Package 'ggInterval'

April 28, 2026

Type Package

Title Visualizing Interval-Valued Data Using 'ggplot2'

Version 0.2.5

Date 2026-04-28

Author Bo-Syue Jiang [aut],
Han-Ming Wu [cre]

Maintainer Han-Ming Wu <wuhm@g.nccu.edu.tw>

Description Extends 'ggplot2' for visualizing interval-valued data with scatter plots, histograms, index plots, boxplots, radar plots, PCA displays, and correlation heatmaps. The package also converts classical data tables into interval-valued data using clustering algorithms or user-defined groupings.

Depends ggplot2, R (>= 4.4.0), tidyverse, RSDA

Suggests testthat (>= 3.0.0), knitr, rmarkdown, data.table, HistDAWass, MAINT.Data, TTR

Imports rlang, R6, dplyr, tidyr, gridExtra, gtools, stringr, prodlim, ggforce, ggpubr, ggthemes, tibble, magrittr, vctrs

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/hanmingwu1103/ggInterval, https://CRAN.R-project.org/package=ggInterval

BugReports https://github.com/hanmingwu1103/ggInterval/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2026-04-28 12:30:02 UTC

Contents

`abalone.i` *abalone.i data example*

Description

abalone.i interval data example.

Usage

```
data(abalone.i)
```

Format

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

Source

Adapted from `MAINT.Data::AbaloneIdt`; the underlying Abalone data are from the UCI Machine Learning Repository.

Examples

```
data(abalone.i)
ggInterval_indexplot(abalone.i, aes(x = Length))
```

`blood.i` *blood.i data example*

Description

blood.i interval data example.

Usage

```
data(blood.i)
```

Format

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

Source

Adapted from `HistDAWass::BLOOD`; see https://CRAN.R-project.org/package=HistDAWass.

References

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Examples

```
data(blood.i)
ggInterval_MMplot(blood.i, 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

Adapted from `RSDA::Cardiological`; see https://CRAN.R-project.org/package=RSDA.

References

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Examples

```
data(Cardiological)
ggInterval_indexplot(Cardiological, aes(x = Syst))
```

Cardiological2 *Cardiological data example*

Description

Cardiological interval data example.

Usage

```
data(Cardiological2)
```

Format

An object of class `symbolic_tbl` (inherits from `tbl_df`, `tbl`, `data.frame`) with 15 rows and 3 columns.

Source

Adapted from RSDA::Cardiological2; see https://CRAN.R-project.org/package=RSDA.

References

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Examples

```
data(Cardiological2)
ggInterval_indexplot(Cardiological2, 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 object, which is represented as an interval or a set in an entry. Object after converting is ggInterval 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,
modalData = NULL)
```

Arguments

data	A classical data frame that you want to be converted into interval-valued 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.
modalData	list, each cell of list contain a set of column index of its modal multi-valued data of the input data. the value of it is a proportion presentation, and sum of each row in these column must be equal to 1. ex 0,1,0 or 0.2,0.3,0.5. the input type of modalData for example is modalData[[1]] = c(2, 3), modalData[[2]] = c(7:10), that 2, 3, 7, 8, 9, 10 columns are modal type of the data. Note: the option is only valid when groupby == "customize".

Value

classic2sym returns an object of class "ggInterval",which has interval-valued data and related outputs 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

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

x1<-runif(10, -30, -10)
y1<-runif(10, -10, 30)
x2<-runif(10, -5, 5)
y2<-runif(10, 10, 50)
x3<-runif(10, -50, 30)
y3<-runif(10, 31, 60)

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)


#example for build modal data
#for the first modal data proportion
a1 <- runif(10, 0,0.4) %>% round(digits = 1)
a2 <- runif(10, 0,0.4) %>% round(digits = 1)

#for the second modal data proportion
b1 <- runif(10, 0,0.4) %>% round(digits = 1)
b2 <- runif(10, 0,0.4) %>% round(digits = 1)

#for interval-valued data
c1 <- runif(10, 10, 20) %>% round(digits = 0)
c2 <- runif(10, -50, -10) %>% round(digits = 0)

#build simulated data
d <- data.frame(a1 = a1, a2 = a2, a3 = 1-(a1+a2),
                c1 = c1, c2 = c2,
                b1 = b1, b2 = b2, b3 = 1-(b1+b2))

#transformation
classic2sym(d, groupby = "customize",
            minData = d$c2,
            maxData = d$c1,
```

```
            modalData = list(1:3, 6:8))#two modal data

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

---

cor *Generic function for the correlation*

Description

This function compute the symbolic correlation

Usage

```
cor(x, ...)

## Default S3 method:
cor(
  x,
  y = NULL,
  use = "everything",
  method = c("pearson", "kendall", "spearman"),
  ...
)

## S3 method for class 'symbolic_tbl'
cor(x, ...)

## S3 method for class 'symbolic_interval'
cor(x, y, method = c("centers", "B", "BD", "BG"), ...)
```

Arguments

x	First symbolic variables.
...	As in R cor function.
y	Second symbolic variables.
use	an optional character string giving a method for computing correlation in the presence of missing values. This must be (an abbreviation of) one of the strings 'everything', 'all.obs', 'complete.obs', 'na.or.complete', or 'pairwise.complete.obs'.
method	The method to use.

Details

Supported interval-valued methods are:

- `"centers"`: correlation of interval centers.
- `"B"`: Billard correlation.
- `"BD"`: Billard-Diday correlation.
- `"BG"`: Bertrand-Goupil correlation.

For `"B"`, `"BD"`, and `"BG"`, the denominator uses the corresponding method-matched standard deviation.

Value

Return a real number in [-1, 1].

References

Bertrand, Patrice and Goupil, Francoise (2000). Descriptive Statistics for Symbolic Data. In Hans-Hermann Bock and Edwin Diday (eds.), *Analysis of Symbolic Data*, pp. 106–124. Berlin and Heidelberg: Springer.

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Billard, Lynne (2008). Sample covariance functions for complex quantitative data. In *Proceedings of the World IASC Conference*, pp. 157–163, Yokohama, Japan.

Rodriguez-Rojas, Oldemar (2000). *Classification et modeles lineaires en analyse des donnees symboliques*. PhD thesis, Universite Paris IX Dauphine.

---

cov *Generic function for the covariance*

Description

This function compute the symbolic covariance.

Usage

```
cov(x, ...)

## Default S3 method:
cov(
  x,
  y = NULL,
  use = "everything",
  method = c("pearson", "kendall", "spearman"),
  ...
)
```

```
## S3 method for class 'symbolic_tbl'
cov(x, ...)

## S3 method for class 'symbolic_interval'
cov(x, y = NULL, method = c("centers", "B", "BD", "BG"), na.rm = FALSE, ...)
```

Arguments

x	First symbolic variables.
...	As in R cov function.
y	Second symbolic variables.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings 'everything', 'all.obs', 'complete.obs', 'na.or.complete', or 'pairwise.complete.obs'.
method	The method to use.
na.rm	As in R cov function.

Details

Supported interval-valued methods are:

- "centers": covariance of interval centers.
- "B": Billard covariance.
- "BD": Billard-Diday covariance.
- "BG": Bertrand-Goupil covariance.

Value

Return a real number.

References

Bertrand, Patrice and Goupil, Francoise (2000). Descriptive Statistics for Symbolic Data. In Hans-Hermann Bock and Edwin Diday (eds.), *Analysis of Symbolic Data*, pp. 106–124. Berlin and Heidelberg: Springer.

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Billard, Lynne (2008). Sample covariance functions for complex quantitative data. In *Proceedings of the World IASC Conference*, pp. 157–163, Yokohama, Japan.

Rodriguez-Rojas, Oldemar (2000). *Classification et modeles lineaires en analyse des donnees symboliques*. PhD thesis, Universite Paris IX Dauphine.

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_radarplot(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

Adapted from `RSDA::facedata`; see https://CRAN.R-project.org/package=RSDA.

References

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Examples

```
data(facedata)
ggInterval_hist(facedata, aes(x = AD))
```

`ggInterval` *A symbolic object by R6 class for interval analysis and ggplot*

Description

This is an object that will be used to make a ggplot object.A ggInterval object contains both classic data that user have and interval data which we transform.More over,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 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:

- `ggInterval$new()`
- `ggInterval$clone()`

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

Usage:

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

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

Usage:

```
ggInterval$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ggInterval_2Dhist` *Visualize a 2-dimension histogram for interval-valued data*

Description

Visualize the joint distribution of two continuous interval-valued variables by dividing the x axis and y axis into rectangles and calculating the frequency of each observation interval in every rectangle.

Usage

```
ggInterval_2Dhist(data = NULL, mapping = aes(NULL),
method = "equal-bin", xBins = 14, yBins = 16,
display = "p", palette = "Blues", direction = 1, tau = 0,
removeZero = FALSE, cell_labels = FALSE, label_rule = "above-mean",
addFreq = NULL)
```

Arguments

data	A ggInterval object. It can also be either an RSDA object or a classical data frame, which will be automatically converted to ggInterval 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	Histogram partition method. Use "equal-bin" for equally spaced bins or "unequal-bin" for non-equidistant bins defined by the observed interval endpoints.
xBins	Number of x-axis bins used when method = "equal-bin".
yBins	Number of y-axis bins used when method = "equal-bin".
display	Metric shown in the cells. Use "p" for relative frequency, "f" for weighted frequency, or "h" for density.
palette	ColorBrewer palette passed to scale_fill_distiller().
direction	Direction passed to scale_fill_distiller().
tau	Non-negative tolerance used when method = "unequal-bin". Consecutive breakpoints whose gaps are smaller than tau are merged.
removeZero	Whether remove data whose frequency is equal to zero.
cell_labels	Logical. If TRUE, add frequency labels to cells.
label_rule	Rule used when cell_labels = TRUE. The default "above-mean" shows only cells whose displayed values exceed the mean cell value. Alternatives are "nonzero" and "all".
addFreq	Deprecated alias for cell_labels.

Value

Return a list containing a ggplot2 object and the corresponding frequency table.

Examples

```
ggInterval_2Dhist(oils, aes(x = GRA, y = FRE),
  xBins = 5, yBins = 5,
  display = "p",
  palette = "Blues",
  cell_labels = TRUE)

ggInterval_2Dhist(oils, aes(x = GRA, y = FRE),
  method = "unequal-bin",
  display = "p",
  palette = "Blues",
  tau = 0.5)
```

ggInterval_2DhistMatrix

2-Dimension histogram matrix

Description

Visualize all continuous interval-valued variables with a matrix of 2D histograms. Each off-diagonal panel shows a 2D histogram for a pair of variables, and each diagonal panel displays the variable name.

Usage

```
ggInterval_2DhistMatrix(data = NULL, mapping = aes(NULL),
method = "equal-bin", xBins = 8, yBins = 8,
display = "p", palette = "Blues", direction = 1, tau = 0,
removeZero = FALSE, cell_labels = FALSE, label_rule = "above-mean",
addFreq = NULL)
```

Arguments

data	A ggInterval object. It can also be either an RSDA object or a classical data frame, which will be automatically converted to ggInterval 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. It is the same as the mapping of ggplot2. This function ignores x and y mappings and plots all continuous interval-valued variables.
method	Histogram partition method. Use "equal-bin" for equally spaced bins or "unequal-bin" for non-equidistant bins defined by the observed interval endpoints.
xBins	Number of x-axis bins used when method = "equal-bin".
yBins	Number of y-axis bins used when method = "equal-bin".
display	Metric shown in the cells. Use "p" for relative frequency, "f" for weighted frequency, or "h" for density.
palette	ColorBrewer palette passed to scale_fill_distiller().
direction	Direction passed to scale_fill_distiller().
tau	Non-negative tolerance used when method = "unequal-bin". Consecutive breakpoints whose gaps are smaller than tau are merged.
removeZero	Whether remove cells whose frequency is equal to zero.
cell_labels	Logical. If TRUE, add cell labels.
label_rule	Rule used when cell_labels = TRUE. The default "above-mean" shows only cells whose displayed values exceed the mean value within the matrix. Alternatives are "nonzero" and "all".
addFreq	Deprecated alias for cell_labels.

Value

Return a ggplot2 object.

Examples

```
ggInterval_2DhistMatrix(
  oils,
  xBins = 5,
  yBins = 5,
  display = "p",
  palette = "Blues",
  cell_labels = TRUE
)

ggInterval_2DhistMatrix(
  oils,
  method = "unequal-bin",
  display = "p",
  palette = "Blues",
  tau = 0.5
)
```

---

ggInterval_3Dscatterplot

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

```
ggInterval_3Dscatterplot(data = NULL,mapping = aes(NULL),scale=FALSE)
```

Arguments

data	A ggInterval object. It can also be either an RSDA object or a classical data frame, which will be automatically converted to ggInterval 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 standardizes the data and FALSE leaves the data unchanged. If variance is too large(or small) or the difference between two variables are too large,it will be distorted or difficult to see, which may happen when variables are measured in different units.

Value

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

Examples

```
ggInterval_3Dscatterplot(facedata[1:5, ], aes(x = BC, y = EH, z = GH))
```

---

ggInterval_boxplot *A interval Box plot*

Description

Visualize the one continuous variable distribution using one of three interval-aware boxplot styles.

Usage

```
ggInterval_boxplot(data = NULL,mapping = aes(NULL),plotAll=FALSE,
width_type = "violin-like")
```

Arguments

data	A ggInterval object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggInterval 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
width_type	Box-width style. Use "violin-like" (default) to let the rectangle widths reflect local interval concentration, "quantile-depth" for progressively narrower quantile boxes, or "side-by-side" for conventional boxplots of the lower and upper bounds shown next to each other.

Value

Return a ggplot2 object.

Examples

```
mydata <- ggInterval::facedata
ggInterval_boxplot(mydata, aes(x = AD))
ggInterval_boxplot(mydata, aes(x = AD), width_type = "quantile-depth")
ggInterval_boxplot(mydata, plotAll = TRUE, width_type = "side-by-side")
```

`ggInterval_corrplot` *Correlation heatmap for interval-valued data*

Description

Visualize pairwise symbolic correlations between interval-valued variables with a heatmap. This plot is especially useful for summarizing multivariate dependence structures before more formal modeling or dimension reduction.

Usage

```
ggInterval_corrplot(
  data = NULL,
  method = c("centers", "B", "BD", "BG"),
  triangle = c("lower", "upper", "full"),
  showValues = TRUE,
  digits = 2,
  showLegend = TRUE
)
```

Arguments

`data`	A ggInterval object. It can also be either an RSDA object or a classical data frame, which will be automatically converted to ggInterval data.
`method`	Correlation method for interval-valued variables. It must be one of `"centers"` (interval centers), `"B"` (Billard), `"BD"` (Billard-Diday), or `"BG"` (Bertrand-Goupil).
`triangle`	Which part of the symmetric correlation matrix to display. It can be `"lower"`, `"upper"`, or `"full"`.
`showValues`	Logical. If `TRUE`, print the correlation values in each cell.
`digits`	Number of digits used when `showValues = TRUE`.
`showLegend`	Logical. If `TRUE`, display the fill legend.

Value

Return a ggplot2 object.

References

Bertrand, Patrice and Goupil, Francoise (2000). Descriptive Statistics for Symbolic Data. In Hans-Hermann Bock and Edwin Diday (eds.), *Analysis of Symbolic Data*, pp. 106–124. Berlin and Heidelberg: Springer.

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Billard, Lynne (2008). Sample covariance functions for complex quantitative data. In *Proceedings of the World IASC Conference*, pp. 157–163, Yokohama, Japan.

Examples

```
ggInterval_corrplot(facedata)

ggInterval_corrplot(facedata, method = "BG", triangle = "full")
```

`ggInterval_CRplot` *Center-range plot for interval data*

Description

Visualize the relation between interval centers and ranges, with mean reference lines and an optional ellipse overlay.

Usage

```
ggInterval_CRplot(data = NULL,mapping = aes(NULL),plotAll=FALSE,
                          addEllipse = TRUE,
                          ellipseFill = "blue",
                          ellipseAlpha = 0.3)
```

Arguments

`data`	A ggInterval object. It can also be either RSDA object or classical data frame,which will be automatically convert to ggInterval 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
`addEllipse`	logical, if TRUE (default), add a shaded ellipse layer.
`ellipseFill`	fill color of the ellipse layer. Default is "blue".
`ellipseAlpha`	alpha level of the ellipse layer. Default is 0.3.

Value

Return a ggplot2 object.

Examples

```
mydata <- ggInterval::facedata
ggInterval_CRplot(mydata, aes(x = AD, col = "blue", shape = 2))
ggInterval_CRplot(mydata, plotAll = TRUE)
ggInterval_CRplot(mydata, plotAll = TRUE, addEllipse = FALSE)
```

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

```
ggInterval_hist(data = NULL,mapping = aes(NULL),method="equal-bin",bins=10,
 plotAll = FALSE, position = "identity", alpha = 0.5)
```

Arguments

data	A ggInterval object.It can also be either RSDA object or classical data frame,which will be automatically convert to ggInterval 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 histogram uses equally spaced intervals. Unequal-bin means the bin widths are determined by the data, so the argument bins is unused in that case.
bins	x axis bins,which mean how many partials the variable
plotAll	boolean, whether plot all variables, default FALSE. will be separate into.
position	"stack" or "identity"
alpha	fill alpha

Value

An object of class ggInterval_hist_result. A direct call displays the histogram, and the returned object stores the ggplot object in $plot together with the accompanying frequency tables.

Examples

```
mydata <- ggInterval::facedata
ggInterval_hist(mydata, aes(x = AD), bins = 10)

hist_obj <- ggInterval_hist(mydata, plotAll = TRUE, bins = 10)
hist_obj
hist_obj$`Table AD`

ggInterval_hist(mydata, aes(x = AD), method = "unequal-bin")
```

`ggInterval_indexImage` *Index image plot for interval-valued data*

Description

Visualize interval-valued observations with color strips. For a single variable, the display acts as a color-based analogue of an index plot. When `plotAll = TRUE`, the function produces a multivariate image plot across all continuous interval-valued variables.

Usage

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

Arguments

`data`	A ggInterval object. It can also be either RSDA object or classical data frame,which will be automatically convert to ggInterval 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`	Logical. If `TRUE`, the color scale is applied separately within each variable (column condition). If `FALSE`, one common color scale is applied to the whole matrix (matrix condition).
`full_strip`	Logical. If `TRUE`, each strip spans the full plotting width. If `FALSE`, strip widths reflect the interval ranges.
`plotAll`	Logical. If `TRUE`, produce a multivariate image plot for all continuous interval-valued variables. The default is `FALSE`.

Value

Return a ggplot2 object.

Examples

```
mydata <- ggInterval::facedata
p <- ggInterval_indexImage(mydata, aes(x = AD),
  full_strip = TRUE, column_condition = TRUE)
# Recommend adding coord_flip() to make the single-variable display clearer.
p + coord_flip()
ggInterval_indexImage(
  mydata,
  plotAll = TRUE,
  full_strip = TRUE,
  column_condition = FALSE
) +
```

```
  scale_colour_distiller(palette = "Blues", direction = 1)
ggInterval_indexImage(mydata, plotAll = TRUE, full_strip = FALSE)
```

`ggInterval_indexplot` *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

```
ggInterval_indexplot(data = NULL,mapping = aes(NULL),
plotAll = FALSE, row_order = "o", user_order = NULL, labels = FALSE)
```

Arguments

`data`	A ggInterval object. It can also be either RSDA object or classical data frame,which will be automatically convert to ggInterval 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`	plot all variables
`row_order`	Row ordering used when `plotAll = TRUE`: `"o"` for the original order, `"c"` for ascending interval centers within each variable panel, `"r"` for ascending interval ranges within each variable panel, and `"u"` for a user-defined permutation supplied through `user_order`.
`user_order`	User-defined row permutation used when `row_order = "u"` and `plotAll = TRUE`. Supply either an integer permutation of `1:nrow(data)` or a character permutation of the row names.
`labels`	Logical. When `TRUE` and `plotAll = TRUE`, repeat the row labels inside each variable panel. This is especially useful for ordered displays with `row_order = "c"` or `"r"`. For ordered displays, the outer left-side row labels are suppressed by default.

Value

Return a ggplot2 object.

Examples

```
mydata <- ggInterval::facedata
Subjects <- substr(rownames(mydata), 1, 3)
ggInterval_indexplot(mydata, aes(x = AD))
ggInterval_indexplot(mydata, aes(x = AD, fill = Subjects))
ggInterval_indexplot(mydata, plotAll = TRUE, row_order = "c")
custom_order <- c(19:21, 22:24, 1:18, 25:27)
ggInterval_indexplot(mydata["AD"], plotAll = TRUE,
                     row_order = "u", user_order = custom_order)
ggInterval_indexplot(mydata, plotAll = TRUE, row_order = "c", labels = TRUE)
```

---

ggInterval_lineplot *Interval-valued line plot*

Description

Visualize interval-valued data along an ordered horizontal axis. A line can connect the centers of intervals at each position, with crossbars or errorbars indicating the interval range. When the horizontal axis is time, the display becomes an interval-valued time-series plot.

Usage

```
ggInterval_lineplot(data = NULL, mapping = aes(NULL),
barWidth = 0.5, add_line = TRUE)

ggInterval_tsplot(data = NULL, mapping = aes(NULL),
barWidth = 0.5, add_line = TRUE)
```

Arguments

`data`	A ggInterval object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggInterval data.
`mapping`	Set of aesthetic mappings created by aes() or aes_(). Must include x for the ordered variable and y for the interval variable. Optional aesthetics include group and fill for multiple line series. It is the same as the mapping of ggplot2.
`barWidth`	The width of the crossbar or errorbar indicating the interval range at each x position. Default is 0.5.
`add_line`	Logical; if TRUE (default), connect interval centers with a line. Set FALSE when adding a customized line layer with `geom_line(...)`.

Value

Return a ggplot2 object.

Examples

```
if (requireNamespace("TTR", quietly = TRUE)) {
  data("ttrc", package = "TTR")
  stock.data <- subset(
    ttrc[, c("Date", "Close", "Low", "High")],
    format(Date, "%Y-%m") %in% c("1985-01", "1985-02", "1985-03")
  )
  stock.data$Month <- factor(
    month.abb[as.integer(format(stock.data$Date, "%m"))],
    levels = month.abb[1:3]
  )
  stock.data$Day <- as.integer(format(stock.data$Date, "%d"))
  stock.i <- classic2sym(
    stock.data,
    groupby = "customize",
    minData = stock.data$Low,
    maxData = stock.data$High
  )
  ggInterval_lineplot(stock.i, aes(y = V1, x = Day), barWidth = 0.6) +
    geom_point(aes(y = Close), shape = 21, fill = "#D95F02",
               color = "black", size = 1.6, stroke = 0.2) +
    coord_cartesian(xlim = c(1, 31), expand = FALSE) +
    facet_wrap(~Month, ncol = 1, scales = "free_y") +
    scale_x_continuous(breaks = c(1, 8, 15, 22, 29)) +
    labs(x = "Day of month", y = "Price") +
    ggthemes::theme_economist() +
    theme(strip.text = element_text(face = "bold"))
}
```

---

ggInterval_MMplot *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_MMplot(data = NULL,mapping = aes(NULL),
          scaleXY = "local",plotAll=FALSE)
```

Arguments

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

```
mydata <- ggInterval::facedata
ggInterval_MMplot(mydata, aes(x = AD))
ggInterval_MMplot(mydata, aes(x = AD, size = 3))
ggInterval_MMplot(mydata, plotAll = TRUE, scaleXY = "global") +
  theme_classic()
```

---

ggInterval_PCA *Vertices PCA for interval data*

Description

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

Usage

```
ggInterval_PCA(data = NULL,mapping = aes(NULL),plot=TRUE,
                       concepts_group=NULL, poly = FALSE, adjust = TRUE,
                       showLabels = TRUE, labelSize = 3,
                       checkOverlap = FALSE)
```

Arguments

data	A ggInterval object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggInterval 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
showLabels	Logical. If TRUE, label interval scores in the plot.
labelSize	Numeric text size used when showLabels = TRUE.
checkOverlap	Logical. Passed to geom_text() to suppress overlapping labels when needed.

Value

A ggplot object for PC1, PC2, and interval scores with related outputs.

- 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

```
Subjects <- substr(rownames(facedata), 1, 3)
p <- ggInterval_PCA(facedata, plot = FALSE, concepts_group = Subjects)
p$ggplotPCA
p$scores_interval
ggInterval_PCA(facedata, poly = TRUE, concepts_group = Subjects)
```

---

ggInterval_radarplot *Radar plot for symbolic data*

Description

Visualize interval-valued and mixed symbolic data with a radar-style display. Interval-valued variables are represented through their lower and upper bounds, while modal multivalued variables can be shown with stacked bars. When `type = "quantile"`, the function summarizes all observations of each interval-valued variable through nested empirical quantile bands.

Usage

```
ggInterval_radarplot(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 ggInterval object. It can also be either RSDA object or classical data frame(not recommended),which will be automatically convert to ggInterval data.
`layerNumber`	Number of concentric guide layers in the radar display.

inOneFig	Logical. If TRUE, draw all selected observations in one figure. Otherwise, generate separate figures for the selected observations.
showLegend	Logical. Whether to show the legend.
showXYLabs	Logical. Whether to show the x- and y-axis labels.
plotPartial	Numeric vector giving the row indices to plot. If NULL, all observations are used. This argument is ignored when type = "quantile" because the quantile plot summarizes all observations.
alpha	Alpha transparency for filled elements.
base_circle	Logical. If TRUE, add inner guide circles.
base_lty	Line type used in the base figure.
addText	Logical. Whether to add interval-valued labels to the plot.
type	Radar representation. Use "default" for polygon intervals, "rect" for rectangle intervals, or "quantile" for the quantile-radar display.
quantileNum	Number of quantile layers when type = "quantile".
Drift	Drift term controlling where the radar values begin.
addText_modal	Logical. Whether to add labels for modal multivalued variables.
addText_modal.p	Logical. Whether to add modal percentages.

Examples

```
# must specify plotPartial to the rows you want to plot
Environment.n <- Environment[, 5:17]
ggInterval_radarplot(Environment.n,
  plotPartial = 2,
  showLegend = FALSE,
  base_circle = TRUE,
  base_lty = 2,
  addText = FALSE
) +
  labs(title = "") +
  scale_fill_manual(values = c("gray50")) +
  scale_color_manual(values = c("red"))

ggInterval_radarplot(Environment,
  plotPartial = 2,
  showLegend = FALSE,
  base_circle = FALSE,
  base_lty = 1,
  addText = TRUE,
  type = "rect"
) +
  labs(title = "") +
  scale_fill_manual(values = c("gray50")) +
  scale_color_manual(values = c("gray50"))

ggInterval_radarplot(
```

```
  facedata,
  base_circle = FALSE,
  base_lty = 1,
  type = "quantile",
  quantileNum = 5,
  showLegend = TRUE,
  Drift = 0
) +
  scale_fill_brewer(palette = "Greys") +
  labs(title = "", fill = "Quantiles")
```

`ggInterval_scatterMatrix` *Scatterplot matrix for interval-valued data*

Description

Visualize all continuous interval-valued variables with a matrix of pairwise interval scatterplots. Each off-diagonal panel shows interval rectangles for one variable pair, whereas each diagonal panel displays the variable name. This function automatically filters out non-interval variables and plots all remaining continuous interval variables, so explicit `x` and `y` mappings are not required. It is not recommended to apply the function to too many variables because the full pairwise matrix becomes computationally expensive and visually crowded.

Usage

```
ggInterval_scatterMatrix(data = NULL, mapping = aes(NULL),
showLegend = FALSE, borderLinewidth = 0.08)
```

Arguments

`data`	A ggInterval object. It can also be either RSDA object or classical data frame,which will be automatically convert to ggInterval 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.
`borderLinewidth`	Numeric border width used for the interval rectangles.

Value

Return a ggplot2 object.

Examples

```
mydata <- ggInterval::facedata
ggInterval_scatterMatrix(mydata[, 1:3])
ggInterval_scatterMatrix(
  mydata[, 1:3],
  aes(fill = "black", alpha = 0.2),
  borderLinewidth = 0.15
)
```

---

ggInterval_scatterplot

Scatter plot for two continuous interval variables

Description

Visualize the distribution of two continuous interval-valued variables with rectangles whose widths and heights represent the corresponding intervals.

Usage

```
ggInterval_scatterplot(data = NULL,mapping = aes(NULL),
showLabels = TRUE, labelSize = 3,
labelPosition = "topright", labelNudgeX = 0, labelNudgeY = 0,
checkOverlap = FALSE, ...)
```

Arguments

data	A ggInterval object.It can also be either RSDA object or classical data frame, which will be automatically convert to ggInterval 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.
showLabels	Logical. If TRUE, add row labels to the rectangles.
labelSize	Numeric text size used when showLabels = TRUE.
labelPosition	Character label position used when showLabels = TRUE. One of "topright", "topleft", "bottomright", or "bottomleft".
labelNudgeX	Numeric horizontal adjustment applied to label positions.
labelNudgeY	Numeric vertical adjustment applied to label positions.
checkOverlap	Logical. Passed to geom_text() to suppress overlapping labels when needed.
...	Others in ggplot2.

Value

Return a ggplot2 object.

Examples

```
Subjects <- substr(rownames(facedata), 1, 3)
ggInterval_scatterplot(facedata, aes(x = AD, y = BC))
ggInterval_scatterplot(
  facedata,
  aes(x = AD, y = BC, fill = Subjects),
  showLabels = TRUE,
  labelSize = 2.6,
  labelPosition = "topright",
  labelNudgeX = 0.8,
  labelNudgeY = 0.15,
  checkOverlap = TRUE,
  col = "black"
)
p <- ggInterval_scatterplot(facedata[1:10, ], aes(x = AD, y = BC, alpha = 0.2))
p + scale_fill_manual(
  labels = rownames(facedata)[1:10],
  values = rainbow(10),
  name = "Group"
)
```

---

iris.i *iris.i data example*

Description

iris.i interval data example.

Usage

```
data(iris.i)
```

Format

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

Examples

```
data(iris.i)
ggInterval_indexplot(iris.i, aes(x = Sepal.Length))
```

mtcars.i *mtcars.i data example*

Description

mtcars.i interval and modal data example.

Usage

```
data(mtcars.i)
```

Format

An object of class `symbolic_tbl` (inherits from `tbl_df`, `tbl`, `data.frame`, `symbolic_tbl`) with 5 rows and 11 columns.

Examples

```
data(mtcars.i)
ggInterval_indexplot(mtcars.i, aes(x = mpg))
```

mushroom *mushroom data example*

Description

mushroom interval data example.

Usage

```
data(mushroom)
```

Format

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

Source

Adapted from `RSDA::mushroom`; see https://CRAN.R-project.org/package=RSDA.

References

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Examples

```
data(mushroom)

ggInterval_scatterplot(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.

Source

Adapted from `RSDA::oils`; see https://CRAN.R-project.org/package=RSDA.

References

Cazes, Pierre; Chouakria, Assia; Diday, Edwin; and Schektman, Youri (1997). Extension de l'analyse en composantes principales a des donnees de type intervalle. *Revue de Statistique Appliquee*, 45(3), 5–24.

Examples

```
data(oils)

ggInterval_scatterplot(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 ggInterval object will do for plot(ggplot) and RSDA's analysis.

Usage

```
RSDA2sym(data=NULL,rawData=NULL)
```

Arguments

data	an interval data, which may transfrom 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 "ggInterval", which has interval-valued data and related outputs 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<-ggInterval::Cardiological
mySym<-RSDA2sym(r)
mySym$intervalData
```

scale *scale for symbolic data table*

Description

scale for symbolic data table

Usage

```
scale(x, ...)

## Default S3 method:
scale(x, center = TRUE, scale = TRUE, ...)

## S3 method for class 'symbolic_tbl'
scale(x, ...)

## S3 method for class 'symbolic_interval'
scale(x, ...)
```

Arguments

x	A ggInterval object. It can also be either RSDA object or classical data frame, which will be automatically convert to ggInterval data.
...	Used by other R function.
center	same as base::scale, either a logical value or numeric-alike vector of length equal to the number of columns of x, where nmeric-alike means that as.numeric(.) will be applied successfully if is.numeric(.) is not true.
scale	same as base::scale, either a logical value or a numeric-alike vector of length equal to the number of columns of x.

Value

Return a scale ggInterval object.

Examples

```
#For all interval-valued
scale(facedata)

#For both interval-valued and modal multi-valued
scale(mtcars.i)
```

sd *Generic function for the standard deviation*

Description

This function compute the symbolic standard deviation.

Usage

```
sd(x, ...)

## Default S3 method:
sd(x, na.rm = FALSE, ...)

## S3 method for class 'symbolic_tbl'
sd(
  x,
  method = c("billard", "centers", "interval", "B", "BD", "BG"),
  na.rm = FALSE,
  ...
)

## S3 method for class 'symbolic_interval'
sd(
  x,
  method = c("billard", "centers", "interval", "B", "BD", "BG"),
  na.rm = FALSE,
  ...
)
```

Arguments

x	First symbolic variables.
...	As in R sd function.
na.rm	As in R sd function.
method	The method to use.

Details

Supported interval-valued methods are:

- "billard": standard deviation based on the Billard univariate variance formula.
- "centers": standard deviation of interval centers.
- "interval": interval-valued standard deviation obtained by standardizing lower and upper bounds separately.
- "B": method-matched standard deviation defined by $\sqrt{C_B(X, X)}$.

- "BD": method-matched standard deviation defined by $\sqrt{C_{BD}(X,X)}$.
- "BG": method-matched standard deviation defined by $\sqrt{C_{BG}(X,X)}$.

Value

Return a real number.

References

Bertrand, Patrice and Goupil, Francoise (2000). Descriptive Statistics for Symbolic Data. In Hans-Hermann Bock and Edwin Diday (eds.), *Analysis of Symbolic Data*, pp. 106–124. Berlin and Heidelberg: Springer.

Billard, Lynne and Diday, Edwin (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Chichester, UK: John Wiley and Sons.

Billard, Lynne (2008). Sample covariance functions for complex quantitative data. In *Proceedings of the World IASC Conference*, pp. 157–163, Yokohama, Japan.

---

summary *summary for symbolic data table*

Description

summary for symbolic data table

Usage

```
summary(object, ...)

## Default S3 method:
summary(object, ...)

## S3 method for class 'symbolic_tbl'
summary(object, ...)

## S3 method for class 'symbolic_interval'
summary(object, ...)

## S3 method for class 'symbolic_modal'
summary(object, summary_fun = "mean", ...)
```

Arguments

object	an object for which a summary is desired.
...	additional arguments affecting the summary produced.
summary_fun	only works when the symbolic_modal class input, it determine which summary function be applied for each modal.

Value

Return a summary table.

Examples

```
#For all interval-valued
summary(facedata)

#For both interval-valued and modal multi-valued
summary(Environment)

summary(Environment$URBANICITY, summary_fun = "quantile")
```

Index