Package ‘ggESDA’

January 10, 2022

Type Package
Title Exploratory Symbolic Data Analysis with ‘ggplot2’
Version 0.1.0
Author Bo-Syue Jiang
Maintainer Bo-Syue Jiang <lukajiang1998@gmail.com>
Description Implements an extension of ‘ggplot2’ and visualizes the symbolic data with multiple plot which can be adjusted by more general and flexible input arguments. It also provides a function to transform the classical data to symbolic data by both clustering algorithm and customized method.
Depends ggplot2, R (>= 3.5.0), tidyverse
Suggests testthat (>= 2.1.0), knitr, rmarkdown
Imports RSDA, rlang, R6, stats, dplyr, tidyr, gridExtra, gtools, stringr, prodlim, ggforce, ggpubr, ggthemes, tibble, magrittr
License GPL (>= 2)
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
VignetteBuilder knitr
URL https://github.com/kiangkiangkiang/ggESDA
BugReports https://github.com/kiangkiangkiang/ggESDA/issues
NeedsCompilation no
Repository CRAN
Date/Publication 2022-01-10 10:02:42 UTC

R topics documented:

  AbaloneIdt .............................................................. 2
  BLOOD ........................................................................ 3
  Cardiological .............................................................. 3
AbaloneIdt data example

Description

AbaloneIdt interval data example.

Usage

data(AbaloneIdt)

Format

An object of class data.frame (inherits from symbolic_tbl) with 24 rows and 7 columns.

References


Examples

data(AbaloneIdt)
ggInterval_index(AbaloneIdt, aes(x = Length))
**BLOOD**  

**BLOOD data example**

**Description**  
BLOOD interval data example.

**Usage**  
data(BLOOD)

**Format**  
An object of class tbl_df (inherits from tbl, data.frame, symbolic_tbl) with 14 rows and 3 columns.

**References**  

**Examples**  
data(BLOOD)  
ggInterval_minmax(BLOOD, aes(x = Hematocrit))

---

**Cardiological**  

**Cardiological data example**

**Description**  
Cardiological interval data example.

**Usage**  
data(Cardiological)

**Format**  
An object of class symbolic_tbl (inherits from tbl_df, tbl, data.frame) with 11 rows and 3 columns.

**Source**  
https://CRAN.R-project.org/package=RSDA
References


Examples

data(Cardiological)
ggInterval_index(Cardiological, aes(x = Syst))

classic2sym

Convert classical data frame into a symbolic data.

Description

A function for converting a classical data, which may present as a data frame or a matrix with one entry one value, into a symbolic data, which is shown as an interval or a set in an entry. Object after converting is ggESDA class containing interval data and raw data (if it exist) and typically statistics.

Usage

classic2sym(data=NULL, groupby = "kmeans", k=5, minData=NULL, maxData=NULL)

Arguments

data A classical data frame that you want to be converted into a interval data

groupby A way to aggregate. It can be either a clustering method or a variable name which exist in input data (necessary factor type). Default "kmeans".

k A number of group, which is used by clustering. Default k = 5.

minData if choose groupby parameter as 'customize', user need to define which data is min data or max data.

maxData if choose groupby parameter as 'customize', user need to define which data is min data or max data.

Value

classic2sym returns an object of class "ggESDA", which have a interval data and others as follows.

- intervalData - The Interval data after converting also known as a RSDA object.
- rawData - Classical data that user input.
- clusterResult - Cluster results. If the groupby method is a clustering method then it will exist.
- statisticsDF - A list contains data frame including some typically statistics in each group.
Examples

```R
# classical data to symbolic data
classic2sym(iris)
classic2sym(mtcars, groupby = "kmeans", k=10)
classic2sym(iris, groupby = "hclust", k=7)
classic2sym(iris, groupby = Species)
```

```R
dx1 <- runif(10, -30, -10)
y1 <- runif(10, -10, 30)
dx2 <- runif(10, -5, 5)
y2 <- runif(10, 10, 50)
dx3 <- runif(10, -50, 30)
y3 <- runif(10, 31, 60)
```

```R
d <- data.frame(min1 = x1, max1 = y1, min2 = x2, max2 = y2, min3 = x3, max3 = y3)
classic2sym(d, groupby = "customize", minData = d[, c(1, 3, 5)], maxData = d[, c(2, 4, 6)])
classic2sym(d, groupby = "customize", minData = d$min1, maxData = d$min2)
```

```R
# extract the data
symObj <- classic2sym(iris)
symObj$intervalData  # interval data
symObj$rawData       # raw data
symObj$clusterResult # cluster result
symObj$statisticsDF  # statistics
```

---

**Environment**

*Environment data example*

**Description**

Environment interval and modal data example.

**Usage**

data(Environment)

**Format**

An object of class `symbolic_tbl` (inherits from `tbl_df`, `tbl`, `data.frame`) with 14 rows and 17 columns.

**Examples**

data(Environment)
ggInterval_radar(Environment, plotPartial = 2, showLegend = FALSE, base_circle = TRUE, base_lty = 2, addText = FALSE)
facedata  

Face Data Example

Description
Symbolic data matrix with all the variables of interval type.

Usage
data('facedata')

Format
$I;AD;AD;$I;BC;BC;........

HUS1:$I;168.86;172.84;$I;58.55;63.39;........
HUS2:$I;169.85;175.03;$I;60.21;64.38;........
HUS3:$I;168.76;175.15;$I;61.4;63.51;........
INC1:$I;155.26;160.45;$I;53.15;60.21;........
INC2:$I;156.26;161.31;$I;51.09;60.07;........
INC3:$I;154.47;160.31;$I;55.08;59.03;........
ISA1:$I;164;168;$I;55.01;60.03;........
ISA2:$I;163;170;$I;54.04;59;........
ISA3:$I;164.01;169.01;$I;55.59.01;........
JPL1:$I;167.11;171.19;$I;61.03;65.01;........
JPL2:$I;169.14;173.18;$I;60.07;65.07;........
JPL3:$I;169.03;170.11;$I;59.01;65.01;........
KHA1:$I;149.34;155.54;$I;54.15;59.14;........
KHA2:$I;149.34;155.32;$I;52.04;58.22;........
KHA3:$I;150.33;157.26;$I;52.09;60.21;........
LOT1:$I;152.64;157.62;$I;51.35;56.22;........
LOT2:$I;154.64;157.62;$I;52.24;56.32;........
LOT3:$I;154.83;157.81;$I;50.36;55.23;........
PHI1:$I;163.08;167.07;$I;66.03;68.07;........
PHI2:$I;164;168.03;$I;65.03;68.12;........
PHI3:$I;161.01;167;$I;64.07;69.01;........
ROM1:$I;167.15;171.24;$I;64.07;68.07;........
ROM2:$I;168.15;172.14;$I;63.13;68.07;........
ROM3:$I;167.11;171.19;$I;63.13;68.03;........

Source
https://CRAN.R-project.org/package=RSDA
References


---

**Description**

This is an object that will be used to make a ggplot object. A ggESDA object contains both classic data that the user has and interval data which we transform. Moreover, some basic statistics from row data will also be recorded in this object, and the interval data which is from RSDA transformation will still contain RSDA properties.

**Public fields**

- `rawData` the data from the user.
- `statisticsDF` contains min max mean median dataframe for each group of symbolic data.
- `intervalData` interval data from RSDA type.
- `clusterResult` clustering result.

**Methods**

**Public methods:**

- `ggESDA$new()`
- `ggESDA$clone()`

**Method** `new()`: initialize all data, check whether satisfy their form.

*Usage:*

```r
ggESDA$new(  
  rawData = NULL,  
  statisticsDF = NULL,  
  intervalData = NULL,  
  clusterResult = NULL  
)
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```r
ggESDA$clone(deep = FALSE)
```

*Arguments:*

depth Whether to make a deep clone.
ggInterval_2Dhist  
visualize a 2-dimension histogram by symbolic data with ggplot package.

Description

Visualize the two continuous variable distribution by dividing both the x axis and y axis into bins, and calculating the frequency of observation interval in each bin.

Usage

```r
ggInterval_2Dhist(data = NULL, mapping = aes(NULL), xBins = 14, yBins = 16, removeZero = FALSE, addFreq = TRUE)
```

Arguments

- `data`: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.
- `mapping`: Set of aesthetic mappings created by `aes()` or `aes()`. If specified and inherit. `aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.
- `xBins`: x axis bins, which mean how many partials x variable will be separate into.
- `yBins`: y axis bins. It is the same as xBins.
- `removeZero`: whether remove data whose frequency is equal to zero
- `addFreq`: where add frequency text in each cells.

Value

Return a ggplot2 object.

Examples

```r
ggInterval_2Dhist(oils, aes(x = GRA, y = FRE), xBins = 5, yBins = 5)
```
**ggInterval_2DhistMatrix**

2-Dimension histogram matrix

**Description**

Visualize the all continuous variable distribution by dividing both the x axis and y axis into bins, and calculating the frequency of observation interval in each bin. Eventually show it by a matrix plot. Note: this function will automatically filter out the discrete variables, and plot all continuous in input data, so it can not be necessary that give the particularly variables in aes such like (aes(x = x, y = y)). It isn’t also recommended to deal with too many variables because the big O in calculating full matrix will be too large.

**Usage**

```r
ggInterval_2DhistMatrix(data = NULL, mapping = aes(NULL), xBins = 8, yBins = 8, removeZero = FALSE, addFreq = TRUE)
```

**Arguments**

- `data` A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.
- `mapping` Set of aesthetic mappings created by aes() or aes(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.
- `xBins` x axis bins, which mean how many bins x variable will be separate into
- `yBins` y axis bins. It is the same as xBins
- `removeZero` whether remove data whose frequency is equal to zero
- `addFreq` where add frequency text in each cells.

**Value**

Return a plot with ggplot2 object

**Examples**

```r
ggInterval_2DhistMatrix(oils, xBins = 5, yBins = 5)
```
ggInterval_3Dscatter  3D scatter plot for interval data

Description
Visualize the three continuous variable distribution by collecting all vertices in each interval to form a shape of cube. Also show the difference between each group.

Usage
```r
ggInterval_3Dscatter(data = NULL, mapping = aes(NULL), scale = FALSE)
```

Arguments
- `data` A ggSDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggSDA data.
- `mapping` Set of aesthetic mappings created by aes() or aes_(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.
- `scale` A boolean variable, TRUE, standardize data. FALSE, not standardize. If variance is too large (or small) or the difference between two variables are too large, it will be distortion or unseeable, which may happen in different units or others. So, a standardize way is necessary.

Value
Return a ggplot2 object (It will still be 2-Dimension).

Examples
```r
ggInterval_3Dscatter(facedata[1:5, ], aes(x = BC, y = EH, z = GH))
```

ggInterval_boxplot  A interval Box plot

Description
Visualize the one continuous variable distribution by box represented by multiple rectangles.

Usage
```r
ggInterval_boxplot(data = NULL, mapping = aes(NULL), plotAll = FALSE)
```
**Arguments**

- **data**: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.

- **mapping**: Set of aesthetic mappings created by aes() or aes_. If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.

- **plotAll**: booleans, if TRUE, plot all variable together

**Value**

Return a ggplot2 object.

**Examples**

```r
p <- ggInterval_boxplot(iris, aes(iris$Petal.Length))
p + scale_fill_manual(values = c("red", "yellow", "green", "blue", "black"), labels = c("0\%", "25\%", "50\%", "75\%", "100\%"), name = "quantile")
```

```r
mydata <- RSDA::facedata
ggInterval_boxplot(mydata, aes(AD, col = "black", alpha = 0.5))
```

```r
myMtcars <- classic2sym(mtcars)
myMtcars <- myMtcars$intervalData
ggInterval_boxplot(myMtcars, aes(disp))
```

---

**ggInterval_centerRange**

*Figure with x-axis = center y-axis = range*

**Description**

Visualize the relation between center and range.

**Usage**

```r
ggInterval_centerRange(data = NULL, mapping = aes(NULL), plotAll = FALSE)
```

**Arguments**

- **data**: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.
ggInterval_hist

mapping Set of aesthetic mappings created by aes() or aes_(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.

plotAll booleans, if TRUE, plot all variable together

Value

Return a ggplot2 object.

Examples

```r
ggInterval_centerRange(iris,aes(iris$Sepal.Length))
```

```r
mydata<-RSDA::facedata
ggInterval_centerRange(mydata,aes(AD,col="blue",pch=2))
```

---

**ggInterval_hist**

*Histogram for symbolic data with equal-bin or unequal-bin.*

Description

Visualize the continuous variable distribution by dividing the x axis into bins, and calculating the frequency of observation interval in each bin.

Usage

```r
ggInterval_hist(data = NULL,mapping = aes(NULL),method="equal-bin",bins=10,
plotAll = FALSE)
```

Arguments

data A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.

mapping Set of aesthetic mappings created by aes() or aes_(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.

method It can be equal-bin(default) or unequal-bin. Equal-bin means the width in histogram is equal, which represent all intervals divided have the same range. Unequal-bin means the range of intervals are not the same, and it can be more general on data. Thus, the bins of unequal-bin method depends on the data, and the argument “bins” will be unused.

bins x axis bins, which mean how many partials the variable

plotAll boolean, whether plot all variables, default FALSE. will be separate into.
**ggInterval_index**

**Value**

Return a ggplot2 object.

**Examples**

```r
ggInterval_hist(mtcars,aes(x=wt))
```

```r
ggInterval_hist(iris,aes(iris$Petal.Length,col="blue",alpha=0.2, fill="red"),bins=30)
```

```r
d<-data.frame(x=rnorm(1000,0,1))
p<-ggInterval_hist(d,aes(x=x),bins=40,method="equal-bin")
p
```

```r
p+scale_fill_manual(values=rainbow(40))+labs(title="myNorm")
```

---

**ggInterval_index**

*Plot the range of each observations*

**Description**

Visualize the range of the variables of each observations by using a kind of margin bar that indicate the minimal and maximal of observations.

**Usage**

```r
ggInterval_index(data = NULL,mapping = aes(NULL))
```

**Arguments**

- **data**
  - A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.

- **mapping**
  - Set of aesthetic mappings created by aes() or aes(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.

**Value**

Return a ggplot2 object.
Examples

# the observations show on the y-axis, values on x-axis
ggInterval_index(iris,aes(x=iris$Sepal.Length))

# change above axis
ggInterval_index(mtcars,aes(y=disp,col="red",fill="grey"))

# symbolic data
mydata <- RSDA::facedata
ggInterval_index(mydata,aes(x=3:13,y=AD))

---

**ggInterval_indexImage**  An index plot presented by color image for interval data.

---

Description

Visualize the range of the variables of each observation by using color image. The index image replace margin bar by color, thus it will be more visible for data.

Usage

```r
ggInterval_indexImage(data = NULL,mapping = aes(NULL),
column_condition=TRUE,full_strip=FALSE, plotAll = FALSE)
```

Arguments

- **data**: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.
- **mapping**: Set of aesthetic mappings created by aes() or aes(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
- **column_condition**: Boolean variables, which mean the color present by column condition (if TRUE) or matrix condition (if FALSE).
- **full_strip**: Boolean variables, which mean the strip present in full figure-width (if TRUE) or only in its variable values (if FALSE).
- **plotAll**: Boolean, which determine if the heatmap type for visualizing full variables is used. default FALSE.

Value

Return a ggplot2 object.
ggInterval_minmax

Examples

d<-data.frame(qq=rnorm(1000,0,1))
ggInterval_indexImage(d,aes(qq))

mydata<-RSDA::facedata
p<-ggInterval_indexImage(mydata,aes(AD),full_strip=TRUE,column_condition = TRUE)
#Recommend to add coord_flip() to make the plot more visible
p+coord_flip()

myIris<-classic2sym(iris,groupby=Species)
myIris<-myIris$intervalData
p<-ggInterval_indexImage(myIris,aes(myIris$Petal.Length),full_strip=FALSE,column_condition=TRUE)
p

ggInterval_indexImage(mtcars,aes(disp))+labs(x="anything")

---

ggInterval_minmax A min-max plot for interval data

Description

Visualize the range of the variables of each observations by marking minimal and maximal point.

Usage

ggInterval_minmax(data = NULL,mapping = aes(NULL),
              scaleXY = "local",plotAll=FALSE)

Arguments

data A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.
mapping Set of aesthetic mappings created by aes() or aes(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
scaleXY default "local", which means limits of x-axis and y-axis depend on their own variable. "global" means limits of them depend on all variables that user input.
plotAll booleans, if TRUE, plot all variable together

Value

Return a ggplot2 object.
**Examples**

```r
ggInterval_minmax(mtcars, aes(disp))

mydata2 <- RSDA::Cardiological
ggInterval_minmax(mydata2, aes(mydata2$Pulse, size=3))

d <- mapply(c(10, 20, 40, 80, 160), c(20, 40, 80, 160, 320), FUN=runif, n=1000)
d <- data.frame(qq=matrix(d, ncol=1))
ggInterval_minmax(d, aes(qq))

myIris <- classic2sym(iris, groupby=Species)
myIris <- myIris$intervalData
ggInterval_minmax(myIris, aes(myIris$Petal.Length)) +
  theme_classic()
```

---

**ggInterval_PCA**  
**Vertice-PCA for interval data**

**Description**

`ggInterval_PCA` performs a principal components analysis on the given numeric interval data and returns the results like `princomp`, `ggplot` object and a interval scores.

**Usage**

```r
ggInterval_PCA(data = NULL, mapping = aes(NULL), plot=TRUE,
  concepts_group=NULL, poly = FALSE, adjust = TRUE)
```

**Arguments**

- **data**: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.
- **mapping**: Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and inherit. `aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. It is the same as the mapping of ggplot2.
- **plot**: Boolean variable. Auto plot (if TRUE). It can also plot by its inner object
- **concepts_group**: color with each group of concept
- **poly**: if plot a poly result
- **adjust**: adjust sign of the principal component

**Value**

A ggplot object for PC1,PC2,and a interval scores and others.

- scores_interval: The interval scores after PCA.
- ggplotPCA: a ggplot object with x-axis and y-axis are PC1 and PC2.
- others: others are the returns values of princomp.
Examples

```r
ggInterval_PCA(iris)

mydata2<-RSDA::Cardiological
ggInterval_PCA(mydata2,aes(col="red",alpha=0.2))

d<-mapply(c(10,20,40,80,160),c(20,40,80,160,320),FUN=runif,n=1000)
d<-data.frame(qq=matrix(d,ncol=4))
ggInterval_PCA(d)

myIris<-classic2sym(iris,Species)
p<-ggInterval_PCA(myIris,plot=FALSE)
p$ggplotPCA
p$scores_interval
```

---

**ggInterval_radar**

*A interval Radar plot*

### Description

Using ggplot2 package to make a radar plot with multiple variables. Each variable contains min values and max values as a symbolic data.

### Usage

```r
ggInterval_radar(data=NULL,layerNumber=3,
inOneFig=TRUE,showLegend=TRUE,showXYLabs=FALSE,
plotPartial=NULL,
alpha=0.5,
base_circle=TRUE,
base_lty=2,
addText=TRUE,
type="default",
quantileNum=4,
Drift=0.5,
addText_modal=TRUE,
addText_modal.p=FALSE)
```

### Arguments

- **data**: A ggESDA object. It can also be either RSDA object or classical data frame (not recommended), which will be automatically convert to ggESDA data.
- **layerNumber**: Number of layer of a concentric circle, usually to visualize the reach of an observation in particular variable.
- **inOneFig**: Whether plot all observations in one figure. If not, it will generate a new windows containing distinct observations.
showLegend  whether show the legend.
showXYLabs whether show the x,y axis labels.
plotPartial a numeric vector, which is the row index from the data. If it is not null, it will extract the row user deciding to draw a radar plot from original data. Notes: the data must be an interval data if the plotPartial is not null.
alpha aesthetic alpha of fill color
base_circle boolean, if true, it will generate inner circle.
base_lty line type in base figure
addText add the value of interval-valued variables in figure
addText_modal add the factor of modal multi-valued variables in figure.
addText_modal.p add the value of modal multi-valued variables in figure.

Examples

mydata<-ggESDA::classic2sym(mtcars,k=4)$intervalData
ggInterval_radar(data=mydata[,c("mpg","disp","drat")])
ggInterval_radar(data=mydata[,c("mpg","disp","drat")],inOneFig = TRUE,plotPartial = c(2,3))

mydata<-ggESDA::classic2sym(iris,groupby = Species)$intervalData
ggInterval_radar(mydata,inOneFig = TRUE)+geom_text(aes(x=0.6,0.6),label="Add anything you want")

---

**ggInterval_scaMatrix** scatter plot for all variable by interval data.

---

**Description**

Visualize the all continuous variable distribution by rectangle for both x-axis and y-axis with a matrix grid. Note: this function will automatically filter out the discrete variables, and plot all continuous in input data, so it can not be necessary that give the particularly variables in aes such like (aes(x = x, y = y)). It isn’t also recommended to deal with too many variables because the big O in calculating full matrix will be too large.

**Usage**

```
ggInterval_scaMatrix(data = NULL,mapping = aes(NULL),showLegend=TRUE)
```
**Arguments**

- **data**: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.

- **mapping**: Set of aesthetic mappings created by aes() or aes_(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

- **showLegend**: whether show the legend.

**Value**

Return a plot with no longer a ggplot2 object, instead of a marrangeGrob object.

**Examples**

```r
a <- rnorm(1000, 0, 5)
b <- runif(1000, -20, -10)
c <- rgamma(1000, 10, 5)
d <- as.data.frame(cbind(norm = a, unif = b, gamma_10_5 = c))
ggInterval_scaMatrix(d)

ggInterval_scaMatrix(mtcars[, c("mpg", "wt", "qsec")],
                     aes(col = "red", lty = 2, fill = "blue", alpha = 0.3))

myIris <- classic2sym(iris, groupby = Species)$intervalData
ggInterval_scaMatrix(myIris[, 1:3])

mydata <- RSDA::Cardiological
ggInterval_scaMatrix(mydata[, 1:3], aes(fill = "black", alpha = 0.2))
```

---

**Description**

Visualize the two continuous variable distribution by rectangle and each of its width and height represents a interval of the data.

**Usage**

```r
ggInterval_scatter(data = NULL, mapping = aes(NULL))
```
Arguments

data: A ggESDA object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggESDA data.

mapping: Set of aesthetic mappings created by aes() or aes_(). If specified and inherit. aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

Value

Return a ggplot2 object.

Examples

```r
a <- rnorm(1000, 0, 5)
b <- runif(1000, -20, -10)
d <- as.data.frame(cbind(norm = a, unif = b))
ggInterval_scatter(d, aes(a, b))

ggInterval_scatter(mtcars[, c("mpg", "wt", "qsec")],
aes(x = mpg, y = wt,
col = "red", lty = 2, fill = "blue", alpha = 0.3))

myIris <- classic2sym(iris, groupby = Species)$intervalData
p <- ggInterval_scatter(myIris, aes(myIris$Petal.Length, myIris$Petal.Width))
p + scale_fill_manual(labels = rownames(myIris),
values = c("red", "blue", "green"),
name = "Group")

mydata <- RSDA::facedata
p <- ggInterval_scatter(mydata[1:10, ], aes(AD, BC, alpha = 0.2))
p + scale_fill_manual(labels = rownames(mydata)[1:10],
values = rainbow(10),
name = "Group")
```

Description

iris.i interval data example.

Usage

data(iris.i)
**mtcars.i**

**Format**
An object of class `data.frame` (inherits from `symbolic_tbl`) with 3 rows and 4 columns.

**Examples**
```r
data(iris.i)
ggInterval_index(iris.i, aes(x = Sepal.Length))
```

---

**mushroom**

**Description**

mushroom interval data example.

**Usage**
```r
data(mushroom)
```

**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`, `symbolic_tbl`) with 23 rows and 3 columns.
References


Examples

data(mushroom)
ggInterval_scatter(mushroom, aes(x = Cap.Widths, y = Stipe.Lengths))

---

oils

**oils data example**

Description

oils interval data example.

Usage

data(oils)

Format

An object of class symbolic_tbl (inherits from tbl_df, tbl, data.frame) with 8 rows and 4 columns.

References


Examples

data(oils)
ggInterval_scatter(oils, aes(x = GRA, y = IOD))
RSDA2sym

RSDA object to symbolic object for ggplot

Description
It will be a good way to unify all symbolic data object in R that collects all useful symbolic analysis tools such like RSDA into the same class for management. In this way, user who wants to do some study in symbolic data will be more convenient for searching packages. Thus, RSDA2sym collecting RSDA object into ggESDA object will do for plot(ggplot) and RSDA's analysis.

Usage
RSDA2sym(data=NULL, rawData=NULL)

Arguments
data
an interval data, which may transform by RSDA::classic.to.sym. Note: data is a necessary parameter, and must have symbolic_tbl class.

rawData
rawData, which can be transformed to interval data, must be a data frame and match to data.

Value
Return an object of class "ggESDA", which have a interval data and others as follows.

• intervalData - The Interval data after converting also known as a RSDA object.
• rawData - Classical data that user input.
• clusterResult - Cluster results. If the groupby method is a clustering method then it will exist.
• statisticsDF - A list contains data frame including some typically statistics in each group.

Examples
r<-RSDA::Cardiological
mySym<-RSDA2sym(r)
mySym$intervalData
Index

* datasets
  - AbaloneIdt, 2
  - BLOOD, 3
  - Cardiological, 3
  - Environment, 5
  - facedata, 6
  - iris.i, 20
  - mtcars.i, 21
  - mushroom, 21
  - oils, 22

AbaloneIdt, 2

BLOOD, 3

Cardiological, 3
classic2sym, 4

Environment, 5

facedata, 6

ggESDA, 7

ggInterval_2Dhist, 8

ggInterval_2DhistMatrix, 9

ggInterval_3Dscatter, 10

ggInterval_boxplot, 10

ggInterval_centerRange, 11

ggInterval_hist, 12

ggInterval_index, 13

ggInterval_indexImage, 14

ggInterval_minmax, 15

ggInterval_PCA, 16

ggInterval_radar, 17

ggInterval_scaMatrix, 18

ggInterval_scatter, 19

iris.i, 20

mtcars.i, 21

mushroom, 21

oils, 22

RSDA2sym, 23