Package 'VizTest'

March 12, 2025

Title Optimal Confidence Intervals for Visual Testing

Version 0.3

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), dplyr, ggplot2, HDInterval

Suggests carData, collapse, knitr, lme4, marginaleffects, patchwork, rmarkdown, sandwich, tidyr, wooldridge

VignetteBuilder knitr

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2025-03-12 17:40:02 UTC

Contents

---

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"), ...)
```

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.
...	Other arguments passed down, currently not implemented.

Value

An object of class "vtcustom" that takes one of two forms:

1. A list with estimates and a variance-covariance matrix. 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. An object of class "vtsim" that has a single element - the data giving the draws from the simulation.

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.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,
  ...,
  ref_lines = "none",
  viz_diff_thresh = 0.02,
  make_plot = TRUE,
  level = c("ce", "max", "min", "median"),
  trans = I
)
```

Arguments

x	Object to be plotted, should be of class viztest
...	Other arguments passed down. Currently not implemented.
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.

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
- `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.

---

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,
        ...
    )
```

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.
`...`	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.

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