Package ‘plothelper’

October 14, 2022

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>New Plots Based on 'ggplot2' and Functions to Create Regular Shapes</td>
</tr>
<tr>
<td>Version</td>
<td>0.1.9</td>
</tr>
<tr>
<td>Date</td>
<td>2020-05-08</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Jiang Wu <a href="mailto:textidea@sina.com">textidea@sina.com</a></td>
</tr>
<tr>
<td>Description</td>
<td>An extension to 'ggplot2' and 'magick'. It contains three groups of functions: Functions in the first group draw 'ggplot2' - based plots: geom_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. Functions in the third group are 'magick' - based functions facilitating image processing.</td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://github.com/githubwwwjjj/plothelper/blob/master/README.md">https://github.com/githubwwwjjj/plothelper/blob/master/README.md</a></td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.6.0), ggplot2 (&gt;= 3.3.0)</td>
</tr>
<tr>
<td>Imports</td>
<td>plyr, ggfittext (&gt;= 0.8.1), magick, grid, gridExtra, scales, farver</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>true</td>
</tr>
<tr>
<td>LazyData</td>
<td>true</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.1.0</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>no</td>
</tr>
<tr>
<td>Author</td>
<td>Jiang Wu [aut, cre] (from Capital Normal University)</td>
</tr>
<tr>
<td>Repository</td>
<td>CRAN</td>
</tr>
<tr>
<td>Date/Publication</td>
<td>2020-05-08 08:40:10 UTC</td>
</tr>
</tbody>
</table>
R topics documented:

ABCxy ................................................................. 3
add_slash_n .......................................................... 4
annotation_shading_polygon ..................................... 5
annotation_transparent_text .................................... 7
ANYxy ................................................................. 11
count_each_column ............................................... 12
elipsexy ............................................................... 13
enlarge_raster ...................................................... 14
geom_circle_cm ....................................................... 15
geom_ellipse_cm ...................................................... 16
geom_multi_raster ................................................ 18
geom_rect_cm ......................................................... 19
geom_shading_bar .................................................. 21
get_click_color ..................................................... 23
get_gg_label .......................................................... 23
GG_shading_bar ..................................................... 24
image_col_numeric ............................................... 26
image_crop_click .................................................. 27
image_keep_color .................................................. 28
image_locator ....................................................... 28
image_modify_hsv .................................................. 29
image_modify_local ............................................... 31
image_modify_local2 ............................................. 32
image_modify_rgb ............................................... 32
image_modify_rgb_v .............................................. 34
image_transparent_inverse .................................... 35
raster_alpha ........................................................ 35
rectxy ................................................................. 36
resize_to_standard ................................................ 37
rotatexy .............................................................. 38
round_text .......................................................... 39
scale_free ............................................................ 40
shading_raster ..................................................... 41
showcolor ............................................................ 42
spathxy ............................................................... 43
stretchxy ............................................................. 44
sunshinexy .......................................................... 45
textgif ................................................................. 46

Index 48
**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**

```r
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. If it has more than 2 columns, only the first 2 columns will be used.
- `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 `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 "g" 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

```r
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)
```

add_slash_n

Adding Slash_n inside Strings

Description

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

Usage

```r
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

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

```r
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 `magick::image_read`.
interpolate  the interpolate argument used by ggplot2::annotation_raster when the
gerater argument is a matrix or raster.

result_interpolate whether to interpolate in the final result which is essentially an output of ggplot2::annotation_raster.
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 magick::image_trim 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 func-
tion only create an image.

width the width which will be passed to magick::image_graph. 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 magick::image_graph. DO SEE Details
below to see how to use this parameter.

res resolution in pixels which will be passed to magick::image_graph. 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))
**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**

```r
annotation_transparent_text(
    label,
    xmin,
    xmax,
)```

```r
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

```r
ymin,
ymax,
bg = "black",
alpha = 0.5,
operator = "out",
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 `magick::image_read`. Default is color black.
- **alpha** it is only used when `bg` is a character vector. Default is 0.5.
- **operator** the argument used by `magick::image_composite`. It should be "out" (default) or "in". The former makes the texts transparent, the latter creates shading texts.
- **interpolate** when `bg` is a matrix, a image or a raster, this parameter is used and will be passed to `ggplot2::annotation_raster` 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 `ggfittext::geom_fit_text`. Default is FALSE.

place position adjustment used by `ggfittext::geom_fit_text`. The value is one of "center", "middle" (= "center"), "topleft", "top", "topright", "right", "bottomright", "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 label_trim a value which will be passed to `magick::image_trim`. However, most of the time you do not need this parameter.

bg_trim whether to trim bg. Most of the time we do want to trim it. However, the `magick::image_trim` function sometimes trims wrongly. So you can turn it off. NOTE: the default value of bg_trim 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 `magick::image_graph`. 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 `magick::image_graph`. 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 `magick::image_graph`. Default is 72.

... arguments which will be passed to `ggfittext::geom_fit_text`. Most often used are angle (0 to 360), lineheight.

Details

height can be used in the following ways:

- (1) an integer which will be directly passed to `magick::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 (ymax-ymin)/(xmax-xmin).
- (4) `height = "image"`, the width and height will be the width and height of bg when the latter 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 = (ymax-ymin)/(xmax-xmin). 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 ellipsexy 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))
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

```r
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 `ellipsexy(n=4, fan=FALSE)`, as the first and 4th points are so close. Similarly, to draw a rectangle, use `ellipsexy(n=5, fan=FALSE)`. 
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 "g" 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

```r
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)
```

describe_raster

Enlarge a Color Matrix

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, space = "rgb")
Arguments

- **x**
  a color matrix or raster. It should have at least 1 row and 1 column with no NAs.

- **n**
  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 c(10, 10).

- **row_first**
  enlarge rows first or enlarge columns first? Default is TRUE. The results are almost the same, so you do not need to change this.

- **space**
  the space parameter used by colorRampPalette. It can be "rgb" (default) or "Lab".

Examples

```r
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), space="Lab")
ggplot()+xlim(0, 10)+ylim(0, 5)+coord_fixed()+
  annotation_raster(mm,
    xmin=0, xmax=10, ymin=0, ymax=5, interpolate=TRUE)
```

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

```r
gem_circle_cm(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  linetype = NULL,
  ...
)
```
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().
- **linetype**: should always be NULL. because it will not be used.
- **...**: 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

```r
library(ggplot2)
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**

*Geom Layer for Ellipse with Absolute Size*

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

```r
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 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**

```r
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))
```

---

**Description**

Unlike `annotation_raster` which draws only 1 raster, this layer draws one or more rasters at the same time. The data must be a tbl object created by package tibble and the reason is that, as we must give each rectangle a vector of colors, the column that contains these vectors of colors must be a list rather than a vector. A list can be a column for tbl object, not for a normal data frame. See examples. Accepted properties are:

- (1) `xmin`.
- (2) `xmax`.
- (3) `ymin`.
- (4) `ymax`.
- (5) `raster`. a list with 1 or more rasters. If you have only 1 raster, you also have to put it into a list. Each raster should be a matrix, a raster object, a character vector or a magick object read into R by `magick::image_read`. You can also use a data frame created by package tibble to combine `xmin`, `xmax`, `ymin`, `ymax`, `raster`.
- (6) `interpolate`. It is the same as that in `annotation_raster` except that the default value is TRUE. It can be used either inside or outside the `aes(...)` function. Its length must be either 1 or the same as the number of rasters.
- (7) `flip`. The default is FALSE. You only need to use TRUE when you use `coord_flip`. Used outside the `aes(...)` function.

**Usage**

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

- **mapping**: aes mapping.
- **data**: data. It should be a tbl object.
- **stat**: stat.
- **position**: position.
- **na.rm**: logical, whether to remove NA values.
- **show.legend**: This will not be used because the layer does not create any legend.
- **inherit.aes**: logical, whether to inherit aes from ggplot().
- **flip**: see description.
- **...**: additional parameters.

Examples

```r
# Example 1: use vectors and a list.
mycolor=list(
c1=matrix(c("red", "blue", "green", "yellow"), nrow=2),
c2=matrix(c("green", "yellow")),
c3=matrix(c("purple", "red")))
xmin=1: 3
xmax=(1: 3)+0.8
ymin=c(0, 1, 2)
ymax=c(1, 3, 5)
ggplot()+
geom_multi_raster(aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax, raster=mycolor))
```

```r
# Example 2: the same as example 1
# except flip=TRUE.
ggplot()+coord_flip()+
geom_multi_raster(aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax, raster=mycolor), flip=TRUE)
```

---

**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**

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

Arguments

mapping    aes mapping.
data        data.
stat        stat.
position    position.
a.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")
**Description**

This function is similar to `geom_bar(aes(x, y), stat=“identity”)` except that it draws bars with shading colors. Unlike `gg_shading_bar` which is a convenient function, this function is used as a ggplot layer. Accepted properties are different from those in `geom_multi_raster` and `gg_shading_bar`.

- (1) x. It is the same as that in `geom_bar`.
- (2) y. It is the same as that in `geom_bar`.
- (3) raster. It should be a list with 1 or more character vectors of colors. If the list only has 1 vector, all the bars will use the same shading pattern. If you have, for example, 5 bars to draw, then you have to put 5 vectors of colors into a list. If you use a data frame, it must be a data frame made by package tibble, and the column for `raster` should be a list.
- (4) width. It is the same as that in `geom_bar`.
- (5) flip. The default is FALSE. You only need to use TRUE when you use `coord_flip`. Use outside the `aes(...)` function.
- (6) 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. It is the same as that in `gg_shading_bar`.
- (7) equal_scale. The default is FALSE. When it is TRUE, a bar will use a certain part of the shading colors according to a global scale. It is the same as that in `gg_shading_bar`.
- (8) smooth. The default is 15. The number of shading colors each bar has. The bigger, the better. It is the same as that in `gg_shading_bar`.
- (9) space. The color space that is used. It can be "rgb" (default) or "Lab".
- (10) orientation. This parameter mimics the same parameter used in `geom_bar`, though acts differently. This enables to flip the x axis and y axis without using `coord_flip`. If it is NA or "x" (default), it supposes x = SOME LABELS and y = SOME VALUES. If it is "y", you must set x = SOME VALUES and y = SOME LABELS. These effects are the same as `geom_bar`.

NOTE: the function does interpolation as default, so you does not need to use `interpolate` parameter. And, unlike `gg_shading_bar`, this function does not draw lines around rectangles.

**Usage**

```r
geom_shading_bar(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
)```
width = 0.9,  
flip = FALSE,  
modify_raster = TRUE,  
smooth = 15,  
equal_scale = FALSE,  
space = "rgb",  
orientation = "x",  
...  
)

Arguments

mapping aes mapping.  
data data. It should be a tbl object.  
stat stat.  
position position. The parameter will not be used here.  
na.rm logical, whether to remove NA values.  
show.legend This will not be used because the layer does not create any legend.  
inherit.aes logical, whether to inherit aes from ggplot().  
width see description.  
flip see description.  
modify_raster see description or gg_shading_bar.  
smooth see description.  
equal_scale see description or gg_shading_bar.  
space see description.  
orientation see description.  
... additional parameters.

Examples

# Example 1: use vectors.  
x=c("b", "a", "c", "d", "e")  
y=c(2, 1, 3, 5, 4)  
raster=list(c("blue", "red"), c("green", "orange"),  
c("cyan", "yellow"), c("purple", "orangered"), c("grey", "red"))  
ggplot()+  
gg_shading_bar(aes(x=x, y=y, raster=raster), smooth=40)  
#  
# Example 2: other parameters  
x=1: 5  
y=c(1, 2, -3, 5, 4)  
raster=list(c("blue", "red"))  
ggplot()+  
gg_shading_bar(aes(x=x, y=y, raster=raster),  
smooth=50, width=0.6, equal_scale=TRUE)+  
scale_x_continuous(breaks=1: 5, labels=letters[1: 5])
get_click_color  

Obtaining the Colors of Positions Clicked

Description
The function draws an image and asks the user to click on the positions whose colors the user wants to know. NOTE: after clicking, you must press Esc button to continue. The result is a vector of colors in hex mode.

Usage
get_click_color(x)

Arguments

x a raster object, or an image loaded by magick::image_read or the filename of that image.

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. It can also be a gg object.

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::expansion`. It is only used when `a` is numeric or `v` is non-NULL.

**add**  
Default is 0. It mimics the `add` argument of `ggplot2::expansion`.

**axis**  
If `gg` is used or `a` is a `ggplot` object, which axis will be checked? It can be "x" or "y" (default).

### Examples

```r
get_gg_label(a=1, b=1000)
# 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=expansion(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

```r
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,
  space = "rgb",
  ...
)
```
Arguments

v a vector of item frequencies. Negative values are OK.
lables a vector of item names. Its length should be equal to that of v. If it is NULL, default names will be used. If it is of class numeric or factor, it will be transformed to a character vector.
raster a list. The length of the list should be equal to that of v. Each element of the list should be a color vector corresponding to a value in v. If it is a vector, it will be automatically transformed to a list. If its length is 1, but the length of v is, say, 3, then it will be automatically repeated for 3 times. Let us suppose v = 5 and raster = list(c("green", "red")), This means the starting side of the bar is green and the far side is red. See examples.

flip default is FALSE and the bars are vertical. When it is TRUE, the bars are horizontal. Note: when using this function, DO NOT USE ggplot2::coord_flip !

change_order when it is "normal" (default), the drawing order is the order of v. 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 v will be used.

equal_scale default is FALSE. When it is TRUE, a bar will use a certain part of the shading colors according to a global scale. See examples.

smooth 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.

space the space parameter used by colorRampPalette. It should be "rgb" (default) or "Lab".

... additional arguments used by ggplot2::coord_flip when flip = TRUE.

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
# (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)

---

## image_col_numeric

### Colorize an Image according to Gray Scale

**Description**

A color image can be converted to one with different degrees of gray. Then, colors in a palette can be added according to the gray degrees. The function is a simple wrapper of `scales::col_numeric`. The pixels which are deliberately assigned "transparent" in the original magick image will always kept unchanged.

### Usage

```r
image_col_numeric(
x, 
  palette = c("purple", "yellow"), 
  n = 256, 
  alpha = FALSE, 
  result = "magick", 
  res = 144 
)
```

### Arguments

- **x**: an image read into R by `magick::image_read`
- **palette**: two or more colors. The default is `c("purple", "yellow")` which means the deeper colors on the image will become purple and the lighter yellow.
- **n**: the max num of colors that will be used. The default is 256. Note, the number of colors that really exist may be smaller than this number.
image_crop_click  

Cut out a Subregion of an Image by Mouse Click

Description

This function is a wrapper of magick::image_crop. While the latter asks you to set a geometry parameter, this function enables you to set the four sides of a subregion only by click the mouse. You must click at least 2 times (that is, click on 2 different points to define a rectangle). After clicking, please press Esc on your keyboard. You can also designate an irregular polygon by mouse with at least 3 clicks. If it is irregular, you MUST click on positions in order (something like that, when you draw a polygon in R, you must input the positions of points in order).

Usage

image_crop_click(x, only_value = FALSE, rectangle = TRUE, trim = FALSE)

Arguments

x  
an image read into R by magick::image_read or an image modified by functions in the magick package.

only_value  
the default is FALSE, which will return the subregion. If you set it to TRUE, the result is only four values with the order: left, right, top, bottom.

rectangle  
whether the cropped area is a rectangle (default is TRUE). If it is FALSE, the subregion can be irregular.

trim  
this is only used when rectangle is FALSE. It decides whether the irregular subregion is to be trimmed. If it is FALSE (default), no trimming will be done. If it is a 0 to 100 value, magick::image_trim will be used, whose fuzz argument is equal to trim. If it is TRUE (not 1), trimming will be done according to the mouse click you have made.
image_keep_color  
Keep Some Colors Unchanged and Make Others into Grayscale

Description

This function keeps pixels with certain colors unchanged and transforms others into grayscale. The function is in fact a wrapper of magick::image_transparent, so it uses the latter’s color and fuzz parameters. NOTE: the function only works for fully opaque or fully transparent (labelled as "transparent") pixels.

Usage

image_keep_color(x, color = NULL, fuzz = 10, result = "magick")

Arguments

x an image read into R by magick::image_read.
color the same as magick::image_transparent. You can use 1 or more colors.
fuzz the same as magick::image_transparent. However, Its length must either be 1 or the same as color.
result if it is "magick" (default), the result is a magick image, if it is "raster", the result is a matrix.

image_locator  
Get the Width and Height of the Mouse Clicked Points

Description

This function simply gets the width and height values of the points on which you click. The result is a list of two vectors, the first vector is for width, the second for height.

Usage

image_locator(x, rectangle = FALSE)

Arguments

x a raster object, or an image loaded by magick::image_read or the filename of that image.
rectangle if it is FALSE (default), the result list contains the width and height values. If it is TRUE, only the left, right, top, bottom values of the rectangle designated by your clicking are returned.
image_modify_hsv

**Modify the H, S, V Values of a Color Vector or an Image**

**Description**

The function modifies the H (0 - 1), S, V values of a vector of colors or an image. The three channels can be modified separately. However, the most frequently used is only the V modification. The ways to modify include: setting values to some specified values (set_*), adding (add_*), multiplying the original values (mult_*), rescaling the original values (rescale_*), using a function to recompute values (fun_*). The most useful way is to use some internal curves that mimic those PS-like apps. DO see Details.

**Usage**

```r
image_modify_hsv(
  x,
  set_h = NULL,
  add_h = NULL,
  mult_h = NULL,
  rescale_h = NULL,
  fun_h = NULL,
  set_s = NULL,
  add_s = NULL,
  mult_s = NULL,
  rescale_s = NULL,
  fun_s = NULL,
  set_v = NULL,
  add_v = NULL,
  mult_v = NULL,
  rescale_v = NULL,
  fun_v = NULL,
  result = "magick",
  res = 144
)
```

**Arguments**

- `x`: an image created by `image_read` or other functions in package magick. Alternatively, it can be a vector of colors.
- `set_h`: set H values with specific values.
- `add_h`: add specific values to current H values.
- `mult_h`: multiply the current values with specific values.
- `rescale_h`: a length 2 numeric vector specifying the desired range of H values, e.g., `rescale_h = c(0.6, 0.95)` which will make the smallest original value to be 0.6, and the largest, 0.95. Alternatively, it can be your own scaling function.
fun_h

your own modifying function (e.g., fun_h = sqrt). Alternatively, it can be a
list that designates how to use internal curves. See Details.

set_s, add_s, mult_s, rescale_s, fun_s

parameters to change S values. Used in the same way as those for H. See above.

set_v, add_v, mult_v, rescale_v, fun_v

parameters to change V values. Used in the same way as those for H. See above.

result

the default is "magick", the output is a magick picture. When it is "raster", a ma-
trix is created which can be use as a raster for ggplot2::annotation_raster.

res when the result is a magick picture, the res parameter used by magick::image_graph.

Default is 144.

Details

fun_* can be a function or a named list which tells the function which internal function is to be
used. You must ensure values used by the function specified by you to be in the range [0, 1] for H,
S, V modification and [0, 255] for R, G, B modification. Also, you’d better make sure the output
values of the function are in

When fun_* is a list, it should be written in the following way:

• (1) fun_* = list(fun = "s", c1 = -2, c2 = 2, domain = c(0, 1)) An "s" curve will be used.
c1 points out how to deal with values below 0.5, c2 with values above 0.5. For c1 and c2,
a value larger than 0 means a curvature towards y = 1, and a value smaller than 0 means a
curvature towards y = 0. So, c1 < 0 and c2 > 0 will make an s shape curve. c1 and c2 can be
any number, though those with absolute values below 4 are quite good (default is -2 and 2). 0
means no change. domain specifies the value domain to put the result. The default is c(0, 1)
which means not to rescale, thus 0.1 is 0.1. However, if you set domain = c(0.5, 1), then 0.1
will be 0.55. If you do not know how to set domain, just ignore it.

• (2) fun_* = list(fun = "circle", value = 0.5) When the fun is "circle" or "c", an arc will
be used. value must be a number between -1 and 1 (default is 0.5). A number larger than 0
means the curvature is towards y = 1, and a number smaller than 0 means it is towards y = 0.
value should not be 0.

• (3) list(fun_* = "linear", x0 = 0.4, y0 = 0.6) This makes a linear modification except
that there is a breakpoint. The default point is (0.4, 0.6) which means: suppose all the original
numbers and output numbers are in the [0, 1] range and the points with their x position smaller
than 0.4 will be put along the line that links (0, 0) and (0.4, 0.6), and, those with x position
larger than 0.4 will be put along the line that links (0.4, 0.6) and (1, 1).

Examples

# First create an image
library(magick)
mycolor=grDevices::hsv(0, s=seq(0.1, 0.9, 0.1),
v=seq(0.1, 0.9, 0.1))
img=image_graph(width=400, height=400)
print(showcolor(mycolor)+theme_void())
dev.off()
# Now increase S values with
# an internal circle curve and
# set V values between [0.5, 1].
res=image_modify_hsv(img,
fun_s=list("circle", value=1),
rescale_v=c(0.5, 1))

---

**image_modify_local**  
Modify Only a Subregion of an Image

**Description**

The function allows you to modify a subregion of your image (or, the opposite, keep the subregion unchanged while modifying other parts). You can set the four sides of the subregion or an irregular polygon by mouse click. If it is irregular, you MUST click in order.

**Usage**

```r
image_modify_local(
  x,
  FUN,
  geometry = "click",
  local = "local",
  rectangle = TRUE,
  trim = FALSE,
  ...
)
```

**Arguments**

- **x**: an image read into R by magick::image_read or an image modified by functions in the magick package.
- **FUN**: the function used to modify x. NOTE: the result of FUN must be of the same class as x and its width and height must not be changed during modification.
- **geometry**: this parameter is different from the one used in package magick. Here, in this function, you can set geometry = "click" if you want to show which part you want to modify by mouse click (see function image_crop_click for how to use mouse click). Otherwise, you can use a length 4 vector with the exact order: left, right, top, bottom.
- **local**: if it is 1 or "local", only a subregion of your image will be modified. If it is 2 or "other", keep the subregion unchanged while modifying other parts. If it is 3 or "subregion", the result is only the modified subregion, not the whole image.
- **rectangle**: if it is TRUE (default), the subregion is a rectangle. If it is FALSE, the subregion can be an irregular polygon designated by your mouse click.
- **trim**: whether to trim the subregion. This is only used when local is 3 or "subregion". It helps remove the transparent parts. See image_crop_click to know how to use this parameter.
image_modify_rgb

Modify R, G, B Values of an Image

Description

The function modifies the R, G, B values of an image and is used in the same way as image_modify_hsv in this package. The three channels can be modified separately. The ways to modify include: setting values to some specified values (set_*), adding (add_*), multiplying the original values (mult_*), rescaling the original values (rescale_*), using a function to recompute values (fun_*). The most useful way is to use some internal curves that mimic those PS-like apps.

image_modify_local2

Modify both a Subregion and the Whole of an Image

Description

The function is similar to image_modify_local but with different parameters. It modifies both a subregion of the image and the whole image, and then combines them. The subregion can be chosen either by numeric values or by mouse click, which is the same as image_modify_local.

Usage

image_modify_local2(x, FUN1, FUN2 = NULL, geometry = "click", rectangle = TRUE)

Arguments

x an image read into R by magick::image_read or an image modified by functions in the magick package.
FUN1 a function to modify a subregion of x. NOTE: the result of these functions must be of the same class as x and should not change the sizes of the subregion.
FUN2 a function to modify the whole image, which must not change the size of the image. If it is NULL (default), nothing will do to the whole image.
geometry this parameter is different from the one used in package magick. Here, in this function, you can set geometry = "click" if you want to show which part is the subregion by mouse click (see function image_crop_click for how to use mouse click). Otherwise, you can use a length 4 vector with the exact order: left, right, top, bottom.
rectangle if it is TRUE (default), the subregion is a rectangle area. If it is FALSE, the subregion is an irregular polygon area, and, now geometry is ignored, you must designate the area by mouse click.
image_modify_rgb

Usage

image_modify_rgb(
  x,
  set_r = NULL,
  add_r = NULL,
  mult_r = NULL,
  rescale_r = NULL,
  fun_r = NULL,
  set_g = NULL,
  add_g = NULL,
  mult_g = NULL,
  rescale_g = NULL,
  fun_g = NULL,
  set_b = NULL,
  add_b = NULL,
  mult_b = NULL,
  rescale_b = NULL,
  fun_b = NULL,
  result = "magick",
  res = 144
)

Arguments

x an image created by magick::image_read or other functions in package magick.
set_r set r values with specific values.
add_r add specific values to current R values.
mult_r multiply the current values with specific values.
rescale_r a length 2 numeric vector specifying the desired range of R values, e.g., rescale_r = c(180, 240) which will make the smallest original value to be 180, and the largest, 240. Alternatively, it can be your own scaling function.
fun_r your own modifying function (e.g., fun_r = sqrt). Alternatively, it can be a list that designates how to use internal curves. See image_modify_hsv.
set_g, add_g, mult_g, rescale_g, fun_g parameters to change G values. Used in the same way as those for R. See above.
set_b, add_b, mult_b, rescale_b, fun_b parameters to change B values. Used in the same way as those for R. See above.
result the default is “magick”, the output is a magick picture. When it is “raster”, a matrix is created which can be use as a raster for ggplot2::annotation_raster.
res when the result is a magick picture, the res parameter used by magick::image_graph. Default is 144.

Details

Several internal curves can be used. Please see the Details part of image_modify_hsv.
image_modify_rgb_v

Modify R, G, B Values according to V values

Description

While the `image_modify_rgb` function modifies R, G, B with reference to the original values, `image_modify_rgb_v` also takes into account the brightness (V) values. It is similar to those apps which divide an image into a bright part and a dark part (and, for example, you can increase red in the bright part and decrease red in the dark part.

Usage

```r
image_modify_rgb_v(
  x,
  fun_r = NULL,
  fun_g = NULL,
  fun_b = NULL,
  alpha = FALSE,
  rescale_v = NULL,
  result = "magick",
  res = 144
)
```

Arguments

- `x` an image created by `magick::image_read` or other functions in package `magick`.
- `fun_r, fun_g, fun_b` a function or a list which designates an internal curve. See the Details part of `image_modify_hsv`.
- `alpha` whether to allow the output colors have transparency. Default is FALSE.
- `rescale_v` You can rescale the V values before modifying colors. A desired range of V values can be given, e.g., `rescale_v = c(0, 2)` which will make the smallest original value to be 0, and the largest, 2. Alternatively, it can be your own scaling function.
- `result` the default is "magick", the output is a magick picture. When it is "raster", a matrix is created which can be use as a raster for `ggplot2::annotation_raster`.
- `res` when the result is a magick picture, the res parameter used by `magick::image_graph`. Default is 144.

Details

This function uses custom functions or internal curves to make modification. See the Details part of `image_modify_hsv` to know how to use them. Note: values will be coerced to be in the [0, 255] range with no warning. For example, the original value is 240 and it becomes 280 in the output, then it will be set to 255 automatically.
image_transparent_inverse

Keep Certain Colors Unchanged and Make Others Transparent

Description
This function is an inverse version of magick::image_transparent. While the latter makes certain colors transparent, the former keeps them unchanged and make others transparent.

Usage
image_transparent_inverse(x, color, fuzz = 0)

Arguments
x a magick image.
color one or more colors you want to keep unchanged.
fuzz color tolerance between 0 and 100. Its length must be 1 or the same as color (which means you can have different fuzz values for different colors). Suppose your color is white. If fuzz=0, then only white will be kept unchanged; if fuzz=10, colors similar to white will also be kept unchanged.

raster_alpha Combine a Matrix of Colors and a Matrix of Alpha Values

Description
The function is a wrapper of scales::alpha. While the latter only works on vectors, the former can combine a matrix of colors and a matrix of alpha values as long as the two have the same numbers of rows and columns.

Usage
raster_alpha(color, alpha, result = "raster", res = 144)

Arguments
color a matrix of colors, a raster or an image read into R by magick::image_read.
alpha either a single value (e.g., 0.4) or a matrix of alpha values. The matrix should have the same numbers of rows and columns as color.
result if it is "raster", the result will be a matrix which can be used by annotation_raster (default), if it is "magick", the result is a magick image.
res the res parameter used by magick::image_graph when result is "magick". Default is 144.
Examples

# A color matrix
colours <- c("red", "yellow", "green", "blue")
colours <- rbind(colours, colours, colours)
# An alpha matrix
alpha <- c(1, 0.6, 0.3, 0.1)
alpha <- rbind(alpha, alpha, alpha)
# Now combine the two
result <- raster_alpha(colours, alpha)

---

rectxy

Generating Coordinates of Multiple Rectangles

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 "bottom-left", x and y are the coordinates of the bottom-left corner.
- **group**: default is TRUE. It indicates whether to add a 3rd column named "g" to label the group number of each group of points. It is useful when using aes(...group=g) with 'ggplot2'.

rectxy
**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**

```r
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**

```r
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 standard = c(0.5, 3), 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
rotatexy

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

```r
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. If it has more than 2 columns, only the first 2 columns will be used.
- **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 "g" 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), xmiddle=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

```r
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

```r
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, 2), c(-1, 0, 0, 2),
```
The function is a simple wrapper of `scales::col_numeric`. The function creates a matrix of colors that can be used to draw a shading rectangle. There are 2 ways to use the function, see the following parameters.

**Usage**

```r
shading_raster(
  nr = NULL,
  nc = NULL,
  middle = NULL,
  palette = c("blue", "red"),
  mat = NULL,
  FUN = NULL
)
```

**Arguments**

- `nr`: method 1 to use this function is to use `nr`, `nc`, `middle`. Suppose there is a matrix with `nr` rows and `nc` columns. A cell whose position in the matrix is designated by `middle`. Then, this cell gets the first color of `palette`, and other cells get shading colors according to their distances between them and `middle`. Method 2 to use this function is to use `mat`. The biggest cell gets the first color and other cells get shading colors.

- `nc`: see `nr`.

- `middle`: see `nr`. The parameter should be a length 2 vector designating the row number and column number of a cell.

- `palette`: two or more colors used to make shading colors.

- `mat`: see `nr`.

- `FUN`: the default NULL makes the colors distributed in a linear way. However, `FUN` can be a single parameter function which transforms the numeric values, such as `log`, `sqrt`.
Examples

# Use method 1.
r=shading_raster(nr=31, nc=60, middle=c(10, 55),
    palette=c("darkorange", "red", "purple"))
 ggplot()+xlim(0, 8)+ylim(0, 6)+
    annotation_raster(r, xmin=-Inf, xmax=Inf,
        ymin=-Inf, ymax=Inf, interpolate=TRUE)
# Use method 2.
r=matrix(c(
    1, 2, 3, 4, 5, 6, 7, 8,
    1, 2, 3, 4, 5, 6, 7, 8,
    1, 1, 1, 1, 1, 1, 1, 1),
    nrow=3, byrow=TRUE)
r=shading_raster(mat=r, palette=c("green", "blue"))

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
c0=c("#83822", "#C03800", "#D3454C",
    "#DC6A30", "#F29856", "#FEEF70",
    "#A5D56D", "#16D670", "#00932F",
    "#03592E", "#04B70", "#007BA9",
    "#EC46BF", "#6A2C8F"
)
showcolor(co, label_size=10)
Reordering Points to Form a "s" Shape

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, form 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

x  a vector of values to be paired with y.

y  a vector of values to be paired with x.

first  the first direction. It may be one of "right", "left", "top", "bottom". Default is "right".

second  the second direction. It may be one of "right", "left", "top", "bottom". Default is "top".

change_line  tail-to-tail or tail-to-head. Default is FALSE which means tail-to-tail. See examples.

stringsAsFactors  to be passed to data.frame.

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**

```r
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. If it has more than 2 columns, only the first 2 columns will be used.
- **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 `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 "g" to label the group number of each group of points. It is useful when using `aes(...group=g)` with `ggplot2`.
sunshinexy 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.
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,
textgif

output = NULL,
...
)

Arguments

text

text_color

bg_color

reflow
default is FALSE. If it is TRUE, ggfittext::geom_fit_text will automatically separate characters into several lines. However, you can separate lines manually by using line break.

width

height

color of the texts. Its length must be the same as that of text.

background color of the texts. It should have the same length as text.

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.

the height of the final gif object. Default is 100.

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 fontface. For Latin words, the built-in families are "serif", "sans" and "mono", and more can be found by typing "?Hershey".

1 (default) for plain, 2 for bold, 3 for italic, 4 for bold italic.

the larger the faster. It should be a factor of 100, say, 2 (default), 4, 5, 10, rather than 3, 6, 7.

if it is NULL (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.

extra arguments used by ggfittext::geom_fit_text, e. g., angle (0 to 360), lineheight.

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, 3
add_slash_n, 4
annotation_shading_polygon, 5
annotation_transparent_text, 7
ANYxy, 11

count_each_column, 12
ellipsexy, 13
enlarge_raster, 14
gem_circlem_cm, 15
gem_ellipses_cm, 16
gem_multi_raster, 18
gem_rect_cm, 19
gem_shading_bar, 21
get_click_color, 23
get_gg_label, 23
gg_shading_bar, 24

image_col_numeric, 26
image_crop_click, 27
image_keep_color, 28
image_locator, 28
image_modify_hsv, 29, 33, 34
image_modify_local, 31
image_modify_local2, 32
image_modify_rgb, 32, 34
image_modify_rgb_v, 34
image_transparent_inverse, 35

raster_alpha, 35
rectxy, 36
resize_to_standard, 37
rotatexy, 38
round_text, 39

scale_free, 40
shading_raster, 41
showcolor, 42
spathxy, 43

stretchxy, 44
sunshinexy, 45
textgif, 46