Package ‘forestplot’

December 12, 2020

Version 1.10.1
Title Advanced Forest Plot Using ‘grid’ Graphics
Description A forest plot that allows for
multiple confidence intervals per row,
custom fonts for each text element,
custom confidence intervals,
text mixed with expressions, and more.
The aim is to extend the use of forest plots beyond meta-analyses.
This is a more general version of the original ‘rmeta’ package’s forestplot()
function and relies heavily on the ‘grid’ package.

License GPL-2
URL https://gforge.se/packages/
BugReports https://github.com/gforge/forestplot/issues
Biarch yes
Depends grid, magrittr, checkmate
Suggests testthat, abind, knitr, rmarkdown, rmeta
Encoding UTF-8
NeedsCompilation no
VignetteBuilder knitr
RoxygenNote 7.1.1
Author Max Gordon [aut, cre],
Thomas Lumley [aut, ctb]
Maintainer Max Gordon <max@gforge.se>
Repository CRAN
Date/Publication 2020-12-12 15:00:10 UTC

R topics documented:

forestplot-package .......................................................... 2
forestplot ................................................................. 3
Description

The forest plot function, `forestplot`, is a more general version of the original `rmeta`-packages `forestplot` implementation. The aim is at using forest plots for more than just meta-analyses.

Details

The forestplot:

1. Allows for multiple confidence intervals per row
2. Custom fonts for each text element
3. Custom confidence intervals
4. Text mixed with expressions
5. Legends both on top/left of the plot and within the graph
6. Custom line height including auto-adapt height
7. Graph width that auto-adapts
8. Flexible arguments
9. and more

Additional functions

The `getTicks` tries to format ticks for plots in a nicer way. The major use is for exponentials where ticks are generated using the $2^n$ since a doubling is a concept easy to grasp for less mathematical-savvy readers.


**Description**

The `forestplot` is based on the `rmeta`-package’s `forestplot` function. This function resolves some limitations of the original functions such as:

- Adding expressions: Allows use of expressions, e.g. `expression(beta)`
- Multiple bands: Using multiple confidence bands for the same label
- Autosize: Adapts to viewport (graph) size

**Usage**

`forestplot(...)`

```r
## Default S3 method:
forestplot(
  labeltext,  # label text
  mean,      # mean
  lower,     # lower limit
  upper,     # upper limit
  align,     # alignment
  is.summary = FALSE,  # is summary
  graph.pos = "right",  # graph position
  hrl_lines,  # horizontal lines
  clip = c(-Inf, Inf),  # clip
  xlab = "",  # x-axis label
  zero = ifelse(xlog, 1, 0),  # zero
  graphwidth = "auto",  # graph width
  colgap,  # color gap
  lineheight = "auto",  # line height
  line.margin,  # line margin
  col = fpColors(),  # colors
  txt_gp = fpTxtGp(),  # text graphics parameters
  xlog = FALSE,  # x-log scale
  xticks,  # x-ticks
  xticks.digits = 2,  # x-ticks digits
  grid = FALSE,  # grid
  lwd.xaxis,  # line width x-axis
  lwd.zero,  # line width zero
  lwd.ci,  # line width confidence interval
  lty.ci = 1,  # line type confidence interval
  ci.vertices,  # confidence interval vertices
  ci.vertices.height = 0.1,  # confidence interval vertices height
  boxsize,  # box size
)```

mar = unit(rep(5, times = 4), "mm"),
title,
legend,
legend_args = fpLegend(),
new_page = getOption("forestplot_new_page", TRUE),
fn.ci_norm = fpDrawNormalCI,
fn.ci_sum = fpDrawSummaryCI,
fn.legend,
shapes_gp = fpShapesGp(),
...
)

Arguments

... Passed on to the fn.ci_norm and fn.ci_sum arguments

labeltext A list, matrix, vector or expression with the names of each row. The list should be wrapped in m x n number to resemble a matrix: list(list("rowname 1 col 1","rowname 2 col 1"),list("r1c2",expression(beta))). You can also provide a matrix although this cannot have expressions by design: matrix(c("rowname 1 col 1","rowname 2 col 1","r1c2","beta"),ncol=2). Use NA:s for blank spaces and if you provide a full column with NA then that column is a empty column that adds some space. Note: If you do not provide the mean/lower/upper arguments the function expects the label text to be a matrix containing the labeltext in the rownames and then columns for mean, lower, and upper.

mean A vector or a matrix with the averages. You can also provide a 2D/3D matrix that is automatically converted to the lower/upper parameters. The values should be in exponentiated form if they follow this interpretation, e.g. use \( \exp(\text{mean}) \) if you have the output from a logistic regression

lower The lower bound of the confidence interval for the forestplot, needs to be the same format as the mean, i.e. matrix/vector of equal columns & length

upper The upper bound of the confidence interval for the forestplot, needs to be the same format as the mean, i.e. matrix/vector of equal columns \& length

align Vector giving alignment (l,r,c) for the table columns

is.summary A vector indicating by TRUE/FALSE if the value is a summary value which means that it will have a different font-style

graph.pos The position of the graph element within the table of text. The position can be \( 1-(\text{ncol(labeltext)}+1) \). You can also choose set the positin to "left" or "right".

hrzl_lines Add horizontal lines to graph. Can either be TRUE or a list of gpar. See line section below for details.

clip Lower and upper limits for clipping confidence intervals to arrows

xlab x-axis label

zero x-axis coordinate for zero line. If you provide a vector of length 2 it will print a rectangle instead of just a line. If you provide NA the line is supressed.
graphwidth  Width of confidence interval graph, see `unit` for details on how to utilize mm etc. The default is `auto`, that is it uses up whatever space that is left after adjusting for text size and legend.

colgap  Sets the gap between columns, defaults to 6 mm but for relative widths. Note that the value should be in `unit`("npc").

lineheight  Height of the graph. By default this is `auto` and adjusts to the space that is left after adjusting for x-axis size and legend. Sometimes it might be desirable to set the line height to a certain height, for instance if you have several forest plots you may want to standardize their line height, then you set this variable to a certain height, note this should be provided as a `unit` object. A good option is to set the line height to `unit(2,"cm")`. A third option is to set line height to "lines" and then you get 50% more than what the text height is as your line height.

line.margin  Set the margin between rows, provided in numeric or `unit` format. When having multiple confidence lines per row setting the correct margin in order to visually separate rows.

col  Set the colors for all the elements. See `fpColors` for details.

txt_gp  Set the fonts etc for all text elements. See `fpTxtGp` for details.

xlog  If `TRUE`, x-axis tick marks are to follow a logarithmic scale, e.g. for logistic regression (OR), survival estimates (HR), Poisson regression etc. Note: This is an intentional break with the original `forestplot` function as I've found that exponentiated ticks/clips/zero effect are more difficult to for non-statisticians and there are sometimes issues with rounding the tick marks properly.

xticks  Optional user-specified x-axis tick marks. Specify `NULL` to use the defaults, numeric(0) to omit the x-axis. By adding a labels-attribute `attr(my_ticks,"labels") <-...` you can dictate the outputted text at each tick. If you specify a boolean vector then ticks indicated with FALSE won't be printed. Note that the labels have to be the same length as the main variable.

xticks.digits  The number of digits to allow in the x-axis if this is created by default.

grid  If you want a discrete gray dashed grid at the level of the ticks you can set this parameter to `TRUE`. If you set the parameter to a vector of values lines will be drawn at the corresponding positions. If you want to specify the `gpar` of the lines then either directly pass a `gpar` object or set the `gp` attribute e.g. `attr(line_vector,"gp") <-gpar(lty=2,col = "red")`

lwd.xaxis  lwd for the xaxis, see `gpar`

lwd.zero  lwd for the vertical line that gives the no-effect line, see `gpar`

lwd.ci  lwd for the confidence bands, see `gpar`

lty.ci  lty for the confidence bands, see `gpar`

ci.vertices  Set this to `TRUE` if you want the ends of the confidence intervals to be shaped as a T. This is set default to `TRUE` if you have any other line type than 1 since there is a risk of a dash occurring at the very end, i.e. showing incorrectly narrow confidence interval.

ci.vertices.height  The height of the vertices. Defaults to `npc` units corresponding to 10% of the row height. Note that the arrows correspond to the vertices heights.
boxsize  Override the default box size based on precision
mar       A numerical vector of the form c(bottom,left,top,right) of the type unit
title     The title of the plot if any
legend    Legend corresponding to the number of bars
legend_args The legend arguments as returned by the `fpLegend` function.
new_page  If you want the plot to appear on a new blank page then set this to TRUE, by
default it is TRUE. If you want to change this behavior for all plots then set the
options(forestplot_new_page = FALSE)
fn.ci_norm You can specify exactly how the line with the box is drawn for the normal (i.e.
non-summary) confidence interval by changing this parameter to your own func-
tion or some of the alternatives provided in the package. It defaults to the box
function `fpDrawNormalCI`
fn.ci_sum  Same as previous argument but for the summary outputs and it defaults to `fpDrawSummaryCI`.
fn.legend  What type of function should be used for drawing the legends, this can be a list
if you want different functions. It defaults to a box if you have anything else
than a single function or the number of columns in the `mean` argument
shapes_gp  Sets graphical parameters (squares and lines widths, styles, etc.) of all shapes
drawn (squares, lines, diamonds, etc.). This overrides `col, lwd.xaxis, lwd.zero, lwd.ci` and `lty.ci`.

Details
See vignette("forestplot") for details.

Value
NULL

Multiple bands
Using multiple bands, i.e. multiple lines, per variable can be interesting when you want to compare
different outcomes. E.g. if you want to compare survival specific to heart disease to overall survival
for smoking it may be useful to have two bands on top of eachother. Another useful implementation
is to show crude and adjusted estimates as separate bands.

Horizontal lines
The argument `hrzl_lines` can be either TRUE or a list with `gpar` elements:

- TRUEA line will be added based upon the is.summary rows. If the first line is a summary it
- `gpar`The same as above but the lines will be formatted according to the `gpar` element
- listThe list must either be numbered, i.e. list("2" = gpar(lty=1)), or have the same
  length as the `NROW(mean) + 1`. If the list is numbered the numbers should not exceed the
  `NROW(mean) + 1`. The no. 1 row designates the top, i.e. the line above the first row, all other
  correspond to the row below. Each element in the list needs to be TRUE, NULL, or `gpar` element.
  The TRUE defaults to a standard line, the NULL skips a line, while `gpar` corresponds to the fully
customized line. Apart from allowing standard `gpar` line descriptions, `lty`, `lwd`, `col`, and more you can also specify `gpar(columns = c(1:3,5))` if you for instance want the line to skip a column.

**Known issues**

The x-axis does not entirely respect the margin. Autosizing boxes is not always the best option, try to set these manually as much as possible.

**API-changes from rmeta-package’s forestplot**

- `xlog`: The xlog outputs the axis in log() format but the input data should be in antilog/exp format
- `col`: The corresponding function is `fpColors` for this package

**Author(s)**

Max Gordon, Thomas Lumley

**See Also**

Other forestplot functions: `fpColors()`, `fpDrawNormalCI()`, `fpLegend()`, `fpShapesGp()`

**Examples**

```
# Simple examples of how to do a forestplot #

ask <- par(ask = TRUE)

# A basic example, create some fake data
row_names <- list(list("test = 1", expression(test >= 2)))
test_data <- data.frame(
  coef = c(1.59, 1.24),
  low = c(1.4, 0.78),
  high = c(1.8, 1.55)
)
forestplot(row_names,
  test_data$coef,
  test_data$low,
  test_data$high,
  zero = 1,
  cex = 2,
  lineheight = "auto",
  xlab = "Lab axis txt"
)

# Print two plots side by side using the grid
# package’s layout option for viewports
grid.newpage()
```
pushViewport(viewport(layout = grid.layout(1, 2)))
pushViewport(viewport(layout.pos.col = 1))
forestplot(row_names,
   test_data$coef,
   test_data$low,
   test_data$high,
   zero = 1,
   cex = 2,
   lineheight = "auto",
   xlab = "Lab axis txt",
   new_page = FALSE)
popViewport()
pushViewport(viewport(layout.pos.col = 2))
forestplot(row_names,
   test_data$coef,
   test_data$low,
   test_data$high,
   zero = 1,
   cex = 2,
   lineheight = "auto",
   xlab = "Lab axis txt",
   new_page = FALSE)
popViewport(2)

# An advanced test
test_data <- data.frame(
   coef1 = c(1, 1.59, 1.3, 1.24),
   coef2 = c(1, 1.7, 1.4, 1.04),
   low1 = c(1, 1.3, 1.1, 0.99),
   low2 = c(1, 1.6, 1.2, 0.7),
   high1 = c(1, 1.94, 1.6, 1.55),
   high2 = c(1, 1.8, 1.55, 1.33)
)

col_no <- grep("coef", colnames(test_data))
row_names <- list(
   list("Category 1", "Category 2", "Category 3", expression(Category >= 4)),
   list("ref",
      substitute(expression(bar(x) == val),
      list(val = round(rowMeans(test_data[2, col_no]), 2)))
   ),
   substitute(
      expression(bar(x) == val),
      list(val = round(rowMeans(test_data[3, col_no]), 2))
   ),
   substitute(
      expression(bar(x) == val),
      list(val = round(rowMeans(test_data[4, col_no]), 2))
   )
)
```r
coef <- with(test_data, cbind(coef1, coef2))
low <- with(test_data, cbind(low1, low2))
high <- with(test_data, cbind(high1, high2))
forestplot(row_names, coef, low, high,
          title = "Cool study",
          zero = c(0.98, 1.02),
          grid = structure(c(2^-.5, 2^.5),
                           gp = gpar(col = "steelblue", lty = 2)
          ),
          boxsize = 0.25,
          col = fpColors(
                        box = c("royalblue", "gold"),
                        line = c("darkblue", "orange"),
                        summary = c("darkblue", "red")
          ),
          xlab = "The estimates",
          new_page = TRUE,
          legend = c("Treatment", "Placebo"),
          legend_args = fpLegend(
                        pos = list("topright"),
                        title = "Group",
                        r = unit(.1, "snpc"),
                        gp = gpar(col = "#CCCCCC", lwd = 1.5)
          )
)

# An example of how the exponential works

test_data <- data.frame(
        coef = c(2.45, 0.43),
        low = c(1.5, 0.25),
        high = c(4, 0.75),
        boxsize = c(0.5, 0.5)
)
row_names <- cbind(
        c("Name", "Variable A", "Variable B"),
        c("HR", test_data$coef)
)
test_data <- rbind(rep(NA, 3), test_data)

forestplot(
        labeltext = row_names,
        test_data[, c("coef", "low", "high")],
        is.summary = c(TRUE, FALSE, FALSE),
        boxsize = test_data$boxsize,
        zero = 1,
        xlog = TRUE,
        col = fpColors(lines = "red", box = "darkred")
)
```
# An example using shapes_gp

```r
forestplot(
  labeltext = cbind(Author = c("Smith et al", "Smooth et al", "Al et al")),
  mean = cbind(1:3, 1.5:3.5),
  lower = cbind(0:2, 0.5:2.5),
  upper = cbind(4:6, 5.5:7.5),
  is.summary = c(FALSE, FALSE, TRUE),
  shapes_gp = fpShapesGp(
    default = gpar(lineend = "square", linejoin = "mitre", lwd = 3, col = "pink"),
    box = gpar(fill = "black", col = "red"), # only one parameter
    lines = list( # as many parameters as CI
      gpar(lwd = 10), gpar(lwd = 5),
      gpar(), gpar(),
      gpar(lwd = 2), gpar(lwd = 1)
    ),
    summary = list( # as many parameters as band per label
      gpar(fill = "violet", col = "gray", lwd = 10),
      gpar(fill = "orange", col = "gray", lwd = 10)
    ),
  ),
  vertices = TRUE
)
```

```
par(ask = ask)
# See vignette for a more detailed description
# vignette("forestplot", package="forestplot")
```

---

**fpColors**  

*A function for the color elements used in forestplot()*

## Description

This function encapsulates all the colors that are used in the `forestplot` function. As there are plenty of color options this function gathers them all in one place.

## Usage

```r
fpColors(
  all.elements,
  box = "black",
  lines = "gray",
  summary = "black",
  zero = "lightgray",
  text = "black",
  axes = "black",
  hrz_lines = "black"
)
```
Arguments

- **all.elements**: A color for all the elements. If set to NULL then it's set to the par("fg") color
- **box**: The color of the box indicating the estimate
- **lines**: The color of the confidence lines
- **summary**: The color of the summary
- **zero**: The color of the zero line
- **text**: The color of the text
- **axes**: The color of the x-axis at the bottom
- **hrz_lines**: The color of the horizontal lines

Details

Further customization of non-text elements can be performed with fpShapesGp passed as shapes_gp parameter to forestplot. The fpColors function is kept for backwards compatibility.

If you have several values per row in a forestplot you can set a color to a vector where the first value represents the first line/box, second the second line/box etc. The vectors are only valid for the box & lines options.

This function is a copy of the meta.colors function in the rmeta package.

Value

- **list**: A list with the elements:
  - **box**: the color of the box(marker
  - **lines**: the color of the lines
  - **summary**: the color of the summary
  - **zero**: the color of the zero vertical line
  - **text**: the color of the text
  - **axes**: the color of the axes

Author(s)

Max Gordon, Thomas Lumley

See Also

Other forestplot functions: forestplot(), fpDrawNormalCI(), fpLegend(), fpShapesGp()

Examples

```r
ask <- par(ask = TRUE)

# An example of how the exponential works
test_data <- data.frame(
  coef = c(2.45, 0.43),
  low = c(1.5, 0.25),
)```
fpDrawNormalCI

Draw standard confidence intervals

Description

A function that is used to draw the different confidence intervals for the non-summary lines. Use the fpDrawNormalCI function as a template if you want to make your own funky line + marker.

Usage

fpDrawNormalCI(
  lower_limit,
  estimate,
  upper_limit,
  size,
  y.offset = 0.5,
  clr.line,
  clr.marker,
  lwd,
  lty = 1,
  vertices,
  vertices.height = 0.1,
  shapes_gp = fpShapesGp(),
  shape_coordinates = structure(c(1, 1), max.coords = c(1, 1)),
  ...
)

fpDrawNormalCI

    lower_limit,
    estimate,
    upper_limit,
    size,
    y.offset = 0.5,
    clr.line,
    clr.marker,
    lwd,
    lty = 1,
    vertices,
    vertices.height = 0.1,
    shapes_gp = fpShapesGp(),
    shape_coordinates = structure(c(1, 1), max.coords = c(1, 1)),
    ...
)

fpDrawCircleCI

    lower_limit,
    estimate,
    upper_limit,
    size,
    y.offset = 0.5,
    clr.line,
    clr.marker,
    lwd,
    lty = 1,
    vertices,
    vertices.height = 0.1,
    shapes_gp = fpShapesGp(),
    shape_coordinates = structure(c(1, 1), max.coords = c(1, 1)),
    ...
)

fpDrawPointCI

    lower_limit,
    estimate,
    upper_limit,
    size,
    y.offset = 0.5,
    clr.line,
    clr.marker,
    lwd,
    lty = 1,
    vertices,
    vertices.height = 0.1,
    pch = 1,
    shapes_gp = fpShapesGp(),
shape_coordinates = structure(c(1, 1), max.coords = c(1, 1)),
...
)

fpDrawSummaryCI(
  lower_limit,
estimate,
upper_limit,
size,
col,
y.offset = 0.5,
shapes_gp = fpShapesGp(),
shape_coordinates = structure(c(1, 1), max.coords = c(1, 1)),
...
)

fpDrawBarCI(
  lower_limit,
estimate,
upper_limit,
size,
col,
y.offset = 0.5,
shapes_gp = fpShapesGp(),
shape_coordinates = structure(c(1, 1), max.coords = c(1, 1)),
...
)

Arguments

lower_limit  The lower limit of the confidence line. A native numeric variable that can actually be outside the boundaries. If you want to see if it is outside then convert it to 'npc' and see if the value ends up more than 1 or less than 0. Here’s how you do the conversion: convertX(unit(upper_limit,"native"),"npc",valueOnly = TRUE) and the convertX together with unit is needed to get the right values while you need to provide the valueOnly as you cannot compare a unit object.
estimate      The estimate indicating the placement of the actual box. Note, this can also be outside bounds and is provided in a numeric format the same way as the lower_limit.
upper_limit   The upper limit of the confidence line. See lower_limit for details.
size          The actual size of the box/diamond/marker. This provided in the 'snpc' format to generate a perfect marker. Although you can provide it alternative units as well, this is useful for the legends to work nicely.
y.offset      If you have multiple lines they need an offset in the y-direction.
clr.line      The color of the line.
clr.marker    The color of the estimate marker
lwd           Line width, see gpar
**lty**
Line type, see `gpar`

**vertices**
Set this to TRUE if you want the ends of the confidence intervals to be shaped as a T. This is set default to TRUE if you have any other line type than 1 since there is a risk of a dash occurring at the very end, i.e. showing incorrectly narrow confidence interval.

**vertices.height**
The height hoft the vertices. Defaults to npc units corresponding to 10% of the row height.

**shapes_gp**
A set of graphical parameters of class `fpShapesGp`

**shape_coordinates**
A vector of length 2 the label (first item of the vector) and the band (second item of the vector) of the confidence interval. This is used together with shapes_gp to retrieve graphical parameters for that item.

... Allows additional parameters for sibling functions

**pch**
Type of point see `grid.points` for details

**col**
The color of the summary object.

**Value**
void The function outputs the line using grid compatible functions and does not return anything.

**Author(s)**
Max Gordon, Thomas Lumley

**See Also**
Other forestplot functions: `forestplot()`, `fpColors()`, `fpLegend()`, `fpShapesGp()`

**Examples**
```r
ask <- par(ask = TRUE)

test_data <- data.frame(
  coef1 = c(1, 1.59, 1.3, 1.24),
  coef2 = c(1, 1.7, 1.4, 1.04)
)

test_data$low1 <- test_data$coef1 - 1.96 * c(0, .2, .1, .15)
test_data$high1 <- test_data$coef1 + 1.96 * c(0, .2, .1, .15)

test_data$low2 <- test_data$coef2 - 1.96 * c(0, .1, .15, .2)
test_data$high2 <- test_data$coef2 + 1.96 * c(0, .1, .15, .2)

col_no <- grep("coef", colnames(test_data))
row_names <- list(
  list("Category 1", "Category 2", "Category 3", expression(Category >= 4)),
  "ref",
```
substitute(
  expression(bar(x) == val),
  list(val = round(rowMeans(test_data[2, col_no]), 2))
),
substitute(
  expression(bar(x) == val),
  list(val = round(rowMeans(test_data[3, col_no]), 2))
),
substitute(
  expression(bar(x) == val),
  list(val = round(rowMeans(test_data[4, col_no]), 2))
)
)

coef <- with(test_data, cbind(coef1, coef2))
low <- with(test_data, cbind(low1, low2))
high <- with(test_data, cbind(high1, high2))

# Change all to diamonds
forestplot(row_names, coef, low, high,
  fn.ci_norm = fpDrawDiamondCI,
  title = "Cool study",
  zero = 1, boxsize = 0.25,
  col = fpColors(
    box = c("royalblue", "gold"),
    line = c("darkblue", "orange"),
    summary = c("darkblue", "red")
  ),
  xlab = "The estimates",
  new_page = TRUE,
  legend = c("Treatment", "Placebo"),
  legend_args = fpLegend(
    title = "Group",
    pos = list("topright", inset = .1),
    r = unit(.1, "npc"),
    gp = gpar(col = "CCCCCC", lwd = 1.5)
  )
)

# Change first to diamonds
forestplot(row_names, coef, low, high,
  fn.ci_norm = c(
    "fpDrawDiamondCI",
    rep("fpDrawNormalCI",
        times = nrow(coef) - 1
    )
  ),
  title = "Cool study",
  zero = 1, boxsize = 0.25,
  col = fpColors(
    box = c("royalblue", "gold"),
    line = c("darkblue", "orange"),
    summary = c("darkblue", "red"))
)
fpLegend

A function for the legend used in forestplot()
Description

This function encapsulates all the legend options that are used in the `forestplot` function. This is in order to limit the crowding among the arguments for the `forestplot` call.

Usage

```r
fpLegend(
  pos = "top",
  gp = NULL,
  r = unit(0, "snpc"),
  padding = unit(ifelse(!is.null(gp), 3, 0), "mm"),
  title = NULL
)
```

Arguments

- **pos**
  - The position of the legend, either at the "top" or the "right" unless positioned inside the plot. If you want the legend to be positioned inside the plot then you have to provide a list with the same x & y qualities as `legend`. For instance if you want the legend to be positioned at the top right corner then use `pos = list("topright")` - this is equivalent to `pos = list(x=1,y=1)`. If you want to have a distance from the edge of the graph then add an inset to the list, e.g. `pos = list("topright","inset"=.1)` - the inset should be either a `unit` element or a value between 0 and 1. The default is to have the boxes aligned vertical, if you want them to be in a line then you can specify the "align" option, e.g. `pos = list("topright","inset"=.1,"align"="horizontal")`.

- **gp**
  - The `gpar` options for the legend. If you want the background color to be light grey then use `gp = gpar(fill = "lightgrey")`. If you want a border then set the `col` argument: `gp = gpar(fill = "lightgrey",col="black")`. You can also use the `lwd` and `lty` argument as usual, `gp = gpar(lwd=2,lty=1)`, will result in a black border box of line type 1 and line width 2.

- **r**
  - The box can have rounded edges, check out `grid.roundrect`. The `r` option should be a `unit` object. This is by default `unit(0, "snpc")` but you can choose any value that you want. The "snpc" unit is the preferred option.

- **padding**
  - The padding for the legend box, only used if box is drawn. This is the distance from the border to the text/boxes of the legend.

- **title**
  - The title of the legend if any

Value

- list
  - Returns a list with all the elements

See Also

- Other forestplot functions: `forestplot()`, `fpColors()`, `fpDrawNormalCI()`, `fpShapesGp()`
fpShapesGp

A function for graphical parameters of the shapes used in forestplot()

Description

This function encapsulates all the non-text elements that are used in the forestplot function. As there are plenty of shapes options this function gathers them all in one place.

Usage

fpShapesGp(
  default = NULL,
  box = NULL,
  lines = NULL,
  vertices = NULL,
  summary = NULL,
  zero = NULL,
  axes = NULL,
  hrz_lines = NULL,
  grid = NULL
)

Arguments

default A fallback gpar for all unspecified attributes. If set to NULL then it defaults to legacy parameters, including the col, lwd.xaxis, lwd.ci and lty.ci parameter of fpColors.

box The graphical parameters (gpar) of the box, circle or point indicating the point estimate, i.e. the middle of the confidence interval (may be a list of gpars)

lines The graphical parameters (gpar) of the confidence lines (may be a list of gpars)

vertices The graphical parameters (gpar) of the vertices (may be a list of gpars). If ci.vertices is set to TRUE in forestplot vertices inherits from lines all its parameters but lty that is set to "solid" by default.

summary The graphical parameters (gpar) of the summary (may be a list of gpars)

zero The graphical parameters (gpar) of the zero line (may not be a list of gpars)

axes The graphical parameters (gpar) of the x-axis at the bottom (may not be a list of gpars)

hrz_lines The graphical parameters (gpar) of the horizontal lines (may not be a list of gpars)

gird The graphical parameters (gpar) of the grid (vertical lines) (may be a list of gpars)
Details

This function obsoletes `fpColors`.

If some, but not all parameters of a shape (e.g. box) are specified in `gpar()` such as setting `lwd` but not line color, the unspecified parameters default to the ones specified in `default`, then, default to legacy parameters of `forestplot` such as `col`.

Parameters `box`, `lines`, `vertices`, `summary` may be set as list containing several gpars. The length of the list must either be equal to the number of bands per label or to the number of bands multiplied by the number of labels, allowing specification of different styles for different parts of the forest plot.

The parameter `grid` can either be a single gpar or a list of gpars with as many elements as there are lines in the grid (as set by the `xticks` or `grid` arguments of `forestplot`)

Parameters `zero`, `axes`, `hrz_lines` must either be `NULL` or gpar but cannot be lists of gpars.

Value

list A list with the elements:

- `default` the gpar for default attributes
- `box` the gpar or list of gpars of the box/marker
- `lines` the gpar or list of gpars of the lines
- `vertices` the gpar or list of gpars of the vertices
- `summary` the gpar or list of gpars of the summary
- `zero` the gpar of the zero vertical line
- `axes` the gpar of the x-axis
- `hrz_lines` the gpar of the horizontal lines
- `grid` the gpar or list of gpars of the grid lines

Author(s)

Andre GILLIBERT

See Also

Other forestplot functions: `forestplot()`, `fpColors()`, `fpDrawNormalCI()`, `fpLegend()`

Examples

```r
ask <- par(ask = TRUE)

# An example of how fpShapesGp works

styles <- fpShapesGp(
  default = gpar(col = "pink", lwd = 2, lineend = "square", linejoin = "mitre"),
  grid = list(
    gpar(col = "blue"),
    gpar(col = "black"),
    gpar(col = "blue"))
```

### Description

This function generates all the `gpar()` elements for the different text elements within the graph. Elements not specified inherit their default settings from the `label` argument.

### Usage

```r
fpTxtGp(label, summary, xlab, title, ticks, legend, legend.title, cex = 1)
```

### Arguments

- **label**  
The text labels (see details below)
- **summary**  
The summary labels (see details below)
- **xlab**  
The `xlab` text
- **title**  
The plot title
- **ticks**  
The ticks associated with the `xlab`
- **legend**  
The legend text
- **legend.title**  
The legend title
- **cex**  
The font size

### Value

A list of the `fpTxtGp` class
**List arguments for** label/summary

You can provide a list of elements for the label and summary in order to specify separate elements. If you provide a list in one dimension the gpar elements are assumed to follow the columns. If you provide a list of 2 dimensions the structure assumes is list[[row]][[column]] and the number of elements should correspond to the number of labels for the label argument, i.e. without the rows marked as summary elements. The same goes for summary arguments.

**Examples**

```r
fpTxtGp(label=gpar(fontfamily="HersheySerif"))
```

---

**getTicks**

**Ticks for plot axis**

**Description**

Gets the ticks in a formatted version. This is since I’m not always that fond of just pretty(1:10/5). In exponential form the ticks are determined from the 2-base, meaning that you get an intuitive feeling for when the value is doubled.

**Usage**

```r
getTicks(low, high = low, clip = c(-Inf, Inf), exp = FALSE, digits = 0)
```

**Arguments**

- `low` lower bound, can be a single number or a vector
- `high` upper bound - optional, you can just have all data in the `low` variable
- `clip` if the ci are clipped
- `exp` If the value should be in exponential form (default)
- `digits` Number of digits - used in exp mode

**Details**

This function is far from perfect and I recommend specifying yourself the ticks that you want.

**Value**

`vector` Returns a vector with the ticks
Examples

test_data <- data.frame(
  coef = c(2, 0.5),
  low = c(1.5, 0.05),
  high = c(3, 0.75),
  boxsize = c(0.5, 0.5)
)

# Exponential form where the exponent base i 2 for easier understanding
getTicks(
  low = test_data$low,
  high = test_data$high,
  clip = c(-Inf, Inf),
  exp = TRUE
)

# Non exponential form with using pretty
getTicks(
  low = test_data$low,
  high = test_data$high,
  clip = c(-Inf, Inf),
  exp = FALSE
)

# A very simple example
getTicks(1:5 * 2.33,
  exp = FALSE
)

# A slightly more advanced exponential version
getTicks(1:10 * .33,
  digits = 2,
  exp = TRUE
)

Description

The data is a list containing the Swedish and the Danish coefficients for health related quality of life (HRQoL) 1 year after total hip arthroplasty surgery. The age is modelled as a spline and is therefore presented as a contrast.

Author(s)

Max Gordon <max@gforge.se>
prDefaultGp

**Construct default parameters from arguments that may include missing arguments**

**Description**

Construct default parameters from arguments that may include missing arguments

**Usage**

```r
prDefaultGp(col, lwd, lty)
```

**Arguments**

- `col` Line color (or missing)
- `lwd` Line width (or missing)
- `lty` Line type (or missing)

**Value**

a `gpar` object containing these three attributes

---

prGetShapeGp

**A function to extract graphical parameters from a fpShapesGp object**

**Description**

A function to extract graphical parameters from a fpShapesGp object

**Usage**

```r
prGetShapeGp(
  shapes_gp,
  coords,
  object,
  default = grid::gpar(),
  nodefault = FALSE
)
```
Arguments

- **shapes_gp**: An object of class `fpShapesGp` specifying all graphical parameters.
- **coords**: A numeric vector of length 2, specifying the label number (first item of the vector) and the confidence band number within this label; that can be >= 2 if there are multiple confidence bands per label. Can be NULL for objects that are used only once (e.g., axes). Vector coords must have an R attribute `max.coords` as numeric vector of length 2 specifying the total number of labels and number of confidence bands by label for the forest plot. The first coordinate specify the label number and the second coordinate (for multi-band forest plots) specifies the band number within the label.
- **object**: One of "box", "lines", "vertices", "summary", "zero", "axes", "hrz_lines" or "grid", referring to the object for which the graphical parameters are requested.
- **default**: Default attributes to rely on when neither found in `shapes_gp$object` nor in `shapes_gp$default`.
- **nodefault**: Logical. If TRUE, do not search attribute in `shapes_gp$default`.

Value

An object of class `gpar`.

Author(s)

Andre GILLIBERT

Description

A function to merge two sets of graphical parameters

Usage

```r
prMergeGp(weak = gpar(), strong = gpar())
```

Arguments

- **weak**: A `gpar`.
- **strong**: Another `gpar`, with parameters taking precedence over weak.

Value

A `gpar` merging attributes of both weak and strong.
Index

* data
  HRQoL, 23
* forestplot functions
  forestplot, 3
  fpColors, 10
  fpDrawNormalCI, 12
  fpLegend, 17
  fpShapesGp, 19
convertX, 14
forestplot, 2, 3, 10, 11, 15, 18–20
forestplot-package, 2
fpColors, 5, 7, 10, 15, 18, 20
fpDrawBarCI (fpDrawNormalCI), 12
fpDrawCircleCI (fpDrawNormalCI), 12
fpDrawDiamondCI (fpDrawNormalCI), 12
fpDrawNormalCI, 6, 7, 11, 12, 18, 20
fpDrawPointCI (fpDrawNormalCI), 12
fpDrawSummaryCI, 6
fpDrawSummaryCI (fpDrawNormalCI), 12
fpLegend, 6, 7, 11, 15, 17, 20
fpShapesGp, 7, 11, 15, 18, 19, 25
fpTxtGp, 5, 21
getTicks, 2, 22
gpar, 4–7, 14, 15, 18, 19, 21, 24, 25
grid.points, 15
grid.roundrect, 18

HRQoL, 23

legend, 18

meta.colors, 11

prDefaultGp, 24
prGetShapeGp, 24
prMergeGp, 25

unit, 5, 6, 14, 18