

Package ‘plotheper’

August 1, 2019

Type Package

Title New Plots Based on 'ggplot2' and Functions to Create Regular Shapes

Version 0.1.4

Date 2019-08-02

Maintainer Jiang Wu <textidea@sina.com>

Description An extension to 'ggplot2' with miscellaneous functions. It contains two groups of functions: Functions in the first group draw 'ggplot2' - based plots: `gg_shading_bar()` draws barplot with shading colors in each bar. `geom_rect_cm()`, `geom_circle_cm()` and `geom_ellipse_cm()` draw rectangles, circles and ellipses with centimeter as their unit. Thus their sizes do not change when the coordinate system or the aspect ratio changes. `annotation_transparent_text()` draws labels with transparent texts. `annotation_shading_polygon()` draws irregular polygons with shading colors. Functions in the second group generate coordinates for regular shapes and make linear transformations.

License GPL-3

URL <https://github.com/githubwwwjjj/plotheper/blob/master/README.md>

Depends R (>= 3.5.0), ggplot2 (>= 3.1.1)

Imports plyr, ggfittext (>= 0.8.1), magick, grid, gridExtra, scales

Encoding UTF-8

LazyLoad true

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Jiang Wu [aut, cre] (from Capital Normal University)

Repository CRAN

Date/Publication 2019-08-01 16:40:09 UTC

R topics documented:

ABCxy	2
add_slash_n	3
annotation_shading_polygon	4
annotation_transparent_text	6
ANYxy	9
count_each_column	10
ellipsex	11
enlarge_raster	13
geom_circle_cm	14
geom_ellipse_cm	15
geom_rect_cm	16
get_gg_label	17
gg_shading_bar	18
rectxy	19
resize_to_standard	21
rotatexy	21
round_text	22
scale_free	23
showcolor	24
spathxy	25
stretchxy	26
sunshinexy	27
textgif	28
Index	30

ABCxy

Symmetrical Transformation

Description

A1 and A2 are symmetrical on the two sides of $Ax+By+C=0$. The input of the function is A1, and the result is A2. The function also works when the line is horizontal or vertical. Note: the two shapes are symmetrical only when `ggplot2::coord_fixed()` is used.

Usage

```
ABCxy(x, A, B, C, p1 = NULL, p2 = NULL, f = NULL, group = TRUE,
      todf = TRUE, checks = TRUE)
```

Arguments

x	the input. It can be a data frame, matrix, tibble object, or a list of these kinds of objects. Each object must have exactly 2 columns and must be numeric without NA.
A	for $Ax+By+C=0$.

B	for $Ax+By+C=0$.
C	for $Ax+By+C=0$.
p1	if A, B, C are not given, you can also give two points p1 and p2 on the supposed $Ax+By+C=0$ line. Note: if A, B, C, p1, p2 are all given, the given A, B, C will be ignored. It must be a vector of length 2. The first element is x and the second is y.
p2	see p1.
f	argument passed to <code>split</code> to divide a data frame into a list of data frames. It should be a vector whose length is equal to the number of rows of x (if x is a data frame).
group	default is TRUE. It indicates whether to add a 3rd column named "d" to label the group number of each group of points. It is useful when using <code>aes(...group=g)</code> with 'ggplot2'.
todf	default is TRUE. It indicates whether to combine the output (a list) into a data frame.
checks	default is TRUE. It indicates whether to check input validity. Do not turn it off unless you are sure that the input is OK.

Value

if `todf = TRUE`, the output will be a data frame with coordinates of possibly several polygons, otherwise, it will be a list of data frames. Data frames have 2 columns named "x" and "y", and if `group = TRUE`, a third column named "g" is added indicating group numbers.

Examples

```
library(ggplot2)
dat1=data.frame(x=c(0, 2, 2, 0), y=c(0, 0, 1, 1))
dat2=ABCxy(dat1, -1, -1, 3)
ggplot()+
  coord_fixed()+
  geom_polygon(data=dat1, aes(x=x, y=y), fill="red")+
  geom_polygon(data=dat2, aes(x=x, y=y), fill="blue")+
  geom_abline(intercept=3, slope=-1)
dat3=ABCxy(dat1, p1=c(0, 1), p2=c(-0.5, 0), todf=TRUE)
ggplot()+
  coord_fixed()+
  geom_polygon(data=dat1, aes(x=x, y=y), fill="red")+
  geom_polygon(data=dat3, aes(x=x, y=y), fill="blue")+
  geom_abline(intercept=1, slope=2)
```

Description

This function simply adds change-line signs inside strings, so that they can be put vertically as the texts of x-axis.

Usage

```
add_slash_n(x, delete_space = TRUE, vertical_line = TRUE)
```

Arguments

`x` a character vector
`delete_space` whether to delete spaces. Default is TRUE.
`vertical_line` whether to change - into |. Default is TRUE.

Examples

```
lab=add_slash_n(c("a b-c", "d - ef ", "n"))
```

annotation_shading_polygon

Layer for Drawing a Single Irregular Polygon with Shading Colors

Description

ggplot2::annotation_raster can only draw shading rectangles. However, this function can draw polygons of any shape with shading colors. See the shape argument and the raster argument.

Usage

```
annotation_shading_polygon(shape = data.frame(c(-1, 1, 0), c(0, 0, 1.732)), xmin = NULL, xmax = NULL, ymin = NULL, ymax = NULL,
  raster = NULL, interpolate = TRUE, result_interpolate = TRUE,
  shape_trim = NULL, raster_trim = NULL, result_trim = NULL,
  result = c("layer", "magick"), width = 800, height = NULL,
  res = 72)
```

Arguments

`shape` the polygon can be a data frame (or matrix object, or tbl_df object) with x and y coordinates (that is, with two columns), a plot created by ggplot or an image read into R by magick::image_read. If it is a plot created by ggplot, its axes can be of numeric, discrete or date/datetime type; however, when the type is date/datetime, the plot should not use ggplot2::coord_fixed.

`xmin` the left side of the position to put the polygon. When shape is something like a data frame, you do not need to set xmin, xmax, ymin and ymax, for the function will generate these values according to the coordinates in the polygon.

xmax	the right side.
ymin	the bottom side.
ymax	the top side.
raster	the shading colors. It can be a raster object, a matrix of colors, a ggplot plot or an image read into R by <code>magick::image_read</code> .
interpolate	the interpolate argument used by <code>ggplot2::annotation_raster</code> when the raster argument is a matrix or raster.
result_interpolate	whether to interpolate in the final result which is essentially an output of <code>ggplot2::annotation_raster</code> . Default is TRUE.
shape_trim	this argument decides whether to trim edges of shape. It should be a number between 0 and 100. Default is NULL. If it is NULL, no trimming will be done.
raster_trim	whether to trim raster. Most of the time we do want to trim the raster. However, the <code>magick::image_trim</code> function sometimes trims wrongly. So you may want to turn it off. Default is NULL.
result_trim	how to trim the final result. If you find your figure loses some parts, you can try to turn this off. Default is NULL.
result	when it is "layer", the function is a ggplot layer. When it is "magick", the function only create an image.
width	the width which will be passed to <code>magick::image_graph</code> . Most of the time you do not need to modify this. Default is 800. HOWEVER, if the final polygon has fuzzy edges, try to enlarge width to make them look better.
height	the height which will be passed to <code>magick::image_graph</code> . DO SEE Details below to see how to use this parameter.
res	resolution in pixels which will be passed to <code>magick::image_graph</code> . Default is 72.

Details

height can be used in the following ways:

- (1) an integer which will be directly passed to `image::graph`.
- (2) a character-like integer, e.g., `height = "0.5"`. Suppose `width = 400`, the height that will be used is $400 * 0.5 = 200$. This effectively prevents the image from becoming too large.
- (3) `height = "coord_fixed"`. the ratio between height and width will be (top-bottom)/(right-left). And top, bottom, right and left are extreme values of shape when the latter is of class `data.frame/matrix/gg`.
- (4) `height = "image"`. the width and height will be the width and height of raster when raster is a magick object.
- (5) `height = NULL`, the default. Now height is computed automatically. A ratio is computed first, $ratio = (top-bottom)/(right-left)$. if the ratio is larger than 5 or smaller than 0.2, then height will be $width * 5$ or $width * 0.2$; else, the height will be treated in the same way as in (3) above. If shape is of class `gg` and it has uses `coord_flip()`, the height will be automatically adjusted. All these works are needed to prevent the image from becoming too large.

Examples

```
# Example 1
poly=ellipsexy(-1, 0, a=1, b=1)
m=matrix(rainbow(7))
ggplot()+
  coord_fixed()+
  annotation_shading_polygon(
    poly, raster=m
  )+
  annotation_shading_polygon(
    poly, raster=m,
    xmin=1, xmax=5,
    ymin=-1, ymax=1,
  )
#
# Example 2, only an image
tt=annotation_shading_polygon(
  poly, result="magick",
  width=280, height=280
)
#
# Example 3, both shape and raster are
# ggplot plots.
p1=ggplot()+geom_tile(aes(x=1: 5, y=1: 5))
p2=ggplot()+geom_polygon(aes(x=c(0, 1, 1, 0),
  y=c(0, 0, 1, 1)), fill="red")+theme_void()
ggplot()+coord_fixed()+
  annotation_shading_polygon(
    shape=p1,
    xmin=1, xmax=10,
    ymin=1, ymax=5,
    raster=p2
  )
```

annotation_transparent_text

Layer for Transparent Text

Description

Suppose there is a colored rectangle with some texts and you want the texts to be transparent so that the colors of the background can be seen. Now you can use this function. The function can be used as a ggplot layer or a generator of image. NOTE: when the function is used as a layer, it uses `ggplot2::annotation_raster` to do the drawing, so you must set limits for the x axis and the y axis. See examples.

Usage

```
annotation_transparent_text(label, xmin, xmax, ymin, ymax, bg = "black",
  alpha = 0.5, interpolate = TRUE, result_interpolate = TRUE,
  expand = c(0.05, 0.05), family = "SimHei", fontface = 1,
  reflow = FALSE, place = "center", label_trim = NULL,
  bg_trim = NULL, result = c("layer", "magick"), width = 800,
  height = NULL, res = 72, ...)
```

Arguments

label	the text.
xmin	the left side of the rectangle.
xmax	the right side of the rectangle.
ymin	the bottom side of the rectangle.
ymax	the top side of the rectangle.
bg	the colors of the rectangle. It can be a character vector of colors, a matrix of colors, an object of raster class or even a image read into R through <code>magick::image_read</code> . Default is color black.
alpha	it is only used when <code>bg</code> is a character vector. Default is 0.5.
interpolate	when <code>bg</code> is a matrix, a image or a raster, this parameter is used and will be passed to <code>ggplot2::annotation_raster</code> to draw a colored rectangle. Default is TRUE.
result_interpolate	whether to use interpolate in the final result. Default is TRUE.
expand	sometimes it is needed to slightly expand the x position and y position to put the text so that they can be shown nicely. It should be two values used by x and y respectively. Default is 0.05 and 0.05.
family	family of text. Default is SimHei which ensures that Chinese texts can be shown. However, you can change it to others, e. g., sans, serif, mono.
fontface	fontface.
reflow	whether to change lines automatically. It will be passed to <code>ggfittext::geom_fit_text</code> . Default is FALSE.
place	position adjustment used by <code>ggfittext::geom_fit_text</code> . The value is one of "center", "middle" (= "center"), "topleft", "top", "topright", "right", "bottom-right", "bottom", "bottomleft", "left".
label_trim	whether to trim label. The default is NULL which means no trimming. But if you want to remove all edges around label, you should give <code>label_trim</code> a value which will be passed to <code>magick::image_trim</code> . However, most of the time you do not need this parameter.
bg_trim	whether to trim <code>bg</code> . Most of the time we do want to trim it. However, the <code>magick::image_trim</code> function sometimes trims wrongly. So you can turn it off. NOTE: the default value of <code>bg_trim</code> is NULL, which means DO NOT TRIM.

result	when it is "layer", the function can be used as a ggplot layer. When it is "magick", the result is only an image which is created by the magick package.
width	the width of the text rectangle. It will be passed to <code>magick::image_graph</code> . Most of the time you do not need to modify this. Default is 800.
height	the height of the text rectangle. It will be passed to <code>magick::image_graph</code> . Default is NULL, which means it will be computed automatically. DO SEE Details below to learn how to handle this parameter.
res	resolution in pixels which will be passed to <code>magick::image_graph</code> . Default is 72.
...	arguments which will be passed to <code>ggfittex::geom_fit_text</code> . Most often used are <code>angle</code> (0 to 360), <code>lineheight</code> .

Details

height can be used in the following ways:

- (1) an integer which will be directly passed to `image::graph`.
- (2) a character-like integer, e.g., `height = "0.5"`. Suppose `width = 400`, the height that will be used is $400 * 0.5 = 200$. This effectively prevents the image from becoming too large.
- (3) `height = "coord_fixed"`. the ratio between height and width will be $(y_{max} - y_{min}) / (x_{max} - x_{min})$.
- (4) `height = "image"`. the width and height will be the width and height of raster when raster is a magick object.
- (5) `height = NULL`, the default. Now height is computed automatically. If `bg` is a magick object, the width and height of the image will be used. If `bg` is not a magick object, a ratio is computed first, $ratio = (y_{max} - y_{min}) / (x_{max} - x_{min})$. if the ratio is larger than 5 or smaller than 0.2, then height will be `width * 5` or `width * 0.2`; else, the height will be treated in the same way as in (3) above. All these works are needed to prevent the image from becoming too large.

Examples

```
# Example 1
m=matrix(rainbow(7), nrow=1)
ggplot()+coord_fixed()+
xlim(0, 7)+ylim(-2, 4)+theme_void()+
annotation_raster(
  raster=m,
  xmin=0, ymin=-3,
  xmax=7, ymax=5,
  interpolate=TRUE
)+
annotation_transparent_text(
  label="R\nDATA\nVISUALIZATION",
  xmin=0, xmax=7,
  ymin=-1, ymax=3,
  family="sans", fontface=2, alpha=0.8,
  place="left", expand=c(0.08, 0.02)
```



```

)
#
# Example 2, this time the result is only an image.
tt=annotation_transparent_text(
  label="abcdefg",
  xmin=1, xmax=8,
  ymin=1, ymax=4,
  alpha=0.6,
  result="magick"
)
#
# Example 3, the rectangle is a matrix.
m=colorRampPalette(c("yellow", "purple"))(10)
ggplot()+coord_fixed(expand=FALSE)+
  theme(panel.background=element_rect(fill="red"))+
  annotation_transparent_text(
    label="hehehaha",
    xmin=1, xmax=8,
    ymin=1, ymax=4,
    bg=m, alpha=1
  )
#
# Example 4, height is too large.
# Now you should explicitly set
# width and height, otherwise, the
# characters will become too flat.
x=c(0, 5, 10)
y=c(0, 500, 1000)
ggplot()+ylim(0, 4000)+
  geom_point(aes(x, y))+
  annotation_transparent_text(label="ha ha\nhe he",
    xmin=0, xmax=10, ymin=1000, ymax=4000, bg="black",
    width=300, height=150
  ) # do not set height=NULL here

```

Description

Given your function to create a multiple of points (for example, points to form a polygon), this function generates x and y coordinates for groups of points of the same type with different parameters. The output of this function can be shown by `ellipsey` and `rectxy` in this package.

Usage

```
ANYxy(myfun = NULL, ..., MoreArgs = NULL, group = TRUE,
      todf = TRUE)
```

Arguments

myfun	your function to generate a single polygon. Note: the value of each argument of your function must be a single-value vector. And the result of your function should be a data frame! . See examples.
...	named parameters used by your function. These parameters will be passed to mapply.
MoreArgs	this will be passed to the MoreArgs argument of mapply.
group	default is TRUE which means a column named "g" will be added to each data frame. This facilitates further drawing using aes(..., group = g).
todf	default is TRUE which means to combine the result into a data frame. Otherwise, the result is a list.

Examples

```
library(ggplot2)
# First, you need a function to generate
# x and y coordinates for a single group
# of points.
x_square=function(start, end, A, B){
  x=seq(start, end, 0.1)
  data.frame(x=x, y=A*(x^2)+B)
}
# All the arguments of your function
# (here, start, end, A, B) should only accept
# vectors of length 1. And, the result of
# your function should be a data frame
# of x and y coordinates
# (here, coordinates of curves).
dat=ANYxy(myfun=x_square,
start=-1, end=1, A=c(1, 2), MoreArgs=list(B=1),
group=TRUE, todf=TRUE)
ggplot(dat)+geom_line(aes(x, y, group=g, color=factor(g)))
```

count_each_column

Counting Each Column and Summarizing in a Matrix

Description

This function counts the frequencies of each element of each column of a data frame or matrix. The frequencies of missing values and the 0 frequencies of non-existent values are also included in the final result.

Usage

```
count_each_column(x, answer = NULL, checks = TRUE)
```

Arguments

x	a data frame or matrix with at least 1 row and 1 column. NOTE: all column should belong to the same class (numeric, character). However, if checks = TRUE, character and factor variables can co-exist and logical values are also OK. If a column has nothing but NA, it should be remove; otherwise, an error will be raised.
answer	the values whose frequencies you want to know, e. g., "agree" and "disagree" in your survey data. Default is NULL which means all possible answers in the whole data will be used.
checks	whether to check the validity of the input data. Default is TRUE. Do not turn it off unless you are sure that your data has no logical variables or factor variables and each column has at least 1 non-missing value.

Examples

```
# values that do not appear in
# the data can also be counted.
# a factor will be transformed into
# a character variable automatically.
x1=c("a", "b", "a", "b", NA)
x2=factor(x1)
x3=c("1", "3", "2", "1", "a")
dat=data.frame(x1, x2, x3, stringsAsFactors=FALSE)
res=count_each_column(dat, answer=c("c", "d", NA, "a"))
# logical value is OK.
x1=c(TRUE, TRUE, TRUE)
x2=c(TRUE, NA, NA)
dat=data.frame(x1, x2)
res=count_each_column(dat)
res=count_each_column(dat, c(TRUE, FALSE))
```

 ellipsexy

Generating Coordinates of Multiple Ellipses or Circles

Description

If radius a is equal to radius b, then the shape will be a circle. Note: the shapes are correct only when `ggplot2::coord_fixed()` is used.

Usage

```
ellipsexy(x = 0, y = 0, a = 2, b = 1, start = 0,
  end = 6.283185, angle = 0, n = 40, xytype = "middle",
  fan = FALSE, group = TRUE, todf = TRUE, checks = TRUE)
```

Arguments

x	the x coordinates of relative points. Its length can be larger than 1. See xytype.
y	the y coordinates of relative points. Its length can be larger than 1. See xytype.
a	the radius that is parallel to x-axis before rotation. Its length can be larger than 1.
b	the radius that is parallel to y-axis before rotation. Its length can be larger than 1.
start	default is 0. The angle of the starting point of the arc. Its length can be larger than 1. Note: "radian = degree * pi / 180".
end	default is 6.283185. The angle of the ending point of the arc. Its length can be larger than 1.
angle	default is 0. The rotation angle in radian. Its length can be larger than 1. Note: "radian = degree * pi / 180". The rotation direction is anti-clockwise.
n	default is 40. The number of points used to draw an arc. The larger, the smoother. It must at least be 4. However, when checks is FALSE, this check is ignored. NOTE: to draw a triangle, you must use <code>ellipsexy(n=4, fan=FALSE)</code> , as the first and 4th points are so close. Similarly, to draw a rectangle, use <code>ellipsexy(n=5, fan=FALSE)</code> .
xytype	should be one of "middle/center" (default), "bottomleft", "middleleft/left/centerleft". It indicates the type of argument of the middle point of an ellipse. If it is "middleleft", x and y are the middle-left coordinates before rotation. If it is "bottomleft", x and y are the coordinates of the bottom-left corner of the rectangle that walls the ellipse.
fan	default is FALSE. If it is TRUE, the coordinates of the middle of an ellipse is added to the output data frame. Meanwhile, if, say, you set n = 50, then n becomes 49 automatically because the last position is reserved for the middle. This helps draw a fan.
group	default is TRUE. It indicates whether to add a 3rd column named "d" to label the group number of each group of points. It is useful when using <code>aes(... group=g)</code> with 'ggplot2'.
todf	default is TRUE. It indicates whether to combine the output (a list) into a data frame.
checks	default is TRUE. It indicates whether to check input validity. Do not turn it off unless you are sure that the input is OK.

Value

if `todf = TRUE`, the output will be a data frame with coordinates of possibly several polygons, otherwise, it will be a list of data frames. Data frames have 2 columns named "x" and "y", and if `group = TRUE`, a third column named "g" is added indicating group numbers.

Examples

```
library(ggplot2)
dat1=ellipsexy(x=1, y=1,
```

```
a=seq(1, 4, length.out=8), angle=seq(0, pi, length.out=8),
xytype="middleleft", n=30, todf=TRUE)
ggplot()+coord_fixed()+
geom_polygon(show.legend=FALSE,
data=dat1, aes(x=x, y=y, group=g, fill=factor(g)), alpha=0.3)
```

enlarge_raster	<i>Enlarge a Color Matrix</i>
----------------	-------------------------------

Description

This is a convenient wrapper of `colorRampPalette` to enlarge a color matrix or raster.

Usage

```
enlarge_raster(x, n = c(10, 10), row_first = TRUE)
```

Arguments

<code>x</code>	a color matrix or raster. It should have at least 1 row and 1 column with no NAs.
<code>n</code>	a vector with 2 numbers. If it has 1 number, the number will be repeated twice. The two numbers indicate how many colors you will get in the result per row and per column. Default is <code>c(10, 10)</code> .
<code>row_first</code>	enlarge rows first or enlarge columns first? Default is TRUE. The results are almost the same, so you do not need to change this.

Examples

```
library(ggplot2)
# the original matrix
m=matrix(c(
"red", "yellow", "green",
"blue", "purple", "cyan",
"black", "orange", "grey"), byrow=TRUE, nrow=3)
# enlarge the matrix
mm=enlarge_raster(m, c(15, 15))
ggplot()+xlim(0, 10)+ylim(0, 5)+coord_fixed()+
annotation_raster(mm,
xmin=0, xmax=10, ymin=0, ymax=5, interpolate=TRUE)
```

 geom_circle_cm

Geom Layer for Circle with Absolute Size

Description

This layer uses centimeter as unit to draw circles so that the size and shape will not be influenced by the change of the coordinate systems (even when a polar system is used). Note: this function does not have `linetype` and `n` arguments.

Usage

```
geom_circle_cm(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, linetype = NULL, ...)
```

Arguments

<code>mapping</code>	aes mapping.
<code>data</code>	data.
<code>stat</code>	stat.
<code>position</code>	position.
<code>na.rm</code>	logical, whether to remove NA values.
<code>show.legend</code>	whether to show legend.
<code>inherit.aes</code>	logical, whether to inherit aes from <code>ggplot()</code> .
<code>linetype</code>	should always be NULL. because it will not be used.
<code>...</code>	additional parameters.

Details

Accepted properties are:

- (1) `rcm` radius in centimeter.
- (2) `color` color of the outline.
- (3) `fill` color inside the shape.
- (4) `alpha` alpha of color and fill.
- (5) `size` line width of the outline.
- (6) `x` x coordinates of the middle points.
- (7) `y` y coordinates of the middle points.

Examples

```
library(ggplot2)
dat=data.frame(x=1: 10, y=rep(5, 10), R=rep(c(0.5, 1), 5))
ggplot(dat)+xlim(0, 11)+ylim(1, 9)+
  geom_circle_cm(aes(x=x, y=y, fill=factor(R)), rcm=dat$R, alpha=0.5)
```

geom_ellipse_cm	<i>Geom Layer for Ellipse with Absolute Size</i>
-----------------	--

Description

This layer uses centimeter as unit to draw ellipse so that its size and shape will not be influenced by the coordinate systems (even when a polar system is used).

Usage

```
geom_ellipse_cm(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...)
```

Arguments

mapping	aes mapping.
data	data.
stat	stat.
position	position.
na.rm	logical, whether to remove NA values.
show.legend	whether to show legend.
inherit.aes	logical, whether to inherit aes from ggplot().
...	additional parameters.

Details

Accepted properties are:

- (1) rcm radius in centimeter.
- (2) ab it means to what extent radius a of an ellipse is larger than radius b. However, its true meaning is the aspect ratio which is used by `gridExtra::ellipseGrob` and indicates the extent to which y dimension is flattened. So, say, when $ab = 2$, radius a is larger than b, but it is not exactly 2 times larger.
- (3) color color of the the outline.
- (4) fill color inside the shape.
- (5) alpha alpha of color and fill.
- (6) size line width of the outline.
- (7) linetype line type.
- (8) angle angle of rotation from 0 degree and in anti-clockwise direction.
- (9) n the number of points to draw the shape. Note: it must be written inside the `aes(...)` function.
- (10) x x coordinates of middle points.
- (11) y y coordinates of middle points.

Examples

```
library(ggplot2)
dat=data.frame(x=c(1, 3, 5, 7, 9), y=rep(5, 5))
ggplot(dat)+xlim(0, 11)+ylim(1, 9)+
geom_ellipse_cm(aes(x=x, y=y), fill="red", ab=seq(1, 4, length.out=5))
ggplot(dat)+xlim(0, 11)+ylim(1, 9)+
geom_ellipse_cm(aes(x=x, y=y, fill=factor(x)), ab=3, angle=c(0, pi/4, pi/3, pi/2, 0.75*pi))
```

geom_rect_cm

*Geom Layer for Rectangle with Absolute Size***Description**

This layer uses centimeter as unit to draw rectangles so that the size and shape will not be influenced by the coordinate systems (even when a polar system is used).

Usage

```
geom_rect_cm(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...)
```

Arguments

mapping	aes mapping.
data	data.
stat	stat.
position	position.
na.rm	logical, whether to remove NA values.
show.legend	whether to show legend.
inherit.aes	logical, whether to inherit aes from ggplot().
...	additional parameters.

Details

Accepted properties are:

- (1) width width in centimeter.
- (2) height height in centimeter.
- (3) color color of the outline.
- (4) fill color inside the shape.
- (5) alpha alpha of color and fill.
- (6) size line width of outline.
- (7) linetype line type.

- (8) hjust horizontal adjustment, default is 0.5 which means no adjustment.
- (9) vjust vertical adjustment, default is 0.5 which means no adjustment.
- (10) x x coordinates of middle points.
- (11) y y coordinates of middle points.

Examples

```
library(ggplot2)
ggplot()+xlim(-0.5, 10.5)+
geom_rect_cm(aes(x=1: 10, y=rep(4, 10)), fill="red", height=rep(1: 2, each=5),
vjust=rep(c(0, 0.5), 5))+
geom_point(aes(x=1: 10, y=rep(4, 10)), color="green")
```

get_gg_label

Checking Min, Max, Labels and Label Positions

Description

Given a numeric vector or a ggplot object, the function will check the range, labels and label positions (the same as major grid lines) that will used on the axis. The result is a length 5 list for min limit, max limit, labels, major grid line positions, all (major and minor) grid line positions.

Usage

```
get_gg_label(a = NULL, b = NULL, v = NULL, gg = NULL,
  mult = 0.05, add = 0, axis = "y")
```

Arguments

a	extreme values of a numeric vector. Note: only one of a, v, gg can be non-NULL.
b	another extreme value if a is not NULL.
v	a numeric vector.
gg	a gg object created by ggplot function. Which value will be checked depends on axis.
mult	default is 0.05 and should be of length 1 or 2. It mimics the mult argument of ggplot2::expand_scale. It is only used when a is numeric or v is non-NULL.
add	default is 0. It mimics the add argument of ggplot2::expand_scale.
axis	if gg is used or a is a ggplot object, which axis will be checked? It can be "x" or "y" (default).

Examples

```
# The following three have the same results.
get_gg_label(a=1, b=1000, mult=0)
get_gg_label(v=c(1, 500, 1000), mult=0)
p=ggplot()+geom_point(aes(1: 3, c(1, 500, 1000)))+
  scale_y_continuous(expand=expand_scale(mult=0))
get_gg_label(gg=p)
```

 gg_shading_bar

Drawing Barplot with Shading Colors

Description

In ordinary barplot, each bar has only one color. This function aims to draw a barplot whose bars have shading effect. Note: unlike `ggplot2::geom_bar`, this function can only deals with a vector of frequencies.

Usage

```
gg_shading_bar(v, labels = NULL, raster = NULL, flip = FALSE,
  change_order = "normal", equal_scale = FALSE, smooth = 15,
  interpolate = TRUE, width = 0.8, color = NA, linetype = 1,
  size = 1, modify_raster = TRUE, ...)
```

Arguments

<code>v</code>	a vector of item frequencies. Negative values are OK.
<code>labels</code>	a vector of item names. Its length should be equal to that of <code>v</code> . If it is <code>NULL</code> , default names will be used. If it is of class <code>numeric</code> or <code>factor</code> , it will be transformed to a character vector.
<code>raster</code>	a list. The length of the list should be equal to that of <code>v</code> . Each element of the list should be a color vector corresponding to a value in <code>v</code> . If it is a vector, it will be automatically transformed to a list. If its length is 1, but the length of <code>v</code> is, say, 3, then it will be automatically repeated for 3 times. Let us suppose <code>v = 5</code> and <code>raster = list(c("green", "red"))</code> . This means the starting side of the bar is green and the far side is red. See examples.
<code>flip</code>	default is <code>FALSE</code> and the bars are vertical. When it is <code>TRUE</code> , the bars are horizontal. Note: when using this function, DO NOT USE <code>ggplot2::coord_flip</code> !
<code>change_order</code>	when it is "normal" (default), the drawing order is the order of <code>v</code> . When it is "big", big values will be drawn first. When it is "small", small values will be drawn first. When it is "rev", the inverse order of <code>v</code> will be used.
<code>equal_scale</code>	default is <code>FALSE</code> . When it is <code>TRUE</code> , a bar will use a certain part of the shading colors according to a global scale. See examples.
<code>smooth</code>	default is 15. The number of shading colors each bar has. The bigger, the better.

interpolate	when it is TRUE (default), it makes the colors smoother.
width	the width of each bar. It should be between 0 and 1.
color	color of the outlines of the bars.
linetype	line type of the outlines of the bars.
size	line width of the outlines of the bars.
modify_raster	if it is TRUE (default), colors will be smoothed using the value of smooth. If raster has enough colors, you can set this to FALSE.
...	additional arguments used by <code>ggplot2::coord_flip</code> when <code>flip = TRUE</code> .

Examples

```
library(ggplot2)
x=c(10, 30, 25, 6)
lab=c("children", "youth", "middle", "aged")
r=list(c("cyan", "red"), c("blue", "yellow"),
c("green", "orange"), c("grey", "black"))
#
## (1) change_order
# change_order = "ordinary", the default
p1=gg_shading_bar(v=x, labels=lab)
# change_order = "big"
p2=gg_shading_bar(v=x, labels=lab, change_order="big")
# flip and let the largest on the top
p3=gg_shading_bar(v=x, labels=lab,
change_order="small", flip=TRUE)
#
## (2) how to use argument raster
p1=gg_shading_bar(v=x, labels=lab, raster=r)
p2=gg_shading_bar(v=x, labels=lab, raster=c("green", "red"))
#
## (3) how to use argument equal_scale
# equal_scale = FALSE
# the far side of each bar is red
gg_shading_bar(c(3, 5), raster=c("green", "red"))
# equal_scale = TRUE
# the far side of the shorter bar
# is not red. Rather, it is something
#' between red and green
gg_shading_bar(c(3, 5), raster=c("green", "red"),
equal_scale=TRUE)
```

Description

Note: the shapes are correct only when `ggplot2::coord_fixed()` is used.

Usage

```
rectxy(x = 0, y = 0, a = 1, b = 1, angle = 0,
       xytype = "middle", group = TRUE, todf = TRUE, checks = TRUE)
```

Arguments

x	the x coordinates of relative points. Its length can be larger than 1. See xytype.
y	the y coordinates of relative points. Its length can be larger than 1. See xytype.
a	the side that is parallel to x-axis before rotation. Its length can be larger than 1.
b	the side that is parallel to y-axis before rotation. Its length can be larger than 1.
angle	default is 0. The rotation angle in radian. Note: "radian = degree * pi / 180". Its length can be larger than 1. The rotation direction is anti-clockwise.
xytype	should be one of "middle/center" (default), "bottomleft", "middleleft/centerleft/left". It indicates the type of argument of the middle point of an shape. If it is "middleleft", x and y are the middle-left coordinates before rotation. If it is "bottomleft", x and y are the coordinates of the bottom-left corner.
group	default is TRUE. It indicates whether to add a 3rd column named "d" to label the group number of each group of points. It is useful when using aes(...group=g) with 'ggplot2'.
todf	default is TRUE. It indicates whether to combine the output (a list) into a data frame.
checks	default is TRUE. It indicates whether to check input validity. Do not turn it off unless you are sure that the input is OK.

Value

if todf = TRUE, the output will be a data frame with coordinates of possibly several polygons, otherwise, it will be a list of data frames. Data frames have 2 columns named "x" and "y", and if group = TRUE, a third column named "g" is added indicating group numbers.

Examples

```
library(ggplot2)
dat1=rectxy(x=4, y=3, a=2, b=1, angle=0, xytype="bottomleft", todf=TRUE)
dat2=rectxy(x=4, y=3, a=2, b=1, angle=pi/6, xytype="bottomleft", todf=TRUE)
ggplot()+
  geom_polygon(data=dat1, aes(x=x, y=y), fill="red", alpha=0.3)+
  geom_polygon(data=dat2, aes(x=x, y=y), fill="blue", alpha=0.3)+
  coord_fixed()
```

resize_to_standard *Resize an Image According to the Other Image or to Ratios*

Description

Simple wrapper of `magick::image_resize`. See the parameters below.

Usage

```
resize_to_standard(x, standard = 0.5, what = "all", scale = TRUE)
```

Arguments

x	the image you want to resize.
standard	either the image whose size is the standard or two ratios. When it specifies two ratios, it should be a numeric vector whose first and second elements are multipliers for width and height. For example, x's width and height are 100 and 60, and <code>standard = c(0.5, 3)</code> , then the result image's width and height will be 50 and 180. If one of the two number is NA, then the dimension represented by this NA will be modified automatically.
what	this parameter is used only when standard is an image. It specifies the way to resize. When it is "width", let x's width be the same as standard; whether its height is automatically scaled depends on scale. When it is "height", let x's height be the same as standard; whether its width is automatically scaled depends on scale. When it is "all" (or "both"), the default, let x's width and height be the same as standard. When it is two number linked with a "_", it means resizing a according to b's width and height multiplied. For example, if it is "3_2" and b's width and height are 50, 70, then the result's width and height are $50 * 3 = 150$, $70 * 2 = 140$. Forms like "_2" or "3_" are also accepted.
scale	Default is TRUE. It is only used when only one of width and height is to be modified. This parameter decides whether the image is automatically scaled.

rotatexy *Rotation Transformation*

Description

A2 (output) is the result of rotating A1 (input) around a point. Note: the two shapes look the same (though with different angles) only when `ggplot2::coord_fixed()` is used.

Usage

```
rotatexy(x, angle = pi/4, xmiddle = 0, ymiddle = 0, f = NULL,
         group = TRUE, todf = TRUE, checks = TRUE)
```

Arguments

x	the input. It can be a data frame, matrix, tibble object, or a list of these kinds of objects. Each object must have exactly 2 columns and must be numeric without NA.
angle	default is pi/4. The rotation angle in radian. Note: "radian = degree * pi / 180". Its length can be larger than 1. The rotation direction is anti-clockwise.
xmiddle	the x coordinates of rotation centers. Its length can be larger than 1.
ymiddle	the y coordinates of rotation centers. Its length can be larger than 1.
f	argument passed to split to divide a data frame into a list of data frames. It should be a vector whose length is equal to the number of rows of x (if x is a data frame).
group	default is TRUE. It indicates whether to add a 3rd column named "d" to label the group number of each group of points. It is useful when using aes(...group=g) with 'ggplot2'.
todf	default is TRUE. It indicates whether to combine the output (a list) into a data frame.
checks	default is TRUE. It indicates whether to check input validity. Do not turn it off unless you are sure that the input is OK.

Value

if todf = TRUE, the output will be a data frame with coordinates of possibly several polygons, otherwise, it will be a list of data frames. Data frames have 2 columns named "x" and "y", and if group = TRUE, a third column named "g" is added indicating group numbers.

Examples

```
library(ggplot2)
dat1=data.frame(x=c(0, 4, 4, 0), y=c(0, 0, 2, 2))
dat2=data.frame(x=c(5, 6, 6, 5), y=c(4, 4, 8, 8))
dat3=rotatexy(list(dat1, dat2), angle=c(pi, pi/4),
xmmiddle=c(0, 5), ymiddle=c(0, 4), todf=TRUE)
ggplot()+
coord_fixed()+
geom_polygon(data=dat1, aes(x=x, y=y), fill="red", alpha=0.2)+
geom_polygon(data=dat2, aes(x=x, y=y), fill="blue", alpha=0.2)+
geom_polygon(show.legend=FALSE, data=dat3,
aes(x=x, y=y, group=g, fill=factor(g)), alpha=0.2)
```

round_text

Converting Numeric Values into Characters with the Same Digits

Description

This simple function is to facilitate something like decimal horizontal adjustment which demands each value has the same digits after the decimal point.

Usage

```
round_text(x, digits = 2, na = NULL)
```

Arguments

x a vector of numeric values.

digits digits which is to be passed to round. It should not be smaller than 0.

na how to show NAs. The default is to show " NA", however, you can change it to "NA" or simply NA.

Examples

```
v=c(3, 3.1, 3.456, 3.452, 3.77, NA, 0, 10.56332)
res=round_text(v, 2, na=NA)
```

scale_free

Scale values into a Certain Location

Description

A simple function to put numeric values into a certain interval. Suppose you have 20, 60, 80, 100, and you want them to be in the interval of [0, 1], so you can get 0, 0.5, 0.75, 1.

Usage

```
scale_free(x, left = 0, right = 1, reverse = FALSE, xmin = NULL,
           xmax = NULL, na.rm = FALSE)
```

Arguments

x a numeric vector or a numeric matrix, data frame, tibble object.

left the smallest value of the the interval. If **x** has **n** columns, then **left** is expected to be of length **n**. However, if it is shorter, it will be repeated to reach that length.

right the largest value of the the interval. If **x** has **n** columns, then **right** is expected to be of length **n**. However, if it is shorter, it will be repeated to reach that length.

reverse whether to assign values in a reverse way. Default is FALSE. If **x** has **n** columns, then **reverse** is expected to be of length **n**. However, if it is shorter, it will be repeated to reach that length.

xmin the min value. Default is NULL, which means use the min value of **x**. However, sometimes the min value of **x** may not be the true min value. Suppose the two scores of a 100-point test are 59, 87, then the true min score is 0 and the true max score is 100. Thus you must add **xmin = 0, xmax = 100**. If **reverse = TRUE** (that is, 0 is better than 100), also add **xmin = 0, xmax = 100**.

xmax the same meaning as **xmin**, but for max value.

na.rm used by min and max. Default is FALSE.

Examples

```

y=scale_free(c(-1, 0, 2))
y=scale_free(c(-1, 0, 2), rev=TRUE)
#
# x is a data frame.
x=data.frame(
  c(-1, 0, 0, 0, 2), c(-1, 0, 0, 0, 2),
  c(-2, 0, 2, 4, 6), c(-2, 0, 2, 4, 6)
)
y=scale_free(x,
  left=0, right=10,
  reverse=c(FALSE, TRUE, FALSE, TRUE)
)
y=scale_free(x,
  left=c(0, 0, 100, 100), right=c(10, 100, 200, 200),
  reverse=c(FALSE, TRUE, FALSE, TRUE)
)

```

showcolor

Show a Color Palette

Description

Simple function to show colors. NOTE: do not add coord_flip().

Usage

```
showcolor(x, label_size = 15, ...)
```

Arguments

x	a character vector of colors.
label_size	size of text on x-axis to show color names.
...	other arguments passed to geom_bar.

Examples

```

# A palette used by David Hockney
co=c("#833822", "#C03800", "#D3454C",
     "#DC6A30", "#F29856", "#FEEF70",
     "#A5D56D", "#16D670", "#00932F",
     "#03592E", "#04B7B0", "#007BA9",
     "#EC46BF", "#6A2C8F"
)
showcolor(co, label_size=10)

```


Description

This is a convenient function to generate points with x and y coordinates (which form a 2-column data.frame). It is much like `expand.grid`. The points generated by `expand.grid` always in this "s" order: the bottom line, from left to right, and the second line, from left to right. However, `spathxy` allows you choose the order you want. See examples.

Usage

```
spathxy(x, y, first = "right", second = "top", change_line = FALSE,
        stringsAsFactors = TRUE)
```

Arguments

<code>x</code>	a vector of values to be paired with <code>y</code> .
<code>y</code>	a vector of values to be paired with <code>x</code> .
<code>first</code>	the first direction. It may be one of "right", "left", "top", "bottom". Default is "right".
<code>second</code>	the second direction. It may be one of "right", "left", "top", "bottom". Default is "top".
<code>change_line</code>	tail-to-tail or tail-to-head. Default is <code>FALSE</code> which means tail-to-tail. See examples.
<code>stringsAsFactors</code>	to be passed to <code>data.frame</code> .

Value

always a 3-column data frame. Column `x` and `y` are coordinates of points; column `index` contains the index number of points.

Examples

```
library(ggplot2)
#
# dat1 is generated by expand.grid
# Note the difference between dat1 and dat2.
# dat3 is the same as dat1.
dat1=expand.grid(1: 3, 1: 7)
colnames(dat1)=c("x", "y")
dat2=spathxy(1: 3, 1: 7,
change_line=FALSE, first="right", second="top")
dat3=spathxy(1: 3, 1: 7,
change_line=TRUE, first="right", second="top")
#
```

```

mycolor=rainbow(nrow(dat1), end=0.6)
ggplot(dat1)+geom_path(aes(x, y), color=mycolor, size=3)
ggplot(dat2)+geom_path(aes(x, y), color=mycolor, size=3)
ggplot(dat3)+geom_path(aes(x, y), color=mycolor, size=3)

```

stretchxy

*Stretching Transformation***Description**

A2 (output) is the result of enlarging (or shrinking) A1 (input) in x dimension and y dimension. Note: the two shapes manifest enlarging or shrinking effect only when `ggplot2::coord_fixed()` is used.

Usage

```

stretchxy(x, xlarge = 2, ylarge = 2, f = NULL, group = TRUE,
         todf = TRUE, checks = TRUE)

```

Arguments

x	the input. It can be a data frame, matrix, tibble object, or a list of these kinds of objects. Each object must have exactly 2 columns and must be numeric without NA.
xlarge	the enlarging extent in x dimension. If it is smaller than 1, the shape will be shrinking.
ylarge	the enlarging extent in y dimension. If it is smaller than 1, the shape will be shrinking.
f	argument passed to <code>split</code> to divide a data frame into a list of data frames. It should be a vector whose length is equal to the number of rows of x (if x is a data frame).
group	default is TRUE. It indicates whether to add a 3rd column named "d" to label the group number of each group of points. It is useful when using <code>aes(...group=g)</code> with 'ggplot2'.
todf	default is TRUE. It indicates whether to combine the output (a list) into a data frame.
checks	default is TRUE. It indicates whether to check input validity. Do not turn it off unless you are sure that the input is OK.

Value

if `todf = TRUE`, the output will be a data frame with coordinates of possibly several polygons, otherwise, it will be a list of data frames. Data frames have 2 columns named "x" and "y", and if `group = TRUE`, a third column named "g" is added indicating group numbers.

Examples

```

library(ggplot2)
dat1=data.frame(x=c(0, 1, 1), y=c(0, 0, 1))
dat2=data.frame(x=c(4, 5, 5, 4), y=c(0, 0, 3, 3))
dat3=stretchxy(list(dat1, dat2), xlarge=3, ylarge=c(3, 2), todf=TRUE)
ggplot()+coord_fixed()+
geom_polygon(data=dat1, aes(x, y), fill="red", alpha=0.3)+
geom_polygon(data=dat2, aes(x, y), fill="blue", alpha=0.3)+
geom_polygon(data=dat3, aes(x, y, fill=g, group=g), fill="blue", alpha=0.3)

```

sunshinexy

*Generating Lines Which Link One Points to Many***Description**

Suppose there is a middle point a, this function simultaneous generates points on lines that start from a to other points.

Usage

```

sunshinexy(x = 0, y = 0, outer = data.frame(1, 1), n = 10,
  delete_n = 0, distance = FALSE, checks = TRUE)

```

Arguments

x	the x coordinate of the middle points. It should be of length 1.
y	the y coordinate of the middle points. It should be of length 1.
outer	the other points. It can be a data frame, It must have exactly 2 columns and must be numeric without NA.
n	default is 10. The number of points per line.
delete_n	default is 0. The number of points to be deleted. Suppose a line has p1, p2, p3, p4, p5 points on it with p1 as the starting point. if delete_n is 2, then p1 and p2 will be deleted. Note: n -delete_n must be larger than 1.
distance	default is FALSE. If it is TRUE, a column named "distance" is added which indicates the distances from the middle point to other points.
checks	default is TRUE. It indicates whether to check input validity. Do not turn it off unless you are sure that the input is OK.

Value

A data frame that has 3 columns. The first and second columns are named "x" and "y", the third column is named "g" indicating group numbers. If distance = TRUE, a fourth column is added which indicates the distances from the middle point to other points.

Examples

```
library(ggplot2)
p=c(1, 1, 0, -1, -1, -1, 0, 1)
q=c(0, 1, 1, 1, 0, -1, -1, -1)
pq=data.frame(cbind(p, q))
dat=sunshinexy(outer=pq, n=20, delete_n=5, distance=TRUE)
ggplot()+coord_fixed()+theme_void()+
geom_point(data=pq, aes(p, q), size=4)+
geom_line(show.legend=FALSE, data=dat, aes(x, y, group=g, color=distance), size=2)+
scale_color_continuous(low="blue", high="red")
```

textgif

Simple Text ".gif" File

Description

This is a wrapper of functions in package `ggfittext` and `magick`. The output is a ".gif" with changing texts and colors. Characters are automatically enlarged or shrunk.

Usage

```
textgif(text, text_color = NULL, bg_color = NULL, reflow = FALSE,
width = 200, height = 100, family = "SimHei", fontface = 1,
fps = 2, output = NULL, ...)
```

Arguments

<code>text</code>	must be a character vector.
<code>text_color</code>	colors of the texts. Its length must be the same as that of <code>text</code> .
<code>bg_color</code>	background color of the texts. It should have the same length as <code>text</code> .
<code>reflow</code>	default is <code>FALSE</code> . If it is <code>TRUE</code> , <code>ggfittext::geom_fit_text</code> will automatically separate characters into several lines. However, you can separate lines manually by using line break.
<code>width</code>	the width of the final gif object. Default is 200. NOTE: how texts are adjusted in the text box depends on the values of width and height.
<code>height</code>	the height of the final gif object. Default is 100.
<code>family</code>	default is "SimHei" so that Chinese characters can be shown. However, some computers may not be able to use this family. And, this family ignores <code>fontface</code> . For Latin words, the built-in families are "serif", "sans" and "mono", and more can be found by typing "?Hershey".
<code>fontface</code>	1 (default) for plain, 2 for bold, 3 for italic, 4 for bold italic.
<code>fps</code>	the larger the faster. It should be a factor of 100, say, 2 (default), 4, 5, 10, rather than 3, 6, 7.
<code>output</code>	if it is <code>NULL</code> (default), an object is created. Otherwise, object will not only be created but also be saved with a file name (".gif") represented by this argument.
<code>...</code>	extra arguments used by <code>ggfittext::geom_fit_text</code> , e. g., <code>angle</code> (0 to 360), <code>lineheight</code> .

Examples

```
mytext=c("AAA", "BBB", "CCC")
color1=c("orange", "red", "white")
color2=c("black", "blue", "green")
g1=textgif(mytext, text_color=color1, bg_color=color2,
width=180, height=120, fps=2, family="serif")
```

Index

ABCxy, [2](#)
add_slash_n, [3](#)
annotation_shading_polygon, [4](#)
annotation_transparent_text, [6](#)
ANYxy, [9](#)

count_each_column, [10](#)

ellipsexy, [11](#)
enlarge_raster, [13](#)

geom_circle_cm, [14](#)
geom_ellipse_cm, [15](#)
geom_rect_cm, [16](#)
get_gg_label, [17](#)
gg_shading_bar, [18](#)

rectxy, [19](#)
resize_to_standard, [21](#)
rotatexy, [21](#)
round_text, [22](#)

scale_free, [23](#)
showcolor, [24](#)
spathxy, [25](#)
stretchxy, [26](#)
sunshinexy, [27](#)

textgif, [28](#)