

Package ‘consort’

January 5, 2023

Type Package

Title Create Consort Diagram

Version 1.1.0

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/>

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

Encoding UTF-8

Imports grid

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

VignetteBuilder knitr

RoxygenNote 7.2.1

Config/testthat/edition 3

NeedsCompilation no

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

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

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2023-01-05 17:40:09 UTC

R topics documented:

consort-package	2
add_box	2
add_label_box	4

add_side_box	5
add_split	7
build_grid	8
build_grviz	9
connect_box	10
consort_plot	11
gen_text	14
get_coords	15
move_box	16
plot.consort	17
textbox	17

Index	19
--------------	-----------

consort-package	<i>Create Consort diagram</i>
-----------------	-------------------------------

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.

add_box	<i>Add nodes</i>
---------	------------------

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 <code>stri_wrap</code> function will be used if <code>stringi</code> package installed as it is a better options for non-Latin language, otherwise <code>strwrap</code> will be used.
...	Other parameters pass to <code>textbox</code> ,

Value

A consort object.

See Also

[add_side_box](#) [add_split](#) [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"
  )
)

```

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"),
  ...
)

```

Arguments

prev_box	A completed diagram created with <code>add_box</code> , <code>add_side_box</code> 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.
...	Other parameters pass to textbox ,

Value

A consort object.

See Also

[add_side_box](#) [add_split](#) [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"))

```

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 <code>stringi</code> 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](#)

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	<i>Add a splitting box</i>
-----------	----------------------------

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, 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.
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 <code>stringi</code> 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](#)

Usage

```
build_grid(x)
```

Arguments

x A consort object.

Value

A gList object

See Also

[gList](#)

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 <- ggplot2::ggsave("consort_diagram.pdf", plot = build_grid(g))

```

build_grviz	<i>Build consort diagram</i>
-------------	------------------------------

Description

Build a grob consort diagram, use this if you want to save plots with [ggsave](#)

Usage

```
build_grviz(x)
```

Arguments

x A consort object.

Value

A Graphviz code

See Also

[grViz](#)

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)")
plot(g, grViz = TRUE)
```

connect_box	<i>Connect grob box with arrow.</i>
-------------	-------------------------------------

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)
```

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 .

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")
```

consort_plot	<i>Self generating consort diagram</i>
--------------	----------------------------------------

Description

Create CONSORT diagram from a participant disposition data.

Usage

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

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.
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.
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.
cex	Multiplier applied to font size, Default is 0.8
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 <code>stri_wrap</code> function will be used if <code>stringi</code> package installed, otherwise <code>strwrap</code> will be used.

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](#)

Examples

```
## Prepare test data
set.seed(1001)
N <- 300

trialno <- sample(c(1000:2000), N)
exc1 <- rep(NA, N)
exc1[sample(1:N, 15)] <- sample(c("Sample not collected", "MRI not collected", "Other"),
  15,
  replace = TRUE, prob = c(0.4, 0.4, 0.2)
)

induc <- rep(NA, N)
induc[is.na(exc1)] <- trialno[is.na(exc1)]

exc2 <- rep(NA, N)
exc2[sample(1:N, 20)] <- sample(c(
  "Sample not collected", "Dead",
  "Other"
), 20,
  replace = TRUE,
  prob = c(0.4, 0.4, 0.2)
)
exc2[is.na(induc)] <- NA

exc <- ifelse(is.na(exc2), exc1, exc2)

arm <- rep(NA, N)
arm[is.na(exc)] <- sample(c("Conc", "Seq"), sum(is.na(exc)), replace = TRUE)
arm3 <- sample(c("Trt A", "Trt B", "Trt C"), N, replace = TRUE)
arm3[is.na(arm)] <- NA

fow1 <- rep(NA, N)
fow1[!is.na(arm)] <- sample(c("Withdraw", "Discontinued", "Death", "Other", NA),
  sum(!is.na(arm)),
  replace = TRUE,
  prob = c(0.05, 0.05, 0.05, 0.05, 0.8)
)
fow2 <- rep(NA, N)
fow2[!is.na(arm) & is.na(fow1)] <- sample(c("Protocol deviation", "Outcome missing", NA),
  sum(!is.na(arm) & is.na(fow1)),
  replace = TRUE,
  prob = c(0.05, 0.05, 0.9)
)

df <- data.frame(trialno, exc1, induc, exc2, exc, arm, arm3, fow1, fow2)
rm(trialno, exc1, induc, exc2, exc, arm, arm3, fow1, fow2, N)
```

```
## Single arm
out <- consort_plot(
  data = df,
  order = c(
    trialno = "Population",
    exc1 = "Excluded",
    arm = "Allocated",
    fow1 = "Lost of Follow-up",
    trialno = "Finished Followup",
    fow2 = "Not evaluable for the final analysis",
    trialno = "Final Analysis"
  ),
  side_box = c("exc1", "fow1", "fow2"),
  cex = 0.9
)

## Two arms
out <- consort_plot(
  data = df,
  order = c(
    trialno = "Population",
    exc = "Excluded",
    arm = "Randomized patient",
    fow1 = "Lost of Follow-up",
    trialno = "Finished Followup",
    fow2 = "Not evaluable",
    trialno = "Final Analysis"
  ),
  side_box = c("exc", "fow1", "fow2"),
  allocation = "arm",
  labels = c(
    "1" = "Screening", "2" = "Randomization",
    "5" = "Final"
  )
)

## Three arms
consort_plot(
  data = df,
  order = c(
    trialno = "Population",
    exc = "Excluded",
    arm3 = "Randomized patient",
    fow1 = "Lost of Follow-up",
    trialno = "Finished Followup",
    fow2 = "Not evaluable",
    trialno = "Final Analysis"
  ),
  side_box = c("exc", "fow1", "fow2"),
  allocation = "arm3",
  labels = c(
    "1" = "Screening", "2" = "Randomization",
    "5" = "Final"
  )
)
```

```

)
)

## Multiple phase
consort_plot(
  data = df,
  order = list(
    trialno = "Population",
    exc1 = "Excluded",
    induc = "Induction",
    exc2 = "Excluded",
    arm3 = "Randomized patient",
    fow1 = "Lost of Follow-up",
    trialno = "Finished Followup",
    fow2 = "Not evaluable",
    trialno = "Final Analysis"
  ),
  side_box = c("exc1", "exc2", "fow1", "fow2"),
  allocation = "arm3",
  labels = c(
    "1" = "Screening", "2" = "Month 4",
    "3" = "Randomization", "5" = "Month 24",
    "6" = "End of study"
  ),
  cex = 0.7
)

```

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)
```

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'. |

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)
```

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	<i>Move a box grob</i>
----------	------------------------

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	<i>Add methods to print function</i>
--------------	--------------------------------------

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](#)

textbox	<i>Create a box with text</i>
---------	-------------------------------

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 = getOption("txt_gp", default = gpar(color = "black", cex = 1)),
  box_fn = roundrectGrob,
  box_gp = getOption("box_gp", default = gpar(fill = "white")),
  name = "textbox"
)

grid.textbox(...)

```

Arguments

<code>text</code>	A character text to be passed to textGrob .
<code>x</code>	A number or unit object specifying x-location.
<code>y</code>	A number or unit object specifying y-location.
<code>just</code>	The justification of the text, "left", "right" and "center". See textGrob for more details.
<code>txt_gp</code>	An object of class gpar style to be applied to the text. This will also be read from global options of "txt_gp". For example, if one wants to set a font size for all the text inside box, <code>options(txt_gp = gpar(cex = 0.8))</code> will do the trick.
<code>box_fn</code>	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. This will also be read from global options of "box_gp".
<code>box_gp</code>	An object of class gpar style to be applied to the box.
<code>name</code>	A character identifier.
<code>...</code>	Parameters passed to <code>textbox</code>

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

`add_box`, [2](#), [6](#), [7](#)
`add_label_box`, [4](#)
`add_side_box`, [3](#), [4](#), [5](#), [7](#), [12](#), [17](#)
`add_split`, [3](#), [4](#), [6](#), [7](#), [12](#), [17](#)

`build_grid`, [8](#)
`build_grviz`, [9](#)

`connect_box`, [10](#)
`consort-package`, [2](#)
`consort_plot`, [11](#)

`editGrob`, [16](#)

`gen_text`, [14](#)
`get_coords`, [15](#)
`ggsave`, [8](#), [9](#)
`gList`, [8](#)
`gpar`, [18](#)
`grid.draw`, [17](#)
`grid.textbox` (`textbox`), [17](#)
`grob`, [17](#)
`grobHeight`, [17](#)
`grobWidth`, [17](#)
`grobX`, [17](#)
`grobY`, [17](#)
`grViz`, [6](#), [9](#), [17](#)

`linesGrob`, [10](#)

`move_box`, [16](#)

`plot.consort`, [17](#)
`print.consort` (`plot.consort`), [17](#)

`stri_wrap`, [3](#), [6](#), [7](#), [11](#)
`strwrap`, [3](#), [6](#), [7](#), [11](#)

`textbox`, [3](#), [4](#), [6](#), [7](#), [12](#), [17](#)
`textGrob`, [18](#)

`unit`, [16](#)