

# Package ‘consort’

July 22, 2025

**Type** Package

**Title** Create Consort Diagram

**Version** 1.2.2

**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** 7.2.3

**Config/testthat/edition** 3

**Depends** R (>= 3.5.0)

**LazyData** true

**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** 2024-06-04 13:40:02 UTC

## Contents

consort-package . . . . .	2
add_box . . . . .	2
add_label_box . . . . .	4
add_side_box . . . . .	5
add_split . . . . .	6
build_grid . . . . .	8
build_grviz . . . . .	9
connect_box . . . . .	10
consort_plot . . . . .	11
dispos.data . . . . .	12
gen_text . . . . .	13
get_coords . . . . .	14
move_box . . . . .	15
plot.consort . . . . .	15
textbox . . . . .	16

<b>Index</b>	<b>18</b>
--------------	-----------

---

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](#) [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"),
  ...
)
```

### 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 <a href="#">textbox</a> ,

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

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

**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`. `build_grid` does not support multiple split for the moment, please use `build_grviz` or `plot(g, grViz = TRUE)` for multiple split nodes instead.

### Usage

```
build_grid(x)
```

### Arguments

`x` A consort object.

### Value

A `gList` object

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

```



```
## End(Not run)
```

---

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

```
## 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	<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 <a href="#">linesGrob</a> .

## 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

*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
)
```

## 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 two splits for example, a character vector with a maximum of length two can be provided. 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
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
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 = "Pariticipants 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 participants randomised to.

**sbujid\_dosed** Participants ID of the participants who had at least one dose of the protocol treatment.

**sbujid\_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)
```

**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(iffelse(mtcars$am == 1, "Automatic", "Manual"), ordered = TRUE),
  vs = factor(iffelse(mtcars$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**

<code>x</code>	A textbox object
----------------	------------------

**Value**

A list of coordinates will return:

<code>left</code>	Left (x-min) side coordinate.
<code>right</code>	Right (x-max) side coordinate.
<code>bottom</code>	Bottom (y-min) side coordinate.
<code>top</code>	Top (y-max) side coordinate.
<code>top.mid</code>	Coordinate vector of top middle, measured by grob.
<code>left.mid</code>	Coordinate vector of left middle, measured by grob.
<code>bottom.mid</code>	Coordinate vector of bottom middle, measured by grob.
<code>right.mid</code>	Coordinate vector of right middle, measured by grob.
<code>x</code>	X (center x) coordinate.
<code>y</code>	Y (center y) coordinate.
<code>width</code>	Width of the textbox, derived with grobWidth.
<code>height</code>	Height of the textbox, derived with grobHeight.
<code>half_width</code>	Half width of the box.
<code>half_height</code>	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 <a href="#">unit</a> .
y	A unit element or a number that can be converted to npc, see <a href="#">unit</a> .
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 <a href="#">grViz</a> to print the plot. Default is FALSE to use <a href="#">grid.draw</a>
...	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**

text	A character text to be passed to <a href="#">textGrob</a> .
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 <a href="#">textGrob</a> for more details.



txt_gp	An object of class <code>gpar</code> 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.
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. This will also be read from global options of "box_gp".
box_gp	An object of class <code>gpar</code> style to be applied to the box.
name	A character identifier.
...	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

- \* **datasets**
  - dispos.data, 12
- add\_box, 2, 4, 6, 7
- add\_label\_box, 3, 4, 6
- add\_side\_box, 3, 4, 5, 7, 12, 16
- add\_split, 3, 4, 6, 6, 12, 16
  
- build\_grid, 8
- build\_grviz, 8, 9
  
- connect\_box, 10
- consort-package, 2
- consort\_plot, 11
  
- dispos.data, 12
  
- editGrob, 15
  
- gen\_text, 13
- get\_coords, 14
- ggsave, 8, 9
- gList, 8
- gpar, 17
- grid.draw, 16
- grid.textbox (textbox), 16
- grob, 16
- grobHeight, 16
- grobWidth, 16
- grobX, 16
- grobY, 16
- grViz, 5, 9, 16
  
- linesGrob, 10
  
- move\_box, 15
  
- plot.consort, 15
- print.consort (plot.consort), 15
  
- stri\_wrap, 3, 5, 7, 11
  
- strwrap, 3, 5, 7, 11
  
- textbox, 3–7, 12, 16
- textGrob, 16
  
- unit, 15