Package 'gtable'

July 2, 2014

Type Package

Title Arrange grobs in tables.

Version 0.1.2

Author Hadley Wickham <h.wickham@gmail.com>

Maintainer Hadley Wickham <h.wickham@gmail.com>

Description Tools to make it easier to work with ``tables'' of grobs.

Depends R (>= 2.14), grid

Suggests testthat, plyr

License GPL-2

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

Repository CRAN

Date/Publication 2012-12-05 13:11:37

NeedsCompilation no

R topics documented:

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

Description

Row and column binding for gtables.

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.

gtable *gtable*

Description

gtable

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.

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 accross 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

gtable_row, gtable_col and gtable_matrix for convenient ways of creating gtables.

Examples

```
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

Add new columns in specified position.

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 row will be added below this position. Defaults to adding col on right. 0 adds on the left.

Examples

```
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 in any way. In the gtable model, grobs always fill up the complete table cell. If you want custom justification you might need to

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.

---

gtable_add_padding *Add padding around edges of table.*

Description

Add padding around edges of 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.

Examples

```
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

Add new rows in specified position.

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.

Examples

```
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.

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

gtable_col *Create a single column gtable.*

Description

Create a single column gtable.

Usage

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

Arguments

`width`	a unit vector giving the width of this column
`vp`	a grid viewport object (or NULL).
`name`	a string giving the name of the table. This is used to name the layout viewport
`heights`	a unit vector giving the height of each row
`grobs`	a single grob or a list of 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.

Examples

```
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

Filter cells by name.

Usage

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

Arguments

`x`	a gtable object
`trim`	if `TRUE`, `gtable_trim` will be used to trim off any empty cells.
`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` and `gregexpr`.
`fixed`	logical. If `TRUE`, `pattern` is a string to be matched as is. Overrides all conflicting arguments.

Examples

```
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

Create a gtable from a matrix of grobs.

Usage

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

Arguments

z	a numeric matrix of the same dimensions as grobs, specifying the order that the grobs are drawn.
vp	a grid viewport object (or NULL).
name	a string giving the name of the table. This is used to name the layout viewport
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
grobs	a single grob or a list of grobs
clip	should drawing be clipped to the specified cells ("on"), the entire table ("inherit"), or not at all ("off")

Examples

```
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

Create a single row gtable.

Usage

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

Arguments

height	a unit vector giving the height of this row
vp	a grid viewport object (or NULL).
name	a string giving the name of the table. This is used to name the layout viewport
widths	a unit vector giving the width of each column
grobs	a single grob or a list of 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.

Examples

```
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

Visualise the layout of a gtable.

Usage

```
gtable_show_layout(x)
```

Arguments

`x`	a gtable object

`gtable_spacer` *Create a row/col spacer gtable.*

Description

Create a row/col spacer gtable.

Usage

```
gtable_row_spacer(widths)

gtable_col_spacer(heights)
```

Arguments

`widths`	unit vector of widths
`heights`	unit vector of heights

gtable_trim *Trim off empty cells.*

Description

Trim off empty cells.

Usage

```
gtable_trim(x)
```

Arguments

x	a gtable object

Examples

```
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).

z_arrange_gtables *Arrange the z values within gtable objects*

Description

This is usually used before rbinding or cbinding the gtables together. The resulting z values will be normalized.

Usage

```
z_arrange_gtables(gtables, z)
```

Arguments

gtables	A list of gtable objects
z	A numeric vector of relative z values

Details

Ties are handled by the "first" method: the first occurrence of a value wins.

z_normalise *Normalise z values within a gtable object*

Description

The z values within a gtable object can be any numeric values. This function will change them to integers (starting from 1), preserving the original order.

Usage

```
z_normalise(x, i = 1)
```

Arguments

x	A gtable object
i	The z value to start counting up from (default is 1)

Details

Ties are handled by the "first" method: the first occurrence of a value wins.

Index