Package 'confinterpret'

October 3, 2017

Type Package

Title Descriptive Interpretations of Confidence Intervals

Version 1.0.0

Description Produces descriptive interpretations of confidence intervals. Includes (extensible) support for various test types, specified as sets of interpretations dependent on where the lower and upper confidence limits sit. Provides plotting functions for graphical display of interpretations.

License AGPL-3

URL https://github.com/jimvine/confinterpret

BugReports https://github.com/jimvine/confinterpret/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests graphics, grDevices, testthat

NeedsCompilation no

Author Jim Vine [aut, cre]

Maintainer Jim Vine <code@jimvine.co.uk>

Repository CRAN

Date/Publication 2017-10-03 11:09:28 UTC

R topics documented:

`confinterpret` *Descriptive interpretations of confidence intervals.*

Description

Produces descriptive interpretations of confidence intervals, depending on the type of test specified by an `interpretation_set`.

Usage

```
confinterpret(ci, interpretation_set, boundaries, comparison_labels = NULL,
  low_to_high = TRUE)
```

Arguments

`ci`	A single row from a matrix of the type returned by `confint()`, containing the confidence interval for the parameter estimate. The two columns provide the lower and upper confidence limits.
`interpretation_set`	List-based object that specifies the boundaries between regions that each of the confidence limits can fall in, and the interpretations to be returned in each of the cases.
`boundaries`	Vector of numbers specifying the values for each of the boundaries defined in the `interpretation_set`. Normally provided in low-to-high order, but see the `low_to_high` parameter for options.
`comparison_labels`	Character vector specifying the labels to be used within the interpretation to describe the comparison. Required if the `interpretation_set` includes a $placeholders entry. Null otherwise.
`low_to_high`	Are the boundaries ordered low-to-high (TRUE) or high-to-low (FALSE)? This can be used to reverse the assessment, including in the cases where only one boundary is supplied. See Details.

Details

Helpful wrapper functions are provided for some commonly used types of test:

Superiority tests `interpret_superiority`

Non-inferiority tests `interpret_noninferiority`

Equivalence tests `interpret_equivalence`

The low_to_high parameter can be set to FALSE to facilitate the situation where the boundaries are ordered high-to-low. This enables the same `interpretation_set` object to be used for both beneficial and harmful outcomes. For an `interpretation_set` that has been defined as if higher numbers are better (for example, proportion of participants recovering from a particular illness after treatment) then the inferiority interpretations will be listed first and the superiority ones last. To use this with a negative outcome (for example, proportion of participants catching an illness after a preventative measure), provide the boundaries in high-to-low order and use `low_to_high = FALSE`. This will also work where a single boundary is specified, and will act to 'reverse' the interpretations.

The use of low_to_high only affects the order of the boundaries (and the regions these implicitly define). It does **not** affect the ordering of the confidence interval: the numerically lower confidence limit should be listed first either way.

Plotting functions are provided to display the results of `confinterpret`. To plot a single result see `plot.interpretation_result`. To plot multiple results on one chart see `plot_interpretation_result_list`.

Value

A list object of class `interpretation_result` with elements stating the interpretation in different formats, plus the parameters used to generate the interpretation.

Examples

```
# Establish a test confidence interval
ci_test <- matrix(c(-0.1,0.1),
                  nrow = 1, dimnames = list("estimate",
                                            c("2.5 %","97.5 %")))
confinterpret(ci_test, interpretations_superiority, 0,
  comparison_labels = c(comparison_intervention = "Treatment as usual",
                        tested_intervention = "New treatment"))
```

`interpretations_equivalence`

Interpretation set for equivalence tests

Description

An `interpretation_set` object used for conducting equivalence tests. A convenient wrapper function, `interpret_equivalence`, is provided, making use of this object.

Usage

```
interpretations_equivalence
```

Format

An object of class `interpretation_set` of length 3.

Details

This `interpretation_set` object contains `placeholders` for descriptive names of the comparison intervention and tested intervention. When used with `confinterpret` these are provided via the `comparison_labels` parameter as a named character vector of length 2, `c(comparison_intervention = "Your control /`
When using the convenience wrapper function, this is handled through the `groups` parameter.

`interpretations_noninferiority`

Interpretation set for non-inferiority tests

Description

An `interpretation_set` object used for conducting non-inferiority tests. A convenient wrapper function, `interpret_noninferiority`, is provided, making use of this object.

Usage

```
interpretations_noninferiority
```

Format

An object of class `interpretation_set` of length 3.

Details

This `interpretation_set` object contains `placeholders` for descriptive names of the comparison intervention and tested intervention. When used with `confinterpret` these are provided via the `comparison_labels` parameter as a named character vector of length 2, `c(comparison_intervention = "Your control /`
When using the convenience wrapper function, this is handled through the `groups` parameter.

interpretations_superiority *Interpretation set for superiority tests*

Description

An `interpretation_set` object used for conducting superiority tests. A convenient wrapper function, `interpret_superiority`, is provided, making use of this object.

Usage

```
interpretations_superiority
```

Format

An object of class `interpretation_set` of length 3.

Details

This `interpretation_set` object contains `placeholders` for descriptive names of the comparison intervention and tested intervention. When used with `confinterpret` these are provided via the `comparison_labels` parameter as a named character vector of length 2, `c(comparison_intervention = "Your control /`
When using the convenience wrapper function, this is handled through the `groups` parameter.

interpretation_result *Interpretation result*

Description

A class to define the result that is returned by an interpretation conducted by `confinterpret`.

Usage

```
interpretation_result(interpretation, ci, interpretation_set,
  interpretation_set_name = deparse(substitute(interpretation_set)),
  boundaries, comparison_labels, low_to_high)
```

Arguments

`interpretation`	A list object from a an `interpretation_set` providing the qualitative interpretation.
`ci`	The confidence interval that was interpreted.
`interpretation_set`	The `interpretation_set` object that was used to conduct the interpretation.

`interpretation_set_name`
: The name of the `interpretation_set` that was used for the interpretation.

`boundaries`
: The boundaries parameter that was used for the interpretation.

`comparison_labels`
: Labels that were used to describe the groups that were compared in the interpretation.

`low_to_high`
: Whether the boundaries were provided in low-to-high or high-to-low order.

Details

The parameters are the ones that were used in conducting the interpretation (typically using `confinterpret` or one of its convenience wrapper functions). See `confinterpret` for more details on how these parameters were used in conducting the interpretation.

Value

A list object of class `interpretation_result` with elements stating the interpretation in different formats (`$interpretation_short`, `$interpretation`, and `$interpretation_md`) and `$parameters`. `$parameters` is list object detailing the parameters that were used to generate the interpretation, and contains `$ci`, `$interpretation_set`, `$interpretation_set_name`, `$boundaries`, `$comparison_labels` and `$low_to_high`.

`interpretation_set` *Interpretation Sets*

Description

A class to define a set of interpretations for confidence intervals, depending on where the lower and upper confidence limits sit.

Usage

```
interpretation_set(boundary_names, placeholders = NULL, interpretations)
```

Arguments

`boundary_names`
: Character vector of boundary names. The length of this vector (i.e., the number of boundary names listed) determines the number of boundaries for use with this interpretation_set, which also determines the number of interpretations that must be provided (see Details).

`placeholders`
: Vector of named character elements, where each item contains a string that is used within the interpretations as a placeholder, enabling a specific value to be substituted. Can be null.

`interpretations`
: An ordered list of interpretations, one for each valid combination of confidence limits. See Details for information on the number and expected ordering of `interpretations` in a given `interpretation_set`.

Details

The set of boundaries specified in an `interpretation_set` can be thought of as establishing a number of regions within which the lower and upper confidence limits can sit. There is 1 more region than the number of boundaries, since the set of regions is effectively 'less than boundary 1', 'between boundary 1 : n-1 and boundary 2 : n' and 'above boundary n'.

The valid combinations are those where the upper confidence limit is in a region greater than or equal to the region of the lower confidence limit. This establishes `sum(1 : n)` valid combinations, where n is the number of regions (i.e., the number of boundaries + 1). An interpretation needs to be provided for each of these combinations.

Interpretations are provided in order. The order is based on first specifying all the cases where the lower confidence limit is in the bottom region, and each of the regions for the upper confidence limit (again, starting from the bottom and increasing to the top); next come all of the cases where the lower confidence limit is in the second-from-bottom region (in this case the valid regions for the upper confidence limit will start at the second-from-bottom and go up to the top region); and so on. So for a 2 region (1 boundary) situation, the interpretations should be provided in the following order:

Order	Lower confidence level	Upper confidence level
1	Region 1	Region 1
2	Region 1	Region 2
3	Region 2	Region 2

For a 3 region (2 boundary) situation, the interpretations should be provided in this order:

Order	Lower confidence level	Upper confidence level
1	Region 1	Region 1
2	Region 1	Region 2
3	Region 1	Region 3
4	Region 2	Region 2
5	Region 2	Region 3
6	Region 3	Region 3

Values for `placeholders` can be specified to enable sections of text in interpretations to be replaced automatically. This can be used, for example, to allow names or descriptions of intervention to be passed to `confinterpret`, so that these can be returned in the final interpretation rather than a generic description. `confinterpret` uses `gsub` with `fixed=TRUE` to do substitutions for `placeholders` entries, so values should be selected that will match accordingly (and will not match extra items). Use of a non-alphanumeric character within a placeholder can help to reduce accidental matches.

A `plot` method is provided for `interpretation_set` objects. See `plot.interpretation_set` for details.

A `print` method is provided for `interpretation_set` objects.

`interpret_equivalence` *Equivalence test interpretations of confidence intervals.*

Description

Conduct equivalence tests on confidence intervals using a standard set of interpretations. Takes a confidence interval around an effect size measure, for example from the results from a randomised controlled trial comparing the outcome for an intervention group to a control group.

Usage

```
interpret_equivalence(ci, actual_null = 0, eq_margin = 0.1,
  groups = c("Control intervention", "Test intervention"),
  beneficial_outcome = TRUE)
```

Arguments

`ci`	A single row from a matrix of the type returned by `confint()`, containing the confidence interval for the parameter estimate. The two columns provide the lower and upper confidence limits.
`actual_null`	The value that precisely zero difference would have in the parameter being examined. For an absolute measure this will typically be 0. For a relative measure it will typically be 1. This is the starting point that the `eq_margin` is applied to in order to establish the region for comparison.
`eq_margin`	Numerical value specifying the equivalence margin to be used.
`groups`	A character vector of length 2 containing short descriptive names of the groups being compared, such as the names of the interventions being compared if the confidence interval is derived from an outcome effect size measure in a randomised controlled trial. Give the name of the intervention given to the comparison or control group first and the new or tested intervention second.
`beneficial_outcome`	Is the outcome to be treated as beneficial (i.e., a higher value of the outcome is superior)? For harmful outcomes (where lower numbers are better), set this to FALSE. If, for example, the outcome is measuring something like prevalence of patients recovering from a disease, that is likely to be beneficial; if it is measuring the prevalence of patients falling ill with a disease it is likely to be **not** beneficial.

Details

Equivalence tests can be specified in analysis plans when the aim is to check whether a new intervention performs the same as an old one. The test is most appropriate where the aim is not to result in a better or worse outcome, but the same as under the previous intervention. One particular use is for testing new versions of medicines, such as generic versions of drugs after the branded version's patent protection has ended. In this situation, if the generic manufacturer is correctly producing the medicine it should result in neither better nor worse outcomes than the branded medicine.

When conducting equivalence tests, an equivalence margin is specified. This is the region around a true null (i.e., no difference) result that is deemed to be within a reasonable range. It is commonly selected to include the range of differences that would be of no practical significance.

You are able to supply descriptive names of the interventions being compared, and these will be inserted into the resultant interpretation. If the comparison / baseline intervention does not have a convenient name (such as "Placebo"), some of these might be suitable:

- "Business as usual"
- "Treatment as usual"
- "No intervention"

(Whilst these may work well as short descriptions for outputting from this function, in your reporting you will still normally want to provide information about what exactly those in a comparison group got.)

This function is provided in the form of a convenience wrapper for `confinterpret`, using `interpretations_equivalence` as its `interpretation_set`.

Value

A list object of class `interpretation_result` with elements stating the interpretation in different formats, plus the parameters used to generate the interpretation.

Examples

```
# Establish a test confidence interval
ci_test <- matrix(c(-0.1, 0.1),
                  nrow = 1, dimnames = list("estimate",
                                            c("2.5 %", "97.5 %")))
interpret_equivalence(ci_test, 0, 0.2, c("Treatment as usual", "New treatment"))
```

`interpret_noninferiority` *Non-inferiority test interpretations of confidence intervals.*

Description

Conduct non-inferiority tests on confidence intervals using a standard set of interpretations. Takes a confidence interval around an effect size measure, for example from the results from a randomised controlled trial comparing the outcome for an intervention group to a control group.

Usage

```
interpret_noninferiority(ci, actual_null = 0, ni_margin = 0.1,
  groups = c("Control intervention", "Test intervention"),
  beneficial_outcome = TRUE)
```

Arguments

`ci`	A single row from a matrix of the type returned by `confint()`, containing the confidence interval for the parameter estimate. The two columns provide the lower and upper confidence limits.
`actual_null`	The value that precisely zero difference would have in the parameter being examined. For an absolute measure this will typically be 0. For a relative measure it will typically be 1. This is the starting point that the `ni_margin` is applied to in order to establish the point for comparison.
`ni_margin`	Numerical value specifying the non-inferiority margin to be used. Provided as a positive number; the value of `beneficial_outcome` defines whether it is added to or subtracted from the `actual_null` value to position the boundary. See Details.
`groups`	A character vector of length 2 containing short descriptive names of the groups being compared, such as the names of the interventions being compared if the confidence interval is derived from an outcome effect size measure in a randomised controlled trial. Give the name of the intervention given to the comparison or control group first and the new or tested intervention second.
`beneficial_outcome`	Is the outcome to be treated as beneficial (i.e., a higher value of the outcome is superior)? For harmful outcomes (where lower numbers are better), set this to FALSE. If, for example, the outcome is measuring something like prevalence of patients recovering from a disease, that is likely to be beneficial; if it is measuring the prevalence of patients falling ill with a disease it is likely to be **not** beneficial.

Details

Non-inferiority tests are typically specified in analysis plans where a new intervention is being compared to an existing one, especially if it has some benefit other than the effect being measured. For example, the new intervention might be cheaper than the old one, or have fewer side effects. In these circumstances, the new intervention may not need to prove itself more effective than the old one, but just to be not substantially worse - i.e., non-inferior.

When conducting non-inferiority tests, a non-inferiority margin is defined. This is effectively the leeway of small, practically insignificant differences by which the new intervention is allowed to under-perform the old one and still be considered non-inferior.

The non-inferiority margin is defined as being a small amount on the inferior side of an actual null result. If using `beneficial_outcome = TRUE` (the default), the non-inferiority margin will extend below `actual_null`; if `beneficial_outcome = FALSE` it extends above it.

You are able to supply descriptive names of the interventions being compared, and these will be inserted into the resultant interpretation. If the comparison / baseline intervention does not have a convenient name (such as "Placebo"), some of these might be suitable:

- "Business as usual"
- "Treatment as usual"
- "No intervention"

(Whilst these may work well as short descriptions for outputting from this function, in your reporting you will still normally want to provide information about what exactly those in a comparison group got.)

This function is provided in the form of a convenience wrapper for `confinterpret`, using `interpretations_noninferiori` as its `interpretation_set`.

Value

A list object of class `interpretation_result` with elements stating the interpretation in different formats, plus the parameters used to generate the interpretation.

Examples

```
# Establish a test confidence interval
ci_test <- matrix(c(-0.05, 0.05),
                  nrow = 1, dimnames = list("estimate",
                                            c("2.5 %", "97.5 %")))
interpret_noninferiority(ci_test, 0, 0.1, c("Treatment as usual",
                                            "New treatment"))
```

`interpret_superiority` *Superiority test interpretations of confidence intervals.*

Description

Conduct superiority tests on confidence intervals using a standard set of interpretations. Takes a confidence interval around an effect size measure, for example from the results from a randomised controlled trial comparing the outcome for an intervention group to a control group.

Usage

```
interpret_superiority(ci, null_value = 0, groups = c("Control intervention",
  "Test intervention"), beneficial_outcome = TRUE)
```

Arguments

`ci`	A single row from a matrix of the type returned by `confint()`, containing the confidence interval for the parameter estimate. The two columns provide the lower and upper confidence limits.
`null_value`	The value that precisely zero difference would have in the parameter being examined. For an absolute measure this will typically be 0. For a relative measure it will typically be 1. For superiority tests this is the point value that the confidence interval is compared at.

`groups` A character vector of length 2 containing short descriptive names of the groups being compared, such as the names of the interventions being compared if the confidence interval is derived from an outcome effect size measure in a randomised controlled trial. Give the name of the intervention given to the comparison or control group first and the new or tested intervention second.

`beneficial_outcome` Is the outcome to be treated as beneficial (i.e., a higher value of the outcome is superior)? For harmful outcomes (where lower numbers are better), set this to FALSE. If, for example, the outcome is measuring something like prevalence of patients recovering from a disease, that is likely to be beneficial; if it is measuring the prevalence of patients falling ill with a disease it is likely to be **not** beneficial.

Details

You are able to supply descriptive names of the interventions being compared, and these will be inserted into the resultant interpretation. If the comparison / baseline intervention does not have a convenient name (such as "Placebo"), some of these might be suitable:

- "Business as usual"
- "Treatment as usual"
- "No intervention"

(Whilst these may work well as short descriptions for outputting from this function, in your reporting you will still normally want to provide information about what exactly those in a comparison group got.)

This function is provided in the form of a convenience wrapper for `confinterpret`, using `interpretations_superiority` as its `interpretation_set`.

Value

A list object of class `interpretation_result` with elements stating the interpretation in different formats, plus the parameters used to generate the interpretation.

Examples

```
# Establish a test confidence interval
ci_test <- matrix(c(-0.1, 0.1),
                  nrow = 1, dimnames = list("estimate",
                                            c("2.5 %", "97.5 %")))
interpret_superiority(ci_test, 0, c("Treatment as usual", "New treatment"))
```

label_ontop_boundaries

Label the boundaries on top of the plot.

Description

If plotting values or ranges may want to call this directly last to ensure it is on top, and specify no labels in the canvas plotting call.

Usage

```
label_ontop_boundaries(boundaries, extra_boundaries = NULL)
```

Arguments

boundaries	Named vector of numerical values of where boundaries should be drawn.
extra_boundaries	Names optional.

plot.interpretation_result

Plot an interpretation_result, as returned by confinterpret()

Description

Produces a diagram that illustrates the confidence interval that was interpreted using confinterpret against a background illustrating the interpretation_set that it was the basis for the interpretation.

Usage

```
## S3 method for class 'interpretation_result'
plot(x, extra_boundaries = NULL,
  estimate = NULL, boundary_values = TRUE, boundary_label_pos = "below",
  interpretation_label_pos = "right", x_axis_pos = "below",
  y_axis_pos = "none", inner_margin = c(-0.1, 0.05, -0.1, 0.05),
  edge_margin = c(0, 0.02, 0, 0.02), edge_type = "gradient",
  interval_type = "norm", interval_value_labels = TRUE,
  estimate_value_labels = TRUE, plot_estimate_marks = TRUE,
  estimate_mark_points = c(0, 0.05, 0, -0.05), ...)
```

Arguments

x	An interpretation_result object, of the type returned by confinterpret.
extra_boundaries	A vector of numerical values specifying the position for displaying additional boundaries, not specified in the interpretation_set. May optionally be named values; if named, the names will be labelled on the plot axis.
estimate	Estimate value that the interval relates to. If not specified, a default of the central point between the two ends of the interval will be assumed.
boundary_values	A logical value indicating whether the values should be appended to the boundaries' names.
boundary_label_pos	Where to put the boundary labels. Options are c("below", "above", "on top", "none"). If you are planning to plot values on the canvas and want the boundary labels on top then you may want to choose "none" and make a call to label_ontop_boundaries() after plotting values.
interpretation_label_pos	Options are c("right", "left", "none")
x_axis_pos	Location of a numerical x axis. Options are c("none", "below", "above").
y_axis_pos	Location of a numerical y axis. Default "none" will almost always be right. Options are c("none", "left", "right").
inner_margin	Numerical vector of the form c(bottom, left, top, right), which gives the amount of inner margin to be added, expressed as a proportion of the plotted area. This is space designed to be past any plotted objects but before the edging (defined separately via edge_margin). See Details.
edge_margin	Numerical vector of the form c(bottom, left, top, right), which gives the amount of 'edge margin' to be added, expressed as a proportion of the plotted width. This is the space designed to be occupied by plot edges (e.g. a gradient fading out). Currently only implemented for left and right; top and bottom values are ignored. See Details.
edge_type	What style of edge to draw at the sides of the plot. Currently supported options are "gradient" (the default) and "zigzag".
interval_type	Set the way the interval is presented. Current options are c("norm", "unif") for a normal distribution-based curve and a box, respectively.
interval_value_labels	Logical value specifying whether interval value labels are to be added.
estimate_value_labels	Logical value specifying whether estimate value labels are to be added.
plot_estimate_marks	Whether to plot marks at the x location of the estimates.
estimate_mark_points	y positions of the ends of the estimate marks as a numeric vector of length 4. Values are, in order: start (relative to centre), end (relative to box top), start (relative to centre), end (relative to box bottom).
...	Further arguments passed to and from methods.

Details

Additional boundaries can be displayed using the `extra_boundaries` parameter. This can be helpful if you want to show a position that is of some practical relevance, but is not defined as a boundary for the purposes of the `interpretation_set`.

If you wish to plot multiple `interpretation_result` objects on one chart, see `plot_interpretation_result_list`.

Plots use the current R Graphics Palette, so you may wish to set that to something attractive before plotting. See ?palette.

Examples

```
# Set a nice colour scheme
grDevices::palette(c("#FF671F99", "#F2A90099", "#0085CA99"))
# Set up a confidence interval to interpret
ci_test <- matrix(c(-0.03, 0.05),
                  nrow = 1,
                  dimnames = list("estimate", c("2.5 %", "97.5 %")))
noninf <- interpret_noninferiority(ci_test, 0, 0.05, c("Treatment as usual",
                                                       "New treatment"))
plot(noninf)
```

`plot.interpretation_set` *Plot a diagram of the valid options for an interpretation_set object*

Description

Produces a diagram that illustrates the set of pairs of lower and upper confidence limits that are valid for a given `interpretation_set` object. The output is presented as a set of regions in different colours with boxes either within regions or spanning them to illustrate where the lower and upper confidence limits sit. The options are labelled alphabetically, and presented in the order in which their associated interpretations should be provided in the `interpretation_set`.

Usage

```
## S3 method for class 'interpretation_set'
plot(x, extra_boundaries = NULL, ...)
```

Arguments

`x`	An `interpretation_set` object.
`extra_boundaries`	A vector of numerical values specifying the position for displaying additional boundaries, not specified in the `interpretation_set`. May optionally be named values; if named, the names will be labelled on the plot axis. See Details for information on specifying locations.
`...`	Further arguments passed to and from methods.

Details

Additional boundaries can be displayed using the `extra_boundaries` parameter. This can be helpful if you want to show a position that is of some practical relevance, but is not defined as a boundary for the purposes of the `interpretation_set`. The boundaries specified by the `interpretation_set` are plotted with spacing 1 and are centred about 0: for an even number of boundaries the central pair of boundaries will be at -0.5 and +0.5; for an odd number of boundaries the central one will be at 0, and the next ones (if any) will be at -1 and +1, and so on.

Plots use the current R Graphics Palette, so you may wish to set that to something attractive before plotting. See `?palette`.

Examples

```
# Set a nice colour scheme
grDevices::palette(c("#FF671F99", "#F2A90099", "#0085CA99"))
# Plot the pre-defined interpretations_equivalence object with an additional
#   central boundary to illustrate where the actual null point is.
plot(interpretations_equivalence, extra_boundaries = c("Actual null" = 0))
```

---

plot_interpretation_result_list

Plotting function for collection of interpretation_result objects

---

Description

Produces a plot presenting a collection of `interpretation_result` objects on a single chart. If the `interpretation_result` objects are named then the names will be used for labelling the relevant intervals on the chart.

Usage

```
plot_interpretation_result_list(x, extra_boundaries = NULL,
  estimates = NULL, boundary_values = TRUE, boundary_label_pos = "below",
  interpretation_label_pos = "right", x_axis_pos = "below",
  y_axis_pos = "none", inner_margin = c(-0.1, 0.05, -0.1, 0.05),
  edge_margin = c(0, 0.02, 0, 0.02), edge_type = "gradient",
  interval_type = "norm", y_scale = 0.75, interval_value_labels = TRUE,
  estimate_value_labels = TRUE, plot_estimate_marks = TRUE, ...)
```

Arguments

`x`	A list of `interpretation_result` objects, length at least 2. The objects may optionally be named. See Details.
`extra_boundaries`	Names optional.

`estimates` Estimate values that the intervals assessed in each `interpretation_result` object relate to. If not specified, a default of the central point between the two ends of each interval will be assumed.

`boundary_values` A logical value indicating whether the values should be appended to the boundaries' names.

`boundary_label_pos` Where to put the boundary labels. Options are `c("below", "above", "on top", "none")`. If you are planning to plot values on the canvas and want the boundary labels on top then you may want to choose "none" and make a call to `label_ontop_boundaries()` after plotting values.

`interpretation_label_pos` Options are `c("right", "left", "none")`

`x_axis_pos` Location of a numerical x axis. Options are `c("none", "below", "above")`.

`y_axis_pos` Location of a numerical y axis. Default "none" will almost always be right. Options are `c("none", "left", "right")`.

`inner_margin` Numerical vector of the form `c(bottom, left, top, right)`, which gives the amount of inner margin to be added, expressed as a proportion of the plotted area. This is space designed to be past any plotted objects but before the edging (defined separately via `edge_margin`). See Details.

`edge_margin` Numerical vector of the form `c(bottom, left, top, right)`, which gives the amount of 'edge margin' to be added, expressed as a proportion of the plotted width. This is the space designed to be occupied by plot edges (e.g. a gradient fading out). Currently only implemented for left and right; top and bottom values are ignored. See Details.

`edge_type` What style of edge to draw at the sides of the plot. Currently supported options are `"gradient"` (the default) and `"zigzag"`.

`interval_type` Set the way the interval is presented. Current options are `c("norm", "unif")` for a normal distribution-based curve and a box, respectively.

`y_scale` How tall the interval plots are to be drawn

`interval_value_labels` Logical value specifying whether interval value labels are to be added.

`estimate_value_labels` Logical value specifying whether estimate value labels are to be added.

`plot_estimate_marks` Whether to plot marks at the x location of the estimates.

`...` Further arguments passed to and from methods.

Details

For a single `interpretation_result` object a `plot()` method is provided; see `plot.interpretation_result`.

To be a valid group of `interpretation_result` objects, each of the items in `x` must be a valid `interpretation_result`, and they must all share some characteristics. Each of the component objects must have been generated using the same `interpretation_set`, with the same boundaries, and the `low_to_high` parameter must be the same. This enables them to be meaningfully plotted on the same canvas.

Examples

```
# Set up some intervals to test:
ci_stage_1 <- matrix(c(0.023, 0.131), nrow = 1,
                     dimnames = list("estimate", c("2.5 %", "97.5 %")))
ci_stage_2 <- matrix(c(-0.016, 0.096), nrow = 1,
                     dimnames = list("estimate", c("2.5 %", "97.5 %")))
# Conduct the interpretations:
interp_stage_1 <-  interpret_noninferiority(ci_stage_1, actual_null = 0,
                                            ni_margin = 0.05,
                                            groups = c("Business as usual",
                                                       "New approach"))
interp_stage_2 <-  interpret_noninferiority(ci_stage_2, actual_null = 0,
                                            ni_margin = 0.05,
                                            groups = c("Business as usual",
                                                       "New approach"))

# Assemble the list object:
interp_1_and_2 <- list("Stage 1" = interp_stage_1,
                       "Stage 2" = interp_stage_2)
# Set a nice colour scheme
grDevices::palette(c("#FF671F99", "#F2A90099", "#0085CA99"))
plot_interpretation_result_list(interp_1_and_2,
                                boundary_label_pos = "on top")
```

`plot_intervals` *Plot intervals*

Description

Plot intervals on a canvas, typically prepared with `plot_region_canvas()`.

Usage

```
plot_intervals(intervals, estimates = NULL, interval_value_labels = FALSE,
  estimate_value_labels = FALSE, interval_labels_offset,
  estimate_labels_offset, interval_type = "norm",
  plot_estimate_marks = FALSE, estimate_mark_points = c(1.2 *
  graphics::strheight("M"), 0.05, -1.2 * graphics::strheight("M"), -0.05), ...)
```

Arguments

`intervals`	The interval(s) to be plotted. Two column matrix.
`estimates`	Estimates for each of the intervals (optional).
`interval_value_labels`	Logical value specifying whether interval value labels are to be added.

estimate_value_labels
: Logical value specifying whether estimate value labels are to be added.

interval_labels_offset
: Amount to offset interval labels by from the centre of the end of the interval's plot. `c(x1, x2, y1, y2)`.

estimate_labels_offset
: Amount to offset estimate labels by. `c(x, y)`. Normally want the estimate to be x-located at its value, but may want a y-offset to move it above or below the plot shape that represents the interval.

interval_type
: Set the way the interval is presented. Current options are `c("norm", "unif")` for a normal distribution-based curve and a box, respectively.

plot_estimate_marks
: Whether to plot marks at the x location of the estimates.

estimate_mark_points
: y positions of the ends of the estimate marks as a numeric vector of length 4. Values are, in order: start (relative to centre), end (relative to box top), start (relative to centre), end (relative to box bottom).

...
: Further parameters to be passed on.

Details

The `estimate_mark_points` parameter can be used to set the length of estimate marks, if they are requested using `plot_estimate_marks = TRUE`. The default is extending a little above and below the interval plot shape and with a gap in the middle big enough for a line of text (a bit bigger than the height of letter "M"). To leave no gap, set the first and third elements to zero, e.g. `estimate_mark_points = c(0, 0.05, 0, -0.05)`. To have the marks not extend outside of the interval shape, set the second and fourth elements to zero, e.g. `estimate_mark_points = c(0, 0, 0, 0)`.

---

plot_intervals_norm *Plot intervals as curved (normal distribution) areas*

Description

Plot intervals as curved (normal distribution) areas

Usage

```
plot_intervals_norm(intervals, estimates = NULL, y_scale = 1,
  interval_value_labels = FALSE, estimate_value_labels = FALSE,
  interval_labels_offset = c(0, 0, 0.15, 0.15),
  estimate_labels_offset = c(0, 0.5 * y_scale), plot_estimate_marks = FALSE,
  estimate_mark_points = c(1.2 * graphics::strheight("M"), 0.05, -1.2 *
  graphics::strheight("M"), -0.05), ...)
```

Arguments

intervals	The interval(s) to be plotted. Two column matrix.
estimates	Estimates for each of the intervals (optional).
y_scale	How tall the interval plots are to be drawn
interval_value_labels	Logical value specifying whether interval value labels are to be added.
estimate_value_labels	Logical value specifying whether estimate value labels are to be added.
interval_labels_offset	Amount to offset interval labels by from the centre of the end of the interval's plot. `c(x1, x2, y1, y2)`.
estimate_labels_offset	Amount to offset estimate labels by. `c(x, y)`. Normally want the estimate to be x-located at its value, but may want a y-offset to move it above or below the plot shape that represents the interval.
plot_estimate_marks	Whether to plot marks at the x location of the estimates.
estimate_mark_points	y positions of the ends of the estimate marks as a numeric vector of length 4. Values are, in order: start (relative to centre), end (relative to box top), start (relative to centre), end (relative to box bottom).
...	Further parameters to be passed on.

---

plot_intervals_unif *Plot intervals as uniform (box) areas*

Description

The current implementation of this function uses boxplot to draw its boxes.

Usage

```
plot_intervals_unif(intervals, estimates = NULL,
  interval_value_labels = FALSE, estimate_value_labels = FALSE,
  interval_labels_offset = c(0, 0, box_halfheight + 0.1, box_halfheight +
  0.1), estimate_labels_offset = c(0, box_halfheight + 0.1),
  plot_estimate_marks = FALSE, estimate_mark_points = c(1.2 *
  graphics::strheight("M"), 0.05, -1.2 * graphics::strheight("M"), -0.05), ...)
```

Arguments

intervals	The interval(s) to be plotted. Two column matrix.
estimates	Estimates for each of the intervals (optional).
interval_value_labels	Logical value specifying whether interval value labels are to be added.
estimate_value_labels	Logical value specifying whether estimate value labels are to be added.
interval_labels_offset	Amount to offset interval labels by from the centre of the end of the interval's plot. c(x1, x2, y1, y2).
estimate_labels_offset	Amount to offset estimate labels by. c(x, y). Normally want the estimate to be x-located at its value, but may want a y-offset to move it above or below the plot shape that represents the interval.
plot_estimate_marks	Whether to plot marks at the x location of the estimates.
estimate_mark_points	y positions of the ends of the estimate marks as a numeric vector of length 4. Values are, in order: start (relative to centre), end (relative to box top), start (relative to centre), end (relative to box bottom).
...	Further parameters to be passed on.

Details

The default value for the estimate_labels_offset parameter is defined in terms of a variable, box_halfheight. Because boxplot, the underlying plotting function, draws boxes different heights depending on the number of boxes drawn, this is set within the function. For one box the box_halfheight is 0.2; otherwise it is 0.4.

---

plot_region_canvas *Plot a canvas backed with regions defined by a set of boundaries*

Description

Produces a plot with all the background elements for plotting interpretation_set objects and similar outputs.

Usage

```
plot_region_canvas(boundaries, extra_boundaries = NULL, values,
  interpretations = NULL, boundary_values = FALSE,
  boundary_label_pos = "below", interpretation_label_pos = "right",
  x_axis_pos = "none", y_axis_pos = "none", inner_margin = c(0, 0.05, 0,
  0.05), edge_margin = c(0, 0.02, 0, 0.02), edge_type = "gradient", ...)
```

Arguments

boundaries	Named vector of numerical values of where boundaries should be drawn.
extra_boundaries	Names optional.
values	A matrix with either one or two columns containing the values of point estimates (one column) or ranges (two columns). Row names can specify labels.
interpretations	Character vector of interpretations to be used for labelling interpretations or NULL. If provided, should be the same length as nrow(values).
boundary_values	A logical value indicating whether the values should be appended to the boundaries' names.
boundary_label_pos	Where to put the boundary labels. Options are c("below", "above", "on top", "none"). If you are planning to plot values on the canvas and want the boundary labels on top then you may want to choose "none" and make a call to label_ontop_boundaries() after plotting values.
interpretation_label_pos	Options are c("right", "left", "none")
x_axis_pos	Location of a numerical x axis. Options are c("none", "below", "above").
y_axis_pos	Location of a numerical y axis. Default "none" will almost always be right. Options are c("none", "left", "right").
inner_margin	Numerical vector of the form c(bottom, left, top, right), which gives the amount of inner margin to be added, expressed as a proportion of the plotted area. This is space designed to be past any plotted objects but before the edging (defined separately via edge_margin). See Details.
edge_margin	Numerical vector of the form c(bottom, left, top, right), which gives the amount of 'edge margin' to be added, expressed as a proportion of the plotted width. This is the space designed to be occupied by plot edges (e.g. a gradient fading out). Currently only implemented for left and right; top and bottom values are ignored. See Details.
edge_type	What style of edge to draw at the sides of the plot. Currently supported options are "gradient" (the default) and "zigzag".
...	Further parameters to be passed on.

Details

If using to plot interpretation_set objects as generic items, the boundaries will typically be at arbitrary values selected for visual clarity. In this case it will typically not make sense to plot a numerical x axis. But boundaries can also be plotted as specific values related to the intended interpretation, and x axis plotting is normally appropriate in this case.

The colours of the background regions are determined by graphics::palette. Normally it will use the first n colours from the palette, where n is the number of regions (which is the number of boundaries + 1). If the left-most boundary is set to be at the edge of the plot (by having no values lower than it and setting inner_margin[2] and edge_margin[2] to 0), then the first colour in palette will be unused.

Similarly, if the right-most boundary is set to be the edge of the plot then there will only be as many regions as boundaries, and elements 1:n-1 of the palette will be used. (And similarly, one fewer regions than boundaries will be drawn if both the first and last boundaries are the edges of the plot.)

A pair of extra margins are defined for the purposes of this plot. Both are technically drawn as part of the plotting area (i.e., not in the area of the actual margin, which normally contains axes etc.). Note that the order of edges used in these margins is the same as the `graphics::par` parameters `mar` and `oma`, but the scaling / units are not. These parameters are specified proportional to the area of active plotting, rather than as lines.)

`strwidthl` *Obtain string widths in (approximate) multiple of lines.*

Description

Obtain string widths in (approximate) multiple of lines.

Usage

```
strwidthl(s)
```

Arguments

`s`	A character vector whose width is to be determined.

`validate_interpretation_result` *Validator for interpretation_result objects*

Description

Checks some features of the passed object to see whether they are as expected for the class. See `interpretation_result` documentation for definition of the class.

Usage

```
validate_interpretation_result(x)
```

Arguments

`x`	An object to be checked to see whether it is a valid interpretation_result.

Value

The interpretation_result object that was input, if no errors are found.

validate_interpretation_set *Validator for interpretation_set objects*

Description

Checks some features of the passed object to see whether they are as expected for the class. See `interpretation_set` documentation for definition of the class.

Usage

```
validate_interpretation_set(interpretation_set)
```

Arguments

`interpretation_set`	An object to be checked to see whether it is a valid interpretation_set.

Value

The interpretation_set object that was input, if no errors are found.

validate_result_list *Validates a collection of interpretation_result objects*

Description

Checks that a collection of interpretation_result objects has been correctly assembled for use in the plotting function.

Usage

```
validate_result_list(x)
```

Arguments

`x`	A list of `interpretation_result` objects, length at least 2. The objects may optionally be named. See Details.

Details

To be a valid group of `interpretation_result` objects, each of the items in `x` must be a valid `interpretation_result`, and they must all share some characteristics. Each of the component objects must have been generated using the same `interpretation_set`, with the same boundaries, and the `low_to_high` parameter must be the same.

Index