Package ‘gridtext’

October 13, 2022

Type Package
Title Improved Text Rendering Support for 'Grid' Graphics
Version 0.1.5
Description Provides support for rendering of formatted text using 'grid' graphics. Text can be formatted via a minimal subset of 'Markdown', 'HTML', and inline 'CSS' directives, and it can be rendered both with and without word wrap.
URL https://wilkelab.org/gridtext/
BugReports https://github.com/wilkelab/gridtext/issues
License MIT + file LICENSE
Depends R (>= 3.5)
Imports curl, grid, grDevices, markdown, rlang, Rcpp, png, jpeg, stringr, xml2
Suggests covr, knitr, rmarkdown, testthat, vdiffr
LinkingTo Rcpp
Encoding UTF-8
RoxygenNote 7.1.1
SystemRequirements C++11
NeedsCompilation yes
Author Claus O. Wilke [aut] (<https://orcid.org/0000-0002-7470-9261>), Brenton M. Wiernik [aut, cre] (<https://orcid.org/0000-0001-9560-6336>, @bmwiernik)
Maintainer Brenton M. Wiernik <brenton@wiernik.org>
Repository CRAN
Date/Publication 2022-09-16 11:16:15 UTC

R topics documented:
gridtext ................................................................. 2
richtext_grob .......................................................... 2
textbox_grob .......................................................... 4
The gridtext package provides two new grobs, `richtext_grob()` and `textbox_grob()`, which support drawing of formatted text labels and formatted text boxes, respectively.

**Usage**

```r
richtext_grob(
  text,
  x = unit(0.5, "npc"),
  y = unit(0.5, "npc"),
  hjust = 0.5,
  vjust = 0.5,
  halign = hjust,
  valign = vjust,
  rot = 0,
  default.units = "npc",
  margin = unit(c(0, 0, 0, 0), "pt"),
  padding = unit(c(0, 0, 0, 0), "pt"),
  r = unit(0, "pt"),
  align_widths = FALSE,
  align_heights = FALSE,
  name = NULL,
  gp = gpar(),
  box_gp = gpar(col = NA),
  vp = NULL,
  use_markdown = TRUE,
  debug = FALSE
)
```
Arguments

- **text**: Character vector containing Markdown/HTML strings to draw.
- **x, y**: Unit objects specifying the location of the reference point.
- **hjust, vjust**: Numerical values specifying the justification of the text boxes relative to x and y. These justification parameters are specified in the internal reference frame of the text boxes, so that, for example, hjust adjusts the vertical justification when the text is rotated 90 degrees to the left or right.
- **halign, valign**: Numerical values specifying the text justification inside the text boxes. If not specified, these default to hjust and vjust.
- **rot**: Angle of rotation for text, in degrees.
- **default.units**: Units of x and y if these are provided only as numerical values.
- **margin, padding**: Unit vectors of four elements each indicating the margin and padding around each text label in the order top, right, bottom, left. Margins are drawn outside the enclosing box (if any), and padding is drawn inside. To avoid rendering artifacts, it is best to specify these values in absolute units (such as points, mm, or inch) rather than in relative units (such as npc).
- **r**: The radius of the rounded corners. To avoid rendering artifacts, it is best to specify this in absolute units (such as points, mm, or inch) rather than in relative units (such as npc).
- **align_widths, align_heights**: Should the widths and heights of all the text boxes be aligned? Default is no.
- **name**: Name of the grob.
- **gp**: Other graphical parameters for drawing.
- **box_gp**: Graphical parameters for the enclosing box around each text label.
- **vp**: Viewport.
- **use_markdown**: Should the text input be treated as markdown? Default is yes.
- **debug**: Should debugging info be drawn? Default is no.

Value

A grid grob that represents the formatted text.

See Also

- textbox_grob()

Examples

```r
library(grid)

text <- c(
  "Some text **in bold.**", "Linebreaks<br>Linebreaks<br>Linebreaks", 
  "*x*<sup>2</sup> + 5*x* + *C*<sub>*i*</sub>", 
  "Some <span style='color:blue'>blue text **in bold.**</span><br>And *italics text.*"<br>
)```
And some <span style='font-size:18pt; color:black'>large</span> text."

x <- c(.2, .1, .7, .9)
y <- c(.8, .4, .1, .5)
rot <- c(0, 0, 45, -45)
gp = gpar(col = c("black", "red"), fontfamily = c("Palatino", "Courier", "Times", "Helvetica"))
box_gp = gpar(col = "black", fill = c("cornsilk", NA, "lightblue1", NA), lty = c(0, 1, 1, 1))
hjust <- c(0.5, 0, 0, 1)
vjust <- c(0.5, 1, 0, 0.5)

g <- richtext_grob(
    text, x, y, hjust = hjust, vjust = vjust, rot = rot,
    padding = unit(c(6, 6, 4, 6), "pt"),
    r = unit(c(0, 2, 4, 8), "pt"),
    gp = gp, box_gp = box_gp
)
grid.newpage()
grid.draw(g)
grid.points(x, y, default.units = "npc", pch = 19, size = unit(5, "pt"))

# multiple text labels with aligned boxes
text <- c("January", "February", "March", "April", "May")
x <- (1:5)/6 + 1/24
y <- rep(0.8, 5)
g <- richtext_grob(
    text, x, y, halign = 0, hjust = 1,
    rot = 45,
    padding = unit(c(3, 6, 1, 3), "pt"),
    r = unit(4, "pt"),
    align_widths = TRUE,
    box_gp = gpar(col = "black", fill = "cornsilk")
)
grid.newpage()
grid.draw(g)
grid.points(x, y, default.units = "npc", pch = 19, size = unit(5, "pt"))

textbox_grob

Draw formatted multi-line text with word wrap

Description

The function textbox_grob() is intended to render multi-line text labels that require automatic word wrapping. It is similar to richtext_grob(), but there are a few important differences. First, while richtext_grob() is vectorized, textbox_grob() is not. It can draw only a single text box at a time. Second, textbox_grob() doesn’t support rendering the text box at arbitrary angles. Only four different orientations are supported, corresponding to a rotation by 0, 90, 180, and 270 degrees.
textbox_grob

Usage

textbox_grob(
  text,
  x = NULL,
  y = NULL,
  width = unit(1, "npc"),
  height = NULL,
  minwidth = NULL,
  maxwidth = NULL,
  minheight = NULL,
  maxheight = NULL,
  hjust = 0.5,
  vjust = 0.5,
  halign = 0,
  valign = 1,
  default.units = "npc",
  margin = unit(c(0, 0, 0, 0), "pt"),
  padding = unit(c(0, 0, 0, 0), "pt"),
  r = unit(0, "pt"),
  orientation = c("upright", "left-rotated", "right-rotated", "inverted"),
  name = NULL,
  gp = gpar(),
  box_gp = gpar(col = NA),
  vp = NULL,
  use_markdown = TRUE
)

Arguments

text Character vector containing Markdown/HTML string to draw.

x, y Unit objects specifying the location of the reference point. If set to NULL (the default), these values are chosen based on the values of hjust and vjust such that the box is appropriately justified in the enclosing viewport.

width, height Unit objects specifying width and height of the grob. A value of NULL means take up exactly the space necessary to render all content. Use a value of unit(1, "npc") to have the box take up all available space.

minwidth, minheight, maxwidth, maxheight Min and max values for width and height. Set to NULL to impose neither a minimum nor a maximum. Note: minheight and maxheight do not work if width = NULL.

hjust, vjust Numerical values specifying the justification of the text box relative to the reference point defined by x and y. These justification parameters are specified in the internal reference frame of the text box, so that, for example, hjust adjusts the vertical justification when the text box is left- or right-rotated.

halign, valign Numerical values specifying the justification of the text inside the text box.

default.units Units of x, y, width, height, minwidth, minheight, maxwidth, maxheight if these are provided only as numerical values.
margin, padding

Unit vectors of four elements each indicating the margin and padding around each text label in the order top, right, bottom, left. Margins are drawn outside the enclosing box (if any), and padding is drawn inside. To avoid rendering artifacts, it is best to specify these values in absolute units (such as points, mm, or inch) rather than in relative units (such as npc).

r

The radius of the rounded corners. To avoid rendering artifacts, it is best to specify this in absolute units (such as points, mm, or inch) rather than in relative units (such as npc).

orientation

Orientation of the box. Allowed values are "upright", "left-rotated", "right-rotated", and "inverted", corresponding to a rotation by 0, 90, 270, and 180 degrees counter-clockwise, respectively.

name

Name of the grob.

gp

Other graphical parameters for drawing.

box_gp

Graphical parameters for the enclosing box around each text label.

vp

Viewport.

use_markdown

Should the text input be treated as markdown?

Value

A grid grob that represents the formatted text.

See Also

richtext_grob()

Examples

library(grid)
g <- textbox_grob("\n**The quick brown fox jumps over the lazy dog.**
The quick brown fox jumps over the lazy dog.
The **quick fox** jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
",
x = unit(0.5, "npc"), y = unit(0.7, "npc"), halign = 0, valign = 1,
gp = gpar(fontsize = 15),
box_gp = gpar(col = "black", fill = "lightcyan1"),
r = unit(5, "pt"),
padding = unit(c(10, 10, 10, 10), "pt"),
margin = unit(c(0, 10, 0, 10), "pt")
)
grid.newpage()
grid.draw(g)

# internal vs. external alignment
g1 <- textbox_grob("The quick brown fox jumps over the lazy dog.
",
hjust = 0, vjust = 1, halign = 0, valign = 1,
width = unit(1.5, "inch"), height = unit(1.5, "inch"),
box_gp = gpar(col = "black", fill = "cornsilk"),
padding = unit(c(2, 2, 2, 2), "pt"),
margin = unit(c(5, 5, 5, 5), "pt")
)
g2 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
hjust = 1, vjust = 1, halign = 0.5, valign = 0.5,
width = unit(1.5, "inch"), height = unit(1.5, "inch"),
box_gp = gpar(col = "black", fill = "cornsilk"),
padding = unit(c(2, 2, 2, 2), "pt"),
margin = unit(c(5, 5, 5, 5), "pt")
)
g3 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
hjust = 0, vjust = 0, halign = 1, valign = 1,
width = unit(1.5, "inch"), height = unit(1.5, "inch"),
box_gp = gpar(col = "black", fill = "cornsilk"),
padding = unit(c(2, 2, 2, 2), "pt"),
margin = unit(c(5, 5, 5, 5), "pt")
)
g4 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
hjust = 1, vjust = 0, halign = 0, valign = 0,
width = unit(1.5, "inch"), height = unit(1.5, "inch"),
box_gp = gpar(col = "black", fill = "cornsilk"),
padding = unit(c(2, 2, 2, 2), "pt"),
margin = unit(c(5, 5, 5, 5), "pt")
)
grid.newpage()
grid.draw(g1)
grid.draw(g2)
grid.draw(g3)
grid.draw(g4)

# internal vs. external alignment, with rotated boxes
g1 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
hjust = 1, vjust = 1, halign = 0, valign = 1,
width = unit(1.5, "inch"), height = unit(1.5, "inch"),
orIENTATION = "left-rotated",
box_gp = gpar(col = "black", fill = "cornsilk"),
padding = unit(c(2, 2, 2, 2), "pt"),
margin = unit(c(5, 5, 5, 5), "pt")
)
g2 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
hjust = 0, vjust = 1, halign = 0.5, valign = 0.5,
width = unit(1.5, "inch"), height = unit(1.5, "inch"),
orIENTATION = "right-rotated",
box_gp = gpar(col = "black", fill = "cornsilk"),
padding = unit(c(2, 2, 2, 2), "pt"),
margin = unit(c(5, 5, 5, 5), "pt")
)
g3 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
  hjust = 1, vjust = 1, halign = 1, valign = 1,
  width = unit(1.5, "inch"), height = unit(1.5, "inch"),
  orientation = "inverted",
  box_gp = gpar(col = "black", fill = "cornsilk"),
  padding = unit(c(2, 2, 2, 2), "pt"),
  margin = unit(c(5, 5, 5, 5), "pt")
)

grid.newpage()
g3 <- textbox_grob(
  "The quick brown fox jumps over the lazy dog.",
  hjust = 1, vjust = 0, halign = 0, valign = 0,
  width = unit(1.5, "inch"), height = unit(1.5, "inch"),
  orientation = "upright",
  box_gp = gpar(col = "black", fill = "cornsilk"),
  padding = unit(c(2, 2, 2, 2), "pt"),
  margin = unit(c(5, 5, 5, 5), "pt")
)

grid.newpage()
grid.draw(g1)
grid.draw(g2)
grid.draw(g3)
grid.draw(g4)
Index

grid::textGrob(), 2
gridtext, 2
grob, 3, 6

plotmath, 2

richtext_grob, 2
richtext_grob(), 2, 4, 6

textbox_grob, 4
textbox_grob(), 2, 3