Package ‘gtable’

September 1, 2022

Title Arrange 'Grobs' in Tables

Version 0.3.1

Description Tools to make it easier to work with ``tables'' of
'grobs'. The 'gtable' package defines a 'gtable' grob class that specifies a
grid along with a list of grobs and their placement in the grid. Further the
package makes it easy to manipulate and combine 'gtable' objects so that
complex compositions can be built up sequentially.

License MIT + file LICENSE

Depends R (>= 3.0)

Imports grid

Suggests covr, testthat, knitr, rmarkdown, ggplot2, profvis

Encoding UTF-8

RoxygenNote 7.2.1

Collate 'new-data-frame.r' 'add-grob.r' 'add-rows-cols.r'
'add-space.r' 'grid.r' 'gtable-layouts.r' 'gtable-package.R'
'gtable.r' 'rbind-cbind.r' 'utils.r' 'trim.r' 'filter.r'
'align.r' 'padding.r' 'z.r'

URL https://gtable.r-lib.org, https://github.com/r-lib/gtable

BugReports https://github.com/r-lib/gtable/issues

VignetteBuilder knitr

NeedsCompilation no

Author Hadley Wickham [aut, cre],
Thomas Lin Pedersen [aut],
RStudio [cph]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2022-09-01 02:40:12 UTC

R topics documented:

`bind` *Row and column binding for gtables.*

Description

These functions are the parallels of the `matrix`/`data.frame` row and column bindings. As such they work in the same way, except they have to take care of additional attributes within the gtables. Most importantly it needs to take care of the sizing of the final gtable, as the different gtables going in may have different widths or heights. By default it tries to calculate the maximum width/height among the supplied gtables, but other options exists. Further, the relative layering of the grobs in each gtable can be modified or left as-is.

Usage

```
## S3 method for class 'gtable'
rbind(..., size = "max", z = NULL)

## S3 method for class 'gtable'
cbind(..., size = "max", z = NULL)
```

Arguments

...	gtables to combine (x and y)
size	How should the widths (for rbind) and the heights (for cbind) be combined across the gtables: take values from first, or last gtable, or compute the min or max values. Defaults to max.
z	A numeric vector indicating the relative z values of each gtable. The z values of each object in the resulting gtable will be modified to fit this order. If NULL, then the z values of obects within each gtable will not be modified.

Value

A gtable object

Examples

```
library(grid)
a <- rectGrob(gp = gpar(fill = "red"))
b <- circleGrob()
c <- linesGrob()

row <- matrix(list(a, b), nrow = 1)
col <- matrix(list(a, b), ncol = 1)
mat <- matrix(list(a, b, c, nullGrob()), nrow = 2)

row_gt <- gtable_matrix("demo", row, unit(c(1, 1), "null"), unit(1, "null"))
col_gt <- gtable_matrix("demo", col, unit(1, "null"), unit(c(1, 1), "null"))
mat_gt <- gtable_matrix("demo", mat, unit(c(1, 1), "null"), unit(c(1, 1), "null"))

# cbind
c_binded <- cbind(mat_gt, col_gt, size = "first")
plot(c_binded)

# rbind
r_binded <- rbind(mat_gt, row_gt, size = "last")
plot(r_binded)

# Dimensions must match along bind direction
try(cbind(mat_gt, row_gt))
```

---

gtable *Create a new grob table.*

Description

A grob table captures all the information needed to layout grobs in a table structure. It supports row and column spanning, offers some tools to automatically figure out the correct dimensions, and makes it easy to align and combine multiple tables.

Usage

```
gtable(
  widths = list(),
  heights = list(),
  respect = FALSE,
  name = "layout",
  rownames = NULL,
  colnames = NULL,
  vp = NULL
)
```

Arguments

widths	a unit vector giving the width of each column
heights	a unit vector giving the height of each row
respect	a logical vector of length 1: should the aspect ratio of height and width specified in null units be respected. See grid.layout() for more details
name	a string giving the name of the table. This is used to name the layout viewport
rownames, colnames	character vectors of row and column names, used for characteric subsetting, particularly for gtable_align, and gtable_join.
vp	a grid viewport object (or NULL).

Details

Each grob is put in its own viewport - grobs in the same location are not combined into one cell. Each grob takes up the entire cell viewport so justification control is not available.

It constructs both the viewports and the gTree needed to display the table.

Value

A gtable object

Components

There are three basics components to a grob table: the specification of table (cell heights and widths), the layout (for each grob, its position, name and other settings), and global parameters.

It's easier to understand how gtable works if in your head you keep the table separate from it's contents. Each cell can have 0, 1, or many grobs inside. Each grob must belong to at least one cell, but can span across many cells.

Layout

The layout details are stored in a data frame with one row for each grob, and columns:

- t top extent of grob
- r right extent of grob

- b bottom extent of
- l left extent of grob
- z the z-order of the grob - used to reorder the grobs before they are rendered
- clip a string, specifying how the grob should be clipped: either "on", "off" or "inherit"
- name, a character vector used to name each grob and its viewport

You should not need to modify this data frame directly - instead use functions like gtable_add_grob.

See Also

Other gtable construction: gtable_col(), gtable_matrix(), gtable_row(), gtable_spacer

Examples

```
library(grid)
a <- gtable(unit(1:3, c("cm")), unit(5, "cm"))
a
gtable_show_layout(a)

# Add a grob:
rect <- rectGrob(gp = gpar(fill = "black"))
a <- gtable_add_grob(a, rect, 1, 1)
a
plot(a)

# gtables behave like matrices:
dim(a)
t(a)
plot(t(a))

# when subsetting, grobs are retained if their extents lie in the
# rows/columns that retained.

b <- gtable(unit(c(2, 2, 2), "cm"), unit(c(2, 2, 2), "cm"))
b <- gtable_add_grob(b, rect, 2, 2)
b[1, ]
b[, 1]
b[2, 2]

# gtable have row and column names
rownames(b) <- 1:3
rownames(b)[2] <- 200
colnames(b) <- letters[1:3]
dimnames(b)
```

`gtable_add_cols` *Add new columns in specified position.*

Description

Insert new columns in a gtable and adjust the grob placement accordingly. If columns are added in the middle of a grob spanning multiple columns, the grob will continue to span them all. If a column is added to the left or right of a grob, the grob will not span the new column(s).

Usage

```
gtable_add_cols(x, widths, pos = -1)
```

Arguments

`x`	a `gtable()` object
`widths`	a unit vector giving the widths of the new columns
`pos`	new columns will be added to the right of this position. Defaults to adding col on right. `0` adds on the left.

Value

A gtable with the new columns added.

See Also

Other gtable manipulation: `gtable_add_grob()`, `gtable_add_padding()`, `gtable_add_rows()`, `gtable_add_space`, `gtable_filter()`

Examples

```
library(grid)
rect <- rectGrob(gp = gpar(fill = "#00000080"))
tab <- gtable(unit(rep(1, 3), "null"), unit(rep(1, 3), "null"))
tab <- gtable_add_grob(tab, rect, t = 1, l = 1, r = 3)
tab <- gtable_add_grob(tab, rect, t = 1, b = 3, l = 1)
tab <- gtable_add_grob(tab, rect, t = 1, b = 3, l = 3)
dim(tab)
plot(tab)

# Grobs will continue to span over new rows if added in the middle
tab2 <- gtable_add_cols(tab, unit(1, "null"), 1)
dim(tab2)
plot(tab2)

# But not when added to left (0) or right (-1, the default)
tab3 <- gtable_add_cols(tab, unit(1, "null"))
tab3 <- gtable_add_cols(tab3, unit(1, "null"), 0)
```

```
dim(tab3)
plot(tab3)
```

`gtable_add_grob` *Add a single grob, possibly spanning multiple rows or columns.*

Description

This only adds grobs into the table - it doesn't affect the table layout in any way. In the gtable model, grobs always fill up the complete table cell. If you want custom justification you might need to define the grob dimension in absolute units, or put it into another gtable that can then be added to the gtable instead of the grob.

Usage

```
gtable_add_grob(
  x,
  grobs,
  t,
  l,
  b = t,
  r = l,
  z = Inf,
  clip = "on",
  name = x$name
)
```

Arguments

`x`	a `gtable()` object
`grobs`	a single grob or a list of grobs
`t`	a numeric vector giving the top extent of the grobs
`l`	a numeric vector giving the left extent of the grobs
`b`	a numeric vector giving the bottom extent of the grobs
`r`	a numeric vector giving the right extent of the grobs
`z`	a numeric vector giving the order in which the grobs should be plotted. Use `Inf` (the default) to plot above or `-Inf` below all existing grobs. By default positions are on the integers, giving plenty of room to insert new grobs between existing grobs.
`clip`	should drawing be clipped to the specified cells (`"on"`), the entire table (`"inherit"`), or not at all (`"off"`)
`name`	name of the grob - used to modify the grob name before it's plotted.

Value

A gtable object with the new grob(s) added

See Also

Other gtable manipulation: gtable_add_cols(), gtable_add_padding(), gtable_add_rows(), gtable_add_space, gtable_filter()

Examples

```
library(grid)

gt <- gtable(widths = unit(c(1, 1), 'null'), heights = unit(c(1, 1), 'null'))
pts <- pointsGrob(x = runif(5), y = runif(5))

# Add a grob to a single cell (top-right cell)
gt <- gtable_add_grob(gt, pts, t = 1, l = 2)

# Add a grob spanning multiple cells
gt <- gtable_add_grob(gt, pts, t = 1, l = 1, b = 2)

plot(gt)
```

---

gtable_add_padding *Add padding around edges of table.*

Description

This is a convenience function for adding an extra row and an extra column at each edge of the table.

Usage

```
gtable_add_padding(x, padding)
```

Arguments

x	a gtable() object
padding	vector of length 4: top, right, bottom, left. Normal recycling rules apply.

Value

A gtable object

See Also

Other gtable manipulation: gtable_add_cols(), gtable_add_grob(), gtable_add_rows(), gtable_add_space, gtable_filter()

Examples

```
library(grid)
gt <- gtable(unit(1, "null"), unit(1, "null"))
gt <- gtable_add_grob(gt, rectGrob(gp = gpar(fill = "black")), 1, 1)

plot(gt)
plot(cbind(gt, gt))
plot(rbind(gt, gt))

pad <- gtable_add_padding(gt, unit(1, "cm"))
plot(pad)
plot(cbind(pad, pad))
plot(rbind(pad, pad))
```

---

gtable_add_rows *Add new rows in specified position.*

Description

Insert new rows in a gtable and adjust the grob placement accordingly. If rows are added in the middle of a grob spanning multiple rows, the grob will continue to span them all. If a row is added above or below a grob, the grob will not span the new row(s).

Usage

```
gtable_add_rows(x, heights, pos = -1)
```

Arguments

x	a gtable() object
heights	a unit vector giving the heights of the new rows
pos	new row will be added below this position. Defaults to adding row on bottom. 0 adds on the top.

Value

A gtable with the new rows added.

See Also

Other gtable manipulation: gtable_add_cols(), gtable_add_grob(), gtable_add_padding(), gtable_add_space, gtable_filter()

Examples

```
library(grid)
rect <- rectGrob(gp = gpar(fill = "#00000080"))
tab <- gtable(unit(rep(1, 3), "null"), unit(rep(1, 3), "null"))
tab <- gtable_add_grob(tab, rect, t = 1, l = 1, r = 3)
tab <- gtable_add_grob(tab, rect, t = 1, b = 3, l = 1)
tab <- gtable_add_grob(tab, rect, t = 1, b = 3, l = 3)
dim(tab)
plot(tab)

# Grobs will continue to span over new rows if added in the middle
tab2 <- gtable_add_rows(tab, unit(1, "null"), 1)
dim(tab2)
plot(tab2)

# But not when added to top (0) or bottom (-1, the default)
tab3 <- gtable_add_rows(tab, unit(1, "null"))
tab3 <- gtable_add_rows(tab3, unit(1, "null"), 0)
dim(tab3)
plot(tab3)
```

---

gtable_add_space *Add row/column spacing.*

Description

Adds `width` space between the columns or `height` space between the rows, effictvely pushing the existing cells apart.

Usage

```
gtable_add_col_space(x, width)

gtable_add_row_space(x, height)
```

Arguments

`x`	a gtable object
`width`	a vector of units of length 1 or ncol - 1
`height`	a vector of units of length 1 or nrow - 1

Value

A gtable with the additional rows or columns added

See Also

Other gtable manipulation: gtable_add_cols(), gtable_add_grob(), gtable_add_padding(), gtable_add_rows(), gtable_filter()

Examples

```
library(grid)

rect <- rectGrob()
rect_mat <- matrix(rep(list(rect), 9), nrow = 3)

gt <- gtable_matrix("rects", rect_mat, widths = unit(rep(1, 3), "null"),
                    heights = unit(rep(1, 3), "null"))

plot(gt)

# Add spacing between the grobs
# same height between all rows
gt <- gtable_add_row_space(gt, unit(0.5, "cm"))

# Different width between the columns
gt <- gtable_add_col_space(gt, unit(c(0.5, 1), "cm"))

plot(gt)
```

---

gtable_col *Create a single column gtable*

Description

This function stacks a list of grobs into a single column gtable of the given width and heights.

Usage

```
gtable_col(name, grobs, width = NULL, heights = NULL, z = NULL, vp = NULL)
```

Arguments

name	a string giving the name of the table. This is used to name the layout viewport
grobs	a single grob or a list of grobs
width	a unit vector giving the width of this column
heights	a unit vector giving the height of each row
z	a numeric vector giving the order in which the grobs should be plotted. Use Inf (the default) to plot above or -Inf below all existing grobs. By default positions are on the integers, giving plenty of room to insert new grobs between existing grobs.
vp	a grid viewport object (or NULL).

Value

A gtable with one column and as many rows as elements in the grobs list.

See Also

Other gtable construction: `gtable_matrix()`, `gtable_row()`, `gtable_spacer`, `gtable()`

Examples

```
library(grid)
a <- rectGrob(gp = gpar(fill = "red"))
b <- circleGrob()
c <- linesGrob()
gt <- gtable_col("demo", list(a, b, c))
gt
plot(gt)
gtable_show_layout(gt)
```

---

gtable_filter *Filter cells by name*

Description

Normally a gtable is considered a matrix when indexing so that indexing is working on the cell layout and not on the grobs it contains. `gtable_filter` allows you to subset the grobs by name and optionally remove rows or columns if left empty after the subsetting

Usage

```
gtable_filter(x, pattern, fixed = FALSE, trim = TRUE, invert = FALSE)
```

Arguments

`x`	a gtable object
`pattern`	character string containing a regular expression (or character string for `fixed = TRUE`) to be matched in the given character vector. Coerced by `as.character` to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. Missing values are allowed except for `regexpr`, `gregexpr` and `regexec`.
`fixed`	logical. If `TRUE`, `pattern` is a string to be matched as is. Overrides all conflicting arguments.
`trim`	if `TRUE`, `gtable_trim()` will be used to trim off any empty cells.
`invert`	Should the filtering be inverted so that cells matching `pattern` is removed instead of kept.

Value

A gtable only containing the matching grobs, potentially stripped of empty columns and rows

See Also

Other gtable manipulation: gtable_add_cols(), gtable_add_grob(), gtable_add_padding(), gtable_add_rows(), gtable_add_space

Examples

```
library(grid)
gt <- gtable(unit(rep(5, 3), c("cm")), unit(5, "cm"))
rect <- rectGrob(gp = gpar(fill = "black"))
circ <- circleGrob(gp = gpar(fill = "red"))

gt <- gtable_add_grob(gt, rect, 1, 1, name = "rect")
gt <- gtable_add_grob(gt, circ, 1, 3, name = "circ")

plot(gtable_filter(gt, "rect"))
plot(gtable_filter(gt, "rect", trim = FALSE))
plot(gtable_filter(gt, "circ"))
plot(gtable_filter(gt, "circ", trim = FALSE))
```

---

gtable_height *Returns the height of a gtable, in the gtable's units*

Description

Note that unlike heightDetails.gtable, this can return relative units.

Usage

```
gtable_height(x)
```

Arguments

x	A gtable object

`gtable_matrix` *Create a gtable from a matrix of grobs.*

Description

This function takes a matrix of grobs and create a gtable matching with the grobs in the same position as they were in the matrix, with the given heights and widths.

Usage

```
gtable_matrix(
  name,
  grobs,
  widths = NULL,
  heights = NULL,
  z = NULL,
  respect = FALSE,
  clip = "on",
  vp = NULL
)
```

Arguments

`name`	a string giving the name of the table. This is used to name the layout viewport
`grobs`	a single grob or a list of grobs
`widths`	a unit vector giving the width of each column
`heights`	a unit vector giving the height of each row
`z`	a numeric matrix of the same dimensions as `grobs`, specifying the order that the grobs are drawn.
`respect`	a logical vector of length 1: should the aspect ratio of height and width specified in null units be respected. See `grid.layout()` for more details
`clip`	should drawing be clipped to the specified cells (`"on"`), the entire table (`"inherit"`), or not at all (`"off"`)
`vp`	a grid viewport object (or NULL).

Value

A gtable of the same dimensions as the grobs matrix.

See Also

Other gtable construction: `gtable_col()`, `gtable_row()`, `gtable_spacer`, `gtable()`

Examples

```
library(grid)
a <- rectGrob(gp = gpar(fill = "red"))
b <- circleGrob()
c <- linesGrob()

row <- matrix(list(a, b, c), nrow = 1)
col <- matrix(list(a, b, c), ncol = 1)
mat <- matrix(list(a, b, c, nullGrob()), nrow = 2)

gtable_matrix("demo", row, unit(c(1, 1, 1), "null"), unit(1, "null"))
gtable_matrix("demo", col, unit(1, "null"), unit(c(1, 1, 1), "null"))
gtable_matrix("demo", mat, unit(c(1, 1), "null"), unit(c(1, 1), "null"))

# Can specify z ordering
z <- matrix(c(3, 1, 2, 4), nrow = 2)
gtable_matrix("demo", mat, unit(c(1, 1), "null"), unit(c(1, 1), "null"), z = z)
```

---

gtable_row *Create a single row gtable.*

Description

This function puts grobs in a list side-by-side in a single-row gtable from left to right witrh the given widths and height.

Usage

```
gtable_row(name, grobs, height = NULL, widths = NULL, z = NULL, vp = NULL)
```

Arguments

name	a string giving the name of the table. This is used to name the layout viewport
grobs	a single grob or a list of grobs
height	a unit vector giving the height of this row
widths	a unit vector giving the width of each column
z	a numeric vector giving the order in which the grobs should be plotted. Use Inf (the default) to plot above or -Inf below all existing grobs. By default positions are on the integers, giving plenty of room to insert new grobs between existing grobs.
vp	a grid viewport object (or NULL).

Value

A gtable with a single row and the same number of columns as elements in the grobs list

See Also

Other gtable construction: gtable_col(), gtable_matrix(), gtable_spacer, gtable()

Examples

```
library(grid)
a <- rectGrob(gp = gpar(fill = "red"))
b <- circleGrob()
c <- linesGrob()
gt <- gtable_row("demo", list(a, b, c))
gt
plot(gt)
gtable_show_layout(gt)
```

---

gtable_show_layout *Visualise the layout of a gtable.*

Description

This function is a simple wrapper around grid::grid.show.layout() that allows you to inspect the layout of the gtable.

Usage

```
gtable_show_layout(x, ...)
```

Arguments

x	a gtable object
...	Arguments passed on to grid::grid.show.layout l A Grid layout object. newpage A logical value indicating whether to move on to a new page before drawing the diagram. vp.ex positive number, typically in $(0, 1]$, specifying the scaling of the layout. bg The colour used for the background. cell.border The colour used to draw the borders of the cells in the layout. cell.fill The colour used to fill the cells in the layout. cell.label A logical indicating whether the layout cells should be labelled. label.col The colour used for layout cell labels. unit.col The colour used for labelling the widths/heights of columns/rows. vp A Grid viewport object (or NULL).

Examples

```
gt <- gtable(widths = grid::unit(c(1, 0.5, 2), c("null", "cm", "null")),
             heights = grid::unit(c(0.2, 1, 3), c("inch", "null", "cm")))
gtable_show_layout(gt)
```

gtable_spacer *Create a row/col spacer gtable.*

Description

Create a zero-column or zero-row gtable with the given heights or widths respectively.

Usage

```
gtable_row_spacer(widths)

gtable_col_spacer(heights)
```

Arguments

widths	unit vector of widths
heights	unit vector of heights

Value

A gtable object

See Also

Other gtable construction: gtable_col(), gtable_matrix(), gtable_row(), gtable()

gtable_trim *Trim off empty cells.*

Description

This function detects rows and columns that does not contain any grobs and removes thewm from the gtable. If the rows and/or columns removed had a non-zero height/width the relative layout of the gtable may change.

Usage

```
gtable_trim(x)
```

Arguments

x	a gtable object

Value

A gtable object

Examples

```
library(grid)
rect <- rectGrob(gp = gpar(fill = "black"))
base <- gtable(unit(c(2, 2, 2), "cm"), unit(c(2, 2, 2), "cm"))

center <- gtable_add_grob(base, rect, 2, 2)
plot(center)
plot(gtable_trim(center))

col <- gtable_add_grob(base, rect, 1, 2, 3, 2)
plot(col)
plot(gtable_trim(col))

row <- gtable_add_grob(base, rect, 2, 1, 2, 3)
plot(row)
plot(gtable_trim(row))
```

---

gtable_width *Returns the width of a gtable, in the gtable's units*

Description

Note that unlike widthDetails.gtable, this can return relative units.

Usage

```
gtable_width(x)
```

Arguments

x	A gtable object

---

is.gtable *Is this a gtable?*

Description

Is this a gtable?

Usage

```
is.gtable(x)
```

Arguments

x	object to test

print.gtable *Print a gtable object*

Description

Print a gtable object

Usage

```
## S3 method for class 'gtable'
print(x, zsort = FALSE, ...)
```

Arguments

x	A gtable object.
zsort	Sort by z values? Default FALSE.
...	Other arguments (not used by this method).

Index