

`legend` The type of legend to be displayed, with possible values `"base"`, `"side"`, or `"none"`. Defaults to `"side"`, when there are more than two groups and `"none"` when only comparing two groups. If the option `"side"` is used the plot is split into two plots, via layout, with the legend displayed in the second plot. This scales better than the base legend (i.e., manually manipulating the size of the plot after it is rendered), but is not compatible with multi-panel plotting (e.g., `par(mfrow = c(2, 2))` for a 2 by 2 plot). When producing multi-panel plots, use `"none"` or `"base"`, the latter of which produces the legend with the base legend function.

`annotate` Logical. Defaults to `FALSE`. When `TRUE` and `legend == "side"` the plot is rendered such that additional annotations can be made on the plot using low level base plotting functions (e.g., arrows). However, if set to `TRUE`, dev.off must be called before a new plot is rendered (i.e., close the current plotting window). Otherwise the plot will be attempted to be rendered in the region designated for the legend. Argument is ignored when `legend != "side"`.

`theme` Visual properties of the plot. There are currently only two themes implemented - a standard plot and a dark theme. If `NULL` (default), the theme will be produced with a standard white background. If `"dark"`, a dark gray background will be used with white text and axes.

`...` Additional arguments passed to plot. Note that it is best to use the full argument rather than partial matching, given the method used to call the plot. While some partial matching is supported (e.g., `m` for `main`, it is generally safest to supply the full argument).

Examples

```
# Produce base empirical cummulative distribution plot
ecdf_plot(mean ~ grade, seda)

# Shade distributions to the right of three cut scores
ecdf_plot(mean ~ grade,
seda,
ref_cut = c(225, 245, 265))

# Add horizontal reference lines
ecdf_plot(mean ~ grade,
seda,
ref_cut = c(225, 245, 265),
ref_hor = TRUE)

# Apply dark theme
ecdf_plot(mean ~ grade,
seda,
ref_cut = c(225, 245, 265),
theme = "dark")
```

`empty_plot` *Create an empty plot*

Description

This function creates an empty plot for further plotting (e.g., via lines). What makes the function unique is that it allows for specification of default `xlab`, `ylab`, and `main` arguments, while allowing the user to override those arguments. Only really useful when used within other functions (e.g., pp_plot).

Usage

```
empty_plot(x, y, default_xlab = NULL, default_ylab = NULL,
  default_main = NULL, default_xlim = NULL, default_ylim = NULL,
  default_xaxt = NULL, default_yaxt = NULL, default_bty = "n",
  theme = "standard", las = c(1, 2), ...)
```

Arguments

`x`	The x variable to be plotted.
`y`	The y variable to be plotted.
`default_xlab`	The default x-label, which can be overridden by the user. Defaults to `NULL`, in which case the label is defined by the default plot function.
`default_ylab`	The default y-label, which can be overridden by the user. Defaults to `NULL`, in which case the label is defined by the default plot function.
`default_main`	The default main title, which can be overridden by the user. Defaults to `NULL`, in which case the title is defined by the default plot function.
`default_xlim`	The default x-axis limits, which can be overridden by the user. Defaults to `NULL`, in which case the limits are defined by the default plot function.
`default_ylim`	The default y-axis limits, which can be overridden by the user. Defaults to `NULL`, in which case the limits are defined by the default plot function.
`default_xaxt`	The default x-axis type, which can be overridden by the user. Defaults to `NULL`, in which case the type is defined by the default plot function.
`default_yaxt`	The default y-axis type, which can be overridden by the user. Defaults to `NULL`, in which case the type is defined by the default plot function.
`default_bty`	The default background type, which can be overridden by the user. Defaults to `"n"`.
`theme`	The theme to be applied.
`las`	The axis option. Defaults to `c(1, 2)` which makes the labels all horizontal.
`...`	Additional arguments supplied to plot (e.g., `xlim`, `ylim`, `cex`, etc.)

hedg_g	*Compute Hedges' g This function calculates effect sizes in terms of Hedges' g, also called the corrected (for sample size) effect size. See coh_d for the uncorrected version. Also see Rhrefhttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC3840331/Lakens (2013) for a discussion on different types of effect sizes and their interpretation. Note that missing data are removed from the calculations of the means and standard deviations.*

Description

Compute Hedges' *g* This function calculates effect sizes in terms of Hedges' *g*, also called the corrected (for sample size) effect size. See coh_d for the uncorrected version. Also see Lakens (2013) for a discussion on different types of effect sizes and their interpretation. Note that missing data are removed from the calculations of the means and standard deviations.

Usage

```
hedg_g(formula, data, ref_group = NULL, tidy = TRUE)
```

Arguments

formula	A formula of the type out ~ group where out is the outcome variable and group is the grouping variable. Note this variable can include any arbitrary number of groups.
data	The data frame that the data in the formula come from.
ref_group	Optional. If the name of the reference group is provided (must be character and match the grouping level exactly), only the estimates corresponding to the given reference group will be returned.
tidy	Logical. Should the data be returned in a tidy data frame? (see Wickham, 2014). If false, effect sizes returned as a vector.

Value

By default the Hedges' *d* for all possible pairings of the grouping factor are returned as a tidy data frame.

Examples

```
# Calculate Hedges' g for all pairwise comparisons
hedg_g(reading ~ condition, star)

# Report only relative to regular-sized classrooms
hedg_g(reading ~ condition,
star,
ref_group = "reg")
```

```
# Return a vector instead of a data frame
hedg_g(reading ~ condition,
star,
ref_group = "reg",
tidy = FALSE)
```

---

pac *Compute the proportion above a specific cut location*

Description

This rather simple function calls cdfs, to compute the empirical cumulative distribution function for all levels of the grouping factor, and then calculates the proportion of the sample above any generic point on the scale for all groups. Alternatively only specific proportions can be returned.

Usage

```
pac(formula, data, cut, ref_group = NULL, diff = TRUE, tidy = TRUE)
```

Arguments

`formula`	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
`data`	The data frame that the data in the formula come from.
`cut`	The point(s) at the scale from which the proportion above should be calculated from.
`ref_group`	Optional. If the name of the reference group is provided (must be character and match the grouping level exactly), only the estimates corresponding to the given reference group will be returned.
`diff`	Logical, defaults to `TRUE`. Should the difference between the groups be returned? If `FALSE` the raw proportion above the cut is returned for each group.
`tidy`	Logical. Should the data be returned in a tidy data frame? (see Wickham, 2014). If false, effect sizes returned as a vector.

Value

Tidy data frame (or vector) of the proportion above the cutoff for each (or selected) groups.

Examples

```
# Compute differences for all pairwise comparisons for each of three cuts
pac(reading ~ condition,
star,
cut = c(450, 500, 550))

# Report raw PAC, instead of differences in PAC
pac(reading ~ condition,
star,
cut = c(450, 500, 550),
diff = FALSE)

# Report differences with regular-sized classrooms as the reference group
pac(reading ~ condition,
star,
cut = c(450, 500, 550),
ref_group = "reg")

# Return a matrix instead of a data frame
# (returns a vector if only one cut is provided)
pac(reading ~ condition,
star,
cut = c(450, 500, 550),
ref_group = "reg",
tidy = FALSE)
```

---

parse_form *Parse formula*

Description

Parse formula

Usage

```
parse_form(formula, data, order = TRUE)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
data	The data frame that the data in the formula come from.
order	Logical. Defaults to `TRUE`. Should the groups be ordered according to their mean?

Value

A list of data split by the grouping factor.

pooled_sd *Compute pooled standard deviation*

Description

Compute pooled standard deviation

Usage

```
pooled_sd(formula, data)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.

Examples

```
pooled_sd(math ~ condition, star)
pooled_sd(reading ~ sex, star)
```

pp_annotate *Annotation function to add AUC/V to a given plot*

Description

Annotation function to add AUC/V to a given plot

Usage

```
pp_annotate(formula, data, ref_group = NULL, x, y, text_size,
  theme = "standard")
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
ref_group	Optional character vector (of length 1) naming the reference group to be plotted on the x-axis. Defaults to the highest scoring group.
x	The x-axis location for the text.

y	The y-axis location for the text.
text_size	Size of the text used in the annotation
theme	The theme for the plot.

---

pp_calcs *Produce calculations necessary for pp_plot.*

Description

Produce calculations necessary for pp_plot.

Usage

```
pp_calcs(formula, data, ref_group = NULL, scheme = "ggplot2")
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
ref_group	Optional character vector (of length 1) naming the reference group to be plotted on the x-axis. Defaults to the highest scoring group.
scheme	What color scheme should the lines follow? Defaults to mimic the ggplot2 color scheme. Other options come from the viridisLite package, and must be installed first. These are the same options available in the package: "viridis", "magma", "inferno", and "plasma". These color schemes work well for color blindness and print well in black and white. Alternatively, colors can be supplied manually through a call to `col` (through `...`).

Value

List with appropriate `probs`, name of the reference group, data for the reference group and all other groups, and data for the x/y axes.

pp_plot *Produces the paired probability plot for two groups*

Description

The paired probability plot maps the probability of obtaining a specific score for each of two groups. The area under the curve (auc) corresponds to the probability that a randomly selected observation from the x-axis group will have a higher score than a randomly selected observation from the y-axis group.

Usage

```
pp_plot(formula, data, ref_group = NULL, cut = NULL, cut_table = FALSE,
  grid = NULL, scheme = "ggplot2", annotate = FALSE, refline = TRUE,
  refline_col = NULL, refline_lty = 2, refline_lwd = 2, text = NULL,
  text_x = 0.8, text_y = 0.2, text_size = 1.5, shade = NULL,
  shade_col = NULL, leg = NULL, n_1 = FALSE, theme = "standard", ...)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
ref_group	Optional character vector (of length 1) naming the reference group to be plotted on the x-axis. Defaults to the highest scoring group.
cut	Integer. Optional vector (or single number) of scores used to annotate the plot. If supplied, line segments will extend from the corresponding x and y axes and meet at the PP curve.
cut_table	Logical. Should a data.frame of the cuts and corresponding proportions be returned? Defaults to FALSE.
grid	Logical. Should gridlines behind the plot be displayed according to the theme?
scheme	What color scheme should the lines follow? Defaults to mimic the ggplot2 color scheme. Other options come from the viridisLite package, and must be installed first. These are the same options available in the package: "viridis", "magma", "inferno", and "plasma". These color schemes work well for color blindness and print well in black and white. Alternatively, colors can be supplied manually through a call to `col` (through `...`).
annotate	Logical. Defaults to `FALSE`. When `TRUE` and `leg == "side"` the plot is rendered such that additional annotations can be made on the plot using low level base plotting functions (e.g., arrows). However, if set to `TRUE`, dev.off must be called before a new plot is rendered (i.e., close the current plotting window). Otherwise the plot will be attempted to be rendered in the region designated for the legend). Argument is ignored when `leg != "side"`.

refline	Logical. Defaults to TRUE. Should a diagonal reference line, representing the point of equal probabilities, be plotted?
refline_col	Color of the reference line.
refline_lty	Line type of the reference line.
refline_lwd	Line width of the reference line.
text	Logical. Should the link{auc} and link{v} statistics be displayed on the plot? Defaults to TRUE when there are two groups. Cannot currently be displayed for more than two groups.
text_x	The x-axis location for the text.
text_y	The y-axis location for the text.
text_size	The size of the text to be displayed. Defaults to 2.
shade	Logical. Should the area under the curve be shaded? Defaults to TRUE if there are only two group. Currently it cannot be produced for more than two groups.
shade_col	The color of the shading. Defaults to the second color in the chosen color scheme.
leg	The type of legend to be displayed, with possible values "base", "side", or "none". Defaults to "side", when there are more than two groups and "none" when only comparing two groups. If the option "side" is used the plot is split into two plots, via layout, with the legend displayed in the second plot. This scales better than the base legend (i.e., manually manipulating the size of the plot after it is rendered), but is not compatible with multi-panel plotting (e.g., par(mfrow = c(2, 2)) for a 2 by 2 plot). When producing multi-panel plots, use "none" or "base", the latter of which produces the legend with the base legend function.
n_1	Logical. Should the lines on the legend be displayed when there is only one curve? Defaults to FALSE, and is relevant when values to cut are provided. Forced to TRUE if legend == "side" and no values to cut are supplied.
theme	Visual properties of the plot. There are currently only two themes implemented - a standard plot and a dark theme. If NULL (default), the theme will be produced with a standard white background. If "dark", a dark gray background will be used with white text and axes.
...	Additional arguments passed to plot. Note that it is best to use the full argument rather than partial matching, given the method used to call the plot. While some partial matching is supported (e.g., m for main, it is generally safest to supply the full argument).

Value

The arguments supplied to the plot are silently returned for testing purposes.

Examples

```
# Prouduce default Probability-Probability plot with two groups
dev.off()
pp_plot(math ~ freelunch, star)
```

```
# Suppress shading and effect-size annotation
pp_plot(math ~ freelunch,
star,
shade = FALSE,
text = FALSE)

# Change color of shading & line, line width, and title
pp_plot(math ~ freelunch,
star,
shade_col = grDevices::rgb(0.1, 0.8, 0.2, 0.5),
col = "purple", lwd = 5,
main = "Probability-Probability Plot")

# Change to dark theme
pp_plot(math ~ freelunch, star, theme = "dark")

# Produce default PP plot w/multiple groups
pp_plot(mean ~ grade, seda)

# Change reference group to third grade
pp_plot(mean ~ grade,
seda,
ref_group = "3")
```

`probs` *Compute probabilities from the empirical CDFs of a grouping variable for each group.*

Description

This formula returns the paired probabilities for any

Usage

```
probs(formula, data, center = FALSE)
```

Arguments

`formula`	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
`data`	The data frame that the data in the formula come from.
`center`	Logical. Should the functions be centered prior to plotting?

Value

A matrix of probabilities with separate columns for each group and `rownames` corresponding to the value the paired probabilities are calculated from.

Examples

```
probs(math ~ condition, star)
```

---

qtile_es *Compute effect sizes by quantile bins*

Description

Returns a data frame with the estimated effect size by the provided percentiles. Currently, the effect size is equivalent to Cohen's d, but future development will allow this to vary.

Usage

```
qtile_es(formula, data, ref_group = NULL, qtiles = seq(0, 1, 0.33))
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
ref_group	Optional character vector (of length 1) naming the reference group to be plotted on the x-axis. Defaults to the highest scoring group.
qtiles	The percentiles to split the data by and calculate effect sizes. Essentially, this is the binning argument. Defaults to `seq(0, 1, .33)`, which splits the distribution into thirds (lower, middle, upper). Any sequence is valid, but it is recommended the bins be even. For example `seq(0, 1, .1)` would split the distributions into deciles.

Examples

```
# Compute effect sizes (Cohen's d) by default quantiles
qtile_es(reading ~ condition, star)

# Compute Cohen's d by quintile
qtile_es(reading ~ condition,
star,
qtiles = seq(0, 1, .2))

# Report effect sizes only relative to regular-sized classrooms
qtile_es(reading ~ condition,
star,
ref_group = "reg",
qtiles = seq(0, 1, .2))
```

qtile_mean_diffs *Compute mean differences by various quantiles*

Description

Compute mean differences by various quantiles

Usage

```
qtile_mean_diffs(formula, data, qtiles = seq(0, 1, 0.33))
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
qtiles	Quantile bins for calculating mean differences

Examples

```
qtile_mean_diffs(reading ~ condition, star)

qtile_mean_diffs(reading ~ condition,
star,
qtiles = seq(0, 1, .2))
```

qtile_n *Compute sample size for each quantile bin for each group*

Description

Compute sample size for each quantile bin for each group

Usage

```
qtile_n(formula, data, qtiles = seq(0, 1, 0.33))
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note the grouping variable must only include only two groups.
data	The data frame that the data in the formula come from.
qtiles	Quantile bins for calculating mean differences

Examples

```
qtile_n(reading ~ condition, star)

qtile_n(reading ~ condition,
star,
qtiles = seq(0, 1, .2))
```

---

seda *Portion of the Stanford Educational Data Archive (SEDA).*

Description

The full SEDA dataset contains mean test scores on statewide testing data in reading and math for every school district in the United States. See a description of the data here. The data represented in this package represent a random sample of 10 cases in the full dataset. To access the full data, please visit the data archive in the above link.

Usage

```
seda
```

Format

A data frame with 32625 rows and 8 columns.

leaid Integer. Local education authority identifier.

leaname Character. Local education authority name.

stateabb Character. State abbreviation.

year Integer. Year the data were collected.

grade Integer. Grade level the data were collected.

subject Character. Whether the data were from reading or mathematics.

mean Double. Mean test score for the LEA in the corresponding subject/grade/year.

se Double. Standard error of the mean.

Source

Sean F. Reardon, Demetra Kalogrides, Andrew Ho, Ben Shear, Kenneth Shores, Erin Fahle. (2016). Stanford Education Data Archive. http://purl.stanford.edu/db586ns4974. For more information, please visit http://seda.stanford.edu.

seg_match *Match segments on a plot*

Description

Given an x and y coordinate, this function will produce segments that extend from -1 to the corresponding xy intersection, with a point at the intersection.

Usage

```
seg_match(x, y, ...)
```

Arguments

x	The x-coordinate.
y	The y-coordinate.
...	Additional parameters passed to segments. Note that whatever parameters are passed here are also passed to points (e.g., col).

Examples

```
plot(1:10, (1:10)^2, type = "l")
seg_match(3, 9)
seg_match(c(6, 8), c(36, 64),
col = c("blue", "green"),
pch = 21,
bg = c("blue", "green"),
lty = 3)
```

star *Data from the Tennessee class size experiment*

Description

These data come from the Ecdat package and represent a cross-section of data from Project STAR (Student/Teacher Achievement Ratio), where students were randomly assigned to classrooms.

Usage

```
star
```

Format

A data frame with 5748 rows and 9 columns.

sid Integer. Student identifier.

schid Integer. School identifier.

condition Character. Classroom type the student was enrolled in (randomly assigned to).

tch_experience Integer. Number of years of teaching experience for the teacher in the classroom in which the student was enrolled.

sex Character. Sex of student: "girl" or "boy".

freelunch Character. Eligibility of the student for free or reduced price lunch: "no" or "yes"

race Character. The identified race of the student: "white", "black", or "other"

math Integer. Math scale score.

reading Integer. Reading scale score.

`themes` *Theme settings*

Description

Parameters for each theme currently implemented.

Usage

```
themes(theme)
```

Arguments

`theme`	The name of the theme.

Value

list with the `par` settings and the primary line color (e.g., white for theme = "dark" and black for theme = "standard").

tpac *Transformed proportion above the cut*

Description

This function transforms calls to pac into standard deviation units. Function assumes that each distribution is distributed normally with common variances. See Ho & Reardon, 2012

Usage

```
tpac(formula, data, cut, ref_group = NULL, diff = TRUE, tidy = TRUE)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
data	The data frame that the data in the formula come from.
cut	The point at the scale from which the proportion above should be calculated from.
ref_group	Optional. If the name of the reference group is provided (must be character and match the grouping level exactly), only the estimates corresponding to the given reference group will be returned.
diff	Logical, defaults to `TRUE`. Should the difference between the groups be returned? If `FALSE` the raw proportion above the cut is returned for each group.
tidy	Logical. Should the data be returned in a tidy data frame? (see Wickham, 2014). If false, effect sizes returned as a matrix or vector (depending on other arguments passed).

Value

A tidy data frame (or vector) of the transformed proportion above the cutoff. Optionally (and by default) all pairwise comparisons are calculated and returned.

Examples

```
# Compute transformed PAC differences for all pairwise comparisons
# for each of three cuts
tpac(reading ~ condition,
star,
cut = c(450, 500, 550))

# Report raw transformed PAC, instead of differences in transformed PAC
tpac(reading ~ condition,
star,
cut = c(450, 500, 550),
diff = FALSE)
```

```
# Report transformed differences with regular-sized classrooms as the
# reference group
tpac(reading ~ condition,
star,
cut = c(450, 500, 550),
ref_group = "reg")

# Return a matrix instead of a data frame
# (returns a vector if only one cut is provided)
tpac(reading ~ condition,
star,
cut = c(450, 500, 550),
ref_group = "reg",
tidy = FALSE)
```

---

v *Calculate the V effect size statistic*

Description

This function calculates the effect size V, as discussed by Ho, 2009. The V statistic is a transformation of auc, interpreted as the average difference between the distributions in standard deviation units.

Usage

```
v(formula, data, ref_group = NULL, tidy = TRUE)
```

Arguments

formula	A formula of the type `out ~ group` where `out` is the outcome variable and `group` is the grouping variable. Note this variable can include any arbitrary number of groups.
data	The data frame that the data in the formula come from.
ref_group	Optional. If the name of the reference group is provided (must be character and match the grouping level exactly), only the estimates corresponding to the given reference group will be returned.
tidy	Logical. Should the data be returned in a tidy data frame? (see Wickham, 2014). If false, effect sizes returned as a vector.

Value

By default the V statistic for all possible pairings of the grouping factor are returned as a tidy data frame. Alternatively, a vector can be returned, and/or only the V corresponding to a specific reference group can be returned.

Examples

```
free_reduced <- rnorm(800, 80, 20)
pay <- rnorm(500, 100, 10)
d <- data.frame(score = c(free_reduced, pay),
frl = c(rep("free_reduced", 800),
rep("pay", 500)))

v(score ~ frl, d)
# Compute V for all pairwise comparisons
v(reading ~ condition, star)

# Specify regular-sized classrooms as the reference group
v(reading ~ condition,
star,
ref_group = "reg")

# Return a vector instead of a data frame
v(reading ~ condition,
star,
ref_group = "reg",
tidy = FALSE)
```

Index