# Package ‘lemon’

January 8, 2019

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Freshing Up your 'ggplot2' Plots</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://github.com/stefan-edwards/lemon">https://github.com/stefan-edwards/lemon</a></td>
</tr>
<tr>
<td>BugReports</td>
<td><a href="https://github.com/stefan-edwards/lemon/issues">https://github.com/stefan-edwards/lemon/issues</a></td>
</tr>
<tr>
<td>Version</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Date</td>
<td>2019-01-08</td>
</tr>
<tr>
<td>Description</td>
<td>Functions for working with legends and axis lines of 'ggplot2', facets that repeat axis lines on all panels, and some 'knitr' extensions.</td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.1.0)</td>
</tr>
<tr>
<td>Imports</td>
<td>ggplot2 (&gt;= 2.2.0), plyr, grid, gridExtra, gtable, knitr (&gt;= 1.12), lattice, scales</td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>LazyData</td>
<td>true</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>6.1.1</td>
</tr>
</tbody>
</table>
| Collate | 'ggplot2.r' 'lemon-plot.r' 'axis-annotation.r' 'brackets.R'
  | 'coord-flex.r' 'coord-capped.r' 'dot.r' 'facet-rep-lab.r'
  | 'facet-wrap.r' 'geom-pointline.r' 'lemon_print.r'
  | 'geom-siderange.r' 'grob_utils.r' 'gtable_show-.r'
  | 'guides-axis.r' 'legends.r' 'lemon.r' 'scale-symmetric.r' |
| Suggests | rmarkdown, stringr, dplyr, testthat |
| VignetteBuilder | knitr |
| NeedsCompilation | no |
| Author | Stefan McKinnon Edwards [aut, ctb, cre]
  | (<https://orcid.org/0000-0002-4628-8148>),
  | Baptiste Auguié [ctb] (For g_legend and grid_arrange_shared_legend),
  | Shaun Jackman [ctb] (For grid_arrange_shared_legend),
  | Hadley Wickham [ctb] (ggplot2 functions),
  | Winston Chang [ctb] (ggplot2 functions) |
| Maintainer | Stefan McKinnon Edwards <sme@iysik.com> |

1
Description

The .dot functions creates functions that allows relative-like specification of paths, but are safe from changing working directory.

Usage

```r
.dot(x, root = getwd(), mustExist = FALSE, relative = FALSE, create = TRUE)

.dot2(names, quiet = FALSE, ...)
```
annotate_y_axis

Arguments

x File path that is appended to BASEDIR.
root Root of your working directory, from which x is relative too.
mustExist Logical value; if TRUE and the resulting path does not exist, it raises an error.
relative For .dot, sets default for the returned function. For the returned function, when TRUE, the function returns a path relative to root.
create Logical value, creates the target directory when TRUE (default).
names Character vector of names
quiet Logical value, suppresses output to stdout() when TRUE.
... Arguments passed on to .dot.

Value

A function that returns file paths constructed from root, x, and ....

Side effect: It creates the directory.

Examples

.data <- .dot('data')
data('input.txt')
data(c('a.txt','b.txt'))

.dot2(c('rawdata','results'))
.rawdata('rawfile.csv')
.results('myresults.txt')

__________________________

annotate_y_axis        Annotations in the axis
__________________________

Description

Annotations in the axis

Usage

annotate_y_axis(label, y, side = waiver(), print_label = TRUE,
                 print_value = TRUE, print_both = TRUE, parsed = FALSE, ...)

annotate_x_axis(label, x, side = waiver(), print_label = TRUE,
                 print_value = TRUE, print_both = TRUE, parsed = FALSE, ...)
Arguments

\begin{itemize}
  \item \textbf{label} \hspace{2em} Text to print
  \item \textbf{y, x} \hspace{2em} Position of the annotation.
  \item \textbf{side} \hspace{2em} left or right, or top or bottom side to print annotation
  \item \textbf{print_label, print_value, print_both} \hspace{2em} Logical; what to show on annotation. Label and/or value. print_both is shortcut for setting both print_label and print_value. When both is TRUE, uses argument sep to separate the label and value.
  \item \textbf{parsed} \hspace{2em} Logical (default FALSE), when TRUE, uses mathplot for outputting expressions. See section "Showing values".
  \item \textbf{...} \hspace{2em} Style settings for label and tick: colour, hjust, vjust, size, fontface, family, rot. When waiver() (default), the relevant theme element is used.
\end{itemize}

Showing values: See \texttt{plotmath} for using mathematical expressions. The function uses a simple replacement strategy where the literal strings \texttt{.y} and \texttt{.val} are replaced by the value after round of to a number of digits, as given by argument digits.

Examples

\begin{verbatim}
library(ggplot2)

p <- ggplot(mtcars, aes(mpg, hp, colour=disp)) + geom_point()

l <- p + annotate_y_axis('mark at', y=200, tick=TRUE)

(l + annotate_x_axis('| good economy ->', x=25, 
  print_value=FALSE, hjust=0, tick=TRUE))

l + annotate_y_axis("x\^2 == .(y)\", y=150, 
  parsed=FALSE, tick=FALSE) + 
  annotate_y_axis("x\^2 + bar(x) == .(y)\", y=mean(mtcars$hp), 
  parsed=TRUE, tick=TRUE)

l + annotate_y_axis("bar(x) == .(y)\", y = mean(mtcars$hp), 
  parsed=TRUE, tick=FALSE) # use double equal signs, or the output becomes '=(...)′ for some reason.

l + annotate_y_axis("this is midway", 
  y=sum(range(mtcars$hp))/2, print_value=FALSE, side='left')

# work around if an axis only contains parsed expressions
p + annotate_y_axis("bar(x) == .(y)\", y = mean(mtcars$hp), parsed=TRUE, tick=FALSE) + 
  annotate_y_axis("some long string", y=100, 
  print_value=FALSE, print_value=FALSE, colour=NA)

# Works together with other functions
p <- p + theme_light() + theme(panel.border=element_blank(), 
  axis.line = element_line(),
  axis.ticks = element_line(colour='black'))

p + coord_capped_cart(bottom='right') + 
  annotate_y_axis('More than I\’can afford', y=125, 
  print_value=FALSE, tick=TRUE)
\end{verbatim}
**brackets_horizontal**  
*Axis brackets instead of axis ticks and lines*

**Description**

To be used with `coord_flex_cart, coord_capped_cart`, etc. for displaying brackets instead of the axis ticks and lines.

**Usage**

```r
brackets_horizontal(direction = c("up", "down"), length = unit(0.05, "npc"), tick.length = waiver())
```

```r
brackets_vertical(direction = c("left", "right"), length = unit(0.05, "npc"), tick.length = waiver())
```

**Arguments**

- **direction**  Which way should the opening side of the brackets point? up, down, left, or right?
- **length**  Length of the unit, parallel with axis line.
- **tick.length**  Height (width) of x-axis (y-axis) bracket. If `waiver()` (default), use `axis.ticks.length` from `theme`.

**Details**

The looks of the brackets are taken from `theme(axis.ticks)`, or `theme(axis.ticks.x)` and `theme(axis.ticks.y)`, respectively.

It does not re-calculate tick marks, but lets `scale_x_*` and `scale_y_*` calculate and draw ticks and labels, and then modifies the ticks with brackets.

Both `length` and `tick.length` accept a numeric scalar instead of a `unit` object that is interpreted as an "npc" unit.

**See Also**

- `unit`

**Examples**

```r
library(ggplot2)
p <- ggplot(mpg, aes(as.factor(cyl), hwy, colour=class)) +
  geom_point(position=position_jitter(width=0.3)) +
  theme_bw() +
  theme(panel.border = element_blank(), axis.line = element_line())
p
```

```r
classical:

p <- p + coord_flex_cart(bottom=brackets_horizontal(length=unit(0.08, 'npc')))  
```
# However getting the correct width is a matter of tweaking either length or
# position_jitter...

# A further adjustment,
p + theme(panel.grid.major.x = element_blank())

---

**coord_capped_cart**

*Cartesian coordinates with capped axis lines.*

**Description**

Caps the axis lines to the outer ticks to e.g. indicate range of values. Methods correspond to `coord_cartesian` and `coord_flip`

**Usage**

coord_capped_cart(xlim = NULL, ylim = NULL, expand = TRUE,
                   top = waiver(), left = waiver(), bottom = waiver(),
                   right = waiver(), gap = 0.01)

coord_capped_flip(xlim = NULL, ylim = NULL, expand = TRUE,
                   top = waiver(), left = waiver(), bottom = waiver(),
                   right = waiver(), gap = 0.01)

capped_horizontal(capped = c("both", "left", "right", "none"),
                   gap = 0.01)

capped_vertical(capped = c("top", "bottom", "both", "none"),
                gap = 0.01)

**Arguments**

- **xlim, ylim** Limits for the x and y axes.
- **expand** If TRUE, the default, adds a small expansion factor to the limits to ensure that data and axes don’t overlap. If FALSE, limits are taken exactly from the data or xlim/ylim.
- **top, left, bottom, right** Either a function returned from `capped_horizontal` or `brackets_horizontal`. If string, it is assumed to be shorthand for capped_horizontal(capped) or similar for vertical.
- **gap** Both ends are always capped by this proportion. Usually a value between 0 and 1.
- **capped** Which end to cap the line. Can be one of (where relevant): both, none, left, right, top, bottom.
Details

This function is a simple override of \texttt{coord\_flex\_cart} and \texttt{coord\_flex\_flip}, which allows short-hand specification of what to cap.

NB! A panel-border is typically drawn on top such that it covers tick marks, grid lines, and axis lines. Many themes also do not draw axis lines. To ensure the modified axis lines are visible, use \texttt{theme(panel.border=element\_blank(), axis.lines=element\_line())}.

Examples

\begin{verbatim}
library(ggplot2)
# Notice how the axis lines of the following plot meet in the lower-left corner.
p <- ggplot(mtcars, aes(x = mpg)) + geom_dotplot() +
    theme_bw() +
    theme(panel.border=element_blank(), axis.line=element_line())

# We can introduce a gap by capping the ends:
p + coord_capped_cart(bottom='none', left='none')

# The lower limit on the y-axis is 0. We can cap the line to this value.
# Notice how the x-axis line extends through the plot when we no long
# define its capping.
p + coord_capped_cart(left='both')

# It it also works on the flipped.
p + coord_capped_flip(bottom='both')

# And on secondary axis, in conjunction with brackets:
p +
    scale_y_continuous(sec.axis = sec_axis(~*100)) +
    scale_x_continuous(sec.axis = sec_axis(-1., name='Madness scale')) +
    coord_capped_cart(bottom='none', left='none', right='both', top=brackets_horizontal())
# Although we cannot recommend the above madness.
\end{verbatim}
Arguments

xlim, ylim  Limits for the x and y axes.
expand  If TRUE, the default, adds a small expansion factor to the limits to ensure that data and axes don’t overlap. If FALSE, limits are taken exactly from the data or xlim/ylim.
top, left, bottom, right  Function for drawing axis lines, ticks, and labels, use e.g. capped_horizontal or brackets_horizontal.
ratio  aspect ratio, expressed as y / x.

Details

NB! A panel-border is typically drawn on top such that it covers tick marks, grid lines, and axis lines. Many themes also do not draw axis lines. To ensure the modified axis lines are visible, use theme(panel.border=element_blank(), axis.line=element_line()).

User defined functions

The provided function in top, right, bottom, and left defaults to render_axis which is defined in 'ggplot2/R/coord.M-rr', which in turns calls guide_axis (see 'ggplot2/R/guides-axis.rr').

The provided function is with the arguments scale_details, axis, scale, position, and theme, and the function should return an absoluteGrob object.

For examples of modifying the drawn object, see e.g. capped_horizontal or brackets_horizontal.

Examples

library(ggplot2)
# A standard plot
p <- ggplot(mtcars, aes(disp, wt)) +
  geom_point() +
  geom_smooth() + theme(panel.border=element_blank(), axis.line=element_line())

# We desire that left axis does not extend beyond '6'
# and the x-axis is unaffected
p + coord_capped_cart(left=’top’)

# Specifying 'bottom' caps the axis with at most the length of 'gap'
p + coord_capped_cart(left=’top’, bottom=’none’)  # We can specify a ridiculus large 'gap', but the lines will always # protrude to the outer most ticks.
p + coord_capped_cart(left='top', bottom='none', gap=2)

# We can use 'capped_horizontal' and 'capped_vertical' to specify for
# each axis individually.
p + coord_capped_cart(left='top', bottom=capped_horizontal('none', gap=2))

# At this point we might as well drop using the short-hand and go full on:
p + coord_flex_cart(left=brackets_vertical(), bottom=capped_horizontal('left'))

# Also works with secondary axes:
p + scale_y_continuous(sec.axis=sec_axis(~5*.name='wt times 5')) +
  coord_flex_cart(left=brackets_vertical(), bottom=capped_horizontal('right'),
  right=capped_vertical('both', gap=0.02))

# Supports the usual 'coord_fixed':
p + coord_flex_fixed(ratio=1.2, bottom=capped_horizontal('right'))

# and coord_flip:
p + coord_flex_flip(ylim=c(2,5), bottom=capped_horizontal('right'))

---

**element_render**

Render a ggplot2 grob or retrieve its gpar object.

**Description**

Helps add the ggplot2-theme's look-and-feel to grid's grob objects. render_gpar returns a gpar-object, element_render returns a grid.grob-object.

**Usage**

```r
element_render(theme, element, ..., name = NULL)
render_gpar(theme, element, ...)
```

**Arguments**

- **theme**: A ggplot2 theme
- **element**: The name of an element in the theme, e.g. "axis.text".
- **...**: Additional arguments sent to grobs (e.g. x or y).
- **name**: Returned grob's name.

**Value**

A grid.grob or gpar object.

**Author(s)**

element_render is from ggplot2 source.
See Also
theme

facet_rep_grid

Description
facet_grid and facet_wrap, but with axis lines and labels preserved on all panels.

Usage

facet_rep_grid(..., repeat.tick.labels = FALSE)
facet_rep_wrap(..., scales = "fixed", repeat.tick.labels = FALSE)

Arguments
...
Arguments used for facet_grid or facet_wrap.
repeat.tick.labels
When FALSE (default), axes on inner panels have their tick labels (i.e. the numbers) removed. Set this to TRUE to keep all labels, or any combination of top, bottom, left, right to keep only those specified. Also accepts 'x' and 'y'.
scales
As for facet_grid, but alters behaviour of repeat.tick.labels.

Details
These two functions are extensions to facet_grid and facet_wrap that keeps axis lines, ticks, and optionally tick labels across all panels.
Examples are given in the vignette "Repeat axis lines on facet panels" vignette.

geom_pointpath

Description

gem_pointpath combines geom_point and geom_path, such that a) when jittering is used, both lines and points stay connected, and b) provides a visual effect by adding a small gap between the point and the end of line. geom_pointline combines geom_point and geom_path.
Usage

geom_pointpath(mapping = NULL, data = NULL, stat = "identity",
position = "identity", na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE, distance = unit(3, "pt"), shorten = 0.5,
threshold = 0.1, lineend = "butt", linejoin = "round",
linemitre = 1, linesize = 0.5, linecolour = waiver(),
linemulti = 1, linecolor = waiver(), arrow = NULL, ...)

geom_pointline(mapping = NULL, data = NULL, stat = "identity",
position = "identity", na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE, distance = unit(3, "pt"), shorten = 0.5,
threshold = 0.1, lineend = "butt", linejoin = "round",
linemitre = 1, linesize = 0.5, linecolour = waiver(),
linemulti = 1, linecolor = waiver(), arrow = NULL, ...)

geom_pointrangeline(mapping = NULL, data = NULL, stat = "identity",
position = "identity", na.rm = FALSE, show.legend = NA,
inherit.aes = TRUE, distance = unit(3, "pt"), lineend = "butt",
linejoin = "round", linemitre = 1, linesize = 0.5,
linecolour = waiver(), linemulti = 1, linecolor = waiver(), arrow = NULL, ...)

Arguments

mapping Set of aesthetic mappings created by aes or aes_.
data The data to be displayed in this layer.
stat The statistical transformation to use on the data for this layer, as a string.
position Position adjustment, either as a string, or the result of a call to a position adjust-
ment function (e.g. position_jitter). Both lines and points gets the same ad-
justment (this is where the function excels over geom_point() + geom_line()).
na.rm If FALSE (default), missing values are removed with a warning. If TRUE, missing
values are silently removed.
show.legend Logical. Should this layer be included in the legends? NA (default), includes if
any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetic, rather than combining with them. This
is most useful for helper functions that define both data and aesthetics and
shouldn’t inherit behaviour from the default plot specification.
distance Gap size between point and end of lines; use unit. Is converted to ’pt’ if given as
simple numeric. When NULL or NA, gapping and shorten/threshold is disabled.
To keep the latter, set to 0.
shorten, threshold When points are closer than threshold, shorten the line by the proportion in
shorten instead of adding a gap by distance.
lineend Line end style (round, butt, square).
linejoin Line join style (round, mintre, bevel).
linemitre Line mitre limit (number greater than 1).
linesize: Width of line.
linecolour, linecolor: When not waiver(), the line is drawn with this colour instead of that set by aesthetic colour.
arrow: Arrow specification, as created by arrow.
... other arguments passed on to layer.

Details

geom_pointpath connects the observations in the same order in which they appear in the data. geom_pointline connects them in order of the variable on the x-axis.

Both geom_pointpath and geom_pointline will only connect observations within the same group! However, if linecolour is not waiver(), connections will be made between groups, but possible in an incorrect order.

Aesthetics

geom_pointline and geom_pointpath understands the following aesthetics (required aesthetics are in bold):

• x
• y
• alpha
• colour – sets colour of point. Only affects line if linecolour=waiver().
• stroke
• shape
• stroke
• group
• linetype
• size – only affects point size. Width of line is set with linesize and cannot be linked to an aesthetic.

Examples

# geom_point examples
library(ggplot2)

p <- ggplot(mtcars, aes(wt, mpg))
p + geom_point() + geom_line()
p + geom_pointline()

p + geom_pointline(linecolour='brown')
p + geom_pointpath()

# Add aesthetic mappings
p + geom_pointline(aes(colour = factor(cyl)))
geom_siderange

Display range of data in side of plot

Description
Projects data onto horizontal or vertical edge of panels.

Usage

```r
geom_siderange(mapping = NULL, data = NULL, stat = "identity", position = "identity", ..., distance = 3, arrow = NULL, lineend = "butt", sides = "bl", start = NA, end = NA, na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

Arguments

- `mapping` Set of aesthetic mappings created by `aes` or `aes_`.
- `data` The data to be displayed in this layer.
- `stat` The statistical transformation to use on the data for this layer, as a string.
position  Position adjustment, either as a string, or the result of a call to a position adjustment function (e.g. `position_jitter`). Both lines and points gets the same adjustment (this is where the function excels over `geom_point()` + `geom_line()`).

... other arguments passed on to `layer`.

distance  Distance between edge of panel and lines, and distance between lines, in multiples of line widths, see description.

arrow  Arrow specification, as created by `arrow`.

lineend  Line end style (round, butt, square).

sides  Character including `top`, `right`, `bottom`, and/or `left`, indicating which side to project data onto.

start, end  Adds a symbol to either end of the siderange. `start` corresponds to minimal value, `end` to maximal value.

na.rm  If `FALSE` (default), missing values are removed with a warning. If `TRUE`, missing values are silently removed.

show.legend  Logical. Should this layer be included in the legends? `NA` (default), includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

inherit.aes  If `FALSE`, overrides the default aesthetic, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification.

Details

The `geom_siderange` projects the data displayed in the panel onto the sides, using the same aesthetics. It has the added capability of potting a symbol at either end of the line, and lines are offset from the edge and each other.

To display a symbol, specify an integer for either `start` or `end`. See the list for `pch` in `points` for values to use. The arguments `start` and `end` also accepts a list object with named entries `pch`, `alpha`, `stroke`, and `fill`, which correspond to the usual aesthetics, as well as a special named entry, `sizer` (note the extra 'r'). This last entry is a multiplier for enlarging the symbol relative to the linewidth, as the aesthetic `size` affects both linewidth and symbol size.

The distance between the panel’s edge and sideranges are specified by the argument `distance`. If a symbol is specified, the linewidth is further expanded to cover the width of the symbol (including `sizer`).

Aesthetics

The `geom` understands the following aesthetics (required are in bold):

- `x`
- `y`
- `alpha`
- `colour`
- `fill` (if a symbol is applied with `start` or `end`)
- `group`


**get_panel_range**

- linetype
- size
- stroke

**See Also**

*geom_rug*

**Examples**

```r
library(ggplot2)

x <- rnorm(25)
df <- data.frame(x=x, y=x+rnorm(25, sd=0.2),
                a=sample(c('horse', 'goat'), 25, replace=TRUE),
                stringsAsFactors = FALSE)
df$y <- ifelse(df$y > 1 & a=='horse', 1, y)
(p <- ggplot(df, aes(x=x, y=y, colour=a)) + geom_point(shape=1))

p + geom_siderange(start=19)

# Capping the sideranges with different symbols:
p + geom_siderange(start=19, end=22, fill='black', sides='b') + geom_siderange(sides='tl')

# It also works with facets
p <- ggplot(mpg, aes(displ, hwy, colour=f1)) + geom_point() +
    facet_wrap(~class, nrow = 4)

p + geom_siderange()
```

**Description**

The names of the internal layout objects from `ggplot_build` changed slightly.

**Usage**

```r
get_panel_y_range(layout, index = 1)
get_panel_x_range(layout, index = 1)
get_panel_params(layout, index = 1)
```
grid_arrange_shared_legend

Share a legend between multiple plots

Arguments

layout layout part from ggplot_build
index Could be panel number?

Description

Extract legend, combines plots using arrangeGrob / grid.arrange, and places legend in a margin.

Usage

grid_arrange_shared_legend(..., ncol = length(list(...)), nrow = 1, position = c("bottom", "right", "top", "left"), plot = TRUE)

Arguments

... Objects to plot. First argument should be a ggplot2 object, as the legend is extracted from this. Other arguments are passed on to arrangeGrob, including named arguments that are not defined for grid_arrange_shared_legend. ggplot2 objects have their legends hidden.
ncol Integer, number of columns to arrange plots in.
nrow Integer, number of rows to arrange plots in.
position 'bottom' or 'right' for positioning legend.
plot Logical, when TRUE (default), draws combined plot on a new page.

Value

gtable of combined plot, invisibly. Draw gtable object using grid.draw.

Author(s)

Originally brought to you by Baptiste Auguié (https://github.com/tidyverse/ggplot2/wiki/Share-a-legend-between-two-ggplot2-graphs) and Shaun Jackman (original). Stefan McKinnon Edwards added left and top margins.

See Also

g_legend, reposition_legend
Examples

library(ggplot2)
dsamp <- diamonds[sample(nrow(diamonds), 300), ]
p1 <- qplot(carat, price, data = dsamp, colour = clarity)
p2 <- qplot(cut, price, data = dsamp, colour = clarity)
p3 <- qplot(color, price, data = dsamp, colour = clarity)
p4 <- qplot(depth, price, data = dsamp, colour = clarity)
grid.arrange_shared_legend(p1, p2, p3, p4, ncol = 4, nrow = 1)
grid.arrange_shared_legend(p1, p2, p3, p4, ncol = 2, nrow = 2)

# Passing on plots in a grob are not touched
grid.arrange_shared_legend(p1, gridExtra::arrangeGrob(p2, p3, p4, ncol=3), ncol=1, nrow=2)

# We can also pass on named arguments to arrangeGrob:
title <- grid::textGrob('This is grob', gp=grid::gpar(fontsize=14, fontface='bold'))
nt <- theme(legend.position=’none’)
grid.arrange_shared_legend(p1, gridExtra::arrangeGrob(p2+nt, p3+nt, p4+nt, ncol=3), ncol=1, nrow=2, top=title)

---

gtable_show_grill Visualise underlying gtable layout.

Description
Visualises the table structure or the names of the gtable’s components.

Usage

gtable_show_grill(x, plot = TRUE)
gtable_show_names(x, plot = TRUE, rect.gp = grid::gpar(col = "black", fill = "white", alpha = 1/4))

Arguments

x A gtable object. If given a ggplot object, it is converted to a gtable object with ggplotGrob.

plot Logical. When TRUE (default), draws resulting gtable object on a new page.

rect.gp Graphical parameters (gpar) for background drop.

Details
These functions are highly similar to gtable_show_layout. gtable_show_grill draws the grid of the underlying table, and places row and column indices in the margin. gtable_show_names replaces the grobs with a semi-transparent rectangle and the component’s name.
Value

Modified gtable object, invisibly.

Examples

```r
library(ggplot2)
library(gtable)
library(grid)

p <- ggplot(mtcars, aes(x, mpg)) + geom_point()
gtable_show_grill(p)
library(ggplot2)
library(gtable)
library(grid)

p <- ggplot(mtcars, aes(x, mpg)) + geom_point()
gtable_show_names(p)
```

---

guidebox_as_column  Guidebox as a column

Description

Takes a plot or legend and returns a single guide-box in a single column, for embedding in e.g. tables.

Usage

```
guidebox_as_column(legend, which.legend = 1, add.title = FALSE)
```

Arguments

- **legend**: A ggplot2 plot or the legend extracted with `g_legend`. **Do not** provide a `ggplotGrob` as it is indistinguishable from a legend.
- **which.legend**: Integer, a legend can contain multiple guide-boxes (or vice versa?). Use this argument to select which to use.
- **add.title**: Does nothing yet.

Value

A `gtable` with keys and labels reordered into a single column and each pair of keys and labels in the same cell.

See Also

- `g_legend`
**g_legend**

**Examples**

```r
library(ggplot2)
library(dplyr)

p <- ggplot(diamonds, aes(x=x, y=y, colour=cut)) + geom_point()
guidebox_as_column(p)
p <- p + guides(colour=guide_legend(ncol=2, byrow=TRUE))
guidebox_as_column(p)
```

**Description**

Extracts the legend (‘guide-box’) from a ggplot2 object.

**Usage**

```r
g_legend(a.gplot)
```

**Arguments**

- `a.gplot`: ggplot2 or gtable object.

**Details**

The extraction is applied after the plot is trained and themes are applied. Modifying the legend is easiest by applying themes etc. to the ggplot2 object, before calling `g_legend`.

An alternative method for extracting the legend is using `gtable::gtable_filter`:

```r
gtable_filter(ggplotGrob(a.ggplot.obj), 'guide-box')
```

This method however returns a `gtable` object which encapsulates the entire legend. The legend itself may be a collection of `gtable`. We have only noticed a problem with this extra layer when using the returned legend with `arrangeGrob` (see examples).

**Value**

`gtable` (grob) object. Draw with `grid.draw`.

**Author(s)**

Baptiste Auguié

**See Also**

- `grid.arrange_shared_legend`, `reposition_legend`, `gtable_filter`
is.small

Is a given unit 'small'?

Description

Uses a holistic approach to determine whether a unit is 'small', i.e. less than 1 cm, 1 line, 10 pt, or 0.4 in.

Usage

is.small(x)

Arguments

x A unit.
Details

Based on arbitrarily chosen definitions of 'small', this function can return TRUE or FALSE if a unit is 'small'.

So far, less than 1 cm, 1 line, 10 pt, or 0.4 inches is defined as being 'small'. Unresolved sizes, such as 'grobheight', 'grobwidth', or 'null' are not small. Units based on arithmetic, such as sum of multiple units, are also not small. NAs are returned for undecided sizes.

Value

Logical or NA.

Description

Collection of misc. functions for changing subtle aspects of ggplots. Works mostly on gtables produced prior to printing.

Functions for axis

See coord_capped_cart and coord_flex_cart. The latter is a shorthand version of the former. It automatically uses capped_horizontal and capped_vertical, but both accepts these as well as brackets_horizontal and brackets_vertical.

Legends

Extract legend  g_legend

Many plots, one legend  grid_arrange_shared_legend

Place legend exactly on plot  reposition_legend

Facets

facet_rep_grid and facet_rep_wrap are extensions to the wellknown facet_grid and facet_wrap
where axis lines and labels are drawn on all panels.

Extending knitr

We automatically load knitr’s knit_print for data frames and dplyr tables to provide automatic pretty printing of data frame using kable.

See lemon_print or vignette('lemon_print', 'lemon').

Relative paths safe from hanging directory: .dot.
lemon_print

Author(s)

Stefan McKinnon Edwards <sme@iysik.com>
Contributions from Baptiste Auguié on g_legend and grid_arrange_shared_legend.
Contributions from Shaun Jackman on grid_arrange_shared_legend.

Source

https://github.com/stefanedwards/lemon

See Also

Useful links:

- https://github.com/stefanedwards/lemon
- Report bugs at https://github.com/stefanedwards/lemon/issues

lemon_print knitr extension: Always use ‘kable’ for data frames.

Description

Convenience function for working with R Notebooks that ensures data frames (and dplyr tables) are printed with kable while allowing RStudio to render the data frame dynamically for inline display.

Usage

lemon_print(x, options, ...)

## S3 method for class 'data.frame'
lemon_print(x, options, ...)

## S3 method for class 'table'
lemon_print(x, options, ...)

Arguments

x an data frame or dplyr table object to be printed
options Current chunk options are passed through this argument.
... Ignored for now.

Details

These functions divert data frame and summary output to kable for nicely printing the output.
For options to kable, they can be given directly as chunk-options (see arguments to kable), or though as a list to a special chunk-option kable.opts.
For more examples, see vignette('lemon_print', package='lemon').
Knitr usage

To use for a single chunk, do

````
{r render=l lemon_print, caption='My data frame'}
data.frame
````

Note: We are not calling the function, but instead refering to it.

An alternate route for specifying kable arguments is as:

````
{r render=l lemon_print, kable.opts=list(align='1')}data.frame
````

The option kable.opts takes precendence over arguments given directly as chunk-options.

To enable as default printing method for all chunks, include

```
knit_print.data.frame <- lemon_print
knit_print.table <- lemon_print
knit_print.grouped_df <- lemon_print # enableds dplyr results
knit_print.tibble <- lemon_print
knit_print.tbl <- lemon_print
```

Note: We are not calling the function, but instead assigning the knit_print functions for some classes.

To disable, temporarily, specify chunk option:

```
{r render=normal_print}
data.frame
```

See Also

knit_print, kable

---

remove_labels_from_axis

Removes labels from axis grobs.

Description

Called from FacetGridRepeatLabels.

Usage

remove_labels_from_axis(axisgrob)
reposition_legend

Reposition a legend onto a panel

Description

Repositions a legend onto a panel, by either taking it from the same ggplot, or by using another. Works on both ggplot2 and gtable objects, and can accept any grob as legend.

Usage

reposition_legend(aplot, position = NULL, legend = NULL, panel = "panel", x = NULL, y = NULL, just = NULL, name = "guide-box", clip = "on", offset = c(0, 0), z = Inf, plot = TRUE)

Arguments

aplot a ggplot2 or gtable object.
positional Where to place the legend in the panel. Overrules just argument.
legend The legend to place, if NULL (default), it is extracted from aplot if this is a ggplot2 object.
panel Name of panel in gtable. See description.
x horizontal coordinate of legend, with 0 at left.
y vertical coordinate of legend, with 0 at bottom.
just 'Anchor point' of legend; it is this point of the legend that is placed at the x and y coordinates.
names, clip, z Parameters forwarded to gtable_add_grob.
offset Numeric vector, sets distance from edge of panel. First element for horizontal distance, second for vertical. Not used by arguments x and y.
plot Logical, when TRUE (default), draws plot with legend repositioned on a new page.

Details

To modify the look of the legend, use themes and the natural ggplot functions found in guide_legend. Positioning is done by argument position which places the panel relative in panel (see below). position resolves to three variables, x, y, and just. x and y is the coordinate in panel, where the anchorpoint of the legend (set via just) is placed. In other words, just='bottom right' places the bottom right corner of the legend at coordinates (x,y).
The positioning can be set by argument position alone, which can be further nudged by setting position, x, and y. Alternatively, manually positioning can be obtained by setting arguments x, y, and just.

Panel name is by default panel, but when using facets it typically takes the form panel-{col}-{row}, but not for wrapped facets. Either print result from ggplotGrob or use gtable_show_names to display all the names of the gtable object.

Panel takes multiple names, and will then use these components’ extremes for placing the legend. If panel is an integer vector of length 2 or 4, these elements are used directly for top-left and bottom-right coordinates.

Value
gtable object, invisibly, with legend repositioned. Can be drawn with grid.draw.

Author(s)
Stefan McKinnon Edwards <sme@iysik.com>

See Also
g_legend, grid.arrange.shared.legend

Examples

library(ggplot2)
dsamp <- diamonds[sample(nrow(diamonds), 1000), ]
(d <- ggplot(dsampl, aes(carat, price)) +
  geom_point(aes(colour = clarity)))

reposition_legend(d + theme(legend.position='bottom'), 'bottom right')

# To change the orientation of the legend, use theme's descriptors.
reposition_legend(d + theme(legend.position='bottom'), 'top left')

# Use odd specifications, here offset the legend with half its height from the bottom.
reposition_legend(d + theme(legend.position='bottom'), x=0.3, y=0, just=c(0, -0.5))

# For using with facets:
reposition_legend(d + facet_grid(.~cut), 'top left', panel = 'panel-3-1')

scale_x_symmetric  Symmetrical position scale for continuous x and y

Description

scale_x_symmetric and scale_y_symmetric are like the default scales for continuous x and y, but ensures that the resulting scale is centered around mid. Does not work when setting limits on the scale.
Usage

scale_x_symmetric(mid = 0, ...)

scale_y_symmetric(mid = 0, ...)

Arguments

mid        Value to center the scale around.
...        Values passed on to scale_continuous.

Examples

library(ggplot2)
df <- expand.grid(a=c(-1,0,1), b=c(-1,0,1))
rnorm2 <- function(x,y,n,sdx,sdy) {
  if (missing(sdy))
    sdy <- sdx
  data.frame(a=x,b=y,x=rnorm(n,n,x,sdx), y=rnorm(n,y,sdy))
}
df <- mapply(rnorm2, df$a, df$b, MoreArgs=list(n=30,sdx=1),SIMPLIFY=FALSE)
df <- do.call(rbind, df)
(p <- ggplot(df, aes(x=x, y=y)) + geom_point() +
  facet_grid(a~b, scales='free_x')
)
p + scale_x_symmetric(mid=0)
Index

*Topic interal
  remove_labels_from_axis, 23
  .dot, 2, 21
  .dot2 (.dot), 2

absoluteGrob, 8
aes, 11, 13
aes_, 11, 13
annotate_x_axis (annotate_y_axis), 3
annotate_y_axis, 3
arrangeGrob, 16, 19
arrow, 12, 14

brackets_horizontal
  (brackets_horizontal), 5
brackets_horizontal, 5, 6–8, 21
brackets_vertical, 21
brackets_vertical
  (brackets_horizontal), 5
capped_horizontal (coord_capped_cart), 6
capped_horizontal, 6–8, 21
capped_horizontal (coord_capped_cart), 6
capped_vertical, 21
capped_vertical (coord_capped_cart), 6
coord_capped_cart, 5, 6, 21
coord_capped_flip (coord_capped_cart), 6
coord_cartesian, 6
coord_flex_cart, 5, 7, 7, 21
coord_flex_fixed (coord_flex_cart), 7
coord_flex_flip, 7
coord_flex_flip (coord_flex_cart), 7
coord_flip, 6
element.render, 9

facet_grid, 10, 21
facet_rep_grid, 10, 21
facet_rep_wrap, 21
facet_rep_wrap (facet_rep_grid), 10
facet_wrap, 10, 21

g_legend, 16, 18, 19, 21, 22, 25
geom_path, 10
geom_point, 10
geom_pointline (geom_pointpath), 10
geom_pointpath, 10
geom_pointrangeline (geom_pointpath), 10
ggeom_rug, 15
ggeom_siderange, 13
get_panel_params (get_panel_range), 15
get_panel_range, 15
get_panel_x_range (get_panel_range), 15
get_panel_y_range (get_panel_range), 15
ggplotGrob, 17, 18, 25
gpar, 9, 17
ggrid.draw, 16, 19, 25
ggrid.grob, 9
ggrid.arrange_shared_legend, 16, 19, 21, 22, 25
ggtable, 18
ggtable.add_grob, 24
ggtable.filter, 19
gggtable.show_grill, 17
gggtable.show_layout, 17
gggtable.show_names, 25
gggtable.show_names (ggtable.show_grill), 17
guide.legend, 24
guidebox.as.column, 18

is_small, 20

kable, 21–23

knit_print, 21, 23

layer, 12, 14

lemon, 21

lemon-package (lemon), 21

lemon_print, 21, 22

plotmath, 4
points, 14
position_jitter, 11, 14

remove_labels_from_axis, 23
render_gpar(element_render), 9
reposition_legend, 16, 19, 21, 24

scale_continuous, 26
scale_x_symmetric, 25
scale_y_symmetric(scale_x_symmetric), 25

theme, 5, 9, 10

unit, 5, 11