Package 'VizTest'

March 3, 2026

Title Optimal Confidence Intervals for Visual Testing

Version 0.7

Description Identifies the optimal confidence level to represent the results of a set of pairwise tests as suggested by Armstrong and Poirier (2025) <doi:10.1017/pan.2024.24>.

Depends R (>= 4.1.0)

Imports dplyr, emmeans, ggplot2, ggtext, HDInterval, multcomp, tidyr

Suggests carData, collapse, factorplot (>= 1.3), forcats, ggnewscale, ggsignif, knitr, lme4, marginaleffects, multcompView, mvtnorm, patchwork, psre, rmarkdown, sandwich, testthat (>= 3.0.0), wooldridge

VignetteBuilder knitr

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Dave Armstrong [aut, cre] (ORCID: <https://orcid.org/0000-0002-9358-2489>), William Poirier [aut] (ORCID: <https://orcid.org/0000-0002-3274-1351>)

Maintainer Dave Armstrong <davearmstrong.ps@gmail.com>

Repository CRAN

Date/Publication 2026-03-03 10:00:31 UTC

Contents

`gen_z` *Calculate z-score for Confidence Interval Overlap*

Description

Calculates the z-score required such that confidence intervals do not overlap under the null hypothesis withe a specified probability.

Usage

```
gen_z(b, v, alpha = 0.05, df = Inf, ...)
```

Arguments

`b`	A vector of estiamtes
`v`	The variance-covariance matrix for `b`.
`alpha`	The desired probability at which the confidence intervals do not overlap under the null hypothesis.
`df`	Degrees of freedom for the t-distribution, defaults to `Inf` indicating a normal distribution.
`...`	Other arguments passed down, currently not implemented.

Value

A list with two elements: `ave_z`: A data frame with one row for each estimate in `b` and the following variables:

- `vij`: observation number
- `s_zb`: standard deviation of the z-scores across all pairs of intervals containing that estimate.
- `min_zb`, `max_zb`: The minimum and maximum z-scores for the pairs of intervals containing that estimate.
- `zb`: The mean z-score for the pairs of intervals containing that estimate.

- ci: The confidence level corresponding to zb. all_z: A data frame with one row for each pair of estimates in b and the following variables:
- i, j: The indices of the two estimates in the pair.
- s_i, s_j: The standard errors of the two estimates in the pair.
- theta: The ratio of the standard errors of the two estimates.
- rho: The correlation between the two estimates.
- zb: The z-score for the pair of estimates.
- ci : The confidence level corresponding to zb.
- olap_ave The probability that the two intervals do not overlap under the null hypothesis.
- olap_84 The probability that two 84% confidence intervals for the estimates in the pair would not overlap under the null hypothesis.

References

Harvey Goldstein and Michael J.R. Healy. (1995) "The Graphical Presentation of A Collection of Means." Journal of the Royal Statistical Society, Series A 158(1): 175-177 doi:10.2307/2983411. David Afshartous and Richard A. Preston. (2010) "Confidence Intervals for Dependent Data: Equating Non-overlap with Statistical Significance." Computational Statistics and Data Analysis 54: 2296-2305 doi:10.1016/j.csda.2010.04.011

Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + hp + disp + vs, data=mtcars)
gen_z(coef(mod), vcov(mod))
```

---

geom_forestpoint *Forest plot points with precision-weighted squares and summary diamonds*

Description

geom_forestpoint() draws the central markers in a forest plot: non-summary rows are rendered as **squares** whose area is controlled by the ggplot2 size aesthetic (typically proportional to precision), while summary rows are rendered as **diamonds** whose width reflects the confidence interval and whose height is derived from that width and bounded by row spacing.

Usage

```
geom_forestpoint(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
```

```
  ...,
  diamond_aspect = 1,
  diamond_row_frac = 0.4,
  diamond_min_frac = 0.06,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes().
data	The data to be displayed in this layer. If NULL, the data are inherited from the plot data as specified in the call to ggplot2::ggplot().
stat	Statistical transformation to use. Defaults to "identity".
position	Position adjustment. Defaults to "identity".
...	Additional arguments passed to the underlying GeomForestPoint ggproto object (e.g., color, fill, linewidth).
diamond_aspect	Numeric. Controls how strongly the diamond height increases with confidence interval width. Larger values produce taller diamonds.
diamond_row_frac	Numeric in (0, 1). Maximum fraction of the vertical row spacing that the diamond half-height may occupy, preventing overlap with adjacent rows.
diamond_min_frac	Numeric in (0, 1). Minimum fraction of the row spacing used as the diamond half-height, ensuring visibility for very narrow confidence intervals.
na.rm	Logical. If TRUE, silently removes missing values.
show.legend	Logical or NA. Whether this layer should be included in the legend.
inherit.aes	Logical. If FALSE, the layer does not inherit aesthetic mappings from the parent plot.

Details

This geom is designed to work with standard ggplot2 scales (e.g., scale_size_area()) and pairs naturally with geom_linerange() for confidence intervals and geom_foreststripe() for background striping.

For non-summary rows, the square side length is derived from the ggplot2 size aesthetic (in mm units), so users can control point sizing using standard size scales such as ggplot2::scale_size_area().

For summary rows, the diamond width is determined by the supplied confidence interval (xmin/xmax), and the height is computed as a bounded function of that width. The diamond height **does not** use the size aesthetic.

Value

A ggplot2 layer object that can be added to a plot.

`geom_foreststripe` *Draw alternating row stripes for forest plots and forest tables*

Description

`geom_foreststripe()` adds alternating horizontal background bands ("zebra striping") to forest plots or forest tables. The stripes are drawn using the y-axis scale (rather than the data itself), which makes the geom robust to missing rows, summary rows, and faceting.

Usage

```
geom_foreststripe(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  n_cols,
  col_gap = 1,
  start = 2L,
  fill = "grey92",
  colour = NA,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `ggplot2::aes()`. Usually left empty; the geom does not require data-driven aesthetics.
`data`	The data to be displayed in this layer. If `NULL`, the layer inherits the plot data; only the y scale is used.
`position`	Position adjustment. Defaults to `"identity"`.
`...`	Additional arguments passed to the underlying `GeomForestStripe`.
`n_cols`	Integer. Number of table columns. Used to determine the horizontal extent of the stripes.
`col_gap`	Numeric. Spacing between table columns on the x axis.
`start`	Integer. Index of the first row to stripe (counting from the top of the plot). Defaults to `2L`, so striping begins on the second row.
`fill`	Fill colour for the stripes.
`colour`	Border colour for the stripes. Defaults to `NA` (no border).
`na.rm`	Logical. If `TRUE`, silently ignores missing values.
`show.legend`	Logical. Whether this layer should be included in the legend. Defaults to `FALSE`.
`inherit.aes`	Logical. If `FALSE`, the layer does not inherit aesthetic mappings from the parent plot.

Details

This geom is designed to work with both the forest figure panel (estimates and confidence intervals) and the forest table panel produced by `geom_foresttable()`. Because stripes are computed from the panel scales, the two panels remain visually aligned when composed with **patchwork**.

The stripes are computed from the panel's y range rather than the data rows. For discrete y scales, this corresponds to the integer row positions used by ggplot2. Every other row is selected starting at `start`.

Horizontal extents are derived from the panel x range, allowing the stripes to span both table columns and forest-plot panels without requiring explicit xmin/xmax aesthetics.

Value

A ggplot2 layer object that can be added to a plot.

See Also

geom_foresttable(), geom_forestpoint(), gg_forest()

`geom_foresttable` *Render a forest-plot table panel as a ggplot2 layer*

Description

`geom_foresttable()` builds a "table" panel for forest plots by expanding the input data to long form (one row per *forest row* × *table column*) and drawing formatted cell text at fixed x positions. Column headers are typically added with `scale_x_foresttable()` (top axis tick labels), which makes the table align cleanly with a forest plot when composing with **patchwork**.

Usage

```
geom_foresttable(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  cols,
  fmt = NULL,
  fmt_default = as.character,
  col_gap = 1,
  col_align = "left",
  col_nudge = 0,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes(). If not supplied, the layer will map x, label, and hjust internally so users generally only need to map y.
data	The data to be displayed in this layer. If NULL, the layer inherits the plot data. If a data frame is provided, it is expanded to long form immediately. If a function is provided, it is used as-is (advanced).
position	Position adjustment. Defaults to "identity".
...	Additional arguments passed to the underlying text geom (GeomForestTableText, which inherits from ggplot2::GeomText), such as colour, family, or fontface.
cols	Character vector of column names to print in the table. Must be non-empty.
fmt	Optional named list of formatting functions. Each function should take a single value and return a length-1 character (or coercible) result. Names should correspond to entries of cols.
fmt_default	Default formatting function used for columns not present in fmt. Defaults to base::as.character().
col_gap	Numeric. Spacing between adjacent table columns on the x axis.
col_align	Character vector of alignments for each column in cols. Each entry must be one of "left", "center", or "right". Length 1 is recycled to length(cols).
col_nudge	Numeric vector of per-column horizontal nudges (in x-axis units) applied to the column positions. Length 1 is recycled to length(cols).
na.rm	Logical. If TRUE, silently removes missing values.
show.legend	Logical. Whether this layer should be included in legends. Defaults to FALSE.
inherit.aes	Logical. If FALSE, the layer does not inherit aesthetic mappings from the parent plot.

Details

The cols argument controls which columns are printed. Formatting can be customized via fmt, a named list of functions (one per column) that convert cell values to character strings.

geom_foresttable() uses ggplot2's feature where the data argument may be a function: when data = NULL, this layer supplies a data function that receives the plot data and expands it to long form while preserving all original columns (including faceting variables and the y mapping). This makes the geom compatible with faceting and grouping while keeping the user-facing API simple.

Column positions are computed as seq_along(cols) * col_gap + col_nudge. Horizontal justification is determined by col_align (left/center/right).

Value

A ggplot2 layer object that can be added to a plot.

See Also

scale_x_foresttable(), gg_forest()

get_letters *Get Letters for Multiple Comparisons*

Description

Gets the letter matrix for a compact letter display. This can be passed to the `letter_plot()` function from the `psre` package to produce plots of confidence intervals with a letter display.

Usage

```
get_letters(x = NULL, ...)
```

Arguments

`x`	An object that can be one of the following classes: an object of class `glht` produced by `glht()` from the `multcomp` package, an object of class `emmGrid` produced by the `emmeans` package, or a list with elements `est` - a vector of estimates and `var` - a variance-covariance matrix for the estimates.
`...`	Additional arguments passed down either to `cld` if the object is an `emmGrid` class or `summary.glht` if the object is a `glht` class. If `x` is a list of estimates and variances, it will be converted to an `emmGrid` object internally and the `emmGrid` method dispatched, `...` will be passed to `cld` in that case. Additional arguments for the `summary.glht()` include `test` and a host of others. See the help file for `summary.glht` for examples. Additional arguments for `cld` include `adjust` for multiplicity corrections and others. See `?emmeans:::cld.emmGrid` for options and details.

Value

A logical index indicating which estimates are in which letter group.

get_viztest_levels *Extract representative difficulty levels from a VizTest result*

Description

`get_viztest_levels()` selects one or more representative stimulus levels from a VizTest result object based on empirical difficulty and agreement. Levels are chosen from rows with the maximum agreement probability (`psame`) and can be returned individually (e.g., lowest, highest) or collectively.

Usage

```
get_viztest_levels(
  x,
  method = c("easiest", "all", "lowest", "highest", "middle"),
  ...
)
```

Arguments

x	A VizTest result object containing a component tab, a data frame with at least the columns level, psame, and easy.
method	Character string specifying which level(s) to return. One of "easiest", "all", "lowest", "highest", or "middle".
...	Reserved for future extensions; currently unused.

Details

The function first subsets x$tab to rows achieving the maximum value of psame (ignoring missing values). From this subset:

- "lowest" returns the smallest level.
- "highest" returns the largest level.
- "middle" returns the median level.
- "easiest" returns the level with the maximum easy value.
- "all" returns a named vector containing all four selections.

Value

A numeric value when method is one of "lowest", "highest", "middle", or "easiest". When method = "all", a named numeric vector with elements lowest, highest, middle, and easiest.

See Also

VizTest

Examples

```
data(mtcars)
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$hp <- scale(mtcars$hp)
mtcars$wt <- scale(mtcars$wt)
mod <- lm(qsec ~ hp + wt + cyl, data=mtcars)
v <- viztest(mod)
v

get_viztest_levels(v, "easiest")
```

gg_forest *Build a paired forest plot + companion table (patchwork-ready)*

Description

gg_forest() creates two aligned ggplot objects: (1) a forest plot with confidence intervals and weighted points (including summary diamonds), and (2) a "table" rendered as text in a ggplot panel with column headers on the top axis. The returned objects share the same y values so they can be combined with **patchwork**. This uses lower level functions in the package like geom_forestpooint, geom_foreststripe, geom_foresttable and scale_x_foresttable that could be used to customize the look of the forest plot further.

Usage

```
gg_forest(
  data,
  y,
  x,
  data_cols,
  xmin_std,
  xmax_std,
  size_prop = 1,
  max_size = 15,
  is_summary = FALSE,
  xmin_ici = NULL,
  xmax_ici = NULL,
  vline = NULL,
  ci_colors = c(ici = "gray65", std = "black"),
  use_log_scale = FALSE,
  stripe_table = TRUE,
  stripe_figure = TRUE,
  start_stripe = 3,
  col_nudge = 0,
  table_format_list = NULL,
  col_align = NULL,
  table_header_size = 16,
  table_text_size = 4,
  fig_xlab = "Estimate",
  ...
)
```

Arguments

data	A data frame containing one row per forest row (study and optional summary rows) and all referenced columns.
y	String. Column name used for the forest-row coordinate.

x	String. Column name containing the point estimate.
data_cols	A names character vector of column names to display in the table. The values identify the variable names and the names are the header labels that will be used in the table.
xmin_std, xmax_std	Strings. Column names for the standard CI bounds.
size_prop	Numeric scalar or string column name giving point-size weights. If a numeric scalar, the all points will recieve the same weight.
max_size	Numeric. Maximum size for points (passed to ggplot2::scale_size_area()). This will control the overall size of the points and may need some tweaking depending on the size of the figure and the number of rows.
is_summary	Logical scalar or string column name indicating summary rows. If a logical scalar, all rows will be get square points (if FALSE) or diamond points (if TRUE).
xmin_ici, xmax_ici	Optional strings for the names of the inferential CI bounds.
vline	Numeric. Value at which a dashed vertical line will be drawn. If NULL, no line is drawn.
ci_colors	Named character vector with entries "std" and "ici".
use_log_scale	Logical. If TRUE, uses a log-scaled x axis.
stripe_table	Logical. Draw alternating row stripes in the table.
stripe_figure	Logical. Draw alternating row stripes in the forest plot.
start_stripe	Integer. Row index at which striping begins.
col_nudge	Numeric. Horizontal nudge applied to table text. This is particularly useful for left-aligned columns where the header text and cell text do not line up natively.
table_format_list	Optional named list of formatting functions for data_cols. This defaults to as.character for factors and characters, sprintf("%d") for integers, and sprintf("%.2f") for numerics. The names of the list should match the values of data_cols.
col_align	Optional character vector giving alignment for each table column ("left", "center", "right").
table_header_size	Numeric. Font size for table headers.
table_text_size	Numeric. Font size for table body text.
fig_xlab	Character. X-axis label for the forest plot.
...	Additional arguments passed to geom_forestpoint().

Details

Column arguments are provided as strings and are evaluated safely using .data[[...]].

Value

An object of class "gg_forest": a list with two ggplot objects (forest, table) suitable for composition with **patchwork**.

See Also

geom_forestpoint(), geom_foresttable(), geom_foreststripe(), scale_x_foresttable()

Examples

```
# Load Packages
library(emmeans)
library(VizTest)
library(dplyr)

# Use built-in Esophageal Cancer Data
data(esoph)

# Aggregate data by age group
ag_data <- aggregate(esoph[,c("ncases", "ncontrols")], list(age = esoph$age), sum)

# Turn counts into integerss (not required, but makes printing nicer)
ag_data$ncases <- as.integer(ag_data$ncases)
ag_data$ncontrols <- as.integer(ag_data$ncontrols)

# Make age into unordered factor
ag_data$age <- factor(as.character(ag_data$age),
                      levels=levels(esoph$age))

# Estimate model of prevalence by age and overall (the summary model)
model1 <- glm(cbind(ncases, ncontrols) ~ age,
              data = ag_data, family = binomial())
model_sum <- glm(cbind(ncases, ncontrols) ~ 1,
                 data = ag_data, family = binomial())

# Make data frame of results for plotting using emmeans
fit <- emmeans(model1, "age")
fit_ci <- confint(fit)

# add in original count data
ag_data <- cbind(fit, ag_data[,c("ncases", "ncontrols")])

# turn coefficients and confidence intervals into odds ratio scale
ag_data$or <- exp(ag_data$emmean)
ag_data$lower <- exp(ag_data$asymp.LCL)
ag_data$upper <- exp(ag_data$asymp.UCL)
# Make summary data frame that we can use for plotting
fit_sum <- data.frame(age= "Summary", emmean = coef(model_sum),
  SE = unname(sqrt(vcov(model_sum))), or = exp(coef(model_sum)))
sum_ci <- confint(model_sum)
fit_sum$lower <- exp(sum_ci[1])
fit_sum$upper <- exp(sum_ci[2])
fit_sum$ncases <- sum(ag_data$ncases)
fit_sum$ncontrols <- sum(ag_data$ncontrols)
rownames(fit_sum) <- NULL

# Find the optimal visual testing intervals
```

```
viztest(fit, include_zero=FALSE, make_plot=FALSE, test_level = .05)

# Add inferential CIs to data (not for summary, though)
fit_ici <- confint(fit, level = .75)
ag_data$lower_ici <- exp(fit_ici$asymp.LCL)
ag_data$upper_ici <- exp(fit_ici$asymp.UCL)

# bind together the age-specific and summary data frames for plotting
ag_data <- dplyr::bind_rows(ag_data, fit_sum)

# identify summary row
ag_data$is_sum <- ag_data$age == "Summary"

# add point-size weight
ag_data$pt_size <- 1/ag_data$SE^2

# make age_label such that ages plot smallest at top and summary at bottom
ag_data$age_label <- factor(ag_data$age, levels=rev(ag_data$age))

# Make gg forest plot
out <- gg_forest(ag_data,
    y = "age_label",
    x = "or",
    xmin_std = "lower",
    xmax_std = "upper",
    xmin_ici = "lower_ici",
    xmax_ici = "upper_ici",
    size_prop = "pt_size",
    is_summary = "is_sum",
    use_log_scale = TRUE,
    data_cols = c("Age" = "age_label",
                  "Controls" = "ncontrols",
                  "Cases"="ncases",
                  "OR" = "or"),
    max_size=5,
    table_header_size = 16,
    table_text_size = 5,
    col_nudge=c(-.085, 0,0,0),
    diamond_aspect=15,
    diamond_row_frac = .9)

# print plot
plot(out, widths=1, 1)
```

---

make_annotations *Make Annotations for Significance Brackets*

Description

Makes a list of annotations for significance brakcets produced by the `geom_signif()` function from the `ggsignif` package. The annotations are added for pairs of estimates whose confidence intervals overlap, but the estimates are nonetheless significantly different from each other.

Usage

```
make_annotations(
  obj,
  type = c("auto", "significant", "insignificant", "discrepancies"),
  tol = 0,
  nudge = NULL,
  ...
)
```

Arguments

`obj`	An object of class `viztest` produced by the `viztest()` function.
`type`	Indicates whether annotations are produced for overlapping intervals that are significantly different from each other or not. The `"auto"` option will find the type that produces the fewest annotations. If `type="discrepancies"`, annotations will be made for pairs of estimates whose test results do not correspond with the (non-)overlaps in the confidence intervals.
`tol`	Tolerance for determining whether intervals are close enough to be considered ambiguous. This also plots significance flags for intervals that do not overlap, but the distance between them is smaller than the tolerance. The default is zero, but increasing the value will potentially produce more significance flags.
`nudge`	A vector of the same length as the number of brackets. This will nudge the y-position of the brakcet by the indicated amount. This will be difficult to specify ahead of time, but can be specified to clean up a plot after an initial run.
`...`	Other arguments, currently ignored.

Examples

```
data(chickwts)
chick_mod <- lm(weight ~ feed, data=chickwts)
library(marginaleffects)
chick_preds <- avg_predictions(chick_mod, variables="feed")
b <- coef(chick_preds)
names(b) <- chick_preds$feed
v <- vcov(chick_preds)
chick_vt_data <- make_vt_data(b, v)
chick_vt <- viztest(chick_vt_data, test_level = 0.0001, include_zero=FALSE)
chick_vt

make_annotations(chick_vt, type="discrepancies")
```

`make_diff_template` *Make Template for Pairwise Significance Input*

Description

Provides a template for producing a binary vector indicating whether each pair of estimates has a significant difference.

Usage

```
make_diff_template(
  estimates,
  include_zero = TRUE,
  include_intercept = FALSE,
  ...
)
```

Arguments

`estimates`	A vector of point estimates (ideally, a named vector).
`include_zero`	Logical indicating whether tests against zero should be included.
`include_intercept`	Logical indicating whether the intercept should be included.
`...`	Other arguments passed down, currently not implemented.

Details

The `viztest()` function uses a normal difference of means test to identify whether there is a significant difference or not. While this test could be done with adjustments for multiplicity or robust standard errors of all different kinds, there may be times when the user would prefer to identify the significant differences manually. The `viztest()` function internally reorders the estimates from largest to smallest so this function does that and then prints the pairs that will correspond with the visual testing grid search being done by `viztest()`.

Please note that the `include_zero` and `include_intercept` arguments should be set the same here as they are in your call to `viztest()`. If they are not, `viztest()` will stop because the results from the comparison of confidence intervals will have different dimensions than the differences that are manually provides.

Value

A two-column data frame containing the names of the larger and smaller parameters in the appropriate order. This can be used to identify the appropriate order in which to specify the `sig_diffs` argument to `viztest()`.

Examples

```
make_diff_template(estimates = c(e1 = 2, e2 = 1, e3 = 3))
```

make_vt_data *Make custom visual testing data*

Description

Makes custom visual testing objects that can be used as input to the `viztest()` function. This is useful in the case where `coef()` and `vcov()` do not function as expected on objects of interest, where the user wants to intervene with some modification to the usual estimates or (more likely) variance-covariance matrix or where normal theory tests may not be as useful (e.g., in the case of simulations of non-normal values). The examples section below shows how this could be leveraged to use a heteroskedasticity-consistent covariance matrix in the test rather than the one returned by `lm()`.

Usage

```
make_vt_data(
  estimates,
  variances = NULL,
  type = c("est_var", "sim"),
  tol = 1e-08,
  ...
)
```

Arguments

`estimates`	A vector of estimates if type is `"est_var"` and or a number of simulations by number of parameters matrix of simulated values if type is `"sim"`.
`variances`	In the case of independent estimates, a vector of variances of the same length as `estimates` if type is `"est_var"`. These will be used as the diagonal elements in a variance-covariance matrix with zero covariances. Alternatively, if type is `"est_var"`, this could be a variance-covariance matrix, with the same number of rows and columns as there are elements in the `estimates` vector. If type is `"sim"`, variances should be `NULL`, but will be disregarded in any event. Also, note, these should be variances of the estimates (e.g., squared standard errors) and not raw variances from the data.
`type`	Indicates the type of input data either estimates with variances or a variance-covariance matrix or data from a simulation.
`tol`	Tolerance for evaluation of symmetry and positive definiteness.
`...`	Other arguments passed down, currently not implemented.

Value

1. If the input is a vector of parameter estimates and a variance-covariance matrix, then a list with estimates and a variance-covariance matrix of class `"vtcustom"` is returned. In this case, the functionms `coef.vtcustom()` and `vcov.vtcustom()` are used to extract the coefficients and variance-covariance matrix in a way that will work with `viztest.default()`.

2. If the input is a matrix of simulation draws, an object of class "vtsim" that has a single element - the data giving the draws from the simulation is returned. In this case, viztest.vtsim() does the relevant testing.

Examples

```
data(mtcars)
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$hp <- scale(mtcars$hp)
mtcars$wt <- scale(mtcars$wt)
mod <- lm(qsec ~ hp + wt + cyl, data=mtcars)
V <- sandwich::vcovHC(mod, "HC3")
vtdat <- make_vt_data(coef(mod), V)
viztest(vtdat,
        test_level = .025,
        include_intercept = FALSE,
        include_zero = FALSE)
```

---

plot.gg_forest *Plotting method for* gg_forest *objects.*

Description

Uses patchwork to combine the forest plot and table components.

Usage

```
## S3 method for class 'gg_forest'
plot(x, ..., widths = c(1.4, 1))
```

Arguments

x	An object of class "gg_foest", produces with the gg_forest() function.
...	Other arguments passed to theme() to control the appearance of the combined plot. For example, you might want to use legend.position = "none" if you don't want a legend for the CI colors.
widths	Numeric vector of length 2 giving the relative widths of the table and forest plot components. The default is c(1.4, 1), which gives the table slightly more width than the forest plot.

Value

A ggplot object combining the forest plot and table components.

plot.viztest *Plot Method for viztest Objects*

Description

Plots the output of viztest objects with optional reference lines

Usage

```
## S3 method for class 'viztest'
plot(
  x,
  ...,
  add_test_level = TRUE,
  ref_lines = "none",
  viz_diff_thresh = 0.02,
  make_plot = TRUE,
  level = c("ce", "max", "min", "median"),
  trans = I,
  est_point_args = list(color = "black", size = 2),
  opt_ci_args = list(),
  test_ci_args = list(),
  ref_line_args = list(color = "gray75", linetype = 3),
  scale_linewidth_args = list(values = c(3.5, 0.5)),
  scale_color_args = list(values = c("gray75", "black")),
  overall_theme = theme_bw,
 theme_arg = list(legend.position = "top", plot.caption = element_textbox_simple(width =
    unit(1, "npc"), halign = 0, margin = margin(1, 0, 0, 0, "lines"))),
  remove_caption = FALSE
)
```

Arguments

x	Object to be plotted, should be of class viztest
...	Other arguments passed down. Currently not implemented.
add_test_level	Add the (1-test level) confidence interval to the plot. For this to work, you must have specified add_test_level in the call to viztest() so that the appropriate confidence intervals can be calculated.
ref_lines	Reference lines to be plotted - one of "all", "ambiguous", "none". This could also be a vector of stimulus names to plot - they should be the same as the names of the estimates in x$est. See details for explanation.
viz_diff_thresh	Threshold for identifying visual difficulty, see details.
make_plot	Logical indicating whether the plot should be constructed or the data returned.
level	Level at which to plot the estimates. Accepts both numeric entries or one of "ce", "max", "min", "median" - defaults to "ce", the cognitively easiest level.

trans	A function to transform the estimates and their confidence intervals like plogis.
est_point_args	A list of arguments to be passed to geom_point() that plots the point estimates.
opt_ci_args	A list of arguments to be passed to geom_linerange() to plot the optimal visual testing intervals.
test_ci_args	A list of arguments to be passed to geom_linerange() to plot the (1-test level) confidence intervals.
ref_line_args	A list of arguments to be passed to geom_segment() to plot the reference lines.
scale_linewidth_args	A list of arguments to be passed to scale_linewidth_manual() to change the thickness of the confidence intervals.
scale_color_args	A list of arguments to be passed to scale_color_manual() to change the default color of the different confidence intervals when add_test_level=TRUE.
overall_theme	A theme function that will be passed to the ggplot call before theme(). Default is theme_bw.
theme_arg	A list of arguments to be passed to theme() to modify the theme of the plot.
remove_caption	Logical indicating whether caption should be removed. By default, it is printed to alert the user.

Details

The ref_lines argument identifies what reference lines will be plotted in the figure. For any particular stimulus, the reference lines run along the upper bound of the stimulus from the stimulus location to the most distant stimulus with overlapping confidence intervals. When ref_lines = "all", all lines are plotted, though in displays with many stimuli, this can make for a messy graph. When "ref_lines = ambiguous" is specified, then only the ones that help discriminate in cases where the result might be visually difficult to discern are plotted. A comparison is determined to be visually difficult if the upper bound of the stimulus in question is within viz_diff_thresh times the difference between the smallest lower bound and the largest upper bound. If ref_lines = "non", then none of the reference lines are plotted. Alternatively, you can specify the names of stimuli whose reference lines will be plotted. These should be the same as the names in the data. The viztest() function returns an object est, which contains the data that are used as input to this function. The variable vbl in The est data frame contains the stimulus names.

Value

By default, a ggplot is returned. If make_plot = FALSE, the data for the plot are returned, but the plot is not constructed. If the data are returned, the following variables are in the dataset:

- vbl - The name of the parameter.
- est - The parameter estimate
- se - The standard error of the estimate
- lwr, upr - The inferential confidence bounds being used
- lwr_add, upr_add - The confidence intervals that come from add_level.
- label - Factor giving the parameter names

- stim_start, stim_end - y-axis bounds of the reference line
- bound_start, bound_end - x-axis values for reference lines
- ambiguous - Logical vector indicating whether the comparison is considered "ambiguous".

Examples

```
data(mtcars)
mod2 <- lm(mpg ~ as.factor(cyl) + vs + am + as.factor(gear), data = mtcars)
v <- viztest(mod2)
plot(v, ref_lines="ambiguous") + ggplot2::theme_classic()
```

---

print.viztest *Print Method for viztest Objects*

Description

Prints a summary of the results from the viztest() function.

Usage

```
## S3 method for class 'viztest'
print(x, ..., best = TRUE, missed_tests = TRUE, level = NULL)
```

Arguments

x	An object of class viztest.
...	Other arguments, currently not implemented.
best	Logical indicating whether the results should be filtered to include only the best level(s) or include all levels
missed_tests	Logical indicating whether the tests not represented by the optimal visual testing intervals should be displayed
level	Which level should be used as the optimal one. If NULL, the easiest optimal level will be used. Easiness is measured by the sum of the overlap in confidence intervals for insignificant tests plus the distance between the lower and upper bound for tests that are significant.

Details

The results are printed in such a way that the range of optional levels is produced including the range along with two candidates for the best levels to use - middle and easiest.

Prints the results from the viztest function

Value

Printed results that give the level(s) that correspond most closely with the pairwise test results. The values returned are the smallest, largest, middle and easiest. By default this function also reports the tests that are not captured by the (non-)overlaps in confidence intervals when each different level is used.

---

reorder_forest *Reorder a factor for a forest plot*

Description

Orders levels of `x` by `bvar` (via `fn`), with "Summary" optionally pinned to the bottom (or top).

Usage

```
reorder_forest(
  x,
  bvar,
  fn = mean,
  descending = TRUE,
  summary_bottom = TRUE,
  ...
)
```

Arguments

`x`	Character/factor column to turn into an ordered factor.
`bvar`	Numeric variable used for ordering (e.g. the point estimate).
`fn`	Aggregation function (default `mean`).
`descending`	Logical; if `TRUE` largest value gets the top row.
`summary_bottom`	Logical; if `TRUE` "Summary" is pinned to row 1 (bottom of a ggplot y-axis).
`...`	Other arguments, currently unimplemented.

---

scale_x_foresttable *X scale for forest tables with column headers on the top axis*

Description

`scale_x_foresttable()` constructs a continuous x scale designed for use with `geom_foresttable()`. It places table column headers on the **top axis tick labels**, sets appropriate breaks and limits based on the requested columns, and ensures consistent spacing so that table panels align cleanly with forest plots when composed using **patchwork**.

Usage

```
scale_x_forestttable(
  cols,
  col_labels = NULL,
  col_gap = 1,
  position = "top",
  expand = ggplot2::expansion(mult = c(0, 0)),
  limits = NULL,
  ...
)
```

Arguments

cols	Character vector of column names corresponding to the table columns. Determines the number, order, and spacing of x-axis breaks.
col_labels	Optional character vector giving display labels for each column. If unnamed, must be the same length as cols and is assumed to be in the same order. If named, names are matched to cols.
col_gap	Numeric. Spacing between adjacent columns on the x axis.
position	Character. Position of the x axis; defaults to "top".
expand	Expansion applied to the x scale. Defaults to no expansion (expansion(mult = c(0, 0))).
limits	Optional numeric vector of length 2 giving explicit x-axis limits. If NULL, limits are computed automatically to bracket all columns.
...	Additional arguments passed to ggplot2::scale_x_continuous().

Details

The scale uses numeric x positions internally (one per column), which allows precise control over column spacing and alignment.

This scale is typically paired with geom_forestttable() and a void theme (theme_void()) to create a table-like panel whose headers are rendered as axis labels. Because headers are true axis labels, ggplot2 can align the table panel perfectly with a forest plot panel when combining plots.

Value

A ggplot2 scale object suitable for addition to a plot.

See Also

geom_forestttable(), gg_forest()

`viztest` *Calculate Correspondence Between Pairwise Test and CI Overlaps*

Description

`viztest()` does a grid search over `range_levels` to find the confidence level(s) such that the (non-)overlaps in confidence intervals corresponds as closely as possible with the results of pairwise tests. To the extent that a level is found that accounts for all pairwise tests, confidence bounds at this level can be added to coefficient or marginal effects plots to enable readers to reliably identify estimates that are statistically different from each other.

Usage

```
viztest(
  obj,
  test_level = 0.05,
  range_levels = c(0.25, 0.99),
  level_increment = 0.01,
 adjust = c("none", "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr"),
  cifun = c("quantile", "hdi"),
  include_intercept = FALSE,
  include_zero = TRUE,
  sig_diffs = NULL,
  tol = 1e-08,
  ...
)
```

Arguments

`obj`	A model object (or any object) where `coef()` and `vcov()` return estimates of coefficients and sampling variability.
`test_level`	The type I error rate of the pairwise tests.
`range_levels`	The range of confidence levels to try.
`level_increment`	Step size of increase between the values of `range_levels`.
`adjust`	Multiplicity adjustment to use when calculating the p-values for normal theory pairwise tests.
`cifun`	For simulation results, the method used to calculate the confidence/credible interval either "quantile" (default) or "hdi" for highest density region.
`include_intercept`	Logical indicating whether the intercept should be included in the tests, defaults to `FALSE`.
`include_zero`	Should univariate tests at zero be included, defaults to `TRUE`.

sig_diffs	An optional vector of values identify whether each pair of values is statistically different (1) or not (0). See Details for more information on specifying this value; there is some added complexity here.
tol	Tolerance for evaluation of symmetry and positive definiteness.
...	Other arguments, currently not implemented.

Details

The algorithm first calculates results of a set of pairwise tests. For objects with estimates and a variance-covariance matrix, normal theory tests are calculated. Optionally, these tests can be subjected to a multiplicity adjustment. In the case of simulation results, something akin to p-values are calculated by identifying the probability that one estimate is larger than another. To mimic the way we use p-values in the frequentist case, we subtract the probability of difference from 1, such that smaller values indicate more confidence in the difference. The algorithm then performs a grid search over `range_levels` at increments of `level_increment`. For each candidate level, the confidence intervals for all parameters are calculated. For each pair of estimates, it identifies whether the confidence intervals (or credible intervals if the input is a matrix of Bayesian simulation draws) overlaps. For each candidate level, it calculates the proportion of times where differences are significant/credible and confidence/credible intervals do not overlap or differences are not significant/credible and the intervals do overlap. The main idea is to find the level(s) such that the (non-)overlaps perfectly correspond with whether the differences are significant.

If such a level can be found, a visual inspection of confidence or credible intervals at that level will identify whether a pair of estimates is statistically different or not.

While most of the parameters are straightforward, the `sig_diffs` argument must be specified such that the stimuli are in order from highest to lowest. This is most easily done by using `make_diff_template()` to identify the appropriate order of the comparisons.

Value

A list (of class "viztest") with the following elements:

1. tab: a data frame with results from the grid search. The data frame has four variables: `level` - is the confidence level used in the grid search; `psame` - the proportion of (non-)overlaps that match the normal theory tests; `pdiff` - the proportion of pairwise tests that are statistically significant; `easy` - the ease with which the comparisons are made.
2. pw_tests: A logical vector indicating which tests are significantly significant.
3. ci_tests: A logical vector indicating whether the confidence intervals are disjoint (`TRUE`) or overlap (`FALSE`).
4. combs: The pairwise combinations of stimuli used in the test. Note, the stimuli are reordered from largest to smallest, so the numbers do not represent the position in the original ordering.
5. param_names: A vector of the names of the parameters reordered by size - largest to smallest.
6. L: The lower confidence bounds from the grid search.
7. U: The upper confidence bounds from the grid search.
8. est: A data frame with the variables `vbl` - the parameter name; `est` - the parameter estimate; `se` - the parameter standard error.
9. call: model call

References

David A. Armstrong II and William Poirier. "Decoupling Visualization and Testing when Presenting Confidence Intervals" Political Analysis doi:10.1017/pan.2024.24.

Examples

```
data(mtcars)
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$hp <- scale(mtcars$hp)
mtcars$wt <- scale(mtcars$wt)
mod <- lm(qsec ~ hp + wt + cyl, data=mtcars)
viztest(mod)
```

Index