

Package: patternplot (via r-universe)

September 3, 2024

Type Package

Title Versatile Pie Charts, Ring Charts, Bar Charts and Box Plots
using Patterns, Colors and Images

Version 1.0.0

Maintainer Chunqiao Luo <chunqiaoluo@gmail.com>

Imports Rcpp (>= 0.11.0), R6 (>= 2.1.2), ggplot2 (>= 2.1.0), jpeg (>= 0.1-8), png (>= 0.1-7), grDevices, utils, RcppParallel, dplyr, gtable, gridExtra

Depends R (>= 3.5)

Suggests knitr, Cairo, RCurl, rmarkdown

Date 2020-04-15

License GPL

Description Creates aesthetically pleasing and informative pie charts, ring charts, bar charts and box plots with colors, patterns, and images.

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

RoxygenNote 7.1.0

NeedsCompilation yes

VignetteBuilder knitr

Author Chunqiao Luo [aut, cre], Shasha Bai [aut]

Encoding UTF-8

Date/Publication 2020-04-21 12:20:09 UTC

Repository <https://chunqiaoluo.r-universe.dev>

RemoteUrl <https://github.com/cran/patternplot>

RemoteRef HEAD

RemoteSha b6789041171a7e2ba0ac76d73c4b6a508519d304

Contents

imagebar	2
imagebar_s	4
imageboxplot	6
imagepie	8
imagering1	10
imagerings2	12
pattern	15
patternbar	16
patternbar_s	18
patternboxplot	20
patternpie	23
patternring1	25
patternrings2	27

Index	32
--------------	-----------

imagebar	<i>Plot a bar chart with bars filled with png and jpeg images.</i>
----------	--

Description

The imagebar function is a tool for creating versatile bar charts by filling the bars with external png and jpeg images.

Usage

```
imagebar(
  data,
  x,
  y,
  group = NULL,
  xlab = "",
  ylab = "",
  label.size = 3.5,
  vjust = -1,
  hjust = -1,
  pattern.type,
  frame.color = "black",
  frame.size = 1,
  legend.type = "h",
  legend.h = 6,
  legend.x.pos = 1.1,
  legend.y.pos = 0.49,
  legend.w = 0.2,
  legend.pixel = 0.3,
  bar.width = 0.9
)
```

Arguments

<code>data</code>	the data to be used.
<code>x</code>	the variable used on x axis.
<code>y</code>	the variable used on y axis.
<code>group</code>	the variable used as the second grouping variable on x axis.
<code>xlab</code>	a character string to give x axis label.
<code>ylab</code>	a character string to give y axis label.
<code>label.size</code>	the font size of legend labels shown above the bars.
<code>vjust</code>	the vertical distance of labels from the top border of each bar.
<code>hjust</code>	the horizontal distance of labels from the top border of each bar.
<code>pattern.type</code>	a list of objects returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the bars.
<code>frame.color</code>	the color of the borders of bars.
<code>frame.size</code>	a numeric value, the line size for the borders of bars.
<code>legend.type</code>	if <code>legend.type='h'</code> , the layout of legends is horizontal; if <code>legend.type='v'</code> , the layout of legends is vertical.
<code>legend.h</code>	a numeric value to change the height of legend boxes.
<code>legend.x.pos</code>	a numeric value to change the position of legends on x axis.
<code>legend.y.pos</code>	a numeric value to change the position of legends on y axis.
<code>legend.w</code>	a numeric value to change the width of legends.
<code>legend.pixel</code>	a numeric value to change the pixel of legend boxes
<code>bar.width</code>	a numeric value to change the width of the bars.

Details

`imagebar` function offers flexible ways of doing bar charts.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `patternbar`

Examples

```

library(patternplot)
library(jpeg)
library(ggplot2)

childcare<-readJPEG(system.file("img", "childcare.jpg", package="patternplot"))
food<-readJPEG(system.file("img", "food.jpg", package="patternplot"))
housing <-readJPEG(system.file("img", "housing.jpg", package="patternplot"))

#Example 1
data <- read.csv(system.file("extdata", "monthlyexp.csv", package="patternplot"))
data<-data[which(data$Location=='City 1'),]
x<-factor(data$Type, c('Housing', 'Food', 'Childcare'))
y<-data$Amount
pattern.type<-list(housing, food, childcare)
imagebar(data,x, y,group=NULL,pattern.type=pattern.type,vjust=-1, hjust=0.5,
frame.color='black',ylab='Monthly Expenses, Dollars')+ggtitle('(A) Bar Chart with Images')

#Example 2
data <- read.csv(system.file("extdata", "monthlyexp.csv", package="patternplot"))
group<-factor(data$Type, c('Housing', 'Food', 'Childcare'))
y<-data$Amount
x<-factor(data$Location, c('City 1', ' City 1'))
pattern.type<-list(housing, food, childcare)
g<-imagebar(data,x,y,group,pattern.type=pattern.type,vjust=-1,hjust=0.5,frame.color='black',
ylab='Monthly Expenses, Dollars')+ggtitle('(B) Image Bar Chart with Two Grouping Variables')

```

imagebar_s

Plot a stacked bar chart with bars filled with png and jpeg images.

Description

The `imagebar_s` function is a tool for creating versatile stacked bar charts by filling the bars with external png and jpeg images.

Usage

```

imagebar_s(
  data,
  x,
  y,
  group,
  xlab = "",
  ylab = "",
  pattern.type,
  label.size = 3.5,
  frame.color = "black",
  frame.size = 1,

```

```

    legend.type = "h",
    legend.h = 6,
    legend.x.pos = 1.1,
    legend.y.pos = 0.49,
    legend.w = 0.2,
    legend.pixel = 20,
    legend.label,
    bar.width = 0.9
)

```

Arguments

<code>data</code>	the data to be used.
<code>x</code>	the variable used on x axis.
<code>y</code>	the variable used on y axis.
<code>group</code>	the variable used as the second grouping variable on x axis.
<code>xlab</code>	a character string to give x axis label.
<code>ylab</code>	a character string to give y axis label.
<code>pattern.type</code>	a list of objects returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the stacked bars.
<code>label.size</code>	the font size of legend labels shown above the stacked bars.
<code>frame.color</code>	the color of the borders of the stacked bars.
<code>frame.size</code>	a numeric value, the line size for the borders of the stacked bars.
<code>legend.type</code>	if <code>legend.type='h'</code> , the layout of legends is horizontal; if <code>legend.type='v'</code> , the layout of legends is vertical.
<code>legend.h</code>	a numeric value to change the height of legend boxes.
<code>legend.x.pos</code>	a numeric value to change the position of legends on x axis.
<code>legend.y.pos</code>	a numeric value to change the position of legends on y axis.
<code>legend.w</code>	a numeric value to change the width of legends.
<code>legend.pixel</code>	a numeric value to change the pixel of legend boxes.
<code>legend.label</code>	a vector to name legend labels.
<code>bar.width</code>	a numeric value to change the width of the bars.

Details

`imagebar_s` function offers flexible ways of doing stacked bar charts.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `patternbar_s`

Examples

```
library(patternplot)
library(jpeg)
library(ggplot2)

childcare<-readJPEG(system.file("img", "childcare.jpg", package="patternplot"))
food<-readJPEG(system.file("img", "food.jpg", package="patternplot"))
housing <-readJPEG(system.file("img", "housing.jpg", package="patternplot"))

data <- read.csv(system.file("extdata", "monthlyexp.csv", package="patternplot"))
x<-data$Location
y<-data$Amount
group<-data$Type
pattern.type<-list(childcare, food, housing)

imagebar_s(data,x,y,group,xlab='',ylab='Monthly Expenses, Dollar',pattern.type=pattern.type,
label.size=3.5,frame.size=1,frame.color='black',legend.type='h',legend.h=6,legend.y.pos=0.49,
legend.pixel=20, legend.w=0.2,legend.x.pos=1.1,
legend.label=c("Childcare", "Food", "Housing"))+scale_y_continuous(limits = c(0, 6800))
```

imageboxplot

Plot a boxplot with boxes filled with png and jpeg images.

Description

The `imageboxplot` function is a tool for creating versatile boxplots by filling the boxplots with external png and jpeg images.

Usage

```
imageboxplot(
  data,
  x,
  y,
  group = NULL,
  xlab = "",
  ylab = "",
  label.size = 3.5,
  pattern.type,
  frame.color = "black",
  linetype = "solid",
```

```

    frame.size = 1,
    outlier.shape = 21,
    outlier.color = "black",
    outlier.size = 1,
    legend.type = "h",
    legend.h = 6,
    legend.x.pos = 1.1,
    legend.y.pos = 0.49,
    legend.w = 0.2,
    legend.pixel = 0.3,
    legend.label
)

```

Arguments

data	the data to be used.
x	the variable used on x axis.
y	the variable used on y axis.
group	the variable used as the second grouping variable on x axis.
xlab	a character string to give x axis label.
ylab	a character string to give y axis label.
label.size	the font size of legend labels.
pattern.type	a list of objects returned by readPNG and readJPEG used to fill boxplots.
frame.color	the color for the borders of boxplots.
linetype	the linetype for the borders of boxplots.
frame.size	a numeric value, the line size for the borders of boxplots.
outlier.shape	the shape of outlier dots.
outlier.color	the color of outlier dots.
outlier.size	the size of outlier dots.
legend.type	if legend.type='h', the layout of legends is horizontal; if legend.type='v', the layout of legends is vertical.
legend.h	a numeric value to fine-tune the width of legend boxes on y axis.
legend.x.pos	a numeric value to change the position of legend text on x axis.
legend.y.pos	a numeric value to change the position of legend text on y axis.
legend.w	a numeric value to change the width of legends.
legend.pixel	a numeric value to change the pixel of legend boxes.
legend.label	a vector to name legend labels.

Details

imageboxplot function offers flexible ways of doing boxplots.

Value

A ggplot object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `patternboxplot`

Examples

```
library(patternplot)
library(jpeg)
library(ggplot2)

Orange<-readJPEG(system.file("img", "oranges.jpg", package="patternplot"))
Strawberry <-readJPEG(system.file("img", "strawberries.jpg", package="patternplot"))
Watermelon<-readJPEG(system.file("img", "watermelons.jpg", package="patternplot"))

#Example 1
data <- read.csv(system.file("extdata", "fruits.csv", package="patternplot"))
x<-data$Fruit
y<-data$Weight
group<-data$Store
pattern.type<-list(Orange, Strawberry, Watermelon)
imageboxplot(data,x,y,group=NULL,pattern.type=pattern.type,
frame.color=c('orange', 'darkred', 'darkgreen'),
legend.label="", ylab='Weight, Pounds')
```

imagepie

Plot a pie chart with slices filled with png and jpeg images.

Description

The `imagepie` function is a tool for creating versatile pie charts by filling the slices with external png and jpeg images.

Usage

```
imagepie(
  group,
  pct,
  label,
  label.size = 4,
  label.color = "black",
  label.distance = 1.35,
```



```

    pattern.type,
    frame.color = "black",
    frame.size = 1
  )

```

Arguments

group	a vector of strings, containing the names of each slice.
pct	a vector of non-negative numbers, containing percentages of each group. The numbers must sum up to 100.
label	a vector of strings, giving the names for the slices shown in the pie chart.
label.size	the font size of labels shown in the pie chart.
label.color	the color of labels shown in the pie chart.
label.distance	the distance of labels from the border of the pie chart.
pattern.type	a list of objects returned by readPNG and readJPEG used to fill slices.
frame.color	the color for the borders of slices.
frame.size	a numeric value, the line size for the borders of slices.

Details

imagepie function offers flexible ways of doing pie charts.

Value

A ggplot object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `patternpie`

Examples

```

library(patternplot)
library(jpeg)
library(ggplot2)
Tomatoes <- readJPEG(system.file("img", "tomatoes.jpg", package="patternplot"))
Peas <- readJPEG(system.file("img", "peas.jpg", package="patternplot"))
Potatoes <- readJPEG(system.file("img", "potatoes.jpg", package="patternplot"))

#Example 1
data <- read.csv(system.file("extdata", "vegetables.csv", package="patternplot"))
pattern.type<-list(Tomatoes,Peas,Potatoes)
imagepie(group=data$group,pct=data$pct,label=data$label,pattern.type=pattern.type,
         label.distance=1.25,frame.color='burlywood4', frame.size=0.8, label.size=6,

```

```
label.color='forestgreen')
```

imagering1

Plot a ring chart using images to fill the ring.

Description

The `imagering1` function is a tool for creating versatile ring charts by filling the ring with images.

Usage

```
imagering1(
  group1,
  pct1,
  pattern.type1,
  pattern.type.inner,
  frame.color = "black",
  frame.size = 1,
  label1,
  label.size1 = 4,
  label.color1 = "black",
  label.distance1 = 1.2,
  r1,
  r2
)
```

Arguments

<code>group1</code>	a vector of strings, containing the names of each slice.
<code>pct1</code>	a vector of non-negative numbers, containing percentages of each group. The numbers must sum up to 100.
<code>pattern.type1</code>	a list of objects returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the ring.
<code>pattern.type.inner</code>	an object returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the inner circle.
<code>frame.color</code>	the color for the borders of the ring.
<code>frame.size</code>	a numeric value, the line size for the borders of the ring.
<code>label1</code>	a vector of strings, giving the names for the slices shown in the ring chart.
<code>label.size1</code>	the font size of labels shown in the ring chart.
<code>label.color1</code>	the color of labels shown in the ring chart.
<code>label.distance1</code>	the distance of labels from the border of the ring chart.
<code>r1</code>	a numeric value, the inner radius of the ring.
<code>r2</code>	a numeric value, the outer radius of the ring.

Details

imaging1 function offers flexible ways of doing ring charts.

Value

A ggplot object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function patternring1

Examples

```
#Example 1
library(patternplot)
library(png)
library(ggplot2)
location<-gsub('\\', '/', tempdir(), fixed=TRUE)
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="darkgreen", pixel=6, res=4)
FarWest<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="darkcyan", pixel=6, res=4)
GreatLakes<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="chocolate", pixel=6, res=4)
Mideast<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="cadetblue1", pixel=6, res=4)
NewEngland<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="darkorchid", pixel=6, res=4)
Plains<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="yellowgreen", pixel=6, res=4)
RockyMountain<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="hotpink", pixel=6, res=4)
Southeast<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="lightslateblue", pixel=6, res=4)
Southwest <-readPNG(paste(location, '/', "blank", ".png", sep=''))

group1<-c('New_England', 'Great_Lakes', 'Plains', 'Rocky_Mountain', 'Far_West',
'Southwest', 'Southeast', 'Mideast')
pct1<-c(12, 11, 17, 15, 8, 11, 16, 10)
label1<-paste(group1, " \n ", pct1, "%", sep="")
```

```

pattern.type1<-list(NewEngland, GreatLakes,Plains, RockyMountain, FarWest,
Southwest, Southeast, Mideast)
pattern.type.inner<-readPNG(system.file("img", "USmap.png", package="patternplot"))

imagering1(group1, pct1, pattern.type1, pattern.type.inner, frame.color='black',
frame.size=1.5, r1=3, r2=4,label1, label.size1=4,label.color1='black', label.distance1=1.2)

```

imagerings2

Plot a ring chart using images to fill the rings.

Description

The `imagerings2` function is a tool for creating versatile ring charts by filling the rings with images.

Usage

```

imagerings2(
  group1,
  group2,
  pct1,
  pct2,
  label1,
  label2,
  label.size1 = 4,
  label.size2 = 4,
  label.color1 = "black",
  label.color2 = "black",
  label.distance1 = 1.2,
  label.distance2 = 1.2,
  pattern.type1,
  pattern.type2,
  pattern.type.inner,
  frame.color = "black",
  frame.size = 1,
  r1,
  r2,
  r3
)

```

Arguments

<code>group1</code>	a vector of strings, containing the names of each slice for the inner ring.
<code>group2</code>	a vector of strings, containing the names of each slice for the outer ring.

<code>pct1</code>	a vector of non-negative numbers, containing percentages of each group for the inner ring. The numbers must sum up to 100.
<code>pct2</code>	a vector of non-negative numbers, containing percentages of each group for the outer ring. The numbers must sum up to 100.
<code>label1</code>	a vector of strings, giving the names for the slices shown in the ring chart for the inner ring.
<code>label2</code>	a vector of strings, giving the names for the slices shown in the ring chart for the outer ring.
<code>label.size1</code>	the font size of labels shown in the ring chart for the inner ring.
<code>label.size2</code>	the font size of labels shown in the ring chart for the outer ring.
<code>label.color1</code>	the color of labels shown in the ring chart for the inner ring.
<code>label.color2</code>	the color of labels shown in the ring chart for the outer ring.
<code>label.distance1</code>	the distance of labels from the border of the ring chart for the inner ring.
<code>label.distance2</code>	the distance of labels from the border of the ring chart for the outer ring.
<code>pattern.type1</code>	a list of objects returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the inner ring.
<code>pattern.type2</code>	a list of objects returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the outer ring.
<code>pattern.type.inner</code>	an object returned by <code>readPNG</code> and <code>readJPEG</code> used to fill the inner circle.
<code>frame.color</code>	the color for the borders of rings.
<code>frame.size</code>	a numeric value, the line size for the borders of rings.
<code>r1</code>	a numeric value, the inner radius of the inner ring.
<code>r2</code>	a numeric value, the outer radius of the inner ring.
<code>r3</code>	a numeric value, the outer radius of the outer ring.

Details

`imagerings2` function offers flexible ways of doing ring charts.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `patternrings2`

Examples

```

#Example 1
library(patternplot)
library(png)
library(ggplot2)
group1<-c("Wind", "Hydro", "Solar", "Coal", "Natural Gas", "Oil")
pct1<-c(12, 15, 8, 22, 18, 25)
label1<-paste(group1, " \n ", pct1, "%", sep="")
location<-gsub('\\', '/', tempdir(), fixed=TRUE)
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="darkolivegreen1", pixel=18, res=12)
Wind<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="white", pixel=18, res=12)
Hydro<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="indianred", pixel=18, res=12)
Solar<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="gray81", pixel=18, res=12)
Coal<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="white", pixel=18, res=12)
NaturalGas<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="sandybrown", pixel=18, res=12)
Oil<-readPNG(paste(location, '/', "blank", ".png", sep=''))
pattern.type1<-list(Wind, Hydro, Solar, Coal, NaturalGas, Oil)

group2<-c("Renewable", "Non-Renewable")
pct2<-c(35, 65)
label2<-paste(group2, " \n ", pct2, "%", sep="")
pattern(type="grid", density=12, color='white', pattern.line.size=5,
background.color="seagreen", pixel=18, res=12)
Renewable<-readPNG(paste(location, '/', "grid", ".png", sep=''))
pattern(type="blank", density=1, color='white', pattern.line.size=1,
background.color="deepskyblue", pixel=18, res=12)
NonRenewable<-readPNG(paste(location, '/', "blank", ".png", sep=''))

pattern.type2<-list(Renewable, NonRenewable)
pattern.type.inner<-readPNG(system.file("img", "earth.png", package="patternplot"))

g<-imagerings2(group1, group2, pct1, pct2, label1, label2, label.size1=3, label.size2=3.5,
label.color1='black', label.color2='black', label.distance1=0.7, label.distance2=1.3,
pattern.type1, pattern.type2, pattern.type.inner, frame.color='skyblue', frame.size=1,
r1=2.2, r2=4.2, r3=5)
g<-g+scale_x_continuous(limits=c(-7, 7))+scale_y_continuous(limits=c(-7, 7))
g

```

pattern	<i>Generate a pattern in png format.</i>
---------	--

Description

The pattern function is a function for generating a pattern in png format.

Usage

```
pattern(
  type = "bricks",
  density = 8,
  pattern.line.size = 10,
  color = "black",
  background.color = "white",
  pixel = 5,
  res = 30
)
```

Arguments

type	pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves', 'Rsymbol_0' to 'Rsymbol_25', and unicode symbols.
density	a numeric number, the density for the lines/dots of a pattern.
pattern.line.size	a numeric value, the line size for the lines/dots of a pattern.
color	color for the lines/dots of pattern.
background.color	color to be filled in the background.
pixel	a numeric value, the pixel resolution of the pattern.
res	a numeric value, the pixel resolution of the pattern.

Details

pattern function generates a pattern in png format.

Value

A ggplot object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

patternbar

Plot a bar chart using patterns and colors to fill the bars.

Description

The patternbar function is a tool for creating versatile bar charts by filling the bars with colors and patterns.

Usage

```
patternbar(  
  data,  
  x,  
  y,  
  group = NULL,  
  xlab = "",  
  ylab = "",  
  label.size = 3.5,  
  vjust = -1,  
  hjust = -1,  
  pattern.type,  
  pattern.line.size = rep(5, ifelse(is.null(group), length(x), length(unique(group)))),  
  pattern.color = rep("black", ifelse(is.null(group), length(x),  
    length(unique(group)))),  
  background.color = rep("white", ifelse(is.null(group), length(x),  
    length(unique(group)))),  
  frame.color = rep("black", ifelse(is.null(group), length(x), length(unique(group)))),  
  frame.size = 1,  
  pixel = 20,  
  density = rep(7, ifelse(is.null(group), length(x), length(unique(group)))),  
  legend.type = "h",  
  legend.h = 6,  
  legend.x.pos = 1.1,  
  legend.y.pos = 0.49,  
  legend.w = 0.2,  
  legend.pixel = 20,  
  bar.width = 0.9  
)
```

Arguments

data	the data to be used.
x	the variable used on x axis.

<code>y</code>	the variable used on y axis.
<code>group</code>	the variable used as the second grouping variable on x axis.
<code>xlab</code>	a character string to give x axis label.
<code>ylab</code>	a character string to give y axis label.
<code>label.size</code>	the font size of legend labels shown above the bars.
<code>vjust</code>	the vertical distance of labels from the top border of each bar.
<code>hjust</code>	the horizontal distance of labels from the top border of each bar.
<code>pattern.type</code>	a vector of patterns to be filled in the bars The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.line.size</code>	a vector of numeric values, the line size for the lines/dots of patterns.
<code>pattern.color</code>	a vector of colors for the lines/dots of patterns.
<code>background.color</code>	a vector of colors to be filled in the bars.
<code>frame.color</code>	the color for the borders of bars.
<code>frame.size</code>	a numeric value, the line size for the borders of bars.
<code>pixel</code>	a numeric value, the pixel resolution of bar chart.
<code>density</code>	a numeric vector, the density for the lines/dots of patterns.
<code>legend.type</code>	if <code>legend.type='h'</code> , the layout of legends is horizontal; if <code>legend.type='v'</code> , the layout of legends is vertical.
<code>legend.h</code>	a numeric value to change the height of legend boxes.
<code>legend.x.pos</code>	a numeric value to change the position of legends on x axis.
<code>legend.y.pos</code>	a numeric value to change the position of legends on y axis.
<code>legend.w</code>	a numeric value to change the width of legends.
<code>legend.pixel</code>	a numeric value to change pixel of legends.
<code>bar.width</code>	a numeric value to change the width of the bars.

Details

`patternbar` function offers flexible ways of doing bar charts.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `imagebar`

Examples

```
#Example 1
library(patternplot)
library(png)
library(ggplot2)
data <- read.csv(system.file("extdata", "monthlyexp.csv", package="patternplot"))
data<-data[which(data$Location=='City 1'),]
x<-factor(data$Type, c('Housing', 'Food', 'Childcare'))
y<-data$Amount
pattern.type<-c('hdashes', 'blank', 'crosshatch')
pattern.color=c('black','black', 'black')
background.color=c('white','white', 'white')
density<-c(20, 20, 10)
patternbar(data,x, y,group=NULL,ylab='Monthly Expenses, Dollars', pattern.type=pattern.type,
pattern.color=pattern.color,background.color=background.color,pattern.line.size=c(5.5, 1, 4),
frame.color=rep('black', 3),density=density)+scale_y_continuous(limits = c(0, 2800))

#Example 2
pattern.color=c('black','white', 'grey20')
background.color=c('lightgreen','lightgreen', 'lightgreen')
patternbar(data,x, y,group=NULL,ylab='Monthly Expenses, Dollars', pattern.type=pattern.type,
pattern.color=pattern.color, background.color=background.color,pattern.line.size=c(5.5, 1, 4),
frame.color=rep('black', 3), density=density)+scale_y_continuous(limits = c(0, 2800))
```

patternbar_s

Plot a stacked bar chart using patterns and colors to fill the bars.

Description

The `patternbar_s` function is a tool for creating versatile stacked bar charts by filling the bars with colors and patterns.

Usage

```
patternbar_s(
  data,
  x,
  y,
  group,
  xlab = "",
  ylab = "",
  label.size = 3.5,
  pattern.type,
  pattern.line.size = rep(10, length(unique(group))),
  pattern.color = rep("black", length(unique(group))),
  background.color = rep("white", length(unique(group))),
  frame.color = "black",
```

```

    frame.size = 1,
    pixel = 20,
    density = rep(12, length(unique(group))),
    legend.type = "h",
    legend.h = 6,
    legend.x.pos = 1.1,
    legend.y.pos = 0.49,
    legend.w = 0.2,
    legend.pixel = 20,
    legend.label,
    bar.width = 0.9
)

```

Arguments

<code>data</code>	the data to be used.
<code>x</code>	the variable used on x axis.
<code>y</code>	the variable used on y axis.
<code>group</code>	the variable used as the second grouping variable on x axis.
<code>xlab</code>	a character string to give x axis label.
<code>ylab</code>	a character string to give y axis label.
<code>label.size</code>	the font size of legend labels shown above the bars.
<code>pattern.type</code>	a vector of patterns to be filled in the bars The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.line.size</code>	a vector of numeric values, the line size for the lines/dots of patterns.
<code>pattern.color</code>	a vector of colors for the lines/dots of patterns.
<code>background.color</code>	a vector of colors to be filled in the bars.
<code>frame.color</code>	the color for the borders of bars.
<code>frame.size</code>	a numeric value, the line size for the borders of bars.
<code>pixel</code>	a numeric value, the pixel resolution of bar chart.
<code>density</code>	a numeric vector, the density for the lines/dots of patterns.
<code>legend.type</code>	if <code>legend.type='h'</code> , the layout of legends is horizontal; if <code>legend.type='v'</code> , the layout of legends is vertical.
<code>legend.h</code>	a numeric value to change the height of legend boxes.
<code>legend.x.pos</code>	a numeric value to change the position of legends on x axis.
<code>legend.y.pos</code>	a numeric value to change the position of legends on y axis.
<code>legend.w</code>	a numeric value to change the width of legends.
<code>legend.pixel</code>	a numeric value to change pixel of legends.
<code>legend.label</code>	a vector to name legend labels.
<code>bar.width</code>	a numeric value to change the width of the bars.

Details

patternbar_s function offers flexible ways of doing stacked bar charts.

Value

A ggplot object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function imagebar_s

Examples

```
#Example 1
library(patternplot)
library(png)
library(ggplot2)
data <- read.csv(system.file("extdata", "monthlyexp.csv", package="patternplot"))
x<-data$Location
y<-data$Amount
group<-data$Type

patternbar_s(data,x, y, group,xlab='', ylab='Monthly Expenses, Dollar', label.size=3.5,
pattern.type=c('Unicode_\u266B', 'nwlines', 'bricks'),pattern.line.size=c(10, 5, 5),frame.size=1,
pattern.color=c('blue', 'green', 'white'),background.color=c('white', 'white', 'orange'),pixel=20,
density=c(10, 10, 10),frame.color='black', legend.type='h', legend.h=10, legend.y.pos=0.49,
legend.pixel=10, legend.w=0.25, legend.x.pos=1.1,
legend.label=c("Childcare", "Food", "Housing" ))+scale_y_continuous(limits = c(0, 6800))
```

patternboxplot

Plot a boxplot using patterns and colors to fill the boxes.

Description

The patternboxplot function is a tool for creating versatile boxplots by filling the boxplots with colors and patterns.

Usage

```
patternboxplot(
  data,
  x,
  y,
  group = NULL,
```

```

xlab = "",
ylab = "",
label.size = 3.5,
linetype = rep("solid", ifelse(is.null(group), length(unique(x)),
  length(unique(group)))),
outlier.shape = 21,
outlier.color = "black",
outlier.size = 1,
pattern.type,
pattern.color = rep("black", ifelse(is.null(group), length(unique(x)),
  length(unique(group)))),
pattern.line.size = rep(5, ifelse(is.null(group), length(x), length(unique(group)))),
background.color = rep("white", ifelse(is.null(group), length(unique(x)),
  length(unique(group)))),
frame.color = rep("black", ifelse(is.null(group), length(unique(x)),
  length(unique(group)))),
frame.size = 1,
pixel = 20,
density = rep(7, ifelse(is.null(group), length(unique(x)), length(unique(group)))),
legend.type = "h",
legend.h = 6,
legend.x.pos = 1.1,
legend.y.pos = 0.49,
legend.w = 0.2,
legend.pixel = 20,
legend.label
)

```

Arguments

<code>data</code>	the data to be used.
<code>x</code>	the variable used on x axis.
<code>y</code>	the variable used on y axis.
<code>group</code>	the variable used as the second grouping variable on x axis.
<code>xlab</code>	a character string to give x axis label.
<code>ylab</code>	a character string to give y axis label.
<code>label.size</code>	the font size of legend labels shown above the boxplots.
<code>linetype</code>	the linetype for the borders of boxplots.
<code>outlier.shape</code>	the shape of outlier dots.
<code>outlier.color</code>	the color of outlier dots.
<code>outlier.size</code>	the size of outlier dots.
<code>pattern.type</code>	a vector of patterns to be filled in the boxes The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.color</code>	a vector of colors for the lines/dots of patterns.

<code>pattern.line.size</code>	a vector of numeric values, the line size for the lines/dots of patterns.
<code>background.color</code>	a vector of colors to be filled in the boxes.
<code>frame.color</code>	the color for the borders of boxes.
<code>frame.size</code>	a numeric value, the line size for the borders of boxes.
<code>pixel</code>	a numeric value, the pixel resolution of boxplot.
<code>density</code>	a numeric vector, the density for the lines/dots of patterns.
<code>legend.type</code>	if <code>legend.type='h'</code> , the layout of legends is horizontal; if <code>legend.type='v'</code> , the layout of legends is vertical.
<code>legend.h</code>	a numeric value to change the height of legend boxes.
<code>legend.x.pos</code>	a numeric value to change the position of legends on x axis.
<code>legend.y.pos</code>	a numeric value to change the position of legends on y axis.
<code>legend.w</code>	a numeric value to change the width of legends.
<code>legend.pixel</code>	a numeric value to change pixel of legends.
<code>legend.label</code>	a vector to name legend labels.

Details

`patternboxplot` function offers flexible ways of doing boxplots.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `imageboxplot`

Examples

```
library(patternplot)
library(png)
library(ggplot2)

#Example 1
data <- read.csv(system.file("extdata", "fruits.csv", package="patternplot"))
group<-data$Fruit
y<-data$Weight
x<-data$Store

pattern.type<-c('nlines', 'blank', 'waves')
pattern.color=c('black','black', 'black')
```

```

background.color=c('white','gray80', 'white')
frame.color=c('black', 'black', 'black')
pattern.line.size<-c(6, 1,6)
density<-c(6, 1, 8)
patternboxplot(data,x, y,group,pattern.type=pattern.type,pattern.line.size=pattern.line.size,
pattern.color=pattern.color,background.color=background.color,frame.color=frame.color,
density=density,legend.h=2,legend.x.pos=1.1,legend.y.pos=0.495,legend.pixel=10, legend.w=0.2,
legend.label=c("Orange", "Strawberry", "Watermelon"))

#Example 2
pattern.color=c('black','white', 'grey20')
background.color=c('gold','lightpink', 'lightgreen')
patternboxplot(data,x, y,group=group,pattern.type=pattern.type,pattern.line.size=pattern.line.size,
pattern.color=pattern.color, background.color=background.color,frame.color=frame.color,
density=density,legend.h=2,legend.x.pos=1.1,legend.y.pos=0.495,legend.pixel=10,legend.w=0.2,
legend.label=c("Orange", "Strawberry", "Watermelon"))

```

patternpie

Plot a pie chart using patterns and colors to fill the slices.

Description

The patternpie function is a tool for creating versatile pie charts by filling the slices with colors and patterns.

Usage

```

patternpie(
  group,
  pct,
  label,
  label.size = 4,
  label.color = "black",
  label.distance = 1.2,
  pattern.type,
  pattern.color = rep("black", length(group)),
  pattern.line.size = rep(5, length(group)),
  background.color = rep("white", length(group)),
  frame.color = "black",
  frame.size = 1,
  density = rep(10, length(group)),
  pixel = 20
)

```

Arguments

group a vector of strings, containing the names of each slice.

<code>pct</code>	a vector of non-negative numbers, containing percentages of each group. The numbers must sum up to 100.
<code>label</code>	a vector of strings, giving the names for the slices shown in the pie chart.
<code>label.size</code>	the font size of labels shown in the pie chart.
<code>label.color</code>	the color of labels shown in the pie chart.
<code>label.distance</code>	the distance of labels from the border of the pie chart.
<code>pattern.type</code>	a vector of patterns to be filled in the slices. The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.color</code>	a vector of colors for the lines/dots of patterns.
<code>pattern.line.size</code>	a vector of numeric values, the line size for the lines/dots of patterns.
<code>background.color</code>	a vector of colors to be filled in the slices.
<code>frame.color</code>	the color for the borders of slices.
<code>frame.size</code>	a numeric value, the line size for the borders of slices.
<code>density</code>	a numeric vector, the density for the lines/dots of patterns.
<code>pixel</code>	a numeric value, the pixel resolution of pie chart.

Details

`patternpie` function offers flexible ways of doing pie charts.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `imagepie`

Examples

```
#Example 1
library(patternplot)
library(png)
library(ggplot2)
data <- read.csv(system.file("extdata", "vegetables.csv", package="patternplot"))
pattern.type<-c('hdashes', 'vdashes', 'bricks')
patternpie(group=data$group,pct=data$pct,label=data$label, label.size=4,label.color='black',
           label.distance=1.2,pattern.type=pattern.type,pattern.line.size=c(5, 5, 2),
           frame.color='black',frame.size=1.5, pixel=12, density=c(12, 12, 10))
```



```
#Example 2
pattern.color<-c('red3','green3', 'white' )
background.color<-c('dodgerblue', 'lightpink', 'orange')
patternpie(group=data$group,pct=data$pct,label=data$label, pattern.type=pattern.type,
pattern.color=pattern.color,background.color=background.color, pattern.line.size=c(5, 5, 2),
          frame.color='grey40',frame.size=1.5, pixel=12, density=c(12, 12, 10))
```

patterning1

Plot a ring chart using patterns and colors to fill the ring.

Description

The patterning1 function is a tool for creating versatile ring charts by filling the ring with colors and patterns.

Usage

```
patterning1(
  group1,
  pct1,
  label1,
  label.size1 = 4,
  label.color1 = "black",
  label.distance1 = 1.2,
  pattern.type1,
  pattern.color1,
  pattern.line.size1 = rep(10, length(group1)),
  background.color1,
  frame.color = "black",
  frame.size = 1,
  density1 = rep(10, length(group1)),
  pixel = 20,
  pattern.type.inner = "blank",
  pattern.color.inner = "white",
  pattern.line.size.inner = 1,
  background.color.inner = "white",
  pixel.inner = 10,
  density.inner = 1,
  r1,
  r2
)
```

Arguments

group1	a vector of strings, containing the names of each slice.
pct1	a vector of non-negative numbers, containing percentages of each group. The numbers must sum up to 100.

<code>label1</code>	a vector of strings, giving the names for the slices shown in the ring chart.
<code>label.size1</code>	the font size of labels shown in the ring chart.
<code>label.color1</code>	the color of labels shown in the ring chart.
<code>label.distance1</code>	the distance of labels from the border of the ring chart.
<code>pattern.type1</code>	a vector of patterns to be filled in the ring. The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.color1</code>	a vector of colors for the lines/dots of patterns filled in the ring.
<code>pattern.line.size1</code>	a vector of numeric values, the line size for the lines/dots of patterns filled in the ring.
<code>background.color1</code>	a vector of colors to be filled in the ring.
<code>frame.color</code>	the color for the borders of the ring.
<code>frame.size</code>	a numeric value, the line size for the borders of the ring.
<code>density1</code>	a numeric vector, the density for the lines/dots of patterns of the ring.
<code>pixel</code>	a numeric value, the pixel resolution of the ring.
<code>pattern.type.inner</code>	a pattern to be filled in the inner circle.
<code>pattern.color.inner</code>	the color for the lines/dots of the pattern of the inner circle.
<code>pattern.line.size.inner</code>	the line size for the lines/dots of the pattern of the inner circle.
<code>background.color.inner</code>	the color to be filled in the inner circle.
<code>pixel.inner</code>	a numeric value, the pixel resolution of the inner circle.
<code>density.inner</code>	a numeric vector, the density for the lines/dots of patterns of the inner circle.
<code>r1</code>	a numeric value, the inner radius of the ring.
<code>r2</code>	a numeric value, the outer radius of the ring.

Details

`patternring1` function offers flexible ways of doing ring charts.

Value

A `ggplot` object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function `imagering1`

Examples

```
library(patternplot)
library(png)
library(ggplot2)
group1<-c('New_England', 'Great_Lakes','Plains', 'Rocky_Mountain',
          'Far_West','Southwest', 'Southeast', 'Mideast')
pct1<-c( 12, 11, 17, 15, 8, 11, 16, 10)
label1<-paste(group1, " \n ", pct1, "%", sep="")

pattern.type1<-c("hdashes", "blank", "grid", "blank", "hlines",
                "blank", "waves", "blank")
pattern.type.inner<-"blank"
pattern.color1<-rep("white", 8)
background.color1<-c("darkgreen", "darkcyan", "chocolate", "cadetblue1",
                    "darkorchid", "yellowgreen", "hotpink", "lightslateblue")
density1<-rep(12, length(group1))
pattern.line.size1=c(12, 1, 10, 1, 20, 1, 12, 1)

g<-patternring1(group1, pct1, label1, label.size1=4,label.color1='black',
label.distance1=1.45, pattern.type1, pattern.color1, pattern.line.size1,
background.color1, frame.color='black',frame.size=1.2, density1, pixel=18,
pattern.type.inner="blank",pattern.color.inner="white", pattern.line.size.inner=1,
background.color.inner="white", pixel.inner=10, density.inner=1, r1=3, r2=4)
g<-g+annotate(geom="text", x=0, y=0,label="2019 Number of Cases \n N=1000",color="black",
size=4)+scale_x_continuous(limits=c(-7, 7))+scale_y_continuous(limits=c(-7, 7))
g
```

patternrings2

Plot a ring chart using patterns and colors to fill the rings.

Description

The `patternrings2` function is a tool for creating versatile ring charts by filling the rings with colors and patterns.

Usage

```
patternrings2(
  group1,
  group2,
  pct1,
  pct2,
  label1,
  label2,
```

```

label.size1 = 4,
label.size2 = 4,
label.color1 = "black",
label.color2 = "black",
label.distance1 = 1.2,
label.distance2 = 1.2,
pattern.type1,
pattern.type2,
pattern.color1,
pattern.color2,
pattern.line.size1 = rep(10, length(group1)),
pattern.line.size2 = rep(10, length(group2)),
background.color1,
background.color2,
density1 = rep(10, length(group1)),
density2 = rep(10, length(group2)),
pixel = 20,
pattern.type.inner = "blank",
pattern.color.inner = "white",
pattern.line.size.inner = 1,
background.color.inner = "white",
pixel.inner = 10,
density.inner = 1,
frame.color = "black",
frame.size = 1,
r1,
r2,
r3
)

```

Arguments

group1	a vector of strings, containing the names of each slice for the inner ring.
group2	a vector of strings, containing the names of each slice for the outer ring.
pct1	a vector of non-negative numbers, containing percentages of each group for the inner ring. The numbers must sum up to 100.
pct2	a vector of non-negative numbers, containing percentages of each group for the outer ring. The numbers must sum up to 100.
label1	a vector of strings, giving the names for the slices shown in the ring chart for the inner ring.
label2	a vector of strings, giving the names for the slices shown in the ring chart for the outer ring.
label.size1	the font size of labels shown in the ring chart for the inner ring.
label.size2	the font size of labels shown in the ring chart for the outer ring.
label.color1	the color of labels shown in the ring chart for the inner ring.
label.color2	the color of labels shown in the ring chart for the outer ring.

<code>label.distance1</code>	the distance of labels from the border of the ring chart for the inner ring.
<code>label.distance2</code>	the distance of labels from the border of the ring chart for the outer ring.
<code>pattern.type1</code>	a vector of patterns to be filled in the slices for the inner ring. The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.type2</code>	a vector of patterns to be filled in the slices for the outer ring. The pattern types include: 'blank', 'bricks', 'vdashes', 'hdashes', 'crosshatch', 'dots', 'grid', 'hlines', 'nelines', 'nwlines', 'vlines', 'waves' and more.
<code>pattern.color1</code>	a vector of colors for the lines/dots of patterns for the inner ring.
<code>pattern.color2</code>	a vector of colors for the lines/dots of patterns for the outer ring.
<code>pattern.line.size1</code>	a vector of numeric values, the line size for the lines/dots of patterns for the inner ring.
<code>pattern.line.size2</code>	a vector of numeric values, the line size for the lines/dots of patterns for the outer ring.
<code>background.color1</code>	a vector of colors to be filled in the slices for the inner ring.
<code>background.color2</code>	a vector of colors to be filled in the slices for the outer ring.
<code>density1</code>	a numeric vector, the density for the lines/dots of patterns for the inner ring.
<code>density2</code>	a numeric vector, the density for the lines/dots of patterns for the outer ring.
<code>pixel</code>	a numeric value, the pixel resolution of ring chart.
<code>pattern.type.inner</code>	a pattern to be filled in the inner circle.
<code>pattern.color.inner</code>	the color for the lines/dots of the pattern.
<code>pattern.line.size.inner</code>	the line size for the lines/dots of the pattern.
<code>background.color.inner</code>	the color to be filled in the inner circle.
<code>pixel.inner</code>	a numeric value, the pixel resolution of the inner circle.
<code>density.inner</code>	a numeric vector, the density for the lines/dots of patterns of the inner circle.
<code>frame.color</code>	the color for the borders of slices.
<code>frame.size</code>	a numeric value, the line size for the borders of slices.
<code>r1</code>	a numeric value, the inner radius of the inner ring.
<code>r2</code>	a numeric value, the outer radius of the inner ring.
<code>r3</code>	a numeric value, the outer radius of the outer ring.

Details

`patternrings2` function offers flexible ways of doing ring charts.

Value

A ggplot object.

Author(s)

Chunqiao Luo (chunqiaoluo@gmail.com)

See Also

Function imagerings2

Examples

```
library(patternplot)
library(png)
library(ggplot2)

group1<-c("Wind", "Hydro", "Solar", "Coal", "Natural Gas", "Oil")
pct1<-c(12, 15, 8, 22, 18, 25)
label1<-paste(group1, " \n ", pct1 , "%", sep="")

group2<-c("Renewable", "Non-Renewable")
pct2<-c(35, 65)
label2<-paste(group2, " \n ", pct2 , "%", sep="")

pattern.type1<-rep(c( "blank"), times=6)
pattern.type2<-c('grid', 'blank')
pattern.type.inner<-"blank"
pattern.color1<-rep('white', length(group1))
pattern.color2<-rep('white', length(group2))

background.color1<-c("darkolivegreen1", "white", "indianred",
                    "gray81", "white", "sandybrown" )
background.color2<-c("seagreen", "deepskyblue")

density1<-rep(10, length(group1))
density2<-rep(10, length(group2))

pattern.line.size1=rep(6, length(group1))
pattern.line.size2=rep(2, length(group2))
pattern.line.size.inner=1

#Example 1: Two rings
g<-patternrings2(group1, group2, pct1,pct2, label1, label2,
label.size1=3, label.size2=3.5, label.color1='black', label.color2='black',
label.distance1=0.75, label.distance2=1.4, pattern.type1, pattern.type2,
pattern.color1,pattern.color2,pattern.line.size1, pattern.line.size2,
background.color1, background.color2,density1=rep(10, length(group1)),
density2=rep(15, length(group2)),pixel=10, pattern.type.inner, pattern.color.inner="black",
pattern.line.size.inner, background.color.inner="white", pixel.inner=6,
density.inner=5, frame.color='black',frame.size=1,r1=2.45, r2=4.25, r3=5)
g<-g+annotate(geom="text", x=0, y=0, label="Earth's Energy",color="black",size=5)
```

```
g<-g+scale_x_continuous(limits=c(-6, 6))+scale_y_continuous(limits=c(-6, 6))
g

#Example 2: Pie in a ring
g<-patternrings2(group1, group2, pct1,pct2, label1, label2, label.size1=3, label.size2=3.5,
label.color1='black', label.color2='black', label.distance1=0.7, label.distance2=1.4,
pattern.type1, pattern.type2, pattern.color1,pattern.color2,pattern.line.size1,
pattern.line.size2, background.color1, background.color2,density1=rep(10, length(group1)),
density2=rep(15, length(group2)),pixel=10, pattern.type.inner, pattern.color.inner="black",
pattern.line.size.inner, background.color.inner="white", pixel.inner=1, density.inner=2,
frame.color='black',frame.size=1, r1=0.005, r2=4, r3=4.75)
g<-g+scale_x_continuous(limits=c(-6, 6))+scale_y_continuous(limits=c(-6, 6))
g
```

Index

imagebar, [2](#)
imagebar_s, [4](#)
imageboxplot, [6](#)
imagepie, [8](#)
imagering1, [10](#)
imagerings2, [12](#)

pattern, [15](#)
patternbar, [16](#)
patternbar_s, [18](#)
patternboxplot, [20](#)
patternpie, [23](#)
patternring1, [25](#)
patternrings2, [27](#)