Package 'consort'

October 5, 2026

Type Package

Title Create Consort Diagram

Version 1.2.4

Description To make it easy to create CONSORT diagrams for the transparent reporting of participant allocation in randomized, controlled clinical trials. This is done by creating a standardized disposition data, and using this data as the source for the creation a standard CONSORT diagram. Human effort by supplying text labels on the node can also be achieved.

License MIT + file LICENSE

URL https://github.com/adayim/consort/,
https://adayim.github.io/consort/

BugReports https://github.com/adayim/consort/issues

Encoding UTF-8

Imports grDevices, grid

Suggests testthat (>= 3.0.0), knitr, DiagrammeR, rmarkdown, covr, stringi

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5.0)

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Alim Dayim [aut, cre] (ORCID: <https://orcid.org/0000-0001-9998-7463>)

Maintainer Alim Dayim <ad938@cam.ac.uk>

Repository CRAN

Date/Publication 2026-10-05 08:00:02 UTC

Contents

add_box . . . 2
add_label_box . . . 3
add_side_box . . . 5
add_split . . . 6
build_grid . . . 8
build_grviz . . . 9
connect_box . . . 10
consort_plot . . . 11
dispos.data . . . 13
gen_text . . . 14
get_coords . . . 15
move_box . . . 16
plot.consort . . . 17
set_consort_defaults . . . 17
textbox . . . 19

Index **21**

`add_box` *Add nodes*

Description

Create/add vertically aligned labeled nodes or side nodes.

Usage

```
add_box(
  prev_box = NULL,
  txt,
  just = c("center", "left", "right"),
  text_width = NULL,
  ...
)
```

Arguments

`prev_box`	Previous node object, the created new node will be vertically aligned with this node. Left this as 'NULL' if this is the first node. The first node will be aligned in the top center.
`txt`	Text in the node. If the 'prev_box' is a horizontally aligned multiple nodes, a vector of with the same length must be provided.
`just`	The justification for the text: left, center or right.

text_width	a positive integer giving the target column for wrapping lines in the output. String will not be wrapped if not defined (default). The stri_wrap function will be used if stringi package installed as it is a better options for non-Latin language, otherwise strwrap will be used.
...	Other parameters pass to textbox,

Value

A consort object.

See Also

add_side_box add_split textbox add_label_box

Examples

```
txt1 <- "Population (n=300)"
txt1_side <- "Excluded (n=15): \n
              \u2022 MRI not collected (n=3)\n
              \u2022 Tissues not collected (n=4)\n
              \u2022 Other (n=8)"

g <- add_box(txt = txt1)

g <- add_side_box(g, txt = txt1_side)

g <- add_box(g, txt = "Randomized (n=200)")

g <- add_split(g, txt = c("Arm A (n=100)", "Arm B (n=100"))
g <- add_side_box(g,
  txt = c(
    "Excluded (n=15):\n
                \u2022 MRI not collected (n=3)\n
                \u2022 Tissues not collected (n=4)\n
                 \u2022 Other (n=8)",
    "Excluded (n=15):\n
                 \u2022 MRI not collected (n=3)\n
                 \u2022 Tissues not collected (n=4)"
  )
)

g <- add_box(g, txt = c("Final analysis (n=100)", "Final analysis (n=100"))
g <- add_label_box(g, txt = c("1" = "Screening", "3" = "Randomized", "4" = "Final analysis"))
```

---

add_label_box *Add a vertically aligned label nodes on the left side.*

Description

In a consort diagram, this can be used to indicate different stage.

Usage

```
add_label_box(
  prev_box,
  txt,
  only_terminal = TRUE,
  just = c("center", "left", "right"),
  box_fn = roundrectGrob,
  ...
)
```

Arguments

prev_box	A completed diagram created with add_box, add_side_box etc.
txt	Text in the node. If a character string is provided, the label will be aligned to the last box if a character is provided. If a named vector, the labels will align to corresponding row of the node. And the names is the number indicating row number of box to horizontally align with and value is the text in the box.
only_terminal	If the txt is only for the terminal box, default. Otherwise, the side box will also be accounted for.
just	The justification for the text: center (default), left or right.
box_fn	The function to create the box grob. Default is roundrectGrob.
...	Other parameters pass to textbox.

Value

A consort object.

See Also

add_side_box add_split textbox add_box

Examples

```
txt1 <- "Population (n=300)"
txt1_side <- "Excluded (n=15): \n
              \u2022 MRI not collected (n=3)\n
              \u2022 Tissues not collected (n=4)\n
              \u2022 Other (n=8)"

g <- add_box(txt = txt1)

g <- add_side_box(g, txt = txt1_side)

g <- add_box(g, txt = "Randomized (n=200)")

g <- add_split(g, txt = c("Arm A (n=100)", "Arm B (n=100")")
g <- add_side_box(g,
  txt = c(
    "Excluded (n=15):\n
```

```
                  \u2022 MRI not collected (n=3)\n
                  \u2022 Tissues not collected (n=4)\n
                   \u2022 Other (n=8)",
    "Excluded (n=15):\n
                   \u2022 MRI not collected (n=3)\n
                   \u2022 Tissues not collected (n=4)"
  )
)

g <- add_box(g, txt = c("Final analysis (n=100)", "Final analysis (n=100"))
g <- add_label_box(g, txt = c("1" = "Screening", "3" = "Randomized", "4" = "Final analysis"))
```

`add_side_box` *Add a side node*

Description

Add an exclusion node on the right side. If the length of text label is two, then the first one will be aligned on the left and the second on the right. Otherwise, all the side nodes will be aligned on the right.

Usage

```
add_side_box(prev_box, txt, side = NULL, text_width = NULL, ...)
```

Arguments

`prev_box`	Previous node object, the created new node will be aligned at the right bottom of the 'prev_box'.
`txt`	Text in the node. If the 'prev_box' is a horizontally aligned multiple nodes, a vector of with the same length must be provided.
`side`	Position of the side box, 'left' or 'right' side of the terminal box. Will be aligned on the left and right side if only two groups, right otherwise. This will be ignored for for 'grViz' plot, see grViz.
`text_width`	a positive integer giving the target column for wrapping lines in the output. String will not be wrapped if not defined (default). The `stri_wrap` function will be used if `stringi` package installed as it is a better options for non-Latin language, otherwise `strwrap` will be used.
`...`	Other parameters pass to textbox,

Value

A `consort` object.

See Also

`add_box` `add_split` `textbox` `add_label_box`

Examples

```
txt1 <- "Population (n=300)"
txt1_side <- "Excluded (n=15): \n
              \u2022 MRI not collected (n=3)\n
              \u2022 Tissues not collected (n=4)\n
              \u2022 Other (n=8)"

g <- add_box(txt = txt1)

g <- add_side_box(g, txt = txt1_side)

g <- add_box(g, txt = "Randomized (n=200)")

g <- add_split(g, txt = c("Arm A (n=100)", "Arm B (n=100"))
g <- add_side_box(g,
  txt = c(
    "Excluded (n=15):\n
                  \u2022 MRI not collected (n=3)\n
                  \u2022 Tissues not collected (n=4)\n
                   \u2022 Other (n=8)",
    "Excluded (n=15):\n
                   \u2022 MRI not collected (n=3)\n
                   \u2022 Tissues not collected (n=4)"
  )
)

g <- add_box(g, txt = c("Final analysis (n=100)", "Final analysis (n=100"))
g <- add_label_box(g, txt = c("1" = "Screening", "3" = "Randomized", "4" = "Final analysis"))
```

`add_split` *Add a splitting box*

Description

This function will create a horizontally aligned nodes. The horizontal coordinate will be automatically calculated if the coordinates not provided.

Usage

```
add_split(
  prev_box,
  txt,
  just = c("center", "left", "right"),
  text_width = NULL,
  ...
)
```

Arguments

prev_box	Previous node that the newly created split box will be aligned.
txt	Text in the node. If the 'prev_box' is a horizontally aligned multiple nodes, a vector of with the same length must be provided.
just	The justification for the text: left, center or right.
text_width	a positive integer giving the target column for wrapping lines in the output. String will not be wrapped if not defined (default). The stri_wrap function will be used if stringi package installed as it is a better options for non-Latin language, otherwise strwrap will be used.
...	Other parameters pass to textbox,

Value

A consort.list object.

See Also

add_box, add_side_box textbox

Examples

```
txt1 <- "Population (n=300)"
txt1_side <- "Excluded (n=15): \n
              \u2022 MRI not collected (n=3)\n
              \u2022 Tissues not collected (n=4)\n
              \u2022 Other (n=8)"

g <- add_box(txt = txt1)

g <- add_side_box(g, txt = txt1_side)

g <- add_box(g, txt = "Randomized (n=200)")

g <- add_split(g, txt = c("Arm A (n=100)", "Arm B (n=100"))
g <- add_side_box(g,
  txt = c(
    "Excluded (n=15):\n
                  \u2022 MRI not collected (n=3)\n
                  \u2022 Tissues not collected (n=4)\n
                   \u2022 Other (n=8)",
    "Excluded (n=15):\n
                   \u2022 MRI not collected (n=3)\n
                   \u2022 Tissues not collected (n=4)"
  )
)

g <- add_box(g, txt = c("Final analysis (n=100)", "Final analysis (n=100"))
g <- add_label_box(g, txt = c("1" = "Screening", "3" = "Randomized", "4" = "Final analysis"))
```

build_grid *Build consort diagram*

Description

Build a grob consort diagram, use this if you want to save plots with ggsave. Splits can be nested to any depth.

Usage

```
build_grid(x, shrink_to_fit = FALSE)
```

Arguments

x	A consort object.
shrink_to_fit	If TRUE, shrink the whole diagram (text included) so it fits the currently open graphics device, never enlarging it beyond its natural size. Default is FALSE, which always lays out the diagram at its natural size – the right choice when the result is passed on to ggsave() rather than drawn directly.

Details

The diagram is laid out at a fixed, device-independent "natural" size, returned (in inches) as the size attribute of the result, which can be passed straight to ggsave's width/ height for an undistorted export, e.g. sz <- attr(build_grid(x), "size"); ggsave(..., width = sz$width, height = sz$height). With shrink_to_fit = TRUE, the diagram is scaled down (text, boxes and spacing together, never enlarged) to fit the currently open graphics device; this is what plot.consort uses so an interactive plot window never ends up with overlapping nodes.

Value

A gList object, with a size attribute giving the diagram's natural width/height in inches.

See Also

gList

Examples

```
## Not run:
txt1 <- "Population (n=300)"
txt1_side <- "Excluded (n=15): \n
              \u2022 MRI not collected (n=3)\n
              \u2022 Tissues not collected (n=4)\n
              \u2022 Other (n=8)"

g <- add_box(txt = txt1)
```

```
g <- add_side_box(g, txt = txt1_side)

g <- add_box(g, txt = "Randomized (n=200)")
gr <- build_grid(g)
sz <- attr(gr, "size")
# ggsave("consort_diagram.pdf", plot = gr, width = sz$width, height = sz$height)

## End(Not run)
```

---

build_grviz *Build consort diagram as Graphviz DOT code*

Description

Build the Graphviz DOT representation of a consort object, use this if you want to render with grViz (see `plot(x, grViz = TRUE)`) or export via export_svg. Splits can be nested to any depth.

Usage

```
build_grviz(x)
```

Arguments

x	A consort object.

Value

A Graphviz DOT code string.

See Also

grViz

Examples

```
## Not run:
txt1 <- "Population (n=300)"
txt1_side <- "Excluded (n=15): \n
              \u2022 MRI not collected (n=3)\n
              \u2022 Tissues not collected (n=4)\n
              \u2022 Other (n=8)"

g <- add_box(txt = txt1)

g <- add_side_box(g, txt = txt1_side)

g <- add_box(g, txt = "Randomized (n=200)")
# plot(g, grViz = TRUE)

## End(Not run)
```

connect_box *Connect grob box with arrow.*

Description

This function is used to create an arrow line to connect two boxes. User should provide the starting and ending side of the arrow.

Usage

```
connect_box(
  start,
  end,
  connect,
  type = c("s", "p"),
  name = NULL,
  arrow_gp = consort_opt("arrow_gp"),
  arrow_length = consort_opt("arrow_length"),
  arrow_type = consort_opt("arrow_type")
)
```

Arguments

start	Starting point of the arrow.
end	Ending point of the arrow.
connect	The connection of the box. It should be the combination of the position. The `t` refers to "top", `l` for "left", `b` for "bottom" and `r` for "right". The first letter is the starting point of the start box, the second is the ending point of the end box. For example, if one wants to connect the left side of the start box with right side of the left side of the end box, the value should be `"lr"`. All the connection will be started in the middle point.
type	Should be one the `"s"` (strait line), or `"p"` (polyline).
name	A character identifier of the line grob, passed to `linesGrob`.
arrow_gp	Graphical parameters `gpar` object for the connector line. Defaults to `grid::gpar(col = "black", lwd = 1)`.
arrow_length	Length of the arrowhead as a `unit` object. Defaults to `unit(0.1, "inches")`.
arrow_type	Arrow type, either `"closed"` or `"open"`. Defaults to `"closed"`.

Value

A lines grob with arrow.

Examples

```
fg1 <- textbox(text = "This is a test")
fg2 <- textbox(text = "This is an other test", 0.7, 0.2)
grid::grid.draw(fg1)
grid::grid.draw(fg2)
connect_box(fg1, fg2, connect = "bl", type = "p")

# Customize line width and color
connect_box(fg1, fg2, connect = "bl", type = "p", arrow_gp = grid::gpar(lwd = 2, col = "red"))

# Or set globally via set_consort_defaults
set_consort_defaults(arrow_gp = grid::gpar(lwd = 2, col = "blue"))
```

---

consort_plot *Self generating consort diagram*

Description

Create CONSORT diagram from a participant disposition data.

Usage

```
consort_plot(
  data,
  orders,
  side_box,
  allocation = NULL,
  labels = NULL,
  kickoff_sidebox = TRUE,
  cex = 0.8,
  text_width = NULL,
  drop_levels = TRUE
)
```

Arguments

data	Data set with disposition information for each participants.
orders	A named vector or a list, names as the variable in the dataset and values as labels in the box. The order of the diagram will be based on this. A list can be used to report multiple variable in a single node, the first variable in a list element will be used to report the total and the exact items will be summarised for the remaining variable. This is limited to non-side box.
side_box	Variable vector, appeared as side box in the diagram. The next box will be the subset of the missing values of these variables.
allocation	Name of the grouping/treatment variable (optional), the diagram will split into branches on this variables forward. For a factorial design, with several splits, a character vector with one variable per split can be provided, the splits are nested in that order. The extra box will be skipped if the values in the orders blank.

labels — Named vector, names is the location of the terminal node. The position location should plus 1 after the allocation variables if the allocation is defined.

kickoff_sidebox — remove (default) the side box observations from the following counting.

cex — Multiplier applied to font size, default is 0.8. Prefer using `set_consort_defaults(txt_gp = gpar(cex = ...))` instead.

text_width — a positive integer giving the target column for wrapping lines in the output. String will not be wrapped if not defined (default). The `stri_wrap` function will be used if `stringi` package installed, otherwise `strwrap` will be used.

drop_levels — If `TRUE` (default), unused factor levels are dropped when tabulating counts, so categories with zero counts are omitted from the boxes. Set to `FALSE` to report zero-count factor levels as `(n=0)`, e.g. to display an exclusion criterion that was applied but excluded nobody. Only applies to factor variables, see `gen_text`.

Details

The calculation of numbers is as in an analogous to Kirchhoff's Laws of electricity. The numbers in terminal nodes must sum to those in the ancestor nodes. All the drop outs will be populated as a side box. Which was different from the official CONSORT diagram template, which has dropout inside a vertical node.

Value

A `consort` object.

See Also

`add_side_box`,`add_split`, `add_side_box` `textbox` `set_consort_defaults`

Examples

```
## Prepare test data
data(dispos.data)

df <- dispos.data[!dispos.data$arm3 %in% "Trt C", ]
p <- consort_plot(data = df,
                  orders = list(c(trialno = "Population"),
                                c(exclusion = "Excluded"),
                                c(arm     = "Randomized patient"),
                                c(arm3     = "",
                                  subjid_notdosed="Participants not treated"),
                                c(followup    = "Pariticpants planned for follow-up",
                                  lost_followup = "Reason for tot followed"),
                                c(assessed = "Assessed for final outcome"),
                                c(no_value = "Reason for not assessed"),
                                c(mitt = "Included in the mITT analysis")),
                  side_box = c("exclusion", "no_value"),
                  allocation = c("arm", "arm3"),
                  labels = c("1" = "Screening", "2" = "Randomization",
                             "5" = "Follow-up", "7" = "Final analysis"),
```

```
                    cex = 0.7)
```

`dispos.data` *Demo clinical trial disposition data*

Description

This is a demo data to demonstrate the use of the package. One row per participant. Participants who are excluded should provide a reason, missing otherwise.

Usage

```
dispos.data
```

Format

A data frame with 300 rows and 11 columns:

trialno Participants ID of the participants

exclusion1, exclusion2 Exclusion reason before and after induction

induction Participants ID of the participants who are included in the induction phase, an extra treatment before randomisation.

exclusion Exclusion reason before randomisation, including before and after induction

arm, arm3 Arms pariticipants randomised to.

sbujid_dosed Participants ID of the participants who had at least one dose of the protocol treatment.

subjid_notdosed Reason for participants not dosed.

followup Participants ID planned for follow-up.

lost_followup Reason for participants not dosed.

assessed Participants ID participants attended assessment.

no_value Reason for participants missing final assessment.

mitt Participants ID included in the mITT analysis.

gen_text *Generate label and bullet points*

Description

This function use the data to generate label and bullet points for the box.

Usage

```
gen_text(
  x,
  label = NULL,
  bullet = FALSE,
  bullet_char = consort_opt("bullet"),
  drop_levels = TRUE
)
```

Arguments

x	A list or a vector to be used. x can be atomic vector, a data.frame or a list. A data.frame is particular useful if the there's a nested reason or a list split nested reasons by group. The nested reasons only support two columns and the bullet will be ignored.
label	A character string as a label at the beginning of the text label. The count for each categories will be returned if no label is provided.
bullet	If shows bullet points. If the value is TRUE, the bullet points will be tabulated, default is FALSE.
bullet_char	A single character used as the bullet symbol. Defaults to consort_opt("bullet"). Can be any Unicode character such as "\u2013" (en-dash), "\u25CB" (circle), "\u25A0" (square), "-", etc.
drop_levels	If TRUE (default), unused factor levels are dropped before tabulation, so categories with zero counts are omitted. Set to FALSE to report zero-count factor levels as (n=0), e.g. to display an exclusion criterion that was applied but excluded nobody. Only applies when x (or the reason column) is a factor.

Value

A character string of vector.

Examples

```
val <- data.frame(
  am = factor(ifelse(mtcars$am == 1, "Automatic", "Manual"), ordered = TRUE),
  vs = factor(ifelse(mtcars$vs == 1, "Straight", "V-shaped"), ordered = TRUE),
  car = row.names(mtcars)
)
```

```
gen_text(val$car, label = "Cars in the data")
gen_text(val$car, label = "Cars in the data", bullet = FALSE)
gen_text(split(val$car, val$am), label = "Cars in the data")
gen_text(split(val$car, val$am), label = "Cars in the data", bullet = FALSE)
gen_text(split(val[,c("vs", "car")], val$am), label = "Cars in the data", bullet = FALSE)
gen_text(val[,c("vs", "car")], label = "Cars in the data", bullet = FALSE)

# Use a custom bullet character
gen_text(val$car, label = "Cars in the data", bullet = TRUE, bullet_char = "-")

# Or set globally via set_consort_defaults
set_consort_defaults(bullet = "\u25CB")
gen_text(val$car, label = "Cars in the data", bullet = TRUE)

# Report zero-count factor levels with (n=0)
reason <- factor(c("Ineligible", NA, "Declined", NA),
                 levels = c("Ineligible", "Declined", "Other"))
gen_text(reason, label = "Excluded", bullet = TRUE)
gen_text(reason, label = "Excluded", bullet = TRUE, drop_levels = FALSE)
```

`get_coords` *Get the coordinates of the textbox object*

Description

This function will get the coordinates of the textbox object.

Usage

```
get_coords(x)
```

Arguments

x	A textbox object

Value

A list of coordinates will return:

`left`	Left (x-min) side coordinate.
`right`	Right (x-max) side coordinate.
`bottom`	Bottom (y-min) side coordinate.
`top`	Top (y-max) side coordinate.
`top.mid`	Coordinate vector of top middle, measured by grob.
`left.mid`	Coordinate vector of left middle, measured by grob.
`bottom.mid`	Coordinate vector of bottom middle, measured by grob.
`right.mid`	Coordinate vector of right middle, measured by grob.

x	X (center x) coordinate.
y	Y (center y) coordinate.
width	Width of the textbox, derived with grobWidth.
height	Height of the textbox, derived with grobHeight.
half_width	Half width of the box.
half_height	Half height of the box.

Examples

```
fg <- textbox(text = "This is a test")
get_coords(fg)
```

---

move_box *Move a box grob*

Description

This function can be used to move the box to a given position with editGrob changing the x and y value.

Usage

```
move_box(obj, x = NULL, y = NULL, pos_type = c("absolute", "relative"))
```

Arguments

obj	A box object.
x	A unit element or a number that can be converted to npc, see unit.
y	A unit element or a number that can be converted to npc, see unit.
pos_type	If the provided coordinates are absolute position the box will be moved to or it's a relative position to it's current.

Value

A box object with updated x and y coordinates.

Examples

```
fg <- textbox(text = "This is a test")
fg2 <- move_box(fg, 0.3, 0.3)
```

plot.consort *Add methods to print function*

Description

Method for plot objects and display the output in on a grid device.

Usage

```
## S3 method for class 'consort'
plot(x, grViz = FALSE, ...)

## S3 method for class 'consort'
print(x, grViz = FALSE, ...)
```

Arguments

x	A consort object.
grViz	If use grViz to print the plot. Default is FALSE to use grid.draw
...	Not used.

Value

None.

See Also

add_side_box,add_split, add_side_box, grid.draw

set_consort_defaults *Set consort diagram default options*

Description

Modify the default graphical parameters and other settings for consort diagrams. Any parameter set to NULL (the default) will remain unchanged. gpar arguments are merged with the current values, so only the properties to change need to be given. Use init_consort_defaults() to restore the package defaults, and get_consort_defaults() to view the current settings.

Usage

```
set_consort_defaults(
  arrow_gp = NULL,
  txt_gp = NULL,
  box_gp = NULL,
  label_txt_gp = NULL,
  label_box_gp = NULL,
  arrow_length = NULL,
  arrow_type = NULL,
  pad_u = NULL,
  bullet = NULL,
  parse_markup = NULL
)

get_consort_defaults()

init_consort_defaults()

## S3 method for class 'consort_defaults'
print(x, ...)
```

Arguments

arrow_gp	A gpar object for the arrow line (colour, line width, line type, etc.). Package default is gpar(col = "black", lwd = 1).
txt_gp	A gpar object for the text inside boxes (font size, colour, font family, etc.). Package default is gpar(cex = 1, col = "black").
box_gp	A gpar object for the box border and fill. Package default is gpar(fill = "white").
label_txt_gp	A gpar object for the label text. Package default is gpar(col = "#4F81BD", cex = 1, fontface = "bold").
label_box_gp	A gpar object for the label box. Package default is gpar(fill = "#A9C7FD").
arrow_length	Numeric, length of the arrowhead in inches. Default is 0.1.
arrow_type	Character, arrow type: "closed" (default) or "open".
pad_u	Numeric, padding between nodes in character units. Default is 3.
bullet	Character, bullet character for side box items, e.g. an en dash or a hyphen. Default is the Unicode bullet (U+2022).
parse_markup	Logical, whether to parse lightweight markup syntax (**bold**, *italic*, ^{superscript}, _{subscript}, __underline__) in node labels. Default is FALSE.
x	A consort_defaults object.
...	Not used.

Value

Invisibly returns the previous defaults (a `consort_defaults` object).

A `consort_defaults` object containing all current default settings.

Examples

```
# Change text color and box fill
old <- set_consort_defaults(
  txt_gp = grid::gpar(col = "navy", cex = 0.9),
  box_gp = grid::gpar(fill = "#F0F0F0")
)

# View current defaults
get_consort_defaults()

# Restore previous defaults
set_consort_defaults(
  txt_gp = old$txt_gp,
  box_gp = old$box_gp
)
get_consort_defaults()
```

---

textbox *Create a box with text*

Description

Create a grob with text inside. To extract the units describing grob boundary location can be accessed with grobX and grobY. The units describing width and height can be accessed with grobWidth and grobHeight.

Usage

```
textbox(
  text,
  x = unit(0.5, "npc"),
  y = unit(0.5, "npc"),
  just = c("center", "left", "right"),
  txt_gp = consort_opt("txt_gp"),
  leftrotate = FALSE,
  box_fn = roundrectGrob,
  box_gp = consort_opt("box_gp"),
  name = "textbox"
)

grid.textbox(...)
```

Arguments

text	A character text to be passed to textGrob.
x	A number or unit object specifying x-location.
y	A number or unit object specifying y-location.
just	The justification of the text, ‘"left"‘, ‘"right‘ and ‘"center"‘. See textGrob for more details.
txt_gp	An object of class gpar style to be applied to the text. Defaults are read from set_consort_defaults. For example, to set a font size for all text inside boxes, use set_consort_defaults(txt_gp = gpar(cex = 0.8)).
leftrotate	If the text box will be rotated 90 degrees counter-clockwise. Default is FALSE.
box_fn	Function to create box for the text. Parameters of ‘x=0.5‘, ‘y=0.5‘ and ‘box_gp‘ will be passed to this function and return a grob object.
box_gp	An object of class gpar style to be applied to the box.
name	A character identifier.
...	Parameters passed to textbox

Value

A text box grob. grid.textbox() returns the value invisibly.

Examples

```
fg <- textbox(text = "This is a test")
grid::grid.draw(fg)
grid.textbox(text = "This is a test")
grid.textbox(text = "This is a test")
```

Index

∗ **datasets**
    dispos.data, 13

add_box, 2, *4, 5*, *7*
add_label_box, *3*, 3, *5*
add_side_box, *3, 4*, 5, *7*, *12*, *17*
add_split, *3–5*, 6, *12*, *17*

build_grid, 8
build_grviz, 9

connect_box, 10
consort_plot, 11

dispos.data, 13

editGrob, *16*
export_svg, *9*

gen_text, *12*, 14
get_consort_defaults
        *(*set_consort_defaults*)*, 17
get_coords, 15
ggsave, *8*
gList, *8*
gpar, *10*, *17, 18*, *20*
grid.draw, *17*
grid.textbox *(*textbox*)*, 19
grob, *19*
grobHeight, *19*
grobWidth, *19*
grobX, *19*
grobY, *19*
grViz, *5*, *9*, *17*

init_consort_defaults
        *(*set_consort_defaults*)*, 17

linesGrob, *10*

move_box, 16

plot.consort, *8*, 17
print.consort *(*plot.consort*)*, 17
print.consort_defaults
        *(*set_consort_defaults*)*, 17

roundrectGrob, *4*

set_consort_defaults, *12*, 17, *20*
stri_wrap, *3*, *5*, *7*, *12*
strwrap, *3*, *5*, *7*, *12*

textbox, *3–5*, *7*, *12*, 19
textGrob, *20*

unit, *10*, *16*