

Package ‘dlookr’

September 2, 2021

Type Package

Title Tools for Data Diagnosis, Exploration, Transformation

Version 0.5.1

Description A collection of tools that support data diagnosis, exploration, and transformation. Data diagnostics provides information and visualization of missing values and outliers and unique and negative values to help you understand the distribution and quality of your data. Data exploration provides information and visualization of the descriptive statistics of univariate variables, normality tests and outliers, correlation of two variables, and relationship between target variable and predictor. Data transformation supports binning for categorizing continuous variables, imputates missing values and outliers, resolving skewness. And it creates automated reports that support these three tasks.

License GPL-2 | file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.2.0)

Imports dplyr (>= 0.7.6), extrafont (>= 0.17), ggplot2 (>= 3.0.0), grid, gridExtra, hrbrthemes (>= 0.8.0), htmltools, kableExtra, knitr (>= 1.22), methods, mice, pagedown (>= 0.15), partykit, purrr, reactable, rlang, rmarkdown, shiny, tibble, tidyr, tidyselect, utils

Suggests DBI, classInt, dbplyr, forecast (>= 8.3), ISLR, nycflights13, randomForest, rpart, prettydoc, RSQLite

Author Choonghyun Ryu [aut, cre]

Maintainer Choonghyun Ryu <choonghyun.ryu@gmail.com>

BugReports <https://github.com/choonghyunryu/dlookr/issues>

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-09-02 21:10:16 UTC

R topics documented:

dlookr-package	4
binning	5
binning_by	7
compare_category	9
compare_numeric	12
correlate	15
correlate.tbl_dbi	17
describe	20
describe.tbl_dbi	22
diagnose	25
diagnose.tbl_dbi	27
diagnose_category	29
diagnose_category.tbl_dbi	32
diagnose_numeric	35
diagnose_numeric.tbl_dbi	37
diagnose_outlier	39
diagnose_outlier.tbl_dbi	41
diagnose_paged_report	43
diagnose_paged_report.tbl_dbi	46
diagnose_report	49
diagnose_report.tbl_dbi	51
diagnose_sparse	54
diagnose_web_report	55
diagnose_web_report.tbl_dbi	58
dlookr_orange_paged	61
dlookr_tmpl_html	61
eda_paged_report	62
eda_paged_report.tbl_dbi	64
eda_report	68
eda_report.tbl_dbi	70
eda_web_report	73
eda_web_report.tbl_dbi	76
entropy	79
extract	79
find_class	80
find_na	82
find_outliers	83
find_skewness	84
get_class	85
get_column_info	86
get_os	87
get_percentile	87
get_transform	88
heartfailure	90
import_liberation	91
imputate_na	91

impute_outlier	93
jobchange	95
jsd	96
kld	97
kurtosis	98
normality	98
normality.tbl_dbi	100
overview	103
performance_bin	104
plot.bins	107
plot.compare_category	108
plot.compare_numeric	109
plot.imputation	110
plot.optimal_bins	112
plot.overview	113
plot.performance_bin	114
plot.relate	115
plot.transform	117
plot.univar_category	118
plot.univar_numeric	119
plot_bar_category	120
plot_box_numeric	123
plot_correlate	124
plot_correlate.tbl_dbi	126
plot_hist_numeric	128
plot_na_hclust	130
plot_na_intersect	131
plot_na_pareto	133
plot_normality	134
plot_normality.tbl_dbi	137
plot_outlier	140
plot_outlier.target_df	142
plot_outlier.tbl_dbi	144
plot_qq_numeric	146
print.relate	148
relate	149
skewness	152
summary.bins	153
summary.compare_category	154
summary.compare_numeric	156
summary.imputation	159
summary.optimal_bins	160
summary.overview	162
summary.performance_bin	163
summary.transform	164
summary.univar_category	165
summary.univar_numeric	167
target_by	168

target_by.tbl_dbi	170
transform	172
transformation_paged_report	173
transformation_report	176
transformation_web_report	178
univar_category	180
univar_numeric	181

Index	185
--------------	------------

dlookr-package	<i>dlookr: Tools for Data Diagnosis, Exploration, Transformation</i>
----------------	--

Description

dlookr provides data diagnosis, data exploration and transformation of variables during data analysis.

Details

It has three main goals:

- When data is acquired, it is possible to judge whether data is erroneous or to select a variable to be corrected or removed through data diagnosis.
- Understand the distribution of data in the EDA process. It can also understand the relationship between target variables and predictor variables for the prediction model.
- Imputes missing value and outlier to standardization and resolving skewness. And, To convert a continuous variable to a categorical variable, bin the continuous variables.

To learn more about dlookr, start with the vignettes: `'browseVignettes(package = "dlookr")'`

Author(s)

Maintainer: Choonghyun Ryu <choonghyun.ryu@gmail.com>

See Also

Useful links:

- Report bugs at <https://github.com/choonghyunryu/dlookr/issues>

binning

*Binning the Numeric Data***Description**

The `binning()` converts a numeric variable to a categorization variable.

Usage

```
binning(
  x,
  nbins,
  type = c("quantile", "equal", "pretty", "kmeans", "bclust"),
  ordered = TRUE,
  labels = NULL,
  approxy.lab = TRUE
)
```

Arguments

<code>x</code>	numeric. numeric vector for binning.
<code>nbins</code>	integer. number of intervals(bins). required. if missing, <code>nclass.Sturges</code> is used.
<code>type</code>	character. binning method. Choose from "quantile", "equal", "equal", "pretty", "kmeans" and "bclust". The "quantile" sets breaks with quantiles of the same interval. The "equal" sets breaks at the same interval. The "pretty" chooses a number of breaks not necessarily equal to <code>nbins</code> using <code>base::pretty</code> function. The "kmeans" uses <code>stats::kmeans</code> function to generate the breaks. The "bclust" uses <code>e1071::bclust</code> function to generate the breaks using bagged clustering. "kmeans" and "bclust" was implemented by <code>classInt::classIntervals()</code> function.
<code>ordered</code>	logical. whether to build an ordered factor or not.
<code>labels</code>	character. the label names to use for each of the bins.
<code>approxy.lab</code>	logical. If TRUE, large number breaks are approximated to pretty numbers. If FALSE, the original breaks obtained by <code>type</code> are used.

Details

This function is useful when used with the `mutate/transmute` function of the `dplyr` package.

Value

An object of `bins` class. Attributes of `bins` class is as follows.

- `type` : binning type, "quantile", "equal", "pretty", "kmeans", "bclust".
- `breaks` : the number of intervals into which `x` is to be cut.
- `levels` : levels of binned value.
- `raw` : raw data, numeric vector corresponding to `x` argument.

"bins" class attributes information

Attributes of the "bins" class that is as follows.

- class : "bins"
- levels : levels of factor or ordered factor
- type : binning method
- breaks : breaks for binning
- raw : raw data before binning

See vignette("transformation") for an introduction to these concepts.

See Also

[binning_by](#), [print.bins](#), [summary.bins](#), [plot.bins](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA

# Binning the carat variable. default type argument is "quantile"
bin <- binning(heartfailure2$platelets)
# Print bins class object
bin

# Summarise bins class object
summary(bin)

# Plot bins class object
plot(bin)

# Using labels argument
bin <- binning(heartfailure2$platelets, nbins = 4,
              labels = c("LQ1", "UQ1", "LQ3", "UQ3"))
bin
# Using another type argument
bin <- binning(heartfailure2$platelets, nbins = 5, type = "equal")
bin
# bin <- binning(heartfailure2$platelets, nbins = 5, type = "pretty")
# bin
# bin <- binning(heartfailure2$platelets, nbins = 5, type = "kmeans")
# bin
# bin <- binning(heartfailure2$platelets, nbins = 5, type = "bclust")
# bin

x <- sample(1:1000, size = 50) * 12345679
bin <- binning(x)
bin
bin <- binning(x, approxy.lab = FALSE)
```

```

bin

# extract binned results
extract(bin)

# -----
# Using pipes & dplyr
# -----
library(dplyr)

# Compare binned frequency by death_event
heartfailure2 %>%
  mutate(platelets_bin = binning(heartfailure2$platelets) %>%
         extract()) %>%
  group_by(death_event, platelets_bin) %>%
  summarise(freq = n()) %>%
  arrange(desc(freq)) %>%
  head(10)

# Compare binned frequency by death_event using Viz
heartfailure2 %>%
  mutate(platelets_bin = binning(heartfailure2$platelets) %>%
         extract()) %>%
  target_by(death_event) %>%
  relate(platelets_bin) %>%
  plot()

```

binning_by

Optimal Binning for Scoring Modeling

Description

The `binning_by()` finding intervals for numerical variable using optimal binning. Optimal binning categorizes a numeric characteristic into bins for ulterior usage in scoring modeling.

Usage

```
binning_by(.data, y, x, p = 0.05, ordered = TRUE, labels = NULL)
```

Arguments

<code>.data</code>	a data frame.
<code>y</code>	character. name of binary response variable(0, 1). The variable must contain only the integers 0 and 1 as element. However, in the case of factor having two levels, it is performed while type conversion is performed in the calculation process.
<code>x</code>	character. name of continuous characteristic variable. At least 5 different values. and Inf is not allowed.

p	numeric. percentage of records per bin. Default 5% (0.05). This parameter only accepts values greater than 0.00 (0%) and lower than 0.50 (50%).
ordered	logical. whether to build an ordered factor or not.
labels	character. the label names to use for each of the bins.

Details

This function is useful when used with the mutate/transmute function of the dplyr package. And this function is implemented using `smbinning()` function of `smbinning` package.

Value

an object of "optimal_bins" class. Attributes of "optimal_bins" class is as follows.

- class : "optimal_bins".
- type : binning type, "optimal".
- breaks : numeric. the number of intervals into which x is to be cut.
- levels : character. levels of binned value.
- raw : numeric. raw data, x argument value.
- ivtable : data.frame. information value table.
- iv : numeric. information value.
- target : integer. binary response variable.

attributes of "optimal_bins" class

Attributes of the "optimal_bins" class that is as follows.

- class : "optimal_bins".
- levels : character. factor or ordered factor levels
- type : character. binning method
- breaks : numeric. breaks for binning
- raw : numeric. before the binned the raw data
- ivtable : data.frame. information value table
- iv : numeric. information value
- target : integer. binary response variable

See vignette("transformation") for an introduction to these concepts.

See Also

[binning](#), [plot.optimal_bins](#).

Examples

```
library(dplyr)

# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA

# optimal binning using character
bin <- binning_by(heartfailure2, "death_event", "creatinine")

# optimal binning using name
bin <- binning_by(heartfailure2, death_event, creatinine)
bin

# performance table
attr(bin, "performance")

# summary optimal_bins class
summary(bin)

# visualize all information for optimal_bins class
# plot(bin)

# visualize WoE information for optimal_bins class
# plot(bin, type = "WoE")

# visualize all information without typographic
# plot(bin, typographic = FALSE)

# extract binned results
# extract(bin) %>%
# head(20)
```

compare_category	<i>Compare categorical variables</i>
------------------	--------------------------------------

Description

The `compare_category()` compute information to examine the relationship between categorical variables.

Usage

```
compare_category(.data, ...)

## S3 method for class 'data.frame'
compare_category(.data, ...)
```

Arguments

`.data` a data.frame or a [tbl_df](#).

`...` one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

It is important to understand the relationship between categorical variables in EDA. `compare_category()` compares relations by pair combination of all categorical variables. and return `compare_category` class that based list object.

Value

An object of the class as `compare` based list. The information to examine the relationship between categorical variables is as follows each components.

- `var1` : factor. The level of the first variable to compare. 'var1' is the name of the first variable to be compared.
- `var2` : factor. The level of the second variable to compare. 'var2' is the name of the second variable to be compared.
- `n` : integer. frequency by var1 and var2.
- `rate` : double. relative frequency.
- `first_rate` : double. relative frequency in first variable.
- `second_rate` : double. relative frequency in second variable.

Attributes of return object

Attributes of `compare_category` class is as follows.

- `variables` : character. List of variables selected for comparison.
- `combination` : matrix. It consists of pairs of variables to compare.

See Also

[summary.compare_category](#), [print.compare_category](#), [plot.compare_category](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

library(dplyr)

# Compare the all categorical variables
```

```
all_var <- compare_category(heartfailure2)

# Print compare_numeric class objects
all_var

# Compare the categorical variables that case of joint the death_event variable
all_var %>%
  "["(grep("death_event", names(all_var)))

# Compare the two categorical variables
two_var <- compare_category(heartfailure2, smoking, death_event)

# Print compare_category class objects
two_var

# Filtering the case of smoking included NA
two_var %>%
  "[["(1) %>%
  filter(!is.na(smoking))

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Summary by returned objects
stat

# component of table
stat$table

# component of chi-square test
stat$chisq

# component of chi-square test
summary(all_var, "chisq")

# component of chi-square test (first, third case)
summary(all_var, "chisq", pos = c(1, 3))

# component of relative frequency table
summary(all_var, "relative")

# component of table without missing values
summary(all_var, "table", na.rm = TRUE)

# component of table include marginal value
margin <- summary(all_var, "table", marginal = TRUE)
margin

# component of chi-square test
summary(two_var, method = "chisq")

# verbose is FALSE
summary(all_var, "chisq", verbose = FALSE)
```

```

#' # Using pipes & dplyr -----
# If you want to use dplyr, set verbose to FALSE
summary(all_var, "chisq", verbose = FALSE) %>%
  filter(p.value < 0.26)

# Extract component from list by index
summary(all_var, "table", na.rm = TRUE, verbose = FALSE) %>%
  "[["(1)

# Extract component from list by name
summary(all_var, "table", na.rm = TRUE, verbose = FALSE) %>%
  "[["("smoking vs death_event")

# plot all pair of variables
# plot(all_var)

# plot a pair of variables
# plot(two_var)

# plot all pair of variables by prompt
# plot(all_var, prompt = TRUE)

# plot a pair of variables
# plot(two_var, las = 1)

```

compare_numeric	<i>Compare numerical variables</i>
-----------------	------------------------------------

Description

The `compare_numeric()` compute information to examine the relationship between numerical variables.

Usage

```

compare_numeric(.data, ...)

## S3 method for class 'data.frame'
compare_numeric(.data, ...)

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

It is important to understand the relationship between numerical variables in EDA. `compare_numeric()` compares relations by pair combination of all numerical variables. and return `compare_numeric` class that based list object.

Value

An object of the class as `compare` based list. The information to examine the relationship between numerical variables is as follows each components. - correlation component : Pearson's correlation coefficient.

- `var1` : factor. The level of the first variable to compare. 'var1' is the name of the first variable to be compared.
- `var2` : factor. The level of the second variable to compare. 'var2' is the name of the second variable to be compared.
- `coef_corr` : double. Pearson's correlation coefficient.

- linear component : linear model summaries

- `var1` : factor. The level of the first variable to compare. 'var1' is the name of the first variable to be compared.
- `var2` : factor. The level of the second variable to compare. 'var2' is the name of the second variable to be compared.
- `r.squared` : double. The percent of variance explained by the model.
- `adj.r.squared` : double. `r.squared` adjusted based on the degrees of freedom.
- `sigma` : double. The square root of the estimated residual variance.
- `statistic` : double. F-statistic.
- `p.value` : double. p-value from the F test, describing whether the full regression is significant.
- `df` : integer degrees of freedom.
- `logLik` : double. the log-likelihood of data under the model.
- `AIC` : double. the Akaike Information Criterion.
- `BIC` : double. the Bayesian Information Criterion.
- `deviance` : double. deviance.
- `df.residual` : integer residual degrees of freedom.

Attributes of return object

Attributes of `compare_numeric` class is as follows.

- `raw` : a data.frame or a `tbl_df`. Data containing variables to be compared. Save it for visualization with `plot.compare_numeric()`.
- `variables` : character. List of variables selected for comparison.
- `combination` : matrix. It consists of pairs of variables to compare.

See Also

[correlate](#), [summary.compare_numeric](#), [print.compare_numeric](#), [plot.compare_numeric](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure[, c("platelets", "creatinine", "sodium")]

library(dplyr)
# Compare the all numerical variables
all_var <- compare_numeric(heartfailure2)

# Print compare_numeric class object
all_var

# Compare the correlation that case of joint the sodium variable
all_var %>%
  "$"(correlation) %>%
  filter(var1 == "sodium" | var2 == "sodium") %>%
  arrange(desc(abs(coef_corr)))

# Compare the correlation that case of abs(coef_corr) > 0.1
all_var %>%
  "$"(correlation) %>%
  filter(abs(coef_corr) > 0.1)

# Compare the linear model that case of joint the sodium variable
all_var %>%
  "$"(linear) %>%
  filter(var1 == "sodium" | var2 == "sodium") %>%
  arrange(desc(r.squared))

# Compare the two numerical variables
two_var <- compare_numeric(heartfailure2, sodium, creatinine)

# Print compare_numeric class objects
two_var

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Just correlation
summary(all_var, method = "correlation")

# Just correlation condition by r > 0.1
summary(all_var, method = "correlation", thres_corr = 0.1)

# linear model summaries condition by R^2 > 0.05
summary(all_var, thres_rs = 0.05)

# verbose is FALSE
summary(all_var, verbose = FALSE)
```

```

# plot all pair of variables
# plot(all_var)

# plot a pair of variables
# plot(two_var)

# plot all pair of variables by prompt
# plot(all_var, prompt = TRUE)

# plot a pair of variables not focuses on typographic elements
# plot(two_var, typographic = FALSE)

```

correlate

Compute the correlation coefficient between two numerical data

Description

The `correlate()` compute Pearson's the correlation coefficient of the numerical data.

Usage

```

correlate(.data, ...)

## S3 method for class 'data.frame'
correlate(.data, ..., method = c("pearson", "kendall", "spearman"))

## S3 method for class 'grouped_df'
correlate(.data, ..., method = c("pearson", "kendall", "spearman"))

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>correlate()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing. See <code>vignette("EDA")</code> for an introduction to these concepts.
<code>method</code>	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.

Details

This function is useful when used with the `group_by()` function of the `dplyr` package. If you want to compute by level of the categorical data you are interested in, rather than the whole observation, you can use `grouped_df` as the `group_by()` function. This function is computed `stats::cor()` function by use = "pairwise.complete.obs" option.

Correlation coefficient information

The information derived from the numerical data compute is as follows.

- `var1` : names of numerical variable
- `var2` : name of the corresponding numeric variable
- `coef_corr` : Pearson's correlation coefficient

See Also

[cor](#), [correlate.tbl_dbi](#).

Examples

```
# Correlation coefficients of all numerical variables
correlate(heartfailure)

# Select the variable to compute
correlate(heartfailure, creatinine, sodium)
correlate(heartfailure, -creatinine, -sodium)
correlate(heartfailure, "creatinine", "sodium")
correlate(heartfailure, 1)
# Non-parametric correlation coefficient by kendall method
correlate(heartfailure, creatinine, method = "kendall")

# Using dplyr::grouped_dt
library(dplyr)

gdata <- group_by(heartfailure, smoking, death_event)
correlate(gdata, "creatinine")
correlate(gdata)

# Using pipes -----
# Correlation coefficients of all numerical variables
heartfailure %>%
  correlate()
# Positive values select variables
heartfailure %>%
  correlate(creatinine, sodium)
# Negative values to drop variables
heartfailure %>%
  correlate(-creatinine, -sodium)
# Positions values select variables
heartfailure %>%
  correlate(1)
```



```

# Positions values select variables
heartfailure %>%
  correlate(-1, -3, -5, -7)
# Non-parametric correlation coefficient by spearman method
heartfailure %>%
  correlate(creatinine, sodium, method = "spearman")

# -----
# Correlation coefficient
# that eliminates redundant combination of variables
heartfailure %>%
  correlate() %>%
  filter(as.integer(var1) > as.integer(var2))

heartfailure %>%
  correlate(creatinine, sodium) %>%
  filter(as.integer(var1) > as.integer(var2))

# Using pipes & dplyr -----
# Compute the correlation coefficient of Sales variable by 'smoking'
# and 'death_event' variables. And extract only those with absolute
# value of correlation coefficient is greater than 0.2
heartfailure %>%
  group_by(smoking, death_event) %>%
  correlate(creatinine) %>%
  filter(abs(coef_corr) >= 0.2)

# extract only those with 'smoking' variable level is "Yes",
# and compute the correlation coefficient of 'Sales' variable
# by 'hblood_pressure' and 'death_event' variables.
# And the correlation coefficient is negative and smaller than 0.5
heartfailure %>%
  filter(smoking == "Yes") %>%
  group_by(hblood_pressure, death_event) %>%
  correlate(creatinine) %>%
  filter(coef_corr < 0) %>%
  filter(abs(coef_corr) > 0.5)

```

correlate.tbl_dbi *Compute the correlation coefficient between two numerical data*

Description

The `correlate()` compute Pearson's the correlation coefficient of the numerical(INTEGER, NUMBER, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```

## S3 method for class 'tbl_dbi'
correlate(

```

```

.data,
...,
in_database = FALSE,
collect_size = Inf,
method = c("pearson", "kendall", "spearman")
)

```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>correlate()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .
<code>method</code>	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated. See <code>vignette("EDA")</code> for an introduction to these concepts.

Details

This function is useful when used with the `group_by()` function of the `dplyr` package. If you want to compute by level of the categorical data you are interested in, rather than the whole observation, you can use `grouped_df` as the `group_by()` function. This function is computed `stats::cor()` function by use = "pairwise.complete.obs" option.

Correlation coefficient information

The information derived from the numerical data compute is as follows.

- `var1` : names of numerical variable
- `var2` : name of the corresponding numeric variable
- `coef_corr` : Pearson's correlation coefficient

See Also

[correlate.data.frame](#), [cor](#).

Examples

```

library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Correlation coefficients of all numerical variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate()

# Positive values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate(platelets, sodium)

# Negative values to drop variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate(-platelets, -sodium, collect_size = 200)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate(1)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate(-1, -2, -3, -5, -6)

# -----
# Correlation coefficient
# that eliminates redundant combination of variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate() %>%
  filter(as.integer(var1) > as.integer(var2))

con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  correlate(platelets, sodium) %>%
  filter(as.integer(var1) > as.integer(var2))

# Using pipes & dplyr -----
# Compute the correlation coefficient of creatinine variable by 'hblood_pressure'
# and 'death_event' variables. And extract only those with absolute
# value of correlation coefficient is greater than 0.2

```

```

con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(hblood_pressure, death_event) %>%
  correlate(creatinine) %>%
  filter(abs(coef_corr) >= 0.2)

# extract only those with 'hblood_pressure' variable level is "Yes",
# and compute the correlation coefficient of 'creatinine' variable
# by 'sex' and 'death_event' variables.
# And the correlation coefficient is negative and smaller than -0.3
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  filter(hblood_pressure == "Yes") %>%
  group_by(sex, death_event) %>%
  correlate(creatinine) %>%
  filter(coef_corr < 0) %>%
  filter(abs(coef_corr) > 0.3)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

describe

Compute descriptive statistic

Description

The `describe()` compute descriptive statistic of numeric variable for exploratory data analysis.

Usage

```

describe(.data, ...)

## S3 method for class 'data.frame'
describe(.data, ..., statistics = NULL, quantiles = NULL)

## S3 method for class 'grouped_df'
describe(.data, ..., statistics = NULL, quantiles = NULL)

```

Arguments

`.data` a `data.frame` or a `tbl_df` or a `grouped_df`.

`...` one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, `describe()` will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

See `vignette("EDA")` for an introduction to these concepts.

statistics	character. the name of the descriptive statistic to calculate. The defaults is <code>c("mean", "sd", "se_mean", "IQR", "skewness", "kurtosis", "quantiles")</code>
quantiles	numeric. list of quantiles to calculate. The values of elements must be between 0 and 1. and to calculate quantiles, you must include "quantiles" in the statistics argument value. The default is <code>c(0, .01, .05, 0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 0.95, 0.99, 1)</code> .

Details

This function is useful when used with the [group_by](#) function of the dplyr package. If you want to calculate the statistic by level of the categorical data you are interested in, rather than the whole statistic, you can use `grouped_df` as the `group_by()` function.

Value

An object of the same class as `.data`.

Descriptive statistic information

The information derived from the numerical data `describe` is as follows.

- `n` : number of observations excluding missing values
- `na` : number of missing values
- `mean` : arithmetic average
- `sd` : standard deviation
- `se_mean` : standard error mean. sd/\sqrt{n}
- `IQR` : interquartile range (Q3-Q1)
- `skewness` : skewness
- `kurtosis` : kurtosis
- `p25` : Q1. 25% percentile
- `p50` : median. 50% percentile
- `p75` : Q3. 75% percentile
- `p01, p05, p10, p20, p30` : 1%, 5%, 20%, 30% percentiles
- `p40, p60, p70, p80` : 40%, 60%, 70%, 80% percentiles
- `p90, p95, p99, p100` : 90%, 95%, 99%, 100% percentiles

See Also

[describe.tbl_dbi](#), [diagnose_numeric.data.frame](#).

Examples

```

# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "sodium"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

# Describe descriptive statistics of numerical variables
describe(heartfailure2)

# Select the variable to describe
describe(heartfailure2, sodium, platelets, statistics = c("mean", "sd", "quantiles"))
describe(heartfailure2, -sodium, -platelets)
describe(heartfailure2, 5, statistics = c("mean", "sd", "quantiles"), quantiles = c(0.01, 0.1))

# Using dplyr::grouped_dt
library(dplyr)

gdata <- group_by(heartfailure2, hblood_pressure, death_event)
describe(gdata, "creatinine")

# Using pipes -----
# Positive values select variables
heartfailure2 %>%
  describe(platelets, sodium, creatinine)

# Negative values to drop variables
heartfailure2 %>%
  describe(-platelets, -sodium, -creatinine)

# Using pipes & dplyr -----
# Find the statistic of all numerical variables by 'hblood_pressure' and 'death_event',
# and extract only those with 'hblood_pressure' variable level is "Yes".
heartfailure2 %>%
  group_by(hblood_pressure, death_event) %>%
  describe() %>%
  filter(hblood_pressure == "Yes")

# extract only those with 'smoking' variable level is "Yes",
# and find 'creatinine' statistics by 'hblood_pressure' and 'death_event'
heartfailure2 %>%
  filter(smoking == "Yes") %>%
  group_by(hblood_pressure, death_event) %>%
  describe(creatinine)

```

describe.tbl_dbi

Compute descriptive statistic

Description

The describe() compute descriptive statistic of numerical(INTEGER, NUMBER, etc.) column of the DBMS table through tbl_dbi for exploratory data analysis.

Usage

```
## S3 method for class 'tbl_dbi'
describe(
  .data,
  ...,
  statistics = NULL,
  quantiles = NULL,
  in_database = FALSE,
  collect_size = Inf
)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>describe()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>statistics</code>	character. the name of the descriptive statistic to calculate. The defaults is <code>c("mean", "sd", "se_mean", "IQR", "skewness", "kurtosis", "quantiles")</code>
<code>quantiles</code>	numeric. list of quantiles to calculate. The values of elements must be between 0 and 1. and to calculate quantiles, you must include "quantiles" in the statistics argument value. The default is <code>c(0, .01, .05, 0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 0.95, 0.99, 1)</code> .
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> . See <code>vignette("EDA")</code> for an introduction to these concepts.

Details

This function is useful when used with the `group_by` function of the `dplyr` package. If you want to calculate the statistic by level of the categorical data you are interested in, rather than the whole statistic, you can use `grouped_df` as the `group_by()` function.

Value

An object of the same class as `.data`.

Descriptive statistic information

The information derived from the numerical data `describe` is as follows.

- `n` : number of observations excluding missing values

- na : number of missing values
- mean : arithmetic average
- sd : standard deviation
- se_mean : standard error mean. sd/sqrt(n)
- IQR : interquartile range (Q3-Q1)
- skewness : skewness
- kurtosis : kurtosis
- p25 : Q1. 25% percentile
- p50 : median. 50% percentile
- p75 : Q3. 75% percentile
- p01, p05, p10, p20, p30 : 1%, 5%, 20%, 30% percentiles
- p40, p60, p70, p80 : 40%, 60%, 70%, 80% percentiles
- p90, p95, p99, p100 : 90%, 95%, 99%, 100% percentiles

See Also

[describe.data.frame](#), [diagnose_numeric.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Positive values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  describe(platelets, creatinine, sodium)

con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  describe(platelets, creatinine, sodium,
    statistics = c("mean", "sd", "quantiles"), quantiles = 0.1)

# Negative values to drop variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  describe(-platelets, -creatinine, -sodium, collect_size = 200)

# Using pipes & dplyr -----
# Find the statistic of all numerical variables by 'smoking' and 'death_event',
# and extract only those with 'smoking' variable level is "Yes".
```



```

con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(smoking, death_event) %>%
  describe() %>%
  filter(smoking == "Yes")

# extract only those with 'sex' variable level is "Male",
# and find 'sodium' statistics by 'smoking' and 'death_event'
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  filter(sex == "Male") %>%
  group_by(smoking, death_event) %>%
  describe(sodium)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

diagnose

Diagnose data quality of variables

Description

The `diagnose()` produces information for diagnosing the quality of the variables of `data.frame` or `tbl_df`.

Usage

```

diagnose(.data, ...)

## S3 method for class 'data.frame'
diagnose(.data, ...)

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

The scope of data quality diagnosis is information on missing values and unique value information. Data quality diagnosis can determine variables that require missing value processing. Also, the unique value information can determine the variable to be removed from the data analysis.

Value

An object of `tbl_df`.

Diagnostic information

The information derived from the data diagnosis is as follows.:

- `variables` : variable names
- `types` : data type of the variable or to select a variable to be corrected or removed through data diagnosis.
 - integer, numeric, factor, ordered, character, etc.
- `missing_count` : number of missing values
- `missing_percent` : percentage of missing values
- `unique_count` : number of unique values
- `unique_rate` : ratio of unique values. `unique_count / number of observation`

See vignette("diagnosis") for an introduction to these concepts.

See Also

[diagnose.tbl_dbi](#), [diagnose_category.data.frame](#), [diagnose_numeric.data.frame](#).

Examples

```
# Diagnosis of all variables
diagnose(jobchange)

# Select the variable to diagnose
diagnose(jobchange, gender, experience, training_hours)
diagnose(jobchange, -gender, -experience, -training_hours)
diagnose(jobchange, "gender", "experience", "training_hours")
diagnose(jobchange, 4, 9, 13)

# Using pipes -----
library(dplyr)

# Diagnosis of all variables
jobchange %>%
  diagnose()
# Positive values select variables
jobchange %>%
  diagnose(gender, experience, training_hours)
# Negative values to drop variables
jobchange %>%
  diagnose(-gender, -experience, -training_hours)
# Positions values select variables
jobchange %>%
  diagnose(4, 9, 13)
# Positions values select variables
```

```

jobchange %>%
  diagnose(-8, -9, -10)

# Using pipes & dplyr -----
# Diagnosis of missing variables
jobchange %>%
  diagnose() %>%
  filter(missing_count > 0)

```

diagnose.tbl_dbi	<i>Diagnose data quality of variables in the DBMS</i>
------------------	---

Description

The `diagnose()` produces information for diagnosing the quality of the column of the DBMS table through `tbl_dbi`.

Usage

```

## S3 method for class 'tbl_dbi'
diagnose(.data, ..., in_database = TRUE, collect_size = Inf)

```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>in_database</code>	a logical. Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory.
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .

Details

The scope of data quality diagnosis is information on missing values and unique value information. Data quality diagnosis can determine variables that require missing value processing. Also, the unique value information can determine the variable to be removed from the data analysis.

Value

An object of `tbl_df`.

Diagnostic information

The information derived from the data diagnosis is as follows.:

- variables : column names
- types : data type of the variable or to select a variable to be corrected or removed through data diagnosis.
 - integer, numeric, factor, ordered, character, etc.
- missing_count : number of missing values
- missing_percent : percentage of missing values
- unique_count : number of unique values
- unique_rate : ratio of unique values. $\text{unique_count} / \text{number of observation}$

See vignette("diagonosis") for an introduction to these concepts.

See Also

[diagnose.data.frame](#), [diagnose_category.tbl_dbi](#), [diagnose_numeric.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy jobchange to the DBMS with a table named TB_JOBCHANGE
copy_to(con_sqlite, jobchange, name = "TB_JOBCHANGE", overwrite = TRUE)

# Using pipes -----
# Diagnosis of all columns
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose()

# Positive values select columns
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose(gender, education_level, company_size)

# Negative values to drop columns
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose(-gender, -education_level, -company_size)

# Positions values select columns, and In-memory mode
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose(1, 3, 8, in_database = FALSE)
```

```

# Positions values select columns, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose(-8, -9, -10, in_database = FALSE, collect_size = 200)

# Using pipes & dplyr -----
# Diagnosis of missing variables
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose() %>%
  filter(missing_count > 0)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

diagnose_category	<i>Diagnose data quality of categorical variables</i>
-------------------	---

Description

The `diagnose_category()` produces information for diagnosing the quality of the variables of `data.frame` or `tbl_df`.

Usage

```

diagnose_category(.data, ...)

## S3 method for class 'data.frame'
diagnose_category(
  .data,
  ...,
  top = 10,
  type = c("rank", "n")[2],
  add_character = TRUE,
  add_date = TRUE
)

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a tbl_df .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_category()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>top</code>	an integer. Specifies the upper top rows or rank to extract. Default is 10.

type	a character string specifying how result are extracted. "rank" that extract top n ranks by decreasing frequency. In this case, if there are ties in rank, more rows than the number specified by the top argument are returned. Default is "n" extract only top n rows by decreasing frequency. If there are too many rows to be returned because there are too many ties, you can adjust the returned rows appropriately by using "n".
add_character	logical. Decide whether to include text variables in the diagnosis of categorical data. The default value is TRUE, which also includes character variables.
add_date	logical. Decide whether to include Date and POSIXct variables in the diagnosis of categorical data. The default value is TRUE, which also includes character variables.

Details

The scope of the diagnosis is the occupancy status of the levels in categorical data. If a certain level of occupancy is close to 100 then the removal of this variable in the forecast model will have to be considered. Also, if the occupancy of all levels is close to 0 variable is likely to be an identifier.

Value

an object of `tbl_df`.

Categorical diagnostic information

The information derived from the categorical data diagnosis is as follows.

- variables : variable names
- levels: level names
- N : number of observation
- freq : number of observation at the levels
- ratio : percentage of observation at the levels
- rank : rank of occupancy ratio of levels

See vignette("diagonosis") for an introduction to these concepts.

See Also

[diagnose_category.tbl_dbi](#), [diagnose.data.frame](#), [diagnose_numeric.data.frame](#), [diagnose_outlier.data.frame](#)

Examples

```
# Diagnosis of categorical variables
diagnose_category(jobchange)

# Select the variable to diagnose
# diagnose_category(jobchange, education_level, company_type)
# diagnose_category(jobchange, -education_level, -company_type)
# diagnose_category(jobchange, "education_level", "company_type")
```

```
# diagnose_category(jobchange, 7)

# Using pipes -----
library(dplyr)

# Diagnosis of all categorical variables
jobchange %>%
  diagnose_category()

# Positive values select variables
jobchange %>%
  diagnose_category(company_type, job_chnge)

# Negative values to drop variables
jobchange %>%
  diagnose_category(-company_type, -job_chnge)

# Positions values select variables
# jobchange %>%
#   diagnose_category(7)

# Positions values select variables
# jobchange %>%
#   diagnose_category(-7)

# Top rank levels with top argument
jobchange %>%
  diagnose_category(top = 2)

# Using pipes & dplyr -----
# Extraction of level that is more than 60% of categorical data
jobchange %>%
  diagnose_category() %>%
  filter(ratio >= 60)

# All observations of enrollee_id have a rank of 1.
# Because it is a unique identifier. Therefore, if you select up to the top rank 3,
# all records are displayed. It will probably fill your screen.

# extract rows that less than equal rank 3
# default of type argument is "n"
jobchange %>%
  diagnose_category(enrollee_id, top = 3)

# extract rows that less than equal rank 3
jobchange %>%
  diagnose_category(enrollee_id, top = 3, type = "rank")

# extract only 3 rows
jobchange %>%
  diagnose_category(enrollee_id, top = 3, type = "n")
```

 diagnose_category.tbl_dbi

Diagnose data quality of categorical variables in the DBMS

Description

The `diagnose_category()` produces information for diagnosing the quality of the character (CHAR, VARCHAR, VARCHAR2, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```
## S3 method for class 'tbl_dbi'
diagnose_category(
  .data,
  ...,
  top = 10,
  type = c("rank", "n")[1],
  in_database = TRUE,
  collect_size = Inf
)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_category()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>top</code>	an integer. Specifies the upper top rank to extract. Default is 10.
<code>type</code>	a character string specifying how result are extracted. Default is "rank" that extract top n ranks by decreasing frequency. In this case, if there are ties in rank, more rows than the number specified by the <code>top</code> argument are returned. "n" extract top n rows by decreasing frequency. If there are too many rows to be returned because there are too many ties, you can adjust the returned rows appropriately by using "n".
<code>in_database</code>	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory.
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .

Details

The scope of the diagnosis is the occupancy status of the levels in categorical data. If a certain level of occupancy is close to 100 then the removal of this variable in the forecast model will have to be considered. Also, if the occupancy of all levels is close to 0 variable is likely to be an identifier.

Value

an object of tbl_df.

Categorical diagnostic information

The information derived from the categorical data diagnosis is as follows.

- variables : variable names
- levels: level names
- N : number of observation
- freq : number of observation at the levels
- ratio : percentage of observation at the levels
- rank : rank of occupancy ratio of levels

See vignette("diagonosis") for an introduction to these concepts.

See Also

[diagnose_category.data.frame](#), [diagnose.tbl_dbi](#), [diagnose_category.tbl_dbi](#), [diagnose_numeric.tbl_dbi](#), [diagnose_outlier.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy jobchange to the DBMS with a table named TB_JOBCHANGE
copy_to(con_sqlite, jobchange, name = "TB_JOBCHANGE", overwrite = TRUE)

# Using pipes -----
# Diagnosis of all categorical variables
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category()

# Positive values select variables
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category(company_type, job_chnge)

# Negative values to drop variables, and In-memory mode
```

```

con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category(-company_type, -job_chnge, in_database = FALSE)

# Positions values select variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category(7, in_database = FALSE, collect_size = 200)

# Positions values select variables
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category(-7)

# Top rank levels with top argument
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category(top = 2)

# Using pipes & dplyr -----
# Extraction of level that is more than 60% of categorical data
con_sqlite %>%
  tbl("TB_JOBCHANGE") %>%
  diagnose_category() %>%
  filter(ratio >= 60)

# Using type argument -----
dfm <- data.frame(alpabet = c(rep(letters[1:5], times = 5), "c"))

# copy dfm to the DBMS with a table named TB_EXAMPLE
copy_to(con_sqlite, dfm, name = "TB_EXAMPLE", overwrite = TRUE)

# extract rows that less than equal rank 10
# default of top argument is 10
con_sqlite %>%
  tbl("TB_EXAMPLE") %>%
  diagnose_category()

# extract rows that less than equal rank 2
con_sqlite %>%
  tbl("TB_EXAMPLE") %>%
  diagnose_category(top = 2, type = "rank")

# extract rows that less than equal rank 2
# default of type argument is "rank"
con_sqlite %>%
  tbl("TB_EXAMPLE") %>%
  diagnose_category(top = 2)

# extract only 2 rows
con_sqlite %>%
  tbl("TB_EXAMPLE") %>%
  diagnose_category(top = 2, type = "n")

```

```
# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)
```

diagnose_numeric	<i>Diagnose data quality of numerical variables</i>
------------------	---

Description

The `diagnose_numeric()` produces information for diagnosing the quality of the numerical data.

Usage

```
diagnose_numeric(.data, ...)

## S3 method for class 'data.frame'
diagnose_numeric(.data, ...)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_numeric()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

The scope of the diagnosis is to calculate a statistic that can be used to understand the distribution of numerical data. `min`, `Q1`, `mean`, `median`, `Q3`, `max` can be used to estimate the distribution of data. If the number of zero or minus is large, it is necessary to suspect the error of the data. If the number of outliers is large, a strategy of eliminating or replacing outliers is needed.

Value

an object of `tbl_df`.

Numerical diagnostic information

The information derived from the numerical data diagnosis is as follows.

- `variables` : variable names
- `min` : minimum
- `Q1` : 25 percentile

- mean : arithmetic average
- median : median. 50 percentile
- Q3 : 75 percentile
- max : maximum
- zero : count of zero values
- minus : count of minus values
- outlier : count of outliers

See vignette("diagnosis") for an introduction to these concepts.

See Also

[diagnose_numeric.tbl_dbi](#), [diagnose.data.frame](#), [diagnose_category.data.frame](#), [diagnose_outlier.data.frame](#)

Examples

```
# Diagnosis of numerical variables
diagnose_numeric(heartfailure)

# Select the variable to diagnose
diagnose_numeric(heartfailure, cpk_enzyme, sodium)
diagnose_numeric(heartfailure, -cpk_enzyme, -sodium)
diagnose_numeric(heartfailure, "cpk_enzyme", "sodium")
diagnose_numeric(heartfailure, 5)

# Using pipes -----
library(dplyr)

# Diagnosis of all numerical variables
heartfailure %>%
  diagnose_numeric()
# Positive values select variables
heartfailure %>%
  diagnose_numeric(cpk_enzyme, sodium)
# Negative values to drop variables
heartfailure %>%
  diagnose_numeric(-cpk_enzyme, -sodium)
# Positions values select variables
heartfailure %>%
  diagnose_numeric(5)
# Positions values select variables
heartfailure %>%
  diagnose_numeric(-1, -5)

# Using pipes & dplyr -----
# List of variables containing outliers
heartfailure %>%
  diagnose_numeric() %>%
  filter(outlier > 0)
```

diagnose_numeric.tbl_dbi

Diagnose data quality of numerical variables in the DBMS

Description

The `diagnose_numeric()` produces information for diagnosing the quality of the numerical (INTEGER, NUMBER, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```
## S3 method for class 'tbl_dbi'
diagnose_numeric(.data, ..., in_database = FALSE, collect_size = Inf)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_numeric()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>in_database</code>	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .

Details

The scope of the diagnosis is the calculate a statistic that can be used to understand the distribution of numerical data. `min`, `Q1`, `mean`, `median`, `Q3`, `max` can be used to estimate the distribution of data. If the number of zero or minus is large, it is necessary to suspect the error of the data. If the number of outliers is large, a strategy of eliminating or replacing outliers is needed.

Value

an object of `tbl_df`.

Numerical diagnostic information

The information derived from the numerical data diagnosis is as follows.

- `variables` : variable names
- `min` : minimum

- Q1 : 25 percentile
- mean : arithmetic average
- median : median. 50 percentile
- Q3 : 75 percentile
- max : maximum
- zero : count of zero values
- minus : count of minus values
- outlier : count of outliers

See vignette("diagnosis") for an introduction to these concepts.

See Also

[diagnose_numeric.data.frame](#), [diagnose.tbl_dbi](#), [diagnose_category.tbl_dbi](#), [diagnose_outlier.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Diagnosis of all numerical variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_numeric()

# Positive values select variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_numeric(age, sodium, collect_size = 200)

# Negative values to drop variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_numeric(-age, -sodium)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_numeric(5)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_numeric(-1, -5)
```

```

# Using pipes & dplyr -----
# List of variables containing outliers
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_numeric() %>%
  filter(outlier > 0)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

diagnose_outlier	<i>Diagnose outlier of numerical variables</i>
------------------	--

Description

The `diagnose_outlier()` produces outlier information for diagnosing the quality of the numerical data.

Usage

```

diagnose_outlier(.data, ...)

## S3 method for class 'data.frame'
diagnose_outlier(.data, ...)

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_outlier()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

The scope of the diagnosis is the provide a outlier information. If the number of outliers is small and the difference between the averages including outliers and the averages not including them is large, it is necessary to eliminate or replace the outliers.

Value

an object of `tbl_df`.

Outlier Diagnostic information

The information derived from the numerical data diagnosis is as follows.

- variables : variable names
- outliers_cnt : number of outliers
- outliers_ratio : percent of outliers
- outliers_mean : arithmetic average of outliers
- with_mean : arithmetic average of with outliers
- without_mean : arithmetic average of without outliers

See vignette("diagonosis") for an introduction to these concepts.

See Also

[diagnose_outlier.tbl_dbi](#), [diagnose.data.frame](#), [diagnose_category.data.frame](#), [diagnose_numeric.data.frame](#)

Examples

```
# Diagnosis of numerical variables
diagnose_outlier(heartfailure)

# Select the variable to diagnose
diagnose_outlier(heartfailure, cpk_enzyme, sodium)
diagnose_outlier(heartfailure, -cpk_enzyme, -sodium)
diagnose_outlier(heartfailure, "cpk_enzyme", "sodium")
diagnose_outlier(heartfailure, 5)

# Using pipes -----
library(dplyr)

# Diagnosis of all numerical variables
heartfailure %>%
  diagnose_outlier()
# Positive values select variables
heartfailure %>%
  diagnose_outlier(cpk_enzyme, sodium)
# Negative values to drop variables
heartfailure %>%
  diagnose_outlier(-cpk_enzyme, -sodium)
# Positions values select variables
heartfailure %>%
  diagnose_outlier(5)
# Positions values select variables
heartfailure %>%
  diagnose_outlier(-1, -5)

# Using pipes & dplyr -----
# outlier_ratio is more than 1%
heartfailure %>%
  diagnose_outlier() %>%
```



```
filter(outliers_ratio > 1)
```

```
diagnose_outlier.tbl_dbi
```

Diagnose outlier of numerical variables in the DBMS

Description

The `diagnose_outlier()` produces outlier information for diagnosing the quality of the numerical (INTEGER, NUMBER, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```
## S3 method for class 'tbl_dbi'
diagnose_outlier(.data, ..., in_database = FALSE, collect_size = Inf)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_outlier()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .

Details

The scope of the diagnosis is the provide a outlier information. If the number of outliers is small and the difference between the averages including outliers and the averages not including them is large, it is necessary to eliminate or replace the outliers.

Value

an object of `tbl_df`.

Outlier Diagnostic information

The information derived from the numerical data diagnosis is as follows.

- variables : variable names
- outliers_cnt : number of outliers
- outliers_ratio : percent of outliers
- outliers_mean : arithmetic average of outliers
- with_mean : arithmetic average of with outliers
- without_mean : arithmetic average of without outliers

See vignette("diagonosis") for an introduction to these concepts.

See Also

[diagnose_outlier.data.frame](#), [diagnose.tbl_dbi](#), [diagnose_category.tbl_dbi](#), [diagnose_numeric.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Diagnosis of all numerical variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_outlier()

# Positive values select variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_outlier(platelets, sodium, collect_size = 200)

# Negative values to drop variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_outlier(-platelets, -sodium)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_outlier(5)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
```

```

diagnose_outlier(-1, -5)

# Using pipes & dplyr -----
# outlier_ratio is more than 1%
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  diagnose_outlier() %>%
  filter(outliers_ratio > 1)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

diagnose_paged_report *Reporting the information of data diagnosis*

Description

The `diagnose_paged_report()` paged report the information for diagnosing the quality of the data.

Usage

```

diagnose_paged_report(.data, ...)

## S3 method for class 'data.frame'
diagnose_paged_report(
  .data,
  output_format = c("pdf", "html"),
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "Data Diagnosis Report",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  abstract_title = "Report Overview",
  abstract = NULL,
  title_color = "white",
  subtitle_color = "gold",
  thres_uniq_cat = 0.5,
  thres_uniq_num = 5,
  flag_content_zero = TRUE,
  flag_content_minus = TRUE,
  flag_content_missing = TRUE,
  cover_img = NULL,
  create_date = Sys.time(),
  logo_img = NULL,
  theme = c("orange", "blue"),
  sample_percent = 100,

```

```

    is_tbl_dbi = FALSE,
    ...
)

```

Arguments

<code>.data</code>	a data.frame or a <code>tbl_df</code> .
<code>...</code>	arguments to be passed to methods.
<code>output_format</code>	report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>rmarkdown::render()</code> and <code>pagedown::chrome_print()</code> . so, you needed Chrome web browser on computer. "html" create html file by <code>rmarkdown::render()</code> .
<code>output_file</code>	name of generated file. default is NULL.
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "Data Diagnosis Report".
<code>subtitle</code>	character. subtitle of report. default is name of data.
<code>author</code>	character. author of report. default is "dlookr".
<code>abstract_title</code>	character. abstract title of report. default is "Report Overview".
<code>abstract</code>	character. abstract of report.
<code>title_color</code>	character. color of title. default is "white".
<code>subtitle_color</code>	character. color of subtitle. default is "gold".
<code>thres_uniq_cat</code>	numeric. threshold to use for "Unique Values - Categorical Variables". default is 0.5.
<code>thres_uniq_num</code>	numeric. threshold to use for "Unique Values - Numerical Variables". default is 5.
<code>flag_content_zero</code>	logical. whether to output "Zero Values" information. the default value is TRUE, and the information is displayed.
<code>flag_content_minus</code>	logical. whether to output "Minus Values" information. the default value is TRUE, and the information is displayed.
<code>flag_content_missing</code>	logical. whether to output "Missing Value" information. the default value is TRUE, and the information is displayed.
<code>cover_img</code>	character. name of cover image.
<code>create_date</code>	Date or POSIXct, character. The date on which the report is generated. The default value is the result of <code>Sys.time()</code> .
<code>logo_img</code>	character. name of logo image file on top right.
<code>theme</code>	character. name of theme for report. support "orange" and "blue". default is "orange".
<code>sample_percent</code>	numeric. Sample percent of data for performing Diagnosis. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
<code>is_tbl_dbi</code>	logical. whether <code>.data</code> is a <code>tbl_dbi</code> object.

Details

Generate generalized data diagnostic reports automatically. You can choose to output to pdf and html files. This is useful for diagnosing a data frame with a large number of variables than data with a small number of variables.

Create an PDF through the Chrome DevTools Protocol. If you want to create PDF, Google Chrome or Microsoft Edge (or Chromium on Linux) must be installed prior to using this function. If not installed, you must use `output_format = "html"`.

Reported information

Reported from the data diagnosis is as follows.

- Overview
 - Data Structures
 - Job Informations
 - Warnings
 - Variables
- Missing Values
 - List of Missing Values
 - Visualization
- Unique Values
 - Categorical Variables
 - Numerical Variables
- Categorical Variable Diagnosis
 - Top Ranks
- Numerical Variable Diagnosis
 - Distribution
 - * Zero Values
 - * Minus Values
 - Outliers
 - * List of Outliers
 - * Individual Outliers

See Also

[diagnose_paged_report.tbl_dbi](#).

Examples

```
## Not run:  
# create dataset  
heartfailure2 <- dlookr::heartfailure  
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "sodium"] <- NA  
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA  
heartfailure2[sample(seq(NROW(heartfailure2)), 2), "time"] <- 0
```

```

heartfailure2[sample(seq(NROW(heartfailure2)), 1), "creatinine"] <- -0.3

# create pdf file. file name is Diagnosis_Paged_Report.pdf
# diagnose_paged_report(heartfailure2)

# create pdf file. file name is Diagn.pdf. and change cover image
# cover <- file.path(system.file(package = "dlookr"), "report", "cover2.jpg")
# diagnose_paged_report(heartfailure2, cover_img = cover, title_color = "gray",
#   output_file = "Diagn.pdf")

# create pdf file. file name is ./Diagn.pdf and not browse
# cover <- file.path(system.file(package = "dlookr"), "report", "cover3.jpg")
# diagnose_paged_report(heartfailure2, output_dir = ".", cover_img = cover,
#   flag_content_missing = FALSE, output_file = "Diagn.pdf", browse = FALSE)

# create pdf file. file name is Diagnosis_Paged_Report.html
# diagnose_paged_report(heartfailure2, output_format = "html")

## End(Not run)

```

diagnose_paged_report.tbl_dbi

Reporting the information of data diagnosis for table of the DBMS

Description

The `diagnose_paged_report()` paged report the information for diagnosing the quality of the DBMS table through `tbl_dbi`.

Usage

```

## S3 method for class 'tbl_dbi'
diagnose_paged_report(
  .data,
  output_format = c("pdf", "html")[1],
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "Data Diagnosis Report",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  abstract_title = "Report Overview",
  abstract = NULL,
  title_color = "white",
  subtitle_color = "gold",
  thres_uniq_cat = 0.5,
  thres_uniq_num = 5,

```

```

    flag_content_zero = TRUE,
    flag_content_minus = TRUE,
    flag_content_missing = TRUE,
    cover_img = NULL,
    create_date = Sys.time(),
    logo_img = NULL,
    theme = c("orange", "blue")[1],
    sample_percent = 100,
    in_database = FALSE,
    collect_size = Inf,
    as_factor = TRUE,
    ...
)

```

Arguments

.data	a tbl_dbi.
output_format	report output type. Choose either "pdf" and "html". "pdf" create pdf file by rmarkdown::render() and pagedown::chrome_print(). so, you needed Chrome web browser on computer. "html" create html file by rmarkdown::render().
output_file	name of generated file. default is NULL.
output_dir	name of directory to generate report file. default is tempdir().
browse	logical. choose whether to output the report results to the browser.
title	character. title of report. default is "Data Diagnosis Report".
subtitle	character. subtitle of report. default is name of data.
author	character. author of report. default is "dlookr".
abstract_title	character. abstract title of report. default is "Report Overview".
abstract	character. abstract of report.
title_color	character. color of title. default is "white".
subtitle_color	character. color of title. default is "gold".
thres_uniq_cat	numeric. threshold to use for "Unique Values - Categorical Variables". default is 0.5.
thres_uniq_num	numeric. threshold to use for "Unique Values - Numerical Variables". default is 5.
flag_content_zero	logical. whether to output "Zero Values" information. the default value is TRUE, and the information is displayed.
flag_content_minus	logical. whether to output "Minus Values" information. the default value is TRUE, and the information is displayed.
flag_content_missing	logical. whether to output "Missing Value" information. the default value is TRUE, and the information is displayed.
cover_img	character. name of cover image.

<code>create_date</code>	Date or POSIXct, character. The date on which the report is generated. The default value is the result of <code>Sys.time()</code> .
<code>logo_img</code>	character. name of logo image on top right.
<code>theme</code>	character. name of theme for report. support "orange" and "blue". default is "orange".
<code>sample_percent</code>	numeric. Sample percent of data for performing Diagnosis. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
<code>in_database</code>	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .
<code>as_factor</code>	logical. whether to convert to factor when importing a character type variable from DBMS table into R.
<code>...</code>	arguments to be passed to methods.

Details

Generate generalized data diagnostic reports automatically. You can choose to output to pdf and html files. This is useful for diagnosing a data frame with a large number of variables than data with a small number of variables.

Create an PDF through the Chrome DevTools Protocol. If you want to create PDF, Google Chrome or Microsoft Edge (or Chromium on Linux) must be installed prior to using this function. If not installed, you must use `output_format = "html"`.

Reported information

Reported from the data diagnosis is as follows.

- Overview
 - Data Structures
 - Job Informations
 - Warnings
 - Variables
- Missing Values
 - List of Missing Values
 - Visualization
- Unique Values
 - Categorical Variables
 - Numerical Variables
- Categorical Variable Diagnosis
 - Top Ranks

- Numerical Variable Diagnosis
 - Distribution
 - * Zero Values
 - * Minus Values
 - Outliers
 - * List of Outliers
 - * Individual Outliers

See Also

[diagnose_paged_report.data.frame.](#)

Examples

```

if (FALSE) {
  library(dplyr)

  # Generate data for the example
  heartfailure2 <- heartfailure
  heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
  heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

  # connect DBMS
  con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

  # copy heartfailure2 to the DBMS with a table named TB_HEARTFAILURE
  copy_to(con_sqlite, heartfailure2, name = "TB_HEARTFAILURE", overwrite = TRUE)

  # reporting the diagnosis information -----
  # create pdf file. file name is Diagnosis_Paged_Report.pdf
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_paged_report()

  # create pdf file. file name is Diagn.pdf, and collect size is 250
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_paged_report(collect_size = 250, output_file = "Diagn.pdf")

  # Disconnect DBMS
  DBI::dbDisconnect(con_sqlite)
}

```

Description

The `diagnose_report()` report the information for diagnosing the quality of the data.

Usage

```
diagnose_report(.data, output_format, output_file, output_dir, ...)
```

```
## S3 method for class 'data.frame'
diagnose_report(
  .data,
  output_format = c("pdf", "html"),
  output_file = NULL,
  output_dir = tempdir(),
  font_family = NULL,
  browse = TRUE,
  ...
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>output_format</code>	report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>knitr::knit()</code> . "html" create html file by <code>rmarkdown::render()</code> .
<code>output_file</code>	name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>...</code>	arguments to be passed to methods.
<code>font_family</code>	character. font family name for figure in pdf.
<code>browse</code>	logical. choose whether to output the report results to the browser.

Details

Generate generalized data diagnostic reports automatically. You can choose to output to pdf and html files. This is useful for diagnosing a data frame with a large number of variables than data with a small number of variables. For pdf output, Korean Gothic font must be installed in Korean operating system.

Reported information

Reported from the data diagnosis is as follows.

- Diagnose Data
 - Overview of Diagnosis
 - * List of all variables quality
 - * Diagnosis of missing data
 - * Diagnosis of unique data(Text and Category)
 - * Diagnosis of unique data(Numerical)

- Detailed data diagnosis
 - * Diagnosis of categorical variables
 - * Diagnosis of numerical variables
 - * List of numerical diagnosis (zero)
 - * List of numerical diagnosis (minus)
- Diagnose Outliers
 - Overview of Diagnosis
 - * Diagnosis of numerical variable outliers
 - * Detailed outliers diagnosis

See vignette("diagonosis") for an introduction to these concepts.

Examples

```
## Not run:
# reporting the diagnosis information -----
# create pdf file. file name is DataDiagnosis_Report.pdf
diagnose_report(heartfailure)
# create pdf file. file name is Diagn.pdf
diagnose_report(heartfailure, output_file = "Diagn.pdf")
# create pdf file. file name is ./Diagn.pdf and not browse
# diagnose_report(heartfailure, output_dir = ".", output_file = "Diagn.pdf",
#   browse = FALSE)
# create html file. file name is Diagnosis_Report.html
diagnose_report(heartfailure, output_format = "html")
# create html file. file name is Diagn.html
diagnose_report(heartfailure, output_format = "html", output_file = "Diagn.html")

## End(Not run)
```

diagnose_report.tbl_dbi

Reporting the information of data diagnosis for table of the DBMS

Description

The `diagnose_report()` report the information for diagnosing the quality of the DBMS table through `tbl_dbi`

Usage

```
## S3 method for class 'tbl_dbi'
diagnose_report(
  .data,
  output_format = c("pdf", "html"),
  output_file = NULL,
```

```

output_dir = tempdir(),
font_family = NULL,
in_database = FALSE,
collect_size = Inf,
...
)

```

Arguments

.data	a tbl_dbi.
output_format	report output type. Choose either "pdf" and "html". "pdf" create pdf file by knitr::knit(). "html" create html file by rmarkdown::render().
output_file	name of generated file. default is NULL.
output_dir	name of directory to generate report file. default is tempdir().
font_family	character. font family name for figure in pdf.
in_database	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in_database = TRUE.
collect_size	a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
...	arguments to be passed to methods.

Details

Generate generalized data diagnostic reports automatically. You can choose to output to pdf and html files. This is useful for diagnosing a data frame with a large number of variables than data with a small number of variables. For pdf output, Korean Gothic font must be installed in Korean operating system.

Reported information

Reported from the data diagnosis is as follows.

- Diagnose Data
 - Overview of Diagnosis
 - * List of all variables quality
 - * Diagnosis of missing data
 - * Diagnosis of unique data(Text and Category)
 - * Diagnosis of unique data(Numerical)
 - Detailed data diagnosis
 - * Diagnosis of categorical variables
 - * Diagnosis of numerical variables
 - * List of numerical diagnosis (zero)
 - * List of numerical diagnosis (minus)
- Diagnose Outliers

- Overview of Diagnosis
 - * Diagnosis of numerical variable outliers
 - * Detailed outliers diagnosis

See vignette("diagonosis") for an introduction to these concepts.

See Also

[diagnose_report.data.frame.](#)

Examples

```

if (FALSE) {
  library(dplyr)

  # Generate data for the example
  heartfailure2 <- heartfailure
  heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
  heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

  # connect DBMS
  con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

  # copy heartfailure2 to the DBMS with a table named TB_HEARTFAILURE
  copy_to(con_sqlite, heartfailure2, name = "TB_HEARTFAILURE", overwrite = TRUE)

  # reporting the diagnosis information -----
  # create pdf file. file name is DataDiagnosis_Report.pdf
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_report()

  # create pdf file. file name is Diagn.pdf, and collect size is 350
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_report(collect_size = 350, output_file = "Diagn.pdf")

  # create html file. file name is Diagnosis_Report.html
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_report(output_format = "html")

  # create html file. file name is Diagn.html
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_report(output_format = "html", output_file = "Diagn.html")

  # Disconnect DBMS
  DBI::dbDisconnect(con_sqlite)
}

```

diagnose_sparese	<i>Diagnosis of level combinations of categorical variables</i>
------------------	---

Description

The `diagnose_sparese()` checks for combinations of levels that do not appear as data among all combinations of levels of categorical variables.

Usage

```
diagnose_sparese(.data, ...)

## S3 method for class 'data.frame'
diagnose_sparese(
  .data,
  ...,
  type = c("all", "sparse")[2],
  add_character = FALSE,
  limit = 500
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>diagnose_sparese()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>type</code>	a character string specifying how result are extracted. "all" that returns a combination of all possible levels. At this time, the frequency of each case is also returned.. Default is "sparse" returns only sparse level combinations.
<code>add_character</code>	logical. Decide whether to include text variables in the diagnosis of categorical data. The default value is TRUE, which also includes character variables.
<code>limit</code>	integer. Conditions to check sparse levels. If the number of all possible combinations exceeds the limit, the calculation ends.

Value

an object of `data.frame`.

Information of sparse levels

The information derived from the sparse levels diagnosis is as follows.

- `variables` : level of categorical variables.
- `N` : number of observation. (optional)

Examples

```

library(dplyr)

# Examples of too many combinations
diagnose_sparese(jobchange)

# Character type is also included in the combination variable
diagnose_sparese(jobchange, add_character = TRUE)

# Combination of two variables
jobchange %>%
  diagnose_sparese(education_level, major_discipline)

# Remove two categorical variables from combination
jobchange %>%
  diagnose_sparese(-city, -education_level)

diagnose_sparese(heartfailure)

# Adjust the threshold of limit to calculate
diagnose_sparese(heartfailure, limit = 50)

# List all combinations, including parese cases
diagnose_sparese(heartfailure, type = "all")

# collaboration with dplyr
heartfailure %>%
  diagnose_sparese(type = "all") %>%
  arrange(desc(n_case)) %>%
  mutate(percent = round(n_case / sum(n_case) * 100, 1))

```

diagnose_web_report *Reporting the information of data diagnosis with html*

Description

The `diagnose_web_report()` report the information for diagnosing the quality of the data.

Usage

```

diagnose_web_report(.data, ...)

## S3 method for class 'data.frame'
diagnose_web_report(
  .data,
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,

```

```

    title = "Data Diagnosis",
    subtitle = deparse(substitute(.data)),
    author = "dlookr",
    title_color = "gray",
    thres_uniq_cat = 0.5,
    thres_uniq_num = 5,
    logo_img = NULL,
    create_date = Sys.time(),
    theme = c("orange", "blue"),
    sample_percent = 100,
    is_tbl_dbi = FALSE,
    ...
  )

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	arguments to be passed to methods.
<code>output_file</code>	name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "Data Diagnosis Report".
<code>subtitle</code>	character. subtitle of report. default is name of data.
<code>author</code>	character. author of report. default is "dlookr".
<code>title_color</code>	character. color of title. default is "gray".
<code>thres_uniq_cat</code>	numeric. threshold to use for "Unique Values - Categorical Variables". default is 0.5.
<code>thres_uniq_num</code>	numeric. threshold to use for "Unique Values - Numerical Variables". default is 5.
<code>logo_img</code>	character. name of logo image file on top left.
<code>create_date</code>	Date or <code>POSIXct</code> , character. The date on which the report is generated. The default value is the result of <code>Sys.time()</code> .
<code>theme</code>	character. name of theme for report. support "orange" and "blue". default is "orange".
<code>sample_percent</code>	numeric. Sample percent of data for performing Diagnosis. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
<code>is_tbl_dbi</code>	logical. whether <code>.data</code> is a <code>tbl_dbi</code> object.

Details

Generate generalized data diagnostic reports automatically. This is useful for diagnosing a data frame with a large number of variables than data with a small number of variables.

Reported information

Reported from the data diagnosis is as follows.

- Overview
 - Data Structures
 - * Data Structures
 - * Data Types
 - * Job Informations
 - Warnings
 - Variables
- Missing Values
 - List of Missing Values
 - Visualization
- Unique Values
 - Categorical Variables
 - Numerical Variables
- Outliers
- Samples
 - Duplicated
 - Heads
 - Tails

See Also

[diagnose_web_report.tbl_dbi](#).

Examples

```
## Not run:
# create dataset
heartfailure2 <- dlookr::heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "sodium"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 2), "time"] <- 0
heartfailure2[sample(seq(NROW(heartfailure2)), 1), "creatinine"] <- -0.3

# create pdf file. file name is Diagnosis_Report.html
# diagnose_web_report(heartfailure2)

# file name is Diagn.html. and change logo image
# logo <- file.path(system.file(package = "dlookr"), "report", "R_logo_html.svg")
# diagnose_web_report(heartfailure2, logo_img = logo, title_color = "black",
#   output_file = "Diagn.html")

# file name is ./Diagn_heartfailure.html, "blue" theme and not browse
# diagnose_web_report(heartfailure2, output_dir = ".", author = "Choonghyun Ryu",
```

```
# output_file = "Diagn_heartfailure.html", theme = "blue", browse = FALSE)
## End(Not run)
```

```
diagnose_web_report.tbl_dbi
```

*Reporting the information of data diagnosis for table of the DBMS
with html*

Description

The `diagnose_web_report()` report the information for diagnosing the quality of the DBMS table through `tbl_dbi`

Usage

```
## S3 method for class 'tbl_dbi'
diagnose_web_report(
  .data,
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "Data Diagnosis",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  title_color = "gray",
  thres_uniq_cat = 0.5,
  thres_uniq_num = 5,
  logo_img = NULL,
  create_date = Sys.time(),
  theme = c("orange", "blue")[1],
  sample_percent = 100,
  in_database = FALSE,
  collect_size = Inf,
  as_factor = TRUE,
  ...
)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>output_file</code>	name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "Data Diagnosis Report".

subtitle	character. subtitle of report. default is name of data.
author	character. author of report. default is "dlookr".
title_color	character. color of title. default is "gray".
thres_uniq_cat	numeric. threshold to use for "Unique Values - Categorical Variables". default is 0.5.
thres_uniq_num	numeric. threshold to use for "Unique Values - Numerical Variables". default is 5.
logo_img	character. name of logo image on top right.
create_date	Date or POSIXct, character. The date on which the report is generated. The default value is the result of Sys.time().
theme	character. name of theme for report. support "orange" and "blue". default is "orange".
sample_percent	numeric. Sample percent of data for performing Diagnosis. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
in_database	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in_database = TRUE.
collect_size	a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
as_factor	logical. whether to convert to factor when importing a character type variable from DBMS table into R.
...	arguments to be passed to methods.

Details

Generate generalized data diagnostic reports automatically. This is useful for diagnosing a data frame with a large number of variables than data with a small number of variables.

Reported information

Reported from the data diagnosis is as follows.

- Overview
 - Data Structures
 - * Data Structures
 - * Data Types
 - * Job Informations
 - Warnings
 - Variables
- Missing Values
 - Top Ranks
- Numerical Variable Diagnosis

- List of Missing Values
- Visualization
- Unique Values
 - Categorical Variables
 - Numerical Variables
- Outliers
- Samples
 - Duplicated
 - Heads
 - Tails

See Also

[diagnose_web_report.data.frame.](#)

Examples

```

if (FALSE) {
  library(dplyr)

  # Generate data for the example
  heartfailure2 <- heartfailure
  heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
  heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

  # connect DBMS
  con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

  # copy heartfailure2 to the DBMS with a table named TB_HEARTFAILURE
  copy_to(con_sqlite, heartfailure2, name = "TB_HEARTFAILURE", overwrite = TRUE)

  # reporting the diagnosis information -----
  # create pdf file. file name is Diagnosis_Report.html
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_web_report()

  # create pdf file. file name is Diagn.html, and collect size is 250
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    diagnose_web_report(collect_size = 250, output_file = "Diagn.html")

  # Disconnect DBMS
  DBI::dbDisconnect(con_sqlite)
}

```

dlookr_orange_paged *Generate paged HTML document*

Description

Generate paged HTML document

Usage

```
dlookr_orange_paged(...)
```

```
dlookr_blue_paged(...)
```

Arguments

... arguments to be passed to pagedown: :[html_paged](#).

Value

document of markdown format.

References

<https://pagedown.rbind.io>

dlookr_templ_html *dlookr HTML template Loads additional style and template file*

Description

dlookr HTML template

Loads additional style and template file

Usage

```
dlookr_templ_html(toc = TRUE, ...)
```

Arguments

toc should a table of contents be displayed?

... additional arguments provided to `html_document`

Value

An R Markdown output format.

References

https://raw.githubusercontent.com/dr-harper/example-rmd-templates/master/R/my_html_format.R

eda_paged_report *Reporting the information of EDA*

Description

The `eda_paged_report()` paged report the information for EDA.

Usage

```
eda_paged_report(.data, ...)

## S3 method for class 'data.frame'
eda_paged_report(
  .data,
  target = NULL,
  output_format = c("pdf", "html"),
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "EDA Report",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  abstract_title = "Report Overview",
  abstract = NULL,
  title_color = "black",
  subtitle_color = "blue",
  cover_img = NULL,
  create_date = Sys.time(),
  logo_img = NULL,
  theme = c("orange", "blue"),
  sample_percent = 100,
  is_tbl_dbi = FALSE,
  ...
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	arguments to be passed to methods.
<code>target</code>	character. target variable.
<code>output_format</code>	report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>rmarkdown::render()</code> and <code>pagedown::chrome_print()</code> . so, you needed Chrome web browser on computer. "html" create html file by <code>rmarkdown::render()</code> .

output_file	name of generated file. default is NULL.
output_dir	name of directory to generate report file. default is tempdir().
browse	logical. choose whether to output the report results to the browser.
title	character. title of report. default is "Data Diagnosis Report".
subtitle	character. subtitle of report. default is name of data.
author	character. author of report. default is "dlookr".
abstract_title	character. abstract title of report. default is "Report Overview".
abstract	character. abstract of report.
title_color	character. color of title. default is "black".
subtitle_color	character. color of subtitle. default is "blue".
cover_img	character. name of cover image.
create_date	Date or POSIXct, character. The date on which the report is generated. The default value is the result of Sys.time().
logo_img	character. name of logo image file on top right.
theme	character. name of theme for report. support "orange" and "blue". default is "orange".
sample_percent	numeric. Sample percent of data for performing Diagnosis. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
is_tbl_dbi	logical. whether .data is a tbl_dbi object.

Details

Generate generalized EDA report automatically. You can choose to output to pdf and html files. This feature is useful for EDA of data with many variables, rather than data with fewer variables.

Create an PDF through the Chrome DevTools Protocol. If you want to create PDF, Google Chrome or Microsoft Edge (or Chromium on Linux) must be installed prior to using this function. If not installed, you must use `output_format = "html"`.

Reported information

The EDA process will report the following information:

- Overview
 - Data Structures
 - Job Informations
- Univariate Analysis
 - Descriptive Statistics
 - * Numerical Variables
 - * Categorical Variables
 - Normality Test
- Bivariate Analysis

- Compare Numerical Variables
- Compare Categorical Variables
- Multivariate Analysis
 - Correlation Analysis
 - * Correlation Coefficient Matrix
 - * Correlation Plot
- Target based Analysis
 - Grouped Numerical Variables
 - Grouped Categorical Variables
 - Grouped Correlation

See Also

[eda_paged_report.tbl_dbi](#).

Examples

```
## Not run:
# create the dataset
heartfailure2 <- dlookr::heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "sodium"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

# create pdf file. file name is EDA_Paged_Report.pdf
# eda_paged_report(heartfailure2, sample_percent = 80)

# create pdf file. file name is EDA.pdf. and change cover image
# cover <- file.path(system.file(package = "dlookr"), "report", "cover1.jpg")
# eda_paged_report(heartfailure2, cover_img = cover, title_color = "gray",
#   output_file = "EDA.pdf")

# create pdf file. file name is ./EDA.pdf and not browse
# cover <- file.path(system.file(package = "dlookr"), "report", "cover3.jpg")
# eda_paged_report(heartfailure2, output_dir = ".", cover_img = cover,
#   flag_content_missing = FALSE, output_file = "EDA.pdf", browse = FALSE)

# create pdf file. file name is EDA_Paged_Report.html
# eda_paged_report(heartfailure2, target = "death_event", output_format = "html")

## End(Not run)
```

eda_paged_report.tbl_dbi

Reporting the information of EDA for table of the DBMS

Description

The `eda_paged_report()` paged report the information for EDA of the DBMS table through `tbl_dbi`

Usage

```
## S3 method for class 'tbl_dbi'
eda_paged_report(
  .data,
  target = NULL,
  output_format = c("pdf", "html")[1],
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "EDA Report",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  abstract_title = "Report Overview",
  abstract = NULL,
  title_color = "black",
  subtitle_color = "blue",
  cover_img = NULL,
  create_date = Sys.time(),
  logo_img = NULL,
  theme = c("orange", "blue")[1],
  sample_percent = 100,
  in_database = FALSE,
  collect_size = Inf,
  as_factor = TRUE,
  ...
)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>target</code>	character. target variable.
<code>output_format</code>	report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>rmarkdown::render()</code> and <code>pagedown::chrome_print()</code> . so, you needed Chrome web browser on computer. "html" create html file by <code>rmarkdown::render()</code> .
<code>output_file</code>	name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "Data Diagnosis Report".
<code>subtitle</code>	character. subtitle of report. default is name of data.
<code>author</code>	character. author of report. default is "dlookr".
<code>abstract_title</code>	character. abstract title of report. default is "Report Overview".

abstract	character. abstract of report.
title_color	character. color of title. default is "black".
subtitle_color	character. color of title. default is "blue".
cover_img	character. name of cover image.
create_date	Date or POSIXct, character. The date on which the report is generated. The default value is the result of Sys.time().
logo_img	character. name of logo image on top right.
theme	character. name of theme for report. support "orange" and "blue". default is "orange".
sample_percent	numeric. Sample percent of data for performing EDA. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
in_database	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in_database = TRUE.
collect_size	a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
as_factor	logical. whether to convert to factor when importing a character type variable from DBMS table into R.
...	arguments to be passed to methods.

Details

Generate generalized EDA report automatically. You can choose to output to pdf and html files. This feature is useful for EDA of data with many variables, rather than data with fewer variables.

Create an PDF through the Chrome DevTools Protocol. If you want to create PDF, Google Chrome or Microsoft Edge (or Chromium on Linux) must be installed prior to using this function. If not installed, you must use output_format = "html".

Reported information

The EDA process will report the following information:

- Overview
 - Data Structures
 - Job Informations
- Univariate Analysis
 - Descriptive Statistics
 - * Numerical Variables
 - * Categorical Variables
 - Normality Test
- Bivariate Analysis
 - Compare Numerical Variables

- Compare Categorical Variables
- Multivariate Analysis
 - Correlation Analysis
 - * Correlation Coefficient Matrix
 - * Correlation Plot
- Target based Analysis
 - Grouped Numerical Variables
 - Grouped Categorical Variables
 - Grouped Correlation

See Also

[eda_paged_report.data.frame.](#)

Examples

```

if (FALSE) {
  library(dplyr)

  # Generate data for the example
  heartfailure2 <- heartfailure
  heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
  heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

  # connect DBMS
  con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

  # copy heartfailure2 to the DBMS with a table named TB_HEARTFAILURE
  copy_to(con_sqlite, heartfailure2, name = "TB_HEARTFAILURE", overwrite = TRUE)

  # reporting the diagnosis information -----
  # create pdf file. file name is EDA_Paged_Report.pdf
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    eda_paged_report(target = "death_event")

  # create pdf file. file name is EDA.pdf, and collect size is 250
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    eda_paged_report(collect_size = 250, output_file = "EDA.pdf")

  # Disconnect DBMS
  DBI::dbDisconnect(con_sqlite)
}

```

 eda_report

Reporting the information of EDA

Description

The `eda_report()` report the information of exploratory data analysis for object inheriting from `data.frame`.

Usage

```
eda_report(.data, ...)

## S3 method for class 'data.frame'
eda_report(
  .data,
  target = NULL,
  output_format = c("pdf", "html"),
  output_file = NULL,
  output_dir = tempdir(),
  font_family = NULL,
  browse = TRUE,
  ...
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	arguments to be passed to methods.
<code>target</code>	target variable.
<code>output_format</code>	character. report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>knitr::knit()</code> . "html" create html file by <code>rmarkdown::render()</code> .
<code>output_file</code>	character. name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	character. name of directory to generate report file. default is <code>tempdir()</code> .
<code>font_family</code>	character. font family name for figure in pdf.
<code>browse</code>	logical. choose whether to output the report results to the browser.

Details

Generate generalized EDA report automatically. You can choose to output as pdf and html files. This feature is useful for EDA of data with many variables, rather than data with fewer variables. For pdf output, Korean Gothic font must be installed in Korean operating system.

Reported information

The EDA process will report the following information:

- Introduction
 - Information of Dataset
 - Information of Variables
 - About EDA Report
- Univariate Analysis
 - Descriptive Statistics
 - Normality Test of Numerical Variables
 - * Statistics and Visualization of (Sample) Data
- Relationship Between Variables
 - Correlation Coefficient
 - * Correlation Coefficient by Variable Combination
 - * Correlation Plot of Numerical Variables
- Target based Analysis
 - Grouped Descriptive Statistics
 - * Grouped Numerical Variables
 - * Grouped Categorical Variables
 - Grouped Relationship Between Variables
 - * Grouped Correlation Coefficient
 - * Grouped Correlation Plot of Numerical Variables

See vignette("EDA") for an introduction to these concepts.

Examples

```
if (FALSE) {
  library(dplyr)

  ## target variable is categorical variable -----
  # reporting the EDA information
  # create pdf file. file name is EDA_Report.pdf
  eda_report(heartfailure, death_event)

  # create pdf file. file name is EDA_heartfailure.pdf
  eda_report(heartfailure, "death_event", output_file = "EDA_heartfailure.pdf")

  # create pdf file. file name is EDA_heartfailure.pdf and not browse
  eda_report(heartfailure, "death_event", output_dir = ".",
            output_file = "EDA_heartfailure.pdf", browse = FALSE)

  # create html file. file name is EDA_Report.html
  eda_report(heartfailure, "death_event", output_format = "html")

  # create html file. file name is EDA_heartfailure.html
```

```

eda_report(heartfailure, death_event, output_format = "html",
           output_file = "EDA_heartfailure.html")

## target variable is numerical variable -----
# reporting the EDA information
eda_report(heartfailure, sodium)

# create pdf file. file name is EDA2.pdf
eda_report(heartfailure, "sodium", output_file = "EDA2.pdf")

# create html file. file name is EDA_Report.html
eda_report(heartfailure, "sodium", output_format = "html")

# create html file. file name is EDA2.html
eda_report(heartfailure, sodium, output_format = "html", output_file = "EDA2.html")

## target variable is null
# reporting the EDA information
eda_report(heartfailure)

# create pdf file. file name is EDA2.pdf
eda_report(heartfailure, output_file = "EDA2.pdf")

# create html file. file name is EDA_Report.html
eda_report(heartfailure, output_format = "html")

# create html file. file name is EDA2.html
eda_report(heartfailure, output_format = "html", output_file = "EDA2.html")
}

```

eda_report.tbl_dbi *Reporting the information of EDA for table of the DBMS*

Description

The `eda_report()` report the information of Exploratory data analysis for object inheriting from the DBMS table through `tbl_dbi`

Usage

```

## S3 method for class 'tbl_dbi'
eda_report(
  .data,
  target = NULL,
  output_format = c("pdf", "html"),
  output_file = NULL,
  font_family = NULL,
  output_dir = tempdir(),

```

```

    in_database = FALSE,
    collect_size = Inf,
    ...
)

```

Arguments

.data	a tbl_dbi.
target	target variable.
output_format	report output type. Choose either "pdf" and "html". "pdf" create pdf file by knitr::knit(). "html" create html file by rmarkdown::render().
output_file	name of generated file. default is NULL.
font_family	character. font family name for figure in pdf.
output_dir	name of directory to generate report file. default is tempdir().
in_database	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in_database = TRUE.
collect_size	a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
...	arguments to be passed to methods.

Details

Generate generalized data EDA reports automatically. You can choose to output to pdf and html files. This is useful for EDA a data frame with a large number of variables than data with a small number of variables. For pdf output, Korean Gothic font must be installed in Korean operating system.

Reported information

The EDA process will report the following information:

- Introduction
 - Information of Dataset
 - Information of Variables
 - About EDA Report
- Univariate Analysis
 - Descriptive Statistics
 - Normality Test of Numerical Variables
 - * Statistics and Visualization of (Sample) Data
- Relationship Between Variables
 - Correlation Coefficient
 - * Correlation Coefficient by Variable Combination
 - * Correlation Plot of Numerical Variables

- Target based Analysis
 - Grouped Descriptive Statistics
 - * Grouped Numerical Variables
 - * Grouped Categorical Variables
 - Grouped Relationship Between Variables
 - * Grouped Correlation Coefficient
 - * Grouped Correlation Plot of Numerical Variables

See vignette("EDA") for an introduction to these concepts.

See Also

[eda_report.data.frame.](#)

Examples

```
if (FALSE) {
  library(dplyr)

  # Generate data for the example
  heartfailure2 <- heartfailure
  heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
  heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

  # connect DBMS
  con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

  # copy heartfailure2 to the DBMS with a table named TB_HEARTFAILURE
  copy_to(con_sqlite, heartfailure2, name = "TB_HEARTFAILURE", overwrite = TRUE)

  ## target variable is categorical variable
  # reporting the EDA information
  # create pdf file. file name is EDA_Report.pdf
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    eda_report(death_event)

  # create pdf file. file name is EDA_TB_HEARTFAILURE.pdf
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    eda_report("death_event", output_file = "EDA_TB_HEARTFAILURE.pdf")

  # create html file. file name is EDA_Report.html
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    eda_report("death_event", output_format = "html")

  # create html file. file name is EDA_TB_HEARTFAILURE.html
  con_sqlite %>%
    tbl("TB_HEARTFAILURE") %>%
    eda_report(death_event, output_format = "html", output_file = "EDA_TB_HEARTFAILURE.html")
}
```



```
## target variable is numerical variable
# reporting the EDA information, and collect size is 250
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report(sodium, collect_size = 250)

# create pdf file. file name is EDA2.pdf
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report("sodium", output_file = "EDA2.pdf")

# create html file. file name is EDA_Report.html
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report("sodium", output_format = "html")

# create html file. file name is EDA2.html
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report(sodium, output_format = "html", output_file = "EDA2.html")

## target variable is null
# reporting the EDA information
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report()

# create pdf file. file name is EDA2.pdf
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report(output_file = "EDA2.pdf")

# create html file. file name is EDA_Report.html
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report(output_format = "html")

# create html file. file name is EDA2.html
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_report(output_format = "html", output_file = "EDA2.html")

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)
}
```

Description

The `eda_web_report()` report the information of exploratory data analysis for object inheriting from `data.frame`.

Usage

```
eda_web_report(.data, ...)

## S3 method for class 'data.frame'
eda_web_report(
  .data,
  target = NULL,
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "EDA",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  title_color = "gray",
  logo_img = NULL,
  create_date = Sys.time(),
  theme = c("orange", "blue"),
  sample_percent = 100,
  is_tbl_dbi = FALSE,
  ...
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	arguments to be passed to methods.
<code>target</code>	character. target variable.
<code>output_file</code>	name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "EDA".
<code>subtitle</code>	character. subtitle of report. default is name of data.
<code>author</code>	character. author of report. default is "dlookr".
<code>title_color</code>	character. color of title. default is "gray".
<code>logo_img</code>	character. name of logo image file on top left.
<code>create_date</code>	Date or <code>POSIXct</code> , character. The date on which the report is generated. The default value is the result of <code>Sys.time()</code> .
<code>theme</code>	character. name of theme for report. support "orange" and "blue". default is "orange".

`sample_percent` numeric. Sample percent of data for performing EDA. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.

`is_tbl_dbi` logical. whether `.data` is a `tbl_dbi` object.

Details

Generate generalized EDA report automatically. This feature is useful for EDA of data with many variables, rather than data with fewer variables.

Reported information

Reported from the EDA is as follows.

- Overview
 - Data Structures
 - Data Types
 - Job Informations
- Univariate Analysis
 - Descriptive Statistics
 - Normality Test
- Bivariate Analysis
 - Compare Numerical Variables
 - Compare Categorical Variables
- Multivariate Analysis
 - Correlation Analysis
 - * Correlation Matrix
 - * Correlation Plot
- Target based Analysis
 - Grouped Numerical Variables
 - Grouped Categorical Variables
 - Grouped Correlation

See Also

[eda_web_report.tbl_dbi](#).

Examples

```
## Not run:  
# create the dataset  
heartfailure2 <- dlookr::heartfailure  
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "sodium"] <- NA  
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA  
  
# create html file. file name is EDA_Report.html
```

```

# eda_web_report(heartfailure2)

# file name is EDA.html. and change logo image
# logo <- file.path(system.file(package = "dlookr"), "report", "R_logo_html.svg")
# eda_web_report(heartfailure2, logo_img = logo, title_color = "black",
#   output_file = "EDA.html")

# file name is ./EDA_heartfailure.html, "blue" theme and not browse
# eda_web_report(heartfailure2, target = "death_event", output_dir = ".",
#   author = "Choonghyun Ryu", output_file = "EDA_heartfailure.html",
#   theme = "blue", browse = FALSE)

## End(Not run)

```

eda_web_report.tbl_dbi

Reporting the information of EDA for table of the DBMS with html

Description

The `eda_web_report()` report the information of exploratory data analysis for the DBMS table through `tbl_dbi`

Usage

```

## S3 method for class 'tbl_dbi'
eda_web_report(
  .data,
  target = NULL,
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "EDA",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  title_color = "gray",
  logo_img = NULL,
  create_date = Sys.time(),
  theme = c("orange", "blue")[1],
  sample_percent = 100,
  in_database = FALSE,
  collect_size = Inf,
  as_factor = TRUE,
  ...
)

```

Arguments

.data	a tbl_dbi.
target	character. target variable.
output_file	name of generated file. default is NULL.
output_dir	name of directory to generate report file. default is tempdir().
browse	logical. choose whether to output the report results to the browser.
title	character. title of report. default is "EDA Report".
subtitle	character. subtitle of report. default is name of data.
author	character. author of report. default is "dlookr".
title_color	character. color of title. default is "gray".
logo_img	character. name of logo image on top right.
create_date	Date or POSIXct, character. The date on which the report is generated. The default value is the result of Sys.time().
theme	character. name of theme for report. support "orange" and "blue". default is "orange".
sample_percent	numeric. Sample percent of data for performing EDA. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.
in_database	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in_database = TRUE.
collect_size	a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
as_factor	logical. whether to convert to factor when importing a character type variable from DBMS table into R.
...	arguments to be passed to methods.

Details

Generate generalized EDA report automatically. This feature is useful for EDA of data with many variables, rather than data with fewer variables.

Reported information

Reported from the EDA is as follows.

- Overview
 - Data Structures
 - Data Types
 - Job Informations
- Univariate Analysis
 - Descriptive Statistics

- Normality Test
- Bivariate Analysis
 - Compare Numerical Variables
 - Compare Categorical Variables
- Multivariate Analysis
 - Correlation Analysis
 - * Correlation Matrix
 - * Correlation Plot
- Target based Analysis
 - Grouped Numerical Variables
 - Grouped Categorical Variables
 - Grouped Correlation

See Also

[eda_web_report.data.frame.](#)

Examples

```

if (FALSE) {
library(dplyr)

# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure2 to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure2, name = "TB_HEARTFAILURE", overwrite = TRUE)

# reporting the diagnosis information -----
# create pdf file. file name is EDA_Report.html
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_web_report(target = "death_event")

# create pdf file. file name is EDA.html, and collect size is 250
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  eda_web_report(collect_size = 250, output_file = "EDA.html")

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)
}

```

entropy	<i>Calculate the entropy</i>
---------	------------------------------

Description

Calculate the Shannon's entropy.

Usage

```
entropy(x)
```

Arguments

x a numeric vector.

Value

numeric. entropy

Examples

```
set.seed(123)
x <- sample(1:10, 20, replace = TRUE)

entropy(x)
```

extract	<i>Extract bins from "bins"</i>
---------	---------------------------------

Description

The extract() extract binned variable from "bins", "optimal_bins" class object.

Usage

```
extract(x)

## S3 method for class 'bins'
extract(x)
```

Arguments

x a bins class or optimal_bins class.

Details

The "bins" and "optimal_bins" class objects use the `summary()` and `plot()` functions to diagnose the performance of binned results. This function is used to extract the binned result if you are satisfied with the result.

Value

factor.

See Also

[binning](#), [binning_by](#).

Examples

```
library(dplyr)

# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA

# optimal binning using binning_by()
bin <- binning_by(heartfailure2, "death_event", "creatinine")
bin

# extract binning result
extract(bin) %>%
  head(20)
```

find_class

Extract variable names or indices of a specific class

Description

The `find_class()` extracts variable information having a certain class from an object inheriting `data.frame`.

Usage

```
find_class(
  df,
  type = c("numerical", "categorical", "categorical2", "date_categorical",
    "date_categorical2"),
  index = TRUE
)
```


Arguments

df	a data.frame or objects inheriting from data.frame
type	character. Defines a group of classes to be searched. "numerical" searches for "numeric" and "integer" classes, "categorical" searches for "factor" and "ordered" classes. "categorical2" adds "character" class to "categorical". "date_categorical" adds result of "categorical2" and "Date", "POSIXct". "date_categorical2" adds result of "categorical" and "Date", "POSIXct".
index	logical. If TRUE is return numeric vector that is variables index. and if FALSE is return character vector that is variables name. default is TRUE.

Value

character vector or numeric vector. The meaning of vector according to data type is as follows.

- character vector : variables name
- numeric vector : variables index

See Also

[get_class](#).

Examples

```
## Not run:
# data.frame
find_class(iris, "numerical")
find_class(iris, "numerical", index = FALSE)
find_class(iris, "categorical")
find_class(iris, "categorical", index = FALSE)

# tbl_df
find_class(ggplot2::diamonds, "numerical")
find_class(ggplot2::diamonds, "numerical", index = FALSE)
find_class(ggplot2::diamonds, "categorical")
find_class(ggplot2::diamonds, "categorical", index = FALSE)

# type is "categorical2"
iris2 <- data.frame(iris, char = "chars",
                   stringsAsFactors = FALSE)
find_class(iris2, "categorical", index = FALSE)
find_class(iris2, "categorical2", index = FALSE)

## End(Not run)
```

find_na	<i>Finding variables including missing values</i>
---------	---

Description

Find the variable that contains the missing value in the object that inherits the data.frame or data.frame.

Usage

```
find_na(.data, index = TRUE, rate = FALSE)
```

Arguments

.data	a data.frame or a tbl_df .
index	logical. When representing the information of a variable including missing values, specify whether or not the variable is represented by an index. Returns an index if TRUE or a variable names if FALSE.
rate	logical. If TRUE, returns the percentage of missing values in the individual variable.

Value

Information on variables including missing values.

See Also

[impute_na](#), [find_outliers](#).

Examples

```
## Not run:
find_na(jobchange)

find_na(jobchange, index = FALSE)

find_na(jobchange, rate = TRUE)

## using dplyr -----
library(dplyr)

# Perform simple data quality diagnosis of variables with missing values.
jobchange %>%
  select(find_na(.)) %>%
  diagnose()

## End(Not run)
```

find_outliers	<i>Finding variables including outliers</i>
---------------	---

Description

Find the numerical variable that contains outliers in the object that inherits the data.frame or data.frame.

Usage

```
find_outliers(.data, index = TRUE, rate = FALSE)
```

Arguments

.data	a data.frame or a tbl_df .
index	logical. When representing the information of a variable including outliers, specify whether or not the variable is represented by an index. Returns an index if TRUE or a variable names if FALSE.
rate	logical. If TRUE, returns the percentage of outliers in the individual variable.

Value

Information on variables including outliers.

See Also

[find_na](#), [imputate_outlier](#).

Examples

```
## Not run:
find_outliers(heartfailure)

find_outliers(heartfailure, index = FALSE)

find_outliers(heartfailure, rate = TRUE)

## using dplyr -----
library(dplyr)

# Perform simple data quality diagnosis of variables with outliers.
heartfailure %>%
  select(find_outliers(.)) %>%
  diagnose()

## End(Not run)
```

find_skewness	<i>Finding skewed variables</i>
---------------	---------------------------------

Description

Find the numerical variable that skewed variable that inherits the data.frame or data.frame.

Usage

```
find_skewness(.data, index = TRUE, value = FALSE, thres = NULL)
```

Arguments

.data	a data.frame or a tbl_df .
index	logical. When representing the information of a skewed variable, specify whether or not the variable is represented by an index. Returns an index if TRUE or a variable names if FALSE.
value	logical. If TRUE, returns the skewness value in the individual variable.
thres	Returns a skewness threshold value that has an absolute skewness greater than thres. The default is NULL to ignore the threshold. but, If value = TRUE, default to 0.5.

Value

Information on variables including skewness.

See Also

[find_na](#), [find_outliers](#).

Examples

```
## Not run:
find_skewness(heartfailure)

find_skewness(heartfailure, index = FALSE)

find_skewness(heartfailure, thres = 0.1)

find_skewness(heartfailure, value = TRUE)

find_skewness(heartfailure, value = TRUE, thres = 0.1)

## using dplyr -----
library(dplyr)

# Perform simple data quality diagnosis of skewed variables
heartfailure %>%
```

```
select(find_skewness(.)) %>%  
diagnose()  
  
## End(Not run)
```

get_class	<i>Extracting a class of variables</i>
-----------	--

Description

The `get_class()` gets class of variables in `data.frame` or `tbl_df`.

Usage

```
get_class(df)
```

Arguments

`df` a `data.frame` or objects inheriting from `data.frame`

Value

a `data.frame` Variables of `data.frame` is as follows.

- `variable` : variables name
- `class` : class of variables

See Also

[find_class](#).

Examples

```
## Not run:  
# data.frame  
get_class(iris)  
  
# tbl_df  
get_class(ggplot2::diamonds)  
  
library(dplyr)  
ggplot2::diamonds %>%  
  get_class() %>%  
  filter(class %in% c("integer", "numeric"))  
  
## End(Not run)
```

get_column_info	<i>Describe column of table in the DBMS</i>
-----------------	---

Description

The `get_column_info()` retrieves the column information of the DBMS table through the `tbl_dbi` object of `dplyr`.

Usage

```
get_column_info(df)
```

Arguments

`df` a `tbl_dbi`.

Value

An object of `data.frame`.

Column information of the DBMS table

- SQLite DBMS connected `RSQLite::SQLite()`:
 - name: column name
 - type: data type in R
- MySQL/MariaDB DBMS connected `RMySQL::MySQL()`:
 - name: column name
 - Sclass: data type in R
 - type: data type of column in the DBMS
 - length: data length in the DBMS
- Oracle DBMS connected `ROracle::dbConnect()`:
 - name: column name
 - Sclass: column type in R
 - type: data type of column in the DBMS
 - len: length of column(`CHAR/VARCHAR/VARCHAR2` data type) in the DBMS
 - precision: precision of column(`NUMBER` data type) in the DBMS
 - scale: decimal places of column(`NUMBER` data type) in the DBMS
 - nullOK: nullability

Examples

```
library(dplyr)
# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  get_column_info

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)
```

get_os	<i>Finding Users Machine's OS</i>
--------	-----------------------------------

Description

Get the operating system that users machines.

Usage

```
get_os()
```

Value

OS names. "windows" or "osx" or "linux"

Examples

```
get_os()
```

get_percentile	<i>Finding percentile</i>
----------------	---------------------------

Description

Find the percentile of the value specified in numeric vector.

Usage

```
get_percentile(x, value, from = 0, to = 1, eps = 1e-06)
```

Arguments

x	numeric. a numeric vector.
value	numeric. a scalar to find percentile value from vector x.
from	numeric. Start interval in logic to find percentile value. default to 0.
to	numeric. End interval in logic to find percentile value. default to 1.
eps	numeric. Threshold value for calculating the approximate value in recursive calling logic to find the percentile value. (epsilon). default to 1e-06.

Value

list. Components of list. is as follows.

- percentile : numeric. Percentile position of value. It has a value between [0, 100].
- is_outlier : logical. Whether value is an outlier.

Examples

```
## Not run:
carat <- ggplot2::diamonds$carat

quantile(carat)

get_percentile(carat, value = 0.5)
get_percentile(carat, value = median(diamonds$carat))
get_percentile(carat, value = 1)
get_percentile(carat, value = 7)

## End(Not run)
```

get_transform	<i>Transform a numeric vector</i>
---------------	-----------------------------------

Description

The get_transform() gets transformation of numeric variable.

Usage

```
get_transform(
  x,
  method = c("log", "sqrt", "log+1", "log+a", "1/x", "x^2", "x^3", "Box-Cox",
    "Yeo-Johnson")
)
```

Arguments

x	numeric. numeric for transform
method	character. transformation method of numeric variable

Details

The supported transformation method is follow.:

- "log" : log transformation. $\log(x)$
- "log+1" : log transformation. $\log(x + 1)$. Used for values that contain 0.
- "log+a" : log transformation. $\log(x + 1 - \min(x))$. Used for values that contain 0.
- "sqrt" : square root transformation.
- "1/x" : $1 / x$ transformation
- "x^2" : x square transformation
- "x^3" : x³ square transformation
- "Box-Cox" : Box-Box transformation
- "Yeo-Johnson" : Yeo-Johnson transformation

Value

numeric. transformed numeric vector.

See Also

[plot_normality.](#)

Examples

```
## Not run:
# log+a transform
get_transform(iris$Sepal.Length, "log+a")

if (requireNamespace("forecast", quietly = TRUE)) {
  # Box-Cox transform
  get_transform(iris$Sepal.Length, "Box-Cox")

  # Yeo-Johnson transform
  get_transform(iris$Sepal.Length, "Yeo-Johnson")
} else {
  cat("If you want to use this feature, you need to install the forecast package.\n")
}

## End(Not run)
```

heartfailure

Heart Failure Data

Description

A dataset containing the ages and other attributes of almost 300 cases.

Usage

`data(heartfailure)`

Format

A data frame with 299 rows and 13 variables. The variables are as follows:

age patient's age.

anaemia decrease of red blood cells or hemoglobin (boolean), Yes, No.

cpk_enzyme level of the CPK(creatinine phosphokinase) enzyme in the blood (mcg/L).

diabetes if the patient has diabetes (boolean), Yes, No.

ejection_fraction percentage of blood leaving the heart at each contraction (percentage).

hblood_pressure high_blood_pressure. if the patient has hypertension (boolean), Yes, No.

platelets platelets in the blood (kiloplatelets/mL).

creatinine level of serum creatinine in the blood (mg/dL).

sodium level of serum sodium in the blood (mEq/L).

sex patient's sex (binary), Male, Female.

smoking if the patient smokes or not (boolean), Yes, No.

time follow-up period (days).

death_event if the patient deceased during the follow-up period (boolean), Yes, No.

Details

Heart failure is a common event caused by Cardiovascular diseases and this dataset contains 12 features that can be used to predict mortality by heart failure.

Source

"Heart Failure Prediction" in Kaggle <<https://www.kaggle.com/andrewmvd/heart-failure-clinical-data>>, License : CC BY 4.0

References

Davide Chicco, Giuseppe Jurman: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making 20, 16 (2020). <<https://doi.org/10.1186/s12911-020-1023-5>>

import_liberation	<i>Import Liberation Sans Narrow fonts</i>
-------------------	--

Description

Import Liberation Sans Narrow font to be used when drawing charts.

Usage

```
import_liberation()
```

Details

The Liberation(tm) Fonts is a font family which aims at metric compatibility with Arial, Times New Roman, and Courier New. It is sponsored by Red Hat. Import fonts is older versions of the Liberation(tm) fonts. This is released as open source under the GNU General Public License version 2 with exceptions. <https://fedoraproject.org/wiki/Licensing/LiberationFontLicense>

imputate_na	<i>Impute Missing Values</i>
-------------	------------------------------

Description

Missing values are imputed with some representative values and statistical methods.

Usage

```
imputate_na(.data, xvar, yvar, method, seed, print_flag, no_attr)
```

Arguments

.data	a data.frame or a tbl_df .
xvar	variable name to replace missing value.
yvar	target variable.
method	method of missing values imputation.
seed	integer. the random seed used in mice. only used "mice" method.
print_flag	logical. If TRUE, mice will print running log on console. Use print_flag=FALSE for silent computation. Used only when method is "mice".
no_attr	logical. If TRUE, return numerical variable or categorical variable. else If FALSE, imputation class.

Details

`imputate_na()` creates an imputation class. The ‘imputation’ class includes missing value position, imputed value, and method of missing value imputation, etc. The ‘imputation’ class compares the imputed value with the original value to help determine whether the imputed value is used in the analysis.

See `vignette("transformation")` for an introduction to these concepts.

Value

An object of imputation class. or numerical variable or categorical variable. if `no_attrs` is `FALSE` then return imputation class, else `no_attrs` is `TRUE` then return numerical vector or factor. Attributes of imputation class is as follows.

- `var_type` : the data type of predictor to replace missing value.
- `method` : method of missing value imputation.
 - predictor is numerical variable.
 - * "mean" : arithmetic mean.
 - * "median" : median.
 - * "mode" : mode.
 - * "knn" : K-nearest neighbors.
 - * "rpart" : Recursive Partitioning and Regression Trees.
 - * "mice" : Multivariate Imputation by Chained Equations.
 - predictor is categorical variable.
 - * "mode" : mode.
 - * "rpart" : Recursive Partitioning and Regression Trees.
 - * "mice" : Multivariate Imputation by Chained Equations.
- `na_pos` : position of missing value in predictor.
- `seed` : the random seed used in mice. only used "mice" method.
- `type` : "missing values". type of imputation.
- `message` : a message tells you if the result was successful.
- `success` : Whether the imputation was successful.

See Also

[imputate_outlier](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

# Replace the missing value of the platelets variable with median
imputate_na(heartfailure2, platelets, method = "median")
```

```

# Replace the missing value of the platelets variable with rpart
# The target variable is death_event.
imputate_na(heartfailure2, platelets, death_event, method = "rpart")

# Replace the missing value of the smoking variable with mode
imputate_na(heartfailure2, smoking, method = "mode")

# Replace the missing value of the smoking variable with mice
# The target variable is death_event.
imputate_na(heartfailure2, smoking, death_event, method = "mice")

## using dplyr -----
library(dplyr)

# The mean before and after the imputation of the platelets variable
heartfailure2 %>%
  mutate(platelets_imp = imputate_na(heartfailure2, platelets, death_event,
                                     method = "knn", no_attrs = TRUE)) %>%
  group_by(death_event) %>%
  summarise(orig = mean(platelets, na.rm = TRUE),
            imputation = mean(platelets_imp))

# If the variable of interest is a numerical variable
platelets <- imputate_na(heartfailure2, platelets, death_event, method = "rpart")
platelets
summary(platelets)

# plot(platelets)

# If the variable of interest is a categorical variable
smoking <- imputate_na(heartfailure2, smoking, death_event, method = "mice")
smoking
summary(smoking)

# plot(smoking)

```

imputate_outlier

Impute Outliers

Description

Outliers are imputed with some representative values and statistical methods.

Usage

```
imputate_outlier(.data, xvar, method, no_attrs)
```

Arguments

<code>.data</code>	a data.frame or a tbl_df .
<code>xvar</code>	variable name to replace missing value.
<code>method</code>	method of missing values imputation.
<code>no_attr</code>	logical. If TRUE, return numerical variable or categorical variable. else If FALSE, imputation class.

Details

`impute_outlier()` creates an imputation class. The ‘imputation’ class includes missing value position, imputed value, and method of missing value imputation, etc. The ‘imputation’ class compares the imputed value with the original value to help determine whether the imputed value is used in the analysis.

See `vignette("transformation")` for an introduction to these concepts.

Value

An object of imputation class. or numerical variable. if `no_attr` is FALSE then return imputation class, else `no_attr` is TRUE then return numerical vector. Attributes of imputation class is as follows.

- `method` : method of missing value imputation.
 - predictor is numerical variable
 - * "mean" : arithmetic mean
 - * "median" : median
 - * "mode" : mode
 - * "capping" : Impute the upper outliers with 95 percentile, and Impute the bottom outliers with 5 percentile.
- `outlier_pos` : position of outliers in predictor.
- `outliers` : outliers. outliers corresponding to `outlier_pos`.
- `type` : "outliers". type of imputation.

See Also

[impute_na](#).

Examples

```
# Replace the outliers of the sodium variable with median.
impute_outlier(heartfailure, sodium, method = "median")

# Replace the outliers of the sodium variable with capping.
impute_outlier(heartfailure, sodium, method = "capping")

## using dplyr -----
library(dplyr)
```

```

# The mean before and after the imputation of the sodium variable
heartfailure %>%
  mutate(sodium_imp = imutate_outlier(heartfailure, sodium,
                                      method = "capping", no_attrs = TRUE)) %>%
  group_by(death_event) %>%
  summarise(orig = mean(sodium, na.rm = TRUE),
            imputation = mean(sodium_imp, na.rm = TRUE))

# If the variable of interest is a numerical variables
sodium <- imutate_outlier(heartfailure, sodium)
sodium
summary(sodium)

# plot(sodium)

```

 jobchange

Job Change of Data Scientists

Description

A dataset containing the gender and other attributes of almost 20000 cases.

Usage

```
data(jobchange)
```

Format

A data frame with 19158 rows and 14 variables. The variables are as follows:

enrollee_id unique ID for candidate

city city code.

city_dev_index developement index of the city (scaled).

gender gender of candidate.

relevent_experience relevant experience of candidate

enrolled_university type of University course enrolled if any.

education_level education level of candidate.

major_discipline education major discipline of candidate.

experience candidate total experience in years.

company_size number of employees in current employer's company.

company_type type of current employer.

last_new_job difference in years between previous job and current job.

training_hours training hours completed.

job_chnge if looking for a job change (boolean), Yes, No.

Details

This dataset designed to understand the factors that lead a person to leave current job for HR researches too.

Source

"HR Analytics: Job Change of Data Scientists" in Kaggle <<https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists>>, License : CC0(Public Domain)

jsd	<i>Jensen-Shannon Divergence</i>
-----	----------------------------------

Description

Computes the Jensen-Shannon divergence between two probability distributions.

Usage

```
jsd(p, q, base = c("log", "log2", "log10"), margin = FALSE)
```

Arguments

p	numeric. probability distributions.
q	numeric. probability distributions.
base	character. log bases. "log", "log2", "log10". default is "log"
margin	logical. Choose whether to return individual values or totals. The default value is FALSE, which returns individual values.

Value

numeric. Jensen-Shannon divergence of probability distributions p and q.

See Also

[kld](#).

Examples

```
# Sample data for probability distributions p.
event <- c(115, 76, 61, 39, 55, 10, 1)
no_event <- c(3, 3, 7, 10, 28, 44, 117)

p <- event / sum(event)
q <- no_event / sum(no_event)

jsd(p, q)
jsd(p, q, base = "log2")
jsd(p, q, margin = TRUE)
```

`kld`*Kullback-Leibler Divergence*

Description

Computes the Kullback-Leibler divergence between two probability distributions.

Usage

```
kld(p, q, base = c("log", "log2", "log10"), margin = FALSE)
```

Arguments

<code>p</code>	numeric. probability distributions.
<code>q</code>	numeric. probability distributions.
<code>base</code>	character. log bases. "log", "log2", "log10". default is "log"
<code>margin</code>	logical. Choose whether to return individual values or totals. The default value is FALSE, which returns individual values.

Value

numeric. Kullback-Leibler divergence of probability distributions `p` and `q`.

See Also

[jsd](#).

Examples

```
# Sample data for probability distributions p.
event <- c(115, 76, 61, 39, 55, 10, 1)
no_event <- c(3, 3, 7, 10, 28, 44, 117)

p <- event / sum(event)
q <- no_event / sum(no_event)

kld(p, q)
kld(p, q, base = "log2")
kld(p, q, margin = TRUE)
```

kurtosis	<i>Kurtosis of the data</i>
----------	-----------------------------

Description

This function calculated kurtosis of given data.

Usage

```
kurtosis(x, na.rm = FALSE)
```

Arguments

x	a numeric vector.
na.rm	logical. Determine whether to remove missing values and calculate them. The default is TRUE.

Value

numeric. calculated kurtosis

See Also

[skewness](#).

Examples

```
set.seed(123)
kurtosis(rnorm(100))
```

normality	<i>Performs the Shapiro-Wilk test of normality</i>
-----------	--

Description

The normality() performs Shapiro-Wilk test of normality of numerical values.

Usage

```
normality(.data, ...)

## S3 method for class 'data.frame'
normality(.data, ..., sample = 5000)

## S3 method for class 'grouped_df'
normality(.data, ..., sample = 5000)
```

Arguments

<code>.data</code>	a data.frame or a tbl_df .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>normality()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>sample</code>	the number of samples to perform the test. See <code>vignette("EDA")</code> for an introduction to these concepts.

Details

This function is useful when used with the [group_by](#) function of the `dplyr` package. If you want to test by level of the categorical data you are interested in, rather than the whole observation, you can use `group_tf` as the `group_by` function. This function is computed [shapiro.test](#) function.

Value

An object of the same class as `.data`.

Normality test information

The information derived from the numerical data test is as follows.

- `statistic` : the value of the Shapiro-Wilk statistic.
- `p_value` : an approximate p-value for the test. This is said in Royston(1995) to be adequate for `p_value < 0.1`.
- `sample` : the number of samples to perform the test. The number of observations supported by the `stats::shapiro.test` function is 3 to 5000.

See Also

[normality.tbl_dbi](#), [diagnose_numeric.data.frame](#), [describe.data.frame](#), [plot_normality.data.frame](#).

Examples

```
# Normality test of numerical variables
normality(heartfailure)

# Select the variable to describe
normality(heartfailure, platelets, sodium)
normality(heartfailure, -platelets, -sodium)
normality(heartfailure, 1)
normality(heartfailure, platelets, sodium, sample = 200)

# death_eventing dplyr::grouped_dt
library(dplyr)
```

```

gdata <- group_by(heartfailure, smoking, death_event)
normality(gdata, "platelets")
normality(gdata, sample = 250)

# death_eventing pipes -----
# Normality test of all numerical variables
heartfailure %>%
  normality()

# Positive values select variables
heartfailure %>%
  normality(platelets, sodium)

# Positions values select variables
heartfailure %>%
  normality(1)

# death_eventing pipes & dplyr -----
# Test all numerical variables by 'smoking' and 'death_event',
# and extract only those with 'smoking' variable level is "No".
heartfailure %>%
  group_by(smoking, death_event) %>%
  normality() %>%
  filter(smoking == "No")

# extract only those with 'sex' variable level is "Male",
# and test 'platelets' by 'smoking' and 'death_event'
heartfailure %>%
  filter(sex == "Male") %>%
  group_by(smoking, death_event) %>%
  normality(platelets)

# Test log(platelets) variables by 'smoking' and 'death_event',
# and extract only p.value greater than 0.01.
heartfailure %>%
  mutate(platelets_income = log(platelets)) %>%
  group_by(smoking, death_event) %>%
  normality(platelets_income) %>%
  filter(p_value > 0.01)

```

normality.tbl_dbi

Performs the Shapiro-Wilk test of normality

Description

The `normality()` performs Shapiro-Wilk test of normality of numerical(INTEGER, NUMBER, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```
## S3 method for class 'tbl_dbi'
normality(.data, ..., sample = 5000, in_database = FALSE, collect_size = Inf)
```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>normality()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>sample</code>	the number of samples to perform the test.
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> . See <code>vignette("EDA")</code> for an introduction to these concepts.

Details

This function is useful when used with the `group_by` function of the `dplyr` package. If you want to test by level of the categorical data you are interested in, rather than the whole observation, you can use `group_tf` as the `group_by` function. This function is computed `shapiro.test` function.

Value

An object of the same class as `.data`.

Normality test information

The information derived from the numerical data test is as follows.

- `statistic` : the value of the Shapiro-Wilk statistic.
- `p_value` : an approximate p-value for the test. This is said in Royston(1995) to be adequate for `p_value < 0.1`.
- `sample` : the numer of samples to perform the test. The number of observations supported by the `stats::shapiro.test` function is 3 to 5000.

See Also

[normality.data.frame](#), [diagnose_numeric.tbl_dbi](#), [describe.tbl_dbi](#), [plot_normality.tbl_dbi](#).

Examples

```

library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Normality test of all numerical variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  normality()

# Positive values select variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  normality(platelets, sodium, collect_size = 200)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  normality(1)

# Using pipes & dplyr -----
# Test all numerical variables by 'smoking' and 'death_event',
# and extract only those with 'smoking' variable level is "Yes".
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(smoking, death_event) %>%
  normality() %>%
  filter(smoking == "Yes")

# extract only those with 'sex' variable level is "Male",
# and test 'sodium' by 'smoking' and 'death_event'
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  filter(sex == "Male") %>%
  group_by(smoking, death_event) %>%
  normality(sodium)

# Test log(sodium) variables by 'smoking' and 'death_event',
# and extract only p.value greater than 0.01.

# SQLite extension functions for log
RSQLite::initExtension(con_sqlite)

con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  mutate(log_sodium = log(sodium)) %>%

```

```

group_by(smoking, death_event) %>%
normality(log_sodium) %>%
filter(p_value > 0.01)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

 overview

Describe overview of data

Description

Inquire basic information to understand the data in general.

Usage

```
overview(.data)
```

Arguments

`.data` a `data.frame` or a `tbl_df`.

Details

`overview()` creates an overview class. The ‘overview’ class includes general information such as the size of the data, the degree of missing values, and the data types of variables.

Value

An object of overview class. The overview class contains `data.frame` and two attributes. `data.frame` has the following 3 variables.: `data.frame` is as follow.:

- `division` : division of information.
 - `size` : indicators of related to data capacity
 - `duplicated` : indicators of related to duplicated value
 - `missing` : indicators of related to missing value
 - `data_type` : indicators of related to data type
- `metrics` : name of metrics.
 - `observations` : number of observations (number of rows)
 - `variables` : number of variables (number of columns)
 - `values` : number of values (number of cells. rows * columns)
 - `memory size` : an estimate of the memory that is being used to store an R object.
 - `duplicate observation`: number of duplicate cases(observations).
 - `complete observation` : number of complete cases(observations). i.e., have no missing values.

- missing observation : number of observations that has missing values.
 - missing variables : number of variables that has missing values.
 - missing values : number of values(cells) that has missing values.
 - numerics : number of variables that is data type is numeric.
 - integers : number of variables that is data type is integer.
 - factors : number of variables that is data type is factor.
 - characters : number of variables that is data type is character.
 - Dates : number of variables that is data type is Date.
 - POSIXcts : number of variables that is data type is POSIXct.
 - others : number of variables that is not above.
- value : value of metrics.

Attributes of overview class is as follows.:

- duplicated : the index of duplicated observations.
- na_col : the data type of predictor to replace missing value.
- info_class : data.frame. variable name and class name that describe the data type of variables.
 - data.frame has a two variables.
 - * variable : variable names
 - * class : data type

See Also

[summary.overview](#), [plot.overview](#).

Examples

```
ov <- overview(jobchange)
ov

summary(ov)

# plot(ov)
```

Description

The `performance_bin()` calculates metrics to evaluate the performance of binned variable for binomial classification model.

Usage

```
performance_bin(y, x, na.rm = FALSE)
```

Arguments

y character or numeric, integer, factor. a binary response variable (0, 1). The variable must contain only the integers 0 and 1 as element. However, in the case of factor/character having two levels, it is performed while type conversion is performed in the calculation process.

x integer or factor, character. At least 2 different values. and Inf is not allowed.

na.rm logical. a logical indicating whether missing values should be removed.

Details

This function is useful when used with the mutate/transmute function of the dplyr package.

Value

an object of "performance_bin" class. vaue of data.frame is as follows.

- Bin : character. bins.
- CntRec : integer. frequency by bins.
- CntPos : integer. frequency of positive by bins.
- CntNeg : integer. frequency of negative by bins.
- CntCumPos : integer. cumulate frequency of positive by bins.
- CntCumNeg : integer. cumulate frequency of negative by bins.
- RatePos : integer. relative frequency of positive by bins.
- RateNeg : integer. relative frequency of negative by bins.
- RateCumPos : numeric. cumulate relative frequency of positive by bins.
- RateCumNeg : numeric. cumulate relative frequency of negative by bins.
- Odds : numeric. odd ratio.
- LnOdds : numeric. loged odd ratio.
- WoE : numeric. weight of evidence.
- IV : numeric. Jeffrey's Information Value.
- JSD : numeric. Jensen-Shannon Divergence.
- AUC : numeric. AUC. area under curve.

Attributes of "performance_bin" class is as follows.

- names : character. variable name of data.frame with "Binning Table".
- class : character. name of class. "performance_bin" "data.frame".
- row.names : character. row name of data.frame with "Binning Table".
- IV : numeric. Jeffrey's Information Value.

- JSD : numeric. Jensen-Shannon Divergence.
- KS : numeric. Kolmogorov-Smirnov Statistics.
- gini : numeric. Gini index.
- HHI : numeric. Herfindahl-Hirschman Index.
- HHI_norm : numeric.normalized Herfindahl-Hirschman Index.
- Cramer_V : numeric. Cramer's V Statistics.
- chisq_test : data.frame. table of significance tests. name is as follows.
 - Bin A : character. first bins.
 - Bin B : character. second bins.
 - statistics : numeric. statistics of Chi-square test.
 - p_value : numeric. p-value of Chi-square test.

See Also

[summary.performance_bin](#), [plot.performance_bin](#), [binning_by](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure

set.seed(123)
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA

# Change the target variable to 0(negative) and 1(positive).
heartfailure2$death_event_2 <- ifelse(heartfailure2$death_event %in% "Yes", 1, 0)

# Binnig from creatinine to platelets_bin.
breaks <- c(0, 1, 2, 10)
heartfailure2$creatinine_bin <- cut(heartfailure2$creatinine, breaks)

# Diagnose performance binned variable
perf <- performance_bin(heartfailure2$death_event_2, heartfailure2$creatinine_bin)
perf
summary(perf)

# plot(perf)

# Diagnose performance binned variable without NA
perf <- performance_bin(heartfailure2$death_event_2, heartfailure2$creatinine_bin, na.rm = TRUE)
perf
summary(perf)

# plot(perf)
```

`plot.bins`*Visualize Distribution for a "bins" object*

Description

Visualize two plots on a single screen. The plot at the top is a histogram representing the frequency of the level. The plot at the bottom is a bar chart representing the frequency of the level.

Usage

```
## S3 method for class 'bins'  
plot(x, typographic = TRUE, ...)
```

Arguments

<code>x</code>	an object of class "bins", usually, a result of a call to <code>binning()</code> .
<code>typographic</code>	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using <code>hrbrthemes</code> package.
<code>...</code>	arguments to be passed to methods, such as graphical parameters (see <code>par</code>).

See Also

[binning](#), [print.bins](#), [summary.bins](#).

Examples

```
# Generate data for the example  
heartfailure2 <- heartfailure  
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA  
  
# Binning the carat variable. default type argument is "quantile"  
bin <- binning(heartfailure2$platelets, nbins = 5)  
plot(bin)  
  
# Using another type arguments  
bin <- binning(heartfailure2$platelets, nbins = 5, type = "equal")  
plot(bin)  
  
#bin <- binning(heartfailure2$platelets, nbins = 5, type = "pretty")  
#plot(bin)  
  
#bin <- binning(heartfailure2$platelets, nbins = 5, type = "kmeans")  
#plot(bin)  
  
bin <- binning(heartfailure2$platelets, nbins = 5, type = "bclust")  
plot(bin)
```

plot.compare_category *Visualize Information for an "compare_category" Object*

Description

Visualize mosaics plot by attribute of compare_category class.

Usage

```
## S3 method for class 'compare_category'
plot(x, prompt = FALSE, na.rm = FALSE, typographic = TRUE, ...)
```

Arguments

x	an object of class "compare_category", usually, a result of a call to compare_category().
prompt	logical. The default value is FALSE. If there are multiple visualizations to be output, if this argument value is TRUE, a prompt is output each time.
na.rm	logical. Specifies whether to include NA when plotting mosaics plot. The default is FALSE, so plot NA.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.
...	arguments to be passed to methods, such as graphical parameters (see par). However, it only support las parameter. las is numeric in 0,1; the style of axis labels. <ul style="list-style-type: none"> • 0 : always parallel to the axis [default], • 1 : always horizontal to the axis,

See Also

[compare_category](#), [print.compare_category](#), [summary.compare_category](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

library(dplyr)

# Compare the all categorical variables
all_var <- compare_category(heartfailure2)

# Print compare_numeric class objects
all_var

# Compare the two categorical variables
```

```
two_var <- compare_category(heartfailure2, smoking, death_event)

# Print compare_category class objects
two_var

# plot all pair of variables
plot(all_var)

# plot a pair of variables
plot(two_var)

# plot all pair of variables by prompt
# plot(all_var, prompt = TRUE)

# plot a pair of variables without NA
plot(two_var, na.rm = TRUE)

# plot a pair of variables
plot(two_var, las = 1)

# plot a pair of variables not focuses on typographic elements
plot(two_var, typographic = FALSE)
```

plot.compare_numeric *Visualize Information for an "compare_numeric" Object*

Description

Visualize scatter plot included box plots by attribute of compare_numeric class.

Usage

```
## S3 method for class 'compare_numeric'
plot(x, prompt = FALSE, typographic = TRUE, ...)
```

Arguments

x	an object of class "compare_numeric", usually, a result of a call to compare_numeric().
prompt	logical. The default value is FALSE. If there are multiple visualizations to be output, if this argument value is TRUE, a prompt is output each time.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hbrthemes package.
...	arguments to be passed to methods, such as graphical parameters (see par). However, it does not support.

See Also

[compare_numeric](#), [print.compare_numeric](#), [summary.compare_numeric](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure[, c("platelets", "creatinine", "sodium")]

library(dplyr)
# Compare the all numerical variables
all_var <- compare_numeric(heartfailure2)

# Print compare_numeric class object
all_var

# Compare the two numerical variables
two_var <- compare_numeric(heartfailure2, sodium, creatinine)

# Print compare_numeric class objects
two_var

# plot all pair of variables
plot(all_var)

# plot a pair of variables
plot(two_var)

# plot all pair of variables by prompt
# plot(all_var, prompt = TRUE)

# plot a pair of variables not focuses on typographic elements
plot(two_var, typographic = FALSE)
```

plot.imputation

Visualize Information for an "imputation" Object

Description

Visualize two kinds of plot by attribute of 'imputation' class. The imputation of a numerical variable is a density plot, and the imputation of a categorical variable is a bar plot.

Usage

```
## S3 method for class 'imputation'
plot(x, typographic = TRUE, ...)
```

Arguments

x	an object of class "imputation", usually, a result of a call to <code>imputate_na()</code> or <code>imputate_outlier()</code> .
typographic	logical. Whether to apply focuses on typographic elements to <code>ggplot2</code> visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using <code>hrbrthemes</code> package.
...	arguments to be passed to methods, such as graphical parameters (see <code>par</code>). only applies when the <code>model</code> argument is TRUE, and is used for ... of the <code>plot.lm()</code> function.

See Also

[imputate_na](#), [imputate_outlier](#), [summary.imputation](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

# Impute missing values -----
# If the variable of interest is a numerical variables
platelets <- imputate_na(heartfailure2, platelets, death_event, method = "rpart")
platelets
summary(platelets)

plot(platelets)

# If the variable of interest is a categorical variables
smoking <- imputate_na(heartfailure2, smoking, death_event, method = "mice")
smoking
summary(smoking)

plot(smoking)

# Impute outliers -----
# If the variable of interest is a numerical variable
platelets <- imputate_outlier(heartfailure2, platelets, method = "capping")
platelets
summary(platelets)

plot(platelets)
```

plot.optimal_bins *Visualize Distribution for an "optimal_bins" Object*

Description

It generates plots for understand distribution, frequency, bad rate, and weight of evidence using optimal_bins.

See vignette("transformation") for an introduction to these concepts.

Usage

```
## S3 method for class 'optimal_bins'
plot(
  x,
  type = c("all", "dist", "freq", "posrate", "WoE"),
  typographic = TRUE,
  rotate_angle = 0,
  ...
)
```

Arguments

x	an object of class "optimal_bins", usually, a result of a call to binning_by().
type	character. options for visualization. Distribution ("dist"), Relative Frequency ("freq"), Positive Rate ("posrate"), and Weight of Evidence ("WoE"). and default "all" draw all plot.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.
rotate_angle	integer. specifies the rotation angle of the x-axis label. This is useful when the x-axis labels are long and overlap. The default is 0 to not rotate the label.
...	further arguments to be passed from or to other methods.

See Also

[binning_by](#), [summary.optimal_bins](#)

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA

# optimal binning using binning_by()
bin <- binning_by(heartfailure2, "death_event", "creatinine")
bin
```



```
# summary optimal_bins class.
summary(bin)

# visualize all information for optimal_bins class
plot(bin)

# rotate the x-axis labels by 45 degrees so that they do not overlap.
plot(bin, rotate_angle = 45)

# visualize WoE information for optimal_bins class
plot(bin, type = "WoE")

# visualize all information with typographic
plot(bin)
```

plot.overview	<i>Visualize Information for an "overview" Object</i>
---------------	---

Description

Visualize a plot by attribute of 'overview' class. Visualize the data type, number of observations, and number of missing values for each variable.

Usage

```
## S3 method for class 'overview'
plot(x, order_type = c("none", "name", "type"), ...)
```

Arguments

x	an object of class "overview", usually, a result of a call to overview().
order_type	character. method of order of bars(variables).
...	further arguments to be passed from or to other methods.

See Also

[overview](#), [summary.overview](#).

Examples

```
ov <- overview(jobchange)
ov

summary(ov)

plot(ov)
```

```
# sort by name of variables
plot(ov, order_type = "name")

# sort by data type of variables
plot(ov, order_type = "type")
```

plot.performance_bin *Visualize Performance for an "performance_bin" Object*

Description

It generates plots for understand frequency, WoE by bins using performance_bin.

Usage

```
## S3 method for class 'performance_bin'
plot(x, typographic = TRUE, ...)
```

Arguments

x	an object of class "performance_bin", usually, a result of a call to performance_bin().
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.
...	further arguments to be passed from or to other methods.

See Also

[performance_bin](#), [summary.performance_bin](#), [binning_by](#), [plot.optimal_bins](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure

set.seed(123)
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA

# Change the target variable to 0(negative) and 1(positive).
heartfailure2$death_event_2 <- ifelse(heartfailure2$death_event %in% "Yes", 1, 0)

# Binnig from creatinine to platelets_bin.
breaks <- c(0, 1, 2, 10)
heartfailure2$creatinine_bin <- cut(heartfailure2$creatinine, breaks)

# Diagnose performance binned variable
```

```

perf <- performance_bin(heartfailure2$death_event_2, heartfailure2$creatinine_bin)
perf
summary(perf)

plot(perf)

# Diagnose performance binned variable without NA
perf <- performance_bin(heartfailure2$death_event_2, heartfailure2$creatinine_bin, na.rm = TRUE)
perf
summary(perf)

plot(perf)
plot(perf, typographic = FALSE)

```

plot.relate

Visualize Information for an "relate" Object

Description

Visualize four kinds of plot by attribute of relate class.

Usage

```

## S3 method for class 'relate'
plot(
  x,
  model = FALSE,
  hex_thres = 1000,
  pal = c("#FFFFFFB2", "#FED976", "#FEB24C", "#FD8D3C", "#FC4E2A", "#E31A1C", "#B10026"),
  typographic = TRUE,
  ...
)

```

Arguments

x	an object of class "relate", usually, a result of a call to relate().
model	logical. This argument selects whether to output the visualization result to the visualization of the object of the lm model to grasp the relationship between the numerical variables.
hex_thres	an integer. Use only when the target and predictor are numeric variables. Used when the number of observations is large. Specify the threshold of the observations to draw hexabin plots that are not scatterplots. The default value is 1000.
pal	Color palette to paint hexabin. Use only when the target and predictor are numeric variables. Applied only when the number of observations is greater than hex_thres.

typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.
...	arguments to be passed to methods, such as graphical parameters (see par). only applies when the model argument is TRUE, and is used for ... of the plot.lm() function.

See Also

[relate](#), [print.relate](#).

Examples

```
# If the target variable is a categorical variable
categ <- target_by(heartfailure, death_event)

# If the variable of interest is a numerical variable
cat_num <- relate(categ, sodium)
cat_num
summary(cat_num)

plot(cat_num)

# If the variable of interest is a categorical variable
cat_cat <- relate(categ, hblood_pressure)
cat_cat
summary(cat_cat)

plot(cat_cat)

##-----
# If the target variable is a numerical variable
num <- target_by(heartfailure, creatinine)

# If the variable of interest is a numerical variable
num_num <- relate(num, sodium)
num_num
summary(num_num)

plot(num_num)

# If the variable of interest is a categorical variable
num_cat <- relate(num, smoking)
num_cat
summary(num_cat)

plot(num_cat)

# Not allow typographic
plot(num_cat, typographic = FALSE)
```

plot.transform *Visualize Information for an "transform" Object*

Description

Visualize two kinds of plot by attribute of ‘transform’ class. The transformation of a numerical variable is a density plot.

Usage

```
## S3 method for class 'transform'  
plot(x, typographic = TRUE, ...)
```

Arguments

x an object of class "transform", usually, a result of a call to transform().

typographic logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hbrthemes package.

... arguments to be passed to methods, such as graphical parameters (see par).

See Also

[transform](#), [summary.transform](#).

Examples

```
# Standardization -----  
creatinine_minmax <- transform(heartfailure$creatinine, method = "minmax")  
creatinine_minmax  
summary(creatinine_minmax)  
  
plot(creatinine_minmax)  
  
# Resolving Skewness -----  
creatinine_log <- transform(heartfailure$creatinine, method = "log")  
creatinine_log  
summary(creatinine_log)  
  
plot(creatinine_log)  
  
plot(creatinine_log, typographic = FALSE)
```

plot.univar_category *Visualize Information for an "univar_category" Object*

Description

Visualize mosaics plot by attribute of univar_category class.

Usage

```
## S3 method for class 'univar_category'  
plot(x, na.rm = TRUE, prompt = FALSE, typographic = TRUE, ...)
```

Arguments

x	an object of class "univar_category", usually, a result of a call to univar_category().
na.rm	logical. Specifies whether to include NA when plotting bar plot. The default is FALSE, so plot NA.
prompt	logical. The default value is FALSE. If there are multiple visualizations to be output, if this argument value is TRUE, a prompt is output each time.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.
...	arguments to be passed to methods, such as graphical parameters (see par). However, it does not support all parameters.

See Also

[univar_category](#), [print.univar_category](#), [summary.univar_category](#).

Examples

```
library(dplyr)  
  
# Calculates the all categorical variables  
all_var <- univar_category(heartfailure)  
  
# Print univar_category class object  
all_var  
  
smoking <- univar_category(heartfailure, smoking)  
  
# Print univar_category class object  
smoking  
  
# plot all variables  
plot(all_var)
```

```
# plot smoking
plot(smoking)
```

plot.univar_numeric *Visualize Information for an "univar_numeric" Object*

Description

Visualize boxplots and histogram by attribute of univar_numeric class.

Usage

```
## S3 method for class 'univar_numeric'
plot(
  x,
  indiv = FALSE,
  viz = c("hist", "boxplot"),
  stand = ifelse(rep(indiv, 4), c("none", "robust", "minmax", "zscore"), c("robust",
    "minmax", "zscore", "none")),
  prompt = FALSE,
  typographic = TRUE,
  ...
)
```

Arguments

x	an object of class "univar_numeric", usually, a result of a call to univar_numeric().
indiv	logical. Select whether to display information of all variables in one plot when there are multiple selected numeric variables. In case of FALSE, all variable information is displayed in one plot. If TRUE, the information of the individual variables is output to the individual plots. The default is FALSE. If only one variable is selected, TRUE is applied.
viz	character. Describe what to plot visualization. "hist" draws a histogram and "boxplot" draws a boxplot. The default is "hist".
stand	character. Describe how to standardize the original data. "robust" normalizes the raw data through transformation calculated by IQR and median. "minmax" normalizes the original data using minmax transformation. "zscore" standardizes the original data using z-Score transformation. "none" does not perform data transformation. The default is "none" if indiv is TRUE, and "robust" if FALSE.
prompt	logical. The default value is FALSE. If there are multiple visualizations to be output, if this argument value is TRUE, a prompt is output each time.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. If TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.
...	arguments to be passed to methods, such as graphical parameters (see par). However, it does not support.

See Also

[univar_numeric](#), [print.univar_numeric](#), [summary.univar_numeric](#).

Examples

```
# Calculates the all categorical variables
all_var <- univar_numeric(heartfailure)

# Print univar_numeric class object
all_var

# Calculates the platelets, sodium variable
univar_numeric(heartfailure, platelets, sodium)

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Summary by returned object
stat

# one plot with all variables
plot(all_var)

# one plot with all normalized variables by Min-Max method
# plot(all_var, stand = "minmax")

# one plot with all variables
# plot(all_var, stand = "none")

# one plot with all robust standardized variables
plot(all_var, viz = "boxplot")

# one plot with all standardized variables by Z-score method
# plot(all_var, viz = "boxplot", stand = "zscore")

# individual boxplot by variables
# plot(all_var, indiv = TRUE, "boxplot")

# individual histogram by variables
plot(all_var, indiv = TRUE, "hist")

# individual histogram by robust standardized variable
# plot(all_var, indiv = TRUE, "hist", stand = "robust")

# plot all variables by prompt
# plot(all_var, indiv = TRUE, "hist", prompt = TRUE)
```


Description

The `plot_bar_category()` to visualizes the distribution of categorical data by level or relationship to specific numerical data by level.

Usage

```
plot_bar_category(.data, ...)

## S3 method for class 'data.frame'
plot_bar_category(
  .data,
  ...,
  top = 10,
  add_character = TRUE,
  title = "Frequency by levels of category",
  each = FALSE,
  typographic = TRUE
)

## S3 method for class 'grouped_df'
plot_bar_category(
  .data,
  ...,
  top = 10,
  add_character = TRUE,
  title = "Frequency by levels of category",
  each = FALSE,
  typographic = TRUE
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> or a <code>grouped_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_bar_category()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>top</code>	an integer. Specifies the upper top rank to extract. Default is 10.
<code>add_character</code>	logical. Decide whether to include text variables in the diagnosis of categorical data. The default value is <code>TRUE</code> , which also includes character variables.
<code>title</code>	character. a main title for the plot.
<code>each</code>	logical. Specifies whether to draw multiple plots on one screen. The default is <code>FALSE</code> , which draws multiple plots on one screen.

`typographic` logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using `hrbrthemes` package.

Details

The distribution of categorical variables can be understood by comparing the frequency of each level. The frequency table helps with this. As a visualization method, a bar graph can help you understand the distribution of categorical data more easily than a frequency table.

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

set.seed(123)
heartfailure2$test <- sample(LETTERS[1:15], 299, replace = TRUE)
heartfailure2$test[1:30] <- NA

# Visualization of all numerical variables
plot_bar_category(heartfailure2)

# Select the variable to diagnose
# plot_bar_category(heartfailure2, "test", "smoking")
# plot_bar_category(heartfailure2, -test, -smoking)

# Visualize the each plots
# plot_bar_category(heartfailure2, each = TRUE)

# Not allow typographic argument
# plot_bar_category(heartfailure2, typographic = FALSE)

# Using pipes -----
library(dplyr)

# Plot of all categorical variables
# heartfailure2 %>%
#   plot_bar_category()

# Visualize just 7 levels of top frequency
# heartfailure2 %>%
#   plot_bar_category(top = 7)

# Visualize only factor, not character
# heartfailure2 %>%
#   plot_bar_category(add_character = FALSE)

# Using group_df -----
# heartfailure2 %>%
#   group_by(death_event) %>%
```

```
# plot_bar_category(top = 5)

# heartfailure2 %>%
#   group_by(death_event) %>%
#   plot_bar_category(each = TRUE, top = 5)
```

plot_box_numeric *Plot Box-Plot of numerical variables*

Description

The `plot_box_numeric()` to visualizes the box plot of numeric data or relationship to specific categorical data.

Usage

```
plot_box_numeric(.data, ...)

## S3 method for class 'data.frame'
plot_box_numeric(
  .data,
  ...,
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE
)

## S3 method for class 'grouped_df'
plot_box_numeric(
  .data,
  ...,
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE
)
```

Arguments

<code>.data</code>	data.frame or a tbl_df or a grouped_df .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_box_numeric()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>title</code>	character. a main title for the plot.

each	logical. Specifies whether to draw multiple plots on one screen. The default is FALSE, which draws multiple plots on one screen.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Details

The box plot helps determine whether the distribution of a numeric variable. `plot_box_numeric()` shows box plots of several numeric variables on one screen. This function can also display a box plot for each level of a specific categorical variable.

Examples

```
# Visualization of all numerical variables
# plot_box_numeric(heartfailure)

# Select the variable to diagnose
# plot_box_numeric(heartfailure, "age", "time")
plot_box_numeric(heartfailure, -age, -time)

# Visualize the each plots
# plot_box_numeric(heartfailure, "age", "time", each = TRUE)

# Not allow the typographic elements
# plot_box_numeric(heartfailure, typographic = FALSE)

# Using pipes -----
library(dplyr)

# Plot of all numerical variables
# heartfailure %>%
#   plot_box_numeric()

# Using groupd_df -----
heartfailure %>%
  group_by(smoking) %>%
  plot_box_numeric()

# heartfailure %>%
#   group_by(smoking) %>%
#   plot_box_numeric(each = TRUE)
```

plot_correlate

Visualize correlation plot of numerical data

Description

The `plot_correlate()` visualize correlation plot for find relationship between two numerical variables.

Usage

```
plot_correlate(.data, ...)

## S3 method for class 'data.frame'
plot_correlate(.data, ..., method = c("pearson", "kendall", "spearman"))

## S3 method for class 'grouped_df'
plot_correlate(.data, ..., method = c("pearson", "kendall", "spearman"))
```

Arguments

.data	a data.frame or a tbl_df .
...	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_correlate()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing. See <code>vignette("EDA")</code> for an introduction to these concepts.
method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.

Details

The scope of the visualization is to provide a correlation information. Since the plot is drawn for each variable, if you specify more than one variable in the `...` argument, the specified number of plots are drawn.

See Also

[plot_correlate.tbl_dbi](#), [plot_outlier.data.frame](#).

Examples

```
# Visualize correlation plot of all numerical variables
plot_correlate(heartfailure)

# Select the variable to compute
plot_correlate(heartfailure, creatinine, sodium)
plot_correlate(heartfailure, -creatinine, -sodium)
plot_correlate(heartfailure, "creatinine", "sodium")
plot_correlate(heartfailure, 1)
plot_correlate(heartfailure, creatinine, sodium, method = "spearman")

# Using dplyr::grouped_dt
library(dplyr)

gdata <- group_by(heartfailure, smoking, death_event)
```

```

plot_correlate(gdata, "creatinine")
plot_correlate(gdata)

# Using pipes -----
# Visualize correlation plot of all numerical variables
heartfailure %>%
  plot_correlate()
# Positive values select variables
heartfailure %>%
  plot_correlate(creatinine, sodium)
# Negative values to drop variables
heartfailure %>%
  plot_correlate(-creatinine, -sodium)
# Positions values select variables
heartfailure %>%
  plot_correlate(1)
# Positions values select variables
heartfailure %>%
  plot_correlate(-1, -3, -5, -7)

# Using pipes & dplyr -----
# Visualize correlation plot of 'creatinine' variable by 'smoking'
# and 'death_event' variables.
heartfailure %>%
  group_by(smoking, death_event) %>%
  plot_correlate(creatinine)

# Extract only those with 'smoking' variable level is "Yes",
# and visualize correlation plot of 'creatinine' variable by 'hblood_pressure'
# and 'death_event' variables.
heartfailure %>%
  filter(smoking == "Yes") %>%
  group_by(hblood_pressure, death_event) %>%
  plot_correlate(creatinine)

```

```
plot_correlate.tbl_dbi
```

Visualize correlation plot of numerical data

Description

The `plot_correlate()` visualize correlation plot for find relationship between two numerical (INTEGER, NUMBER, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```

## S3 method for class 'tbl_dbi'
plot_correlate(
  .data,

```

```

    ...,
    in_database = FALSE,
    collect_size = Inf,
    method = c("pearson", "kendall", "spearman")
  )

```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_correlate()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported in <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .
<code>method</code>	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated. See vignette("EDA") for an introduction to these concepts.

Details

The scope of the visualization is the provide a correlation information. Since the plot is drawn for each variable, if you specify more than one variable in the `...` argument, the specified number of plots are drawn.

See Also

[plot_correlate.data.frame](#), [plot_outlier.tbl_dbi](#).

Examples

```

library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Visualize correlation plot of all numerical variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%

```

```

plot_correlate()

# Positive values select variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_correlate(platelets, sodium, collect_size = 200)

# Negative values to drop variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_correlate(-platelets, -sodium)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_correlate(1)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_correlate(-1, -2, -3, -5, -6)

# Using pipes & dplyr -----
# Visualize correlation plot of 'sodiumsodium' variable by 'smoking'
# and 'death_event' variables.
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(smoking, death_event) %>%
  plot_correlate(sodium)

# Extract only those with 'smoking' variable level is "Yes",
# and visualize correlation plot of 'sodium' variable by 'sex'
# and 'death_event' variables.
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  filter(smoking == "Yes") %>%
  group_by(sex, death_event) %>%
  plot_correlate(sodium)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

plot_hist_numeric

Plot histogram of numerical variables

Description

The `plot_hist_numeric()` to visualizes the histogram of numeric data or relationship to specific categorical data.

Usage

```
plot_hist_numeric(.data, ...)

## S3 method for class 'data.frame'
plot_hist_numeric(
  .data,
  ...,
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE
)

## S3 method for class 'grouped_df'
plot_hist_numeric(
  .data,
  ...,
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE
)
```

Arguments

<code>.data</code>	data.frame or a tbl_df or a grouped_df .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_hist_numeric()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>title</code>	character. a main title for the plot.
<code>each</code>	logical. Specifies whether to draw multiple plots on one screen. The default is <code>FALSE</code> , which draws multiple plots on one screen.
<code>typographic</code>	logical. Whether to apply focuses on typographic elements to <code>ggplot2</code> visualization. The default is <code>TRUE</code> . if <code>TRUE</code> provides a base theme that focuses on typographic elements using <code>hrbrthemes</code> package.

Details

The histogram helps determine whether the distribution of a numeric variable. `plot_hist_numeric()` shows box plots of several numeric variables on one screen. This function can also display a histogram for each level of a specific categorical variable. The bin-width is set to the Freedman-Diaconis rule ($2 * IQR(x) / \text{length}(x)^{1/3}$)

Examples

```
# Visualization of all numerical variables
# plot_hist_numeric(heartfailure)
```

```

# Select the variable to diagnose
# plot_hist_numeric(heartfailure, "age", "time")
plot_hist_numeric(heartfailure, -age, -time)

# Visualize the each plots
# plot_hist_numeric(heartfailure, "age", "time", each = TRUE)

# Not allow the typographic elements
# plot_hist_numeric(heartfailure, typographic = FALSE)

# Using pipes -----
library(dplyr)

# Plot of all numerical variables
# heartfailure %>%
#   plot_hist_numeric()

# Using groupd_df -----
heartfailure %>%
  group_by(smoking) %>%
  plot_hist_numeric()

# heartfailure %>%
#   group_by(smoking) %>%
#   plot_hist_numeric(each = TRUE)

```

plot_na_hclust

Combination chart for missing value

Description

Visualize distribution of missing value by combination of variables.

Usage

```

plot_na_hclust(
  x,
  main = NULL,
  col.left = "#009E73",
  col.right = "#56B4E9",
  typographic = TRUE
)

```

Arguments

x data frames, or objects to be coerced to one.
main character. Main title.

col.left	character. The color of left legend that is frequency of NA. default is "#009E73".
col.right	character. The color of right legend that is percentage of NA. default is "#56B4E9".
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Details

Rows are variables containing missing values, and columns are observations. These data structures were grouped into similar groups by applying hclust. So, it was made possible to visually examine how the missing values are distributed for each combination of variables.

Examples

```
# Generate data for the example
set.seed(123L)
jobchange2 <- jobchange[sample(nrow(jobchange), size = 1000), ]

# Visualize hcluster chart for variables with missing value.
plot_na_hclust(jobchange2)

# Change the main title.
plot_na_hclust(jobchange2, main = "Distribution of missing value")

# Not support typographic elements
plot_na_hclust(jobchange2, typographic = FALSE)
```

plot_na_intersect *Plot the combination variables that is include missing value*

Description

Visualize the combinations of missing value across cases.

Usage

```
plot_na_intersect(
  x,
  only_na = TRUE,
  n_interacts = NULL,
  n_vars = NULL,
  main = NULL,
  typographic = TRUE
)
```

Arguments

x	data frames, or objects to be coerced to one.
only_na	logical. The default value is FALSE. If TRUE, only variables containing missing values are selected for visualization. If FALSE, included complete case.
n_intersects	integer. Specifies the number of combinations of variables including missing values. The combination of variables containing many missing values is chosen first.
n_vars	integer. Specifies the number of variables that contain missing values to be visualized. The default value is NULL, which visualizes variables containing all missing values. If this value is greater than the number of variables containing missing values, all variables containing missing values are visualized. Variables containing many missing values are chosen first.
main	character. Main title.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Details

The visualization consists of four parts. The bottom left, which is the most basic, visualizes the case of cross(intersection)-combination. The x-axis is the variable including the missing value, and the y-axis represents the case of a combination of variables. And on the marginal of the two axes, the frequency of the case is expressed as a bar graph. Finally, the visualization at the top right expresses the number of variables including missing values in the data set, and the number of observations including missing values and complete cases .

Examples

```
# Generate data for the example
set.seed(123L)
jobchange2 <- jobchange[sample(nrow(jobchange), size = 1000), ]

# Visualize the combination variables that is include missing value.
plot_na_intersect(jobchange2)

# Diagnose the data with missing_count using diagnose() function
library(dplyr)

jobchange2 %>%
  diagnose %>%
  arrange(desc(missing_count))

# Visualize the combination variables that is include missing value
plot_na_intersect(jobchange2)

# Visualize variables containing missing values and complete case
plot_na_intersect(jobchange2, only_na = FALSE)

# Using n_vars argument
```

```

plot_na_intersect(jobchange2, n_vars = 5)

# Using n_intersects argument
plot_na_intersect(jobchange2, only_na = FALSE, n_intersects = 7)

# Not allow typographic elements
plot_na_intersect(jobchange2, typographic = FALSE)

```

plot_na_pareto *Pareto chart for missing value*

Description

Visualize pareto chart for variables with missing value.

Usage

```

plot_na_pareto(
  x,
  only_na = FALSE,
  relative = FALSE,
  main = NULL,
  col = "black",
  grade = list(Good = 0.05, OK = 0.1, NotBad = 0.2, Bad = 0.5, Remove = 1),
  plot = TRUE,
  typographic = TRUE
)

```

Arguments

x	data frames, or objects to be coerced to one.
only_na	logical. The default value is FALSE. If TRUE, only variables containing missing values are selected for visualization. If FALSE, all variables are included.
relative	logical. If this argument is TRUE, it sets the unit of the left y-axis to relative frequency. In case of FALSE, set it to frequency.
main	character. Main title.
col	character. The color of line for display the cumulative percentage.
grade	list. Specifies the cut-off to set the grade of the variable according to the ratio of missing values. The default values are Good: [0, 0.05], OK: (0.05, 0.1], NotBad: (0.1, 0.2], Bad: (0.2, 0.5], Remove: (0.5, 1].
plot	logical. If this value is TRUE then visualize plot. else if FALSE, return aggregate information about missing values.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Examples

```
# Generate data for the example
set.seed(123L)
jobchange2 <- jobchange[sample(nrow(jobchange), size = 1000), ]

# Diagnose the data with missing_count using diagnose() function
library(dplyr)

jobchange2 %>%
  diagnose %>%
  arrange(desc(missing_count))

# Visualize pareto chart for variables with missing value.
plot_na_pareto(jobchange2)

# Visualize pareto chart for variables with missing value.
plot_na_pareto(jobchange2, col = "blue")

# Visualize only variables containing missing values
plot_na_pareto(jobchange2, only_na = TRUE)

# Display the relative frequency
plot_na_pareto(jobchange2, relative = TRUE)

# Change the grade
plot_na_pareto(jobchange2, grade = list(High = 0.1, Middle = 0.6, Low = 1))

# Change the main title.
plot_na_pareto(jobchange2, relative = TRUE, only_na = TRUE,
               main = "Pareto Chart for jobchange")

# Return the aggregate information about missing values.
plot_na_pareto(jobchange2, only_na = TRUE, plot = FALSE)

# Not support typographic elements
plot_na_pareto(jobchange2, typographic = FALSE)
```

plot_normality

Plot distribution information of numerical data

Description

The `plot_normality()` visualize distribution information for normality test of the numerical data.

Usage

```
plot_normality(.data, ...)
```

```

## S3 method for class 'data.frame'
plot_normality(
  .data,
  ...,
  left = c("log", "sqrt", "log+1", "log+a", "1/x", "x^2", "x^3", "Box-Cox",
           "Yeo-Johnson"),
  right = c("sqrt", "log", "log+1", "log+a", "1/x", "x^2", "x^3", "Box-Cox",
            "Yeo-Johnson"),
  col = "steelblue",
  typographic = TRUE
)

## S3 method for class 'grouped_df'
plot_normality(
  .data,
  ...,
  left = c("log", "sqrt", "log+1", "log+a", "1/x", "x^2", "x^3", "Box-Cox",
           "Yeo-Johnson"),
  right = c("sqrt", "log", "log+1", "log+a", "1/x", "x^2", "x^3", "Box-Cox",
            "Yeo-Johnson"),
  col = "steelblue",
  typographic = TRUE
)

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_normality()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing. See <code>vignette("EDA")</code> for an introduction to these concepts.
<code>left</code>	character. Specifies the data transformation method to draw the histogram in the lower left corner. The default is "log".
<code>right</code>	character. Specifies the data transformation method to draw the histogram in the lower right corner. The default is "sqrt".
<code>col</code>	a color to be used to fill the bars. The default is "steelblue".
<code>typographic</code>	logical. Whether to apply focuses on typographic elements to <code>ggplot2</code> visualization. The default is <code>TRUE</code> . if <code>TRUE</code> provides a base theme that focuses on typographic elements using <code>hrbrthemes</code> package.

Details

The scope of the visualization is the provide a distribution information. Since the plot is drawn for each variable, if you specify more than one variable in the `...` argument, the specified number of plots are drawn.

The argument values that left and right can have are as follows.:

- "log" : log transformation. $\log(x)$
- "log+1" : log transformation. $\log(x + 1)$. Used for values that contain 0.
- "log+a" : log transformation. $\log(x + 1 - \min(x))$. Used for values that contain 0.
- "sqrt" : square root transformation.
- "1/x" : $1 / x$ transformation
- "x^2" : x square transformation
- "x^3" : x^3 square transformation
- "Box-Cox" : Box-Box transformation
- "Yeo-Johnson" : Yeo-Johnson transformation

Distribution information

The plot derived from the numerical data visualization is as follows.

- histogram by original data
- q-q plot by original data
- histogram by log transfer data
- histogram by square root transfer data

See Also

[plot_normality.tbl_dbi](#), [plot_outlier.data.frame](#).

Examples

```
# Visualization of all numerical variables
heartfailure2 <- heartfailure[, c("creatinine", "platelets", "sodium", "sex", "smoking")]
plot_normality(heartfailure2)

# Select the variable to plot
plot_normality(heartfailure2, platelets, sodium)
plot_normality(heartfailure2, -platelets, -sodium, col = "gray")
plot_normality(heartfailure2, 1)

# Change the method of transformation
plot_normality(heartfailure2, platelets, right = "1/x")

if (requireNamespace("forecast", quietly = TRUE)) {
  plot_normality(heartfailure2, platelets, left = "Box-Cox", right = "Yeo-Johnson")
} else {
  cat("If you want to use this feature, you need to install the rpart package.\n")
}
# Not allow typographic elements
plot_normality(heartfailure2, platelets, typographic = FALSE)
```



```

# Using dplyr::grouped_df
library(dplyr)

gdata <- group_by(heartfailure2, sex, smoking)
plot_normality(gdata)
plot_normality(gdata, "creatinine")

# Using pipes -----
# Visualization of all numerical variables
heartfailure2 %>%
  plot_normality()

# Positive values select variables
heartfailure2 %>%
  plot_normality(platelets, sodium)

# Positions values select variables
# heartfailure2 %>%
#   plot_normality(1)

# Using pipes & dplyr -----
# Plot 'creatinine' variable by 'sex' and 'smoking'
heartfailure2 %>%
  group_by(sex, smoking) %>%
  plot_normality(creatinine)

# extract only those with 'sex' variable level is "Male",
# and plot 'platelets' by 'smoking'
if (requireNamespace("forecast", quietly = TRUE)) {
  heartfailure2 %>%
    filter(sex == "Male") %>%
    group_by(smoking) %>%
    plot_normality(platelets, right = "Box-Cox")
} else {
  cat("If you want to use this feature, you need to install the rpart package.\n")
}

```

```
plot_normality.tbl_dbi
```

Plot distribution information of numerical data

Description

The `plot_normality()` visualize distribution information for normality test of the numerical (INTEGER, NUMBER, etc.) column of the DBMS table through `tbl_dbi`.

Usage

```
## S3 method for class 'tbl_dbi'
```

```

plot_normality(
  .data,
  ...,
  in_database = FALSE,
  collect_size = Inf,
  left = c("log", "sqrt", "log+1", "1/x", "x^2", "x^3", "Box-Cox", "Yeo-Johnson"),
  right = c("sqrt", "log", "log+1", "1/x", "x^2", "x^3", "Box-Cox", "Yeo-Johnson"),
  col = "steelblue",
  typographic = TRUE
)

```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_normality()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing. See <code>vignette("EDA")</code> for an introduction to these concepts.
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .
<code>left</code>	character. Specifies the data transformation method to draw the histogram in the lower left corner. The default is "log".
<code>right</code>	character. Specifies the data transformation method to draw the histogram in the lower right corner. The default is "sqrt".
<code>col</code>	a color to be used to fill the bars. The default is "steelblue".
<code>typographic</code>	logical. Whether to apply focuses on typographic elements to <code>ggplot2</code> visualization.

Details

The scope of the visualization is the provide a distribution information. Since the plot is drawn for each variable, if you specify more than one variable in the `...` argument, the specified number of plots are drawn.

The argument values that `left` and `right` can have are as follows.:

- "log" : log transformation. $\log(x)$
- "log+1" : log transformation. $\log(x + 1)$. Used for values that contain 0.
- "sqrt" : square root transformation.
- "1/x" : $1 / x$ transformation

- "x^2" : x square transformation
- "x^3" : x^3 square transformation
- "Box-Cox" : Box-Box transformation
- "Yeo-Johnson" : Yeo-Johnson transformation

Distribution information

The plot derived from the numerical data visualization is as follows.

- histogram by original data
- q-q plot by original data
- histogram by log transfer data
- histogram by square root transfer data

See Also

[plot_normality.data.frame](#), [plot_outlier.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Visualization of all numerical variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_normality()

# Positive values select variables, and In-memory mode and collect size is 200
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_normality(platelets, sodium, collect_size = 200)

# Positions values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_normality(1)

# Not allow the typographic elements
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_normality(1, typographic = FALSE)
```

```

# Using pipes & dplyr -----
# Plot 'sodium' variable by 'smoking' and 'death_event'
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(smoking, death_event) %>%
  plot_normality(sodium)

# Plot using left and right arguments
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(smoking, death_event) %>%
  plot_normality(sodium, left = "Box-Cox", right = "log")

# extract only those with 'smoking' variable level is "Yes",
# and plot 'sodium' by 'death_event'
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  filter(smoking == "Yes") %>%
  group_by(death_event) %>%
  plot_normality(sodium)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

plot_outlier

Plot outlier information of numerical data diagnosis

Description

The `plot_outlier()` visualize outlier information for diagnosing the quality of the numerical data.

Usage

```

plot_outlier(.data, ...)

## S3 method for class 'data.frame'
plot_outlier(.data, ..., col = "steelblue", typographic = TRUE)

```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_outlier()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

col	a color to be used to fill the bars. The default is "steelblue".
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Details

The scope of the diagnosis is the provide a outlier information. Since the plot is drawn for each variable, if you specify more than one variable in the ... argument, the specified number of plots are drawn.

Outlier diagnostic information

The plot derived from the numerical data diagnosis is as follows.

- With outliers box plot
- Without outliers box plot
- With outliers histogram
- Without outliers histogram

See vignette("diagonosis") for an introduction to these concepts.

See Also

[plot_outlier.tbl_dbi](#), [diagnose_outlier.data.frame](#).

Examples

```
# Visualization of all numerical variables
# plot_outlier(heartfailure)

# Select the variable to diagnose
plot_outlier(heartfailure, cpk_enzyme, sodium)
# plot_outlier(heartfailure, -cpk_enzyme, -sodium)
# plot_outlier(heartfailure, "cpk_enzyme", "sodium")
# plot_outlier(heartfailure, 7)

# Using the col argument
# plot_outlier(heartfailure, cpk_enzyme, col = "gray")

# Not allow typographic argument
# plot_outlier(heartfailure, cpk_enzyme, typographic = FALSE)

# Using pipes -----
library(dplyr)

# Visualization of all numerical variables
# heartfailure %>%
#   plot_outlier()
```

```

# Positive values select variables
heartfailure %>%
  plot_outlier(cpk_enzyme, sodium)

# Negative values to drop variables
# heartfailure %>%
#   plot_outlier(-cpk_enzyme, -sodium)

# Positions values select variables
# heartfailure %>%
#   plot_outlier(7)

# Positions values select variables
# heartfailure %>%
#   plot_outlier(-1, -5)

# Using pipes & dplyr -----
# Visualization of numerical variables with a ratio of
# outliers greater than 5%
# heartfailure %>%
#   plot_outlier(heartfailure %>%
#     diagnose_outlier() %>%
#     filter(outliers_ratio > 5) %>%
#     select(variables) %>%
#     pull())

```

plot_outlier.target_df

Plot outlier information of target_df

Description

The `plot_outlier()` visualize outlier information for diagnosing the quality of the numerical data with `target_df` class.

Usage

```

## S3 method for class 'target_df'
plot_outlier(.data, ..., typographic = TRUE)

```

Arguments

<code>.data</code>	a <code>target_df</code> . reference target_by .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_outlier()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

typographic logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Details

The scope of the diagnosis is the provide a outlier information. Since the plot is drawn for each variable, if you specify more than one variable in the ... argument, the specified number of plots are drawn.

Outlier diagnostic information

The plot derived from the numerical data diagnosis is as follows.

- With outliers box plot by target variable
- Without outliers box plot by target variable
- With outliers density plot by target variable
- Without outliers density plot by target variable

See Also

[plot_outlier.data.frame.](#)

Examples

```
# the target variable is a categorical variable
categ <- target_by(heartfailure, death_event)

plot_outlier(categ, sodium)
plot_outlier(categ, sodium, typographic = FALSE)

# death_eventing dplyr
library(dplyr)
heartfailure %>%
  target_by(death_event) %>%
  plot_outlier(sodium, cpk_enzyme)

# death_eventing DBMS tables -----
# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# If the target variable is a categorical variable
categ <- target_by(con_sqlite %>% tbl("TB_HEARTFAILURE") , death_event)

plot_outlier(categ, sodium)
```

plot_outlier.tbl_dbi *Plot outlier information of numerical data diagnosis in the DBMS*

Description

The plot_outlier() visualize outlier information for diagnosing the quality of the numerical(INTEGER, NUMBER, etc.) column of the DBMS table through tbl_dbi.

Usage

```
## S3 method for class 'tbl_dbi'
plot_outlier(
  .data,
  ...,
  col = "steelblue",
  in_database = FALSE,
  collect_size = Inf,
  typographic = TRUE
)
```

Arguments

.data	a tbl_dbi.
...	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, plot_outlier() will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
col	a color to be used to fill the bars. The default is "lightblue".
in_database	Specifies whether to perform in-database operations. If TRUE, most operations are performed in the DBMS. if FALSE, table data is taken in R and operated in-memory. Not yet supported in_database = TRUE.
collect_size	a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
typographic	logical. Whether to apply focuses on typographic elements to ggplot2 visualization. The default is TRUE. if TRUE provides a base theme that focuses on typographic elements using hrbrthemes package.

Details

The scope of the diagnosis is the provide a outlier information. Since the plot is drawn for each variable, if you specify more than one variable in the ... argument, the specified number of plots are drawn.

Outlier diagnostic information

The plot derived from the numerical data diagnosis is as follows.

- With outliers box plot
- Without outliers box plot
- With outliers histogram
- Without outliers histogram

See vignette("diagonosis") for an introduction to these concepts.

See Also

[plot_outlier.data.frame](#), [diagnose_outlier.tbl_dbi](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# Using pipes -----
# Visualization of all numerical variables
# con_sqlite %>%
#   tbl("TB_HEARTFAILURE") %>%
#   plot_outlier()

# Positive values select variables
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  plot_outlier(platelets, sodium)

# Negative values to drop variables, and In-memory mode and collect size is 200
# con_sqlite %>%
#   tbl("TB_HEARTFAILURE") %>%
#   plot_outlier(-platelets, -sodium, collect_size = 200)

# Positions values select variables
# con_sqlite %>%
#   tbl("TB_HEARTFAILURE") %>%
#   plot_outlier(6)

# Positions values select variables
# con_sqlite %>%
#   tbl("TB_HEARTFAILURE") %>%
#   plot_outlier(-1, -5)

# Not allow the typographic elements
```

```

# con_sqlite %>%
#   tbl("TB_HEARTFAILURE") %>%
#   plot_outlier(-1, -5, typographic = FALSE)

# Using pipes & dplyr -----
# Visualization of numerical variables with a ratio of
# outliers greater than 1%
# con_sqlite %>%
#   tbl("TB_HEARTFAILURE") %>%
#   plot_outlier(con_sqlite %>%
#     tbl("TB_HEARTFAILURE") %>%
#     diagnose_outlier() %>%
#     filter(outliers_ratio > 1) %>%
#     select(variables) %>%
#     pull())

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

plot_qq_numeric	<i>Plot Q-Q plot of numerical variables</i>
-----------------	---

Description

The `plot_qq_numeric()` to visualizes the Q-Q plot of numeric data or relationship to specific categorical data.

Usage

```

plot_qq_numeric(.data, ...)

## S3 method for class 'data.frame'
plot_qq_numeric(
  .data,
  ...,
  col_point = "steelblue",
  col_line = "black",
  title = "Q-Q plot by numerical variables",
  each = FALSE,
  typographic = TRUE
)

## S3 method for class 'grouped_df'
plot_qq_numeric(
  .data,
  ...,
  col_point = "steelblue",

```

```

  col_line = "black",
  title = "Q-Q plot by numerical variables",
  each = FALSE,
  typographic = TRUE
)

```

Arguments

<code>.data</code>	data.frame or a <code>tbl_df</code> or a <code>grouped_df</code> .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. If the first expression is negative, <code>plot_qq_numeric()</code> will automatically start with all variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.
<code>col_point</code>	character. a color of points in Q-Q plot.
<code>col_line</code>	character. a color of line in Q-Q plot.
<code>title</code>	character. a main title for the plot.
<code>each</code>	logical. Specifies whether to draw multiple plots on one screen. The default is <code>FALSE</code> , which draws multiple plots on one screen.
<code>typographic</code>	logical. Whether to apply focuses on typographic elements to <code>ggplot2</code> visualization. The default is <code>TRUE</code> . if <code>TRUE</code> provides a base theme that focuses on typographic elements using <code>hrbrthemes</code> package.

Details

The The Q-Q plot helps determine whether the distribution of a numeric variable is normally distributed. `plot_qq_numeric()` shows Q-Q plots of several numeric variables on one screen. This function can also display a Q-Q plot for each level of a specific categorical variable.

Examples

```

# Visualization of all numerical variables
# plot_qq_numeric(heartfailure)

# Select the variable to diagnose
# plot_qq_numeric(heartfailure, "age", "time")
plot_qq_numeric(heartfailure, -age, -time)

# Not allow the typographic elements
# plot_qq_numeric(heartfailure, "age", typographic = FALSE)

# Using pipes -----
library(dplyr)

# Plot of all numerical variables
# heartfailure %>%
#   plot_qq_numeric()

```

```

# Using groupd_df -----
heartfailure %>%
  group_by(smoking) %>%
  plot_qq_numeric()

#heartfailure %>%
#   group_by(smoking) %>%
#   plot_qq_numeric(each = TRUE)

```

print.relate

Summarizing relate information

Description

print and summary method for "relate" class.

Usage

```

## S3 method for class 'relate'
print(x, ...)

```

Arguments

x an object of class "relate", usually, a result of a call to relate().
... further arguments passed to or from other methods.

Details

print.relate() tries to be smart about formatting four kinds of relate. summary.relate() tries to be smart about formatting four kinds of relate.

See Also

[plot.relate](#).

Examples

```

## Not run:
# If the target variable is a categorical variable
categ <- target_by(heartfailure, death_event)

# If the variable of interest is a categorical variable
cat_cat <- relate(categ, hblood_pressure)

# Print bins class object
cat_cat

summary(cat_cat)

```

```

## End(Not run)

# If the target variable is a categorical variable
categ <- target_by(heartfailure, death_event)

# If the variable of interest is a numerical variable
cat_num <- relate(categ, sodium)
cat_num
summary(cat_num)

# plot(cat_num)

# If the variable of interest is a categorical variable
cat_cat <- relate(categ, hblood_pressure)
cat_cat
summary(cat_cat)

# plot(cat_cat)

##-----
# If the target variable is a numerical variable
num <- target_by(heartfailure, creatinine)

# If the variable of interest is a numerical variable
num_num <- relate(num, sodium)
num_num
summary(num_num)

# plot(num_num)

# If the variable of interest is a categorical variable
num_cat <- relate(num, smoking)
num_cat
summary(num_cat)

# plot(num_cat)

# Not allow typographic
# plot(num_cat, typographic = FALSE)

```

relate

Relationship between target variable and variable of interest

Description

The relationship between the target variable and the variable of interest (predictor) is briefly analyzed.

Usage

```
relate(.data, predictor)

## S3 method for class 'target_df'
relate(.data, predictor)
```

Arguments

`.data` a `target_df`.

`predictor` variable of interest. predictor.
See vignette("relate") for an introduction to these concepts.

Details

Returns the four types of results that correspond to the combination of the target variable and the data type of the variable of interest.

- target variable: categorical variable
 - predictor: categorical variable
 - * contingency table
 - * `c("xtabs", "table")` class
 - predictor: numerical variable
 - * descriptive statistic for each levels and total observation.
- target variable: numerical variable
 - predictor: categorical variable
 - * ANOVA test. "lm" class.
 - predictor: numerical variable
 - * simple linear model. "lm" class.

Value

An object of the class `relate`. Attributes of `relate` class is as follows.

- `target` : name of target variable
- `predictor` : name of predictor
- `model` : levels of binned value.
- `raw` : `table_df` with two variables `target` and `predictor`.

Descriptive statistic information

The information derived from the numerical data describe is as follows.

- `mean` : arithmetic average
- `sd` : standard deviation
- `se_mean` : standrd error mean. sd/\sqrt{n}

- IQR : interquartile range (Q3-Q1)
- skewness : skewness
- kurtosis : kurtosis
- p25 : Q1. 25% percentile
- p50 : median. 50% percentile
- p75 : Q3. 75% percentile
- p01, p05, p10, p20, p30 : 1%, 5%, 20%, 30% percentiles
- p40, p60, p70, p80 : 40%, 60%, 70%, 80% percentiles
- p90, p95, p99, p100 : 90%, 95%, 99%, 100% percentiles

See Also

[print.relate](#), [plot.relate](#).

Examples

```
# If the target variable is a categorical variable
categ <- target_by(heartfailure, death_event)

# If the variable of interest is a numerical variable
cat_num <- relate(categ, sodium)
cat_num
summary(cat_num)

# plot(cat_num)

# If the variable of interest is a categorical variable
cat_cat <- relate(categ, hblood_pressure)
cat_cat
summary(cat_cat)

# plot(cat_cat)

##-----
# If the target variable is a numerical variable
num <- target_by(heartfailure, creatinine)

# If the variable of interest is a numerical variable
num_num <- relate(num, sodium)
num_num
summary(num_num)

# plot(num_num)

# If the variable of interest is a categorical variable
num_cat <- relate(num, smoking)
num_cat
summary(num_cat)
```

```
# plot(num_cat)

# Not allow typographic
# plot(num_cat, typographic = FALSE)
```

skewness

Skewness of the data

Description

This function calculated skewness of given data.

Usage

```
skewness(x, na.rm = TRUE)
```

Arguments

x	a numeric vector.
na.rm	logical. Determine whether to remove missing values and calculate them. The default is TRUE.

Value

numeric. calculated skewness.

See Also

[kurtosis](#), [find_skewness](#).

Examples

```
set.seed(123)
skewness(rnorm(100))
```

`summary.bins`*Summarizing Binned Variable*

Description

summary method for "bins" and "optimal_bins".

Usage

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

```
## S3 method for class 'bins'  
print(x, ...)
```

Arguments

<code>object</code>	an object of "bins" and "optimal_bins", usually, a result of a call to <code>binning()</code> .
<code>...</code>	further arguments passed to or from other methods.
<code>x</code>	an object of class "bins" and "optimal_bins", usually, a result of a call to <code>binning()</code> .

Details

`print.bins()` prints the information of "bins" and "optimal_bins" objects nicely. This includes frequency of bins, binned type, and number of bins. `summary.bins()` returns `data.frame` including frequency and relative frequency for each levels(bins).

See vignette("transformation") for an introduction to these concepts.

Value

The function `summary.bins()` computes and returns a `data.frame` of summary statistics of the binned given in object. Variables of data frame is as follows.

- `levels` : levels of factor.
- `freq` : frequency of levels.
- `rate` : relative frequency of levels. it is not percentage.

See Also

[binning](#)

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA

# Binning the carat variable. default type argument is "quantile"
bin <- binning(heartfailure2$platelets)

# Print bins class object
bin

# Summarize bins class object
summary(bin)
```

```
summary.compare_category
```

Summarizing compare_category information

Description

print and summary method for "compare_category" class.

Usage

```
## S3 method for class 'compare_category'
summary(
  object,
  method = c("all", "table", "relative", "chisq"),
  pos = NULL,
  na.rm = TRUE,
  marginal = FALSE,
  verbose = TRUE,
  ...
)

## S3 method for class 'compare_category'
print(x, ...)
```

Arguments

object	an object of class "compare_category", usually, a result of a call to compare_category().
method	character. Specifies the type of information to be aggregated. "table" create contingency table, "relative" create relative contingency table, and "chisq" create information of chi-square test. and "all" aggregates all information. The default is "all"

pos	integer. Specifies the pair of variables to be summarized by index. The default is NULL, which aggregates all variable pairs.
na.rm	logical. Specifies whether to include NA when counting the contingency tables or performing a chi-square test. The default is TRUE, where NA is removed and aggregated.
marginal	logical. Specifies whether to add marginal values to the contingency table. The default value is FALSE, so no marginal value is added.
verbose	logical. Specifies whether to output additional information during the calculation process. The default is to output information as TRUE. In this case, the function returns the value with invisible(). If FALSE, the value is returned by return().
...	further arguments passed to or from other methods.
x	an object of class "compare_category", usually, a result of a call to compare_category().

Details

print.compare_category() displays only the information compared between the variables included in compare_category. The "type", "variables" and "combination" attributes are not displayed. When using summary.compare_category(), it is advantageous to set the verbose argument to TRUE if the user is only viewing information from the console. It is also advantageous to specify FALSE if you want to manipulate the results.

See Also

[plot.compare_category](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

library(dplyr)

# Compare the all categorical variables
all_var <- compare_category(heartfailure2)

# Print compare_category class objects
all_var

# Compare the two categorical variables
two_var <- compare_category(heartfailure2, smoking, death_event)

# Print compare_category class objects
two_var

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)
```

```

# Summary by returned objects
stat

# component of table
stat$table

# component of chi-square test
stat$chisq

# component of chi-square test
summary(all_var, "chisq")

# component of chi-square test (first, third case)
summary(all_var, "chisq", pos = c(1, 3))

# component of relative frequency table
summary(all_var, "relative")

# component of table without missing values
summary(all_var, "table", na.rm = TRUE)

# component of table include marginal value
margin <- summary(all_var, "table", marginal = TRUE)
margin

# component of chi-square test
summary(two_var, method = "chisq")

# verbose is FALSE
summary(all_var, "chisq", verbose = FALSE)

#' # Using pipes & dplyr -----
# If you want to use dplyr, set verbose to FALSE
summary(all_var, "chisq", verbose = FALSE) %>%
  filter(p.value < 0.26)

# Extract component from list by index
summary(all_var, "table", na.rm = TRUE, verbose = FALSE) %>%
  "[ "(1)

# Extract component from list by name
summary(all_var, "table", na.rm = TRUE, verbose = FALSE) %>%
  "[ "(smoking vs death_event")

```

```
summary.compare_numeric
```

Summarizing compare_numeric information

Description

print and summary method for "compare_numeric" class.

Usage

```
## S3 method for class 'compare_numeric'
summary(
  object,
  method = c("all", "correlation", "linear"),
  thres_corr = 0.3,
  thres_rs = 0.1,
  verbose = TRUE,
  ...
)

## S3 method for class 'compare_numeric'
print(x, ...)
```

Arguments

object	an object of class "compare_numeric", usually, a result of a call to compare_numeric().
method	character. Select statistics to be aggregated. "correlation" calculates the Pearson's correlation coefficient, and "linear" returns the aggregation of the linear model. "all" returns both information. However, the difference between summary.compare_numeric() and compare_numeric() is that only cases that are greater than the specified threshold are returned. "correlation" returns only cases with a correlation coefficient greater than the thres_corr argument value. "linear" returns only cases with R ² greater than the thres_rs argument.
thres_corr	numeric. This is the correlation coefficient threshold of the correlation coefficient information to be returned. The default is 0.3.
thres_rs	numeric. R ² threshold of linear model summaries information to return. The default is 0.1.
verbose	logical. Specifies whether to output additional information during the calculation process. The default is to output information as TRUE. In this case, the function returns the value with invisible(). If FALSE, the value is returned by return().
...	further arguments passed to or from other methods.
x	an object of class "compare_numeric", usually, a result of a call to compare_numeric().

Details

print.compare_numeric() displays only the information compared between the variables included in compare_numeric. When using summary.compare_numeric(), it is advantageous to set the verbose argument to TRUE if the user is only viewing information from the console. It is also advantageous to specify FALSE if you want to manipulate the results.

Value

An object of the class as compare based list. The information to examine the relationship between numerical variables is as follows each components. - correlation component : Pearson's correlation coefficient.

- var1 : factor. The level of the first variable to compare. 'var1' is the name of the first variable to be compared.
- var2 : factor. The level of the second variable to compare. 'var2' is the name of the second variable to be compared.
- coef_corr : double. Pearson's correlation coefficient.

- linear component : linear model summaries

- var1 : factor. The level of the first variable to compare. 'var1' is the name of the first variable to be compared.
- var2 : factor. The level of the second variable to compare. 'var2' is the name of the second variable to be compared.
- r.squared : double. The percent of variance explained by the model.
- adj.r.squared : double. r.squared adjusted based on the degrees of freedom.
- sigma : double. The square root of the estimated residual variance.
- statistic : double. F-statistic.
- p.value : double. p-value from the F test, describing whether the full regression is significant.
- df : integer degrees of freedom.
- logLik : double. the log-likelihood of data under the model.
- AIC : double. the Akaike Information Criterion.
- BIC : double. the Bayesian Information Criterion.
- deviance : double. deviance.
- df.residual : integer residual degrees of freedom.

See Also

[plot.compare_numeric](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure[, c("platelets", "creatinine", "sodium")]

library(dplyr)
# Compare the all numerical variables
all_var <- compare_numeric(heartfailure2)

# Print compare_numeric class object
all_var

# Compare the correlation that case of joint the sodium variable
```

```

all_var %>%
  "$"(correlation) %>%
  filter(var1 == "sodium" | var2 == "sodium") %>%
  arrange(desc(abs(coef_corr)))

# Compare the correlation that case of abs(coef_corr) > 0.1
all_var %>%
  "$"(correlation) %>%
  filter(abs(coef_corr) > 0.1)

# Compare the linear model that case of joint the sodium variable
all_var %>%
  "$"(linear) %>%
  filter(var1 == "sodium" | var2 == "sodium") %>%
  arrange(desc(r.squared))

# Compare the two numerical variables
two_var <- compare_numeric(heartfailure2, sodium, creatinine)

# Print compare_numeric class objects
two_var

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Just correlation
summary(all_var, method = "correlation")

# Just correlation condition by r > 0.1
summary(all_var, method = "correlation", thres_corr = 0.1)

# linear model summaries condition by R^2 > 0.05
summary(all_var, thres_rs = 0.05)

# verbose is FALSE
summary(all_var, verbose = FALSE)

```

summary.imputation *Summarizing imputation information*

Description

print and summary method for "imputation" class.

Usage

```

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

```

Arguments

object an object of class "imputation", usually, a result of a call to `imputate_na()` or `imputate_outlier()`.

... further arguments passed to or from other methods.

Details

`summary.imputation()` tries to be smart about formatting two kinds of imputation.

See Also

[imputate_na](#), [imputate_outlier](#), [summary.imputation](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(NROW(heartfailure2)), 20), "platelets"] <- NA
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "smoking"] <- NA

# Impute missing values -----
# If the variable of interest is a numerical variables
platelets <- imputate_na(heartfailure2, platelets, death_event, method = "rpart")
platelets
summary(platelets)
plot(platelets)

# If the variable of interest is a categorical variables
smoking <- imputate_na(heartfailure2, smoking, death_event, method = "mice")
smoking
summary(smoking)

# plot(smoking)

# Impute outliers -----
# If the variable of interest is a numerical variable
platelets <- imputate_outlier(heartfailure2, platelets, method = "capping")
platelets
summary(platelets)

# plot(platelets)
```


Description

summary method for "optimal_bins". summary metrics to evaluate the performance of binomial classification model.

Usage

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

Arguments

object an object of class "optimal_bins", usually, a result of a call to binning_by().
... further arguments to be passed from or to other methods.

Details

print() to print only binning table information of "optimal_bins" objects. summary.performance_bin() includes general metrics and result of significance tests life follows.:

- Binning Table : Metrics by bins.
 - CntRec, CntPos, CntNeg, RatePos, RateNeg, Odds, WoE, IV, JSD, AUC.
- General Metrics.
 - Gini index.
 - Jeffrey's Information Value.
 - Jensen-Shannon Divergence.
 - Kolmogorov-Smirnov Statistics.
 - Herfindahl-Hirschman Index.
 - normalized Herfindahl-Hirschman Index.
 - Cramer's V Statistics.
- Table of Significance Tests.

Value

NULL.

See Also

[binning_by](#), [plot.optimal_bins](#)

Examples

```
library(dplyr)  
  
# Generate data for the example  
heartfailure2 <- heartfailure  
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA  
  
# optimal binning
```

```
bin <- binning_by(heartfailure2, "death_event", "creatinine")
bin

# summary optimal_bins class
summary(bin)

# performance table
attr(bin, "performance")

# visualize all information for optimal_bins class
# plot(bin)

# visualize WoE information for optimal_bins class
# plot(bin, type = "WoE")

# visualize all information without typographic
# plot(bin, typographic = FALSE)

# extract binned results
extract(bin) %>%
  head(20)
```

summary.overview

Summarizing overview information

Description

print and summary method for "overview" class.

Usage

```
## S3 method for class 'overview'
summary(object, html = FALSE, ...)
```

Arguments

object	an object of class "overview", usually, a result of a call to overview().
html	logical. whether to send summary results to html. The default is FALSE, which prints to the R console.
...	further arguments passed to or from other methods.

Details

summary.overview() tries to be smart about formatting 14 information of overview.

See Also

[overview](#), [plot.overview](#).

Examples

```
ov <- overview(jobchange)
ov

summary(ov)
```

summary.performance_bin

Summarizing Performance for Binned Variable

Description

summary method for "performance_bin". summary metrics to evaluate the performance of binomial classification model.

Usage

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

Arguments

object an object of class "performance_bin", usually, a result of a call to performance_bin().
... further arguments to be passed from or to other methods.

Details

print() to print only binning table information of "performance_bin" objects. summary.performance_bin() includes general metrics and result of significance tests life follows.:

- Binning Table : Metrics by bins.
 - CntRec, CntPos, CntNeg, RatePos, RateNeg, Odds, WoE, IV, JSD, AUC.
- General Metrics.
 - Gini index.
 - Jeffrey's Information Value.
 - Jensen-Shannon Divergence.
 - Kolmogorov-Smirnov Statistics.
 - Herfindahl-Hirschman Index.
 - normalized Herfindahl-Hirschman Index.
 - Cramer's V Statistics.
- Table of Significance Tests.

Value

NULL.

See Also

[performance_bin](#), [plot.performance_bin](#), [binning_by](#), [summary.optimal_bins](#).

Examples

```
# Generate data for the example
heartfailure2 <- heartfailure

set.seed(123)
heartfailure2[sample(seq(NROW(heartfailure2)), 5), "creatinine"] <- NA

# Change the target variable to 0(negative) and 1(positive).
heartfailure2$death_event_2 <- ifelse(heartfailure2$death_event %in% "Yes", 1, 0)

# Binnig from creatinine to platelets_bin.
breaks <- c(0, 1, 2, 10)
heartfailure2$creatinine_bin <- cut(heartfailure2$creatinine, breaks)

# Diagnose performance binned variable
perf <- performance_bin(heartfailure2$death_event_2, heartfailure2$creatinine_bin)
perf
summary(perf)

# plot(perf)

# Diagnose performance binned variable without NA
perf <- performance_bin(heartfailure2$death_event_2, heartfailure2$creatinine_bin, na.rm = TRUE)
perf
summary(perf)

# plot(perf)
```

summary.transform

Summarizing transformation information

Description

print and summary method for "transform" class.

Usage

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

Arguments

object an object of class "transform", usually, a result of a call to transform().
 ... further arguments passed to or from other methods.

Details

summary.transform compares the distribution of data before and after data transformation.

See Also

[transform](#), [plot.transform](#).

Examples

```
# Standardization -----
creatinine_minmax <- transform(heartfailure$creatinine, method = "minmax")
creatinine_minmax
summary(creatinine_minmax)

# plot(creatinine_minmax)

# Resolving Skewness -----
creatinine_log <- transform(heartfailure$creatinine, method = "log")
creatinine_log
summary(creatinine_log)

# plot(creatinine_log)

# plot(creatinine_log, typographic = FALSE)
```

summary.univar_category

Summarizing univar_category information

Description

print and summary method for "univar_category" class.

Usage

```
## S3 method for class 'univar_category'
summary(object, na.rm = TRUE, ...)

## S3 method for class 'univar_category'
print(x, ...)
```

Arguments

object	an object of class "univar_category", usually, a result of a call to univar_category().
na.rm	logical. Specifies whether to include NA when performing a chi-square test. The default is TRUE, where NA is removed and aggregated.
...	further arguments passed to or from other methods.
x	an object of class "univar_category", usually, a result of a call to univar_category().

Details

print.univar_category() displays only the information of variables included in univar_category. The "variables" attribute is not displayed.

Value

An object of the class as individual variables based list. The information to examine the relationship between categorical variables is as follows each components.

- variable : factor. The level of the variable. 'variable' is the name of the variable.
- statistic : numeric. the value the chi-squared test statistic.
- p.value : numeric. the p-value for the test.
- df : integer. the degrees of freedom of the chi-squared test.

See Also

[plot.univar_category](#).

Examples

```
library(dplyr)

# Calculates the all categorical variables
all_var <- univar_category(heartfailure)

# Print univar_category class object
all_var

# Calculates the only smoking variable
all_var %>%
  "["(names(all_var) %in% "smoking")

smoking <- univar_category(heartfailure, smoking)

# Print univar_category class object
smoking

# Filtering the case of smoking included NA
smoking %>%
  "[["(1) %>%
  filter(!is.na(smoking))
```

```
# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Summary by returned object
stat
```

```
summary.univar_numeric
```

Summarizing univar_numeric information

Description

print and summary method for "univar_numeric" class.

Usage

```
## S3 method for class 'univar_numeric'
summary(object, stand = c("robust", "minmax", "zscore"), ...)

## S3 method for class 'univar_numeric'
print(x, ...)
```

Arguments

object	an object of class "univar_numeric", usually, a result of a call to univar_numeric().
stand	character Describe how to standardize the original data. "robust" normalizes the raw data through transformation calculated by IQR and median. "minmax" normalizes the original data using minmax transformation. "zscore" standardizes the original data using z-Score transformation. The default is "robust".
...	further arguments passed to or from other methods.
x	an object of class "univar_numeric", usually, a result of a call to univar_numeric().

Details

print.univar_numeric() displays only the information of variables included in univar_numeric The "variables" attribute is not displayed.

Value

An object of the class as indivisual variabes based list. The statistics returned by summary.univar_numeric() are different from the statistics returned by univar_numeric(). univar_numeric() is the statistics for the original data, but summary. univar_numeric() is the statistics for the standardized data. A component named "statistics" is a tibble object with the following statistics.:

- variable : factor. The level of the variable. 'variable' is the name of the variable.

- n : number of observations excluding missing values
- na : number of missing values
- mean : arithmetic average
- sd : standard deviation
- se_mean : standard error mean. sd/\sqrt{n}
- IQR : interquartile range (Q3-Q1)
- skewness : skewness
- kurtosis : kurtosis
- median : median. 50% percentile

See Also

[plot.univar_numeric.](#)

Examples

```
# Calculates the all categorical variables
all_var <- univar_numeric(heartfailure)

# Print univar_numeric class object
all_var

# Calculates the platelets, sodium variable
univar_numeric(heartfailure, platelets, sodium)

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Summary by returned object
stat

# Statistics of numerical variables normalized by Min-Max method
summary(all_var, stand = "minmax")

# Statistics of numerical variables standardized by Z-score method
summary(all_var, stand = "zscore")
```

target_by

Target by one variables

Description

In the data analysis, a target_df class is created to identify the relationship between the target variable and the other variable.

Usage

```
target_by(.data, target, ...)  
  
## S3 method for class 'data.frame'  
target_by(.data, target, ...)
```

Arguments

.data	a data.frame or a tbl_df .
target	target variable.
...	arguments to be passed to methods.

Details

Data analysis proceeds with the purpose of predicting target variables that correspond to the facts of interest, or examining associations and relationships with other variables of interest. Therefore, it is a major challenge for EDA to examine the relationship between the target variable and its corresponding variable. Based on the derived relationships, analysts create scenarios for data analysis. `target_by()` inherits the [grouped_df](#) class and returns a `target_df` class containing information about the target variable and the variable.

See vignette("EDA") for an introduction to these concepts.

Value

an object of `target_df` class. Attributes of `target_df` class is as follows.

- `type_y` : the data type of target variable.

See Also

[relate](#).

Examples

```
# If the target variable is a categorical variable  
categ <- target_by(heartfailure, death_event)  
  
# If the variable of interest is a numerical variable  
cat_num <- relate(categ, sodium)  
cat_num  
summary(cat_num)  
  
# plot(cat_num)  
  
# If the variable of interest is a categorical variable  
cat_cat <- relate(categ, hblood_pressure)  
cat_cat  
summary(cat_cat)
```

```

# plot(cat_cat)

##-----
# If the target variable is a numerical variable
num <- target_by(heartfailure, creatinine)

# If the variable of interest is a numerical variable
num_num <- relate(num, sodium)
num_num
summary(num_num)

# plot(num_num)

# If the variable of interest is a categorical variable
num_cat <- relate(num, smoking)
num_cat
summary(num_cat)

# plot(num_cat)

# Not allow typographic
# plot(num_cat, typographic = FALSE)

```

target_by.tbl_dbi	<i>Target by one column in the DBMS</i>
-------------------	---

Description

In the data analysis, a `target_df` class is created to identify the relationship between the target column and the other column of the DBMS table through `tbl_dbi`

Usage

```

## S3 method for class 'tbl_dbi'
target_by(.data, target, in_database = FALSE, collect_size = Inf, ...)

```

Arguments

<code>.data</code>	a <code>tbl_dbi</code> .
<code>target</code>	target variable.
<code>in_database</code>	Specifies whether to perform in-database operations. If <code>TRUE</code> , most operations are performed in the DBMS. if <code>FALSE</code> , table data is taken in R and operated in-memory. Not yet supported <code>in_database = TRUE</code> .
<code>collect_size</code>	a integer. The number of data samples from the DBMS to R. Applies only if <code>in_database = FALSE</code> .
<code>...</code>	arguments to be passed to methods.

Details

Data analysis proceeds with the purpose of predicting target variables that correspond to the facts of interest, or examining associations and relationships with other variables of interest. Therefore, it is a major challenge for EDA to examine the relationship between the target variable and its corresponding variable. Based on the derived relationships, analysts create scenarios for data analysis. `target_by()` inherits the `grouped_df` class and returns a `target_df` class containing information about the target variable and the variable.

See `vignette("EDA")` for an introduction to these concepts.

Value

an object of `target_df` class. Attributes of `target_df` class is as follows.

- `type_y` : the data type of target variable.

See Also

[target_by.data.frame](#), [relate](#).

Examples

```
library(dplyr)

# connect DBMS
con_sqlite <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# copy heartfailure to the DBMS with a table named TB_HEARTFAILURE
copy_to(con_sqlite, heartfailure, name = "TB_HEARTFAILURE", overwrite = TRUE)

# If the target variable is a categorical variable
categ <- target_by(con_sqlite %>% tbl("TB_HEARTFAILURE") , death_event)

# If the variable of interest is a numerical variable
cat_num <- relate(categ, sodium)
cat_num
summary(cat_num)
plot(cat_num)

# If the variable of interest is a categorical column
cat_cat <- relate(categ, hblood_pressure)
cat_cat
summary(cat_cat)
plot(cat_cat)

##-----
# If the target variable is a categorical column,
# and In-memory mode and collect size is 200
num <- target_by(con_sqlite %>% tbl("TB_HEARTFAILURE"), death_event, collect_size = 250)

# If the variable of interest is a numerical column
num_num <- relate(num, creatinine)
```

```

num_num
summary(num_num)
plot(num_num)
plot(num_num, hex_thres = 200)

# If the variable of interest is a categorical column
num_cat <- relate(num, smoking)
num_cat
summary(num_cat)
plot(num_cat)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

```

transform

Data Transformations

Description

Performs variable transformation for standardization and resolving skewness of numerical variables.

Usage

```

transform(
  x,
  method = c("zscore", "minmax", "log", "log+1", "sqrt", "1/x", "x^2", "x^3",
    "Box-Cox", "Yeo-Johnson")
)

```

Arguments

`x` numeric vector for transformation.
`method` method of transformations.

Details

`transform()` creates an transform class. The ‘transform’ class includes original data, transformed data, and method of transformation.

See vignette("transformation") for an introduction to these concepts.

Value

An object of transform class. Attributes of transform class is as follows.

- `method` : method of transformation data.
 - Standardization
 - * "zscore" : z-score transformation. $(x - \mu) / \sigma$

- * "minmax" : minmax transformation. $(x - \min) / (\max - \min)$
- Resolving Skewness
 - * "log" : log transformation. $\log(x)$
 - * "log+1" : log transformation. $\log(x + 1)$. Used for values that contain 0.
 - * "sqrt" : square root transformation.
 - * "1/x" : $1 / x$ transformation
 - * "x^2" : x square transformation
 - * "x^3" : x^3 square transformation
 - * "Box-Cox" : Box-Box transformation
 - * "Yeo-Johnson" : Yeo-Johnson transformation

See Also

[summary.transform](#), [plot.transform](#).

Examples

```
# Standardization -----
creatinine_minmax <- transform(heartfailure$creatinine, method = "minmax")
creatinine_minmax
summary(creatinine_minmax)
plot(creatinine_minmax)

# Resolving Skewness -----
creatinine_log <- transform(heartfailure$creatinine, method = "log")
creatinine_log
summary(creatinine_log)

# plot(creatinine_log)

# plot(creatinine_log, typographic = FALSE)

# Using dplyr -----
library(dplyr)

heartfailure %>%
  mutate(creatinine_log = transform(creatinine, method = "log+1")) %>%
  lm(sodium ~ creatinine_log, data = .)
```

transformation_paged_report

Reporting the information of transformation

Description

The `eda_paged_report()` paged report the information for data transformation.

Usage

```
transformation_paged_report(
  .data,
  target = NULL,
  output_format = c("pdf", "html"),
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "Transformation Report",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  abstract_title = "Report Overview",
  abstract = NULL,
  title_color = "white",
  subtitle_color = "tomato1",
  cover_img = NULL,
  create_date = Sys.time(),
  logo_img = NULL,
  theme = c("orange", "blue"),
  sample_percent = 100,
  ...
)
```

Arguments

<code>.data</code>	a data.frame or a <code>tbl_df</code> .
<code>target</code>	character. target variable.
<code>output_format</code>	report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>rmarkdown::render()</code> and <code>pagedown::chrome_print()</code> . so, you needed Chrome web browser on computer. "html" create html file by <code>rmarkdown::render()</code> .
<code>output_file</code>	name of generated file. default is NULL.
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "Data Diagnosis Report".
<code>subtitle</code>	character. subtitle of report. default is name of data.
<code>author</code>	character. author of report. default is "dlookr".
<code>abstract_title</code>	character. abstract title of report. default is "Report Overview".
<code>abstract</code>	character. abstract of report.
<code>title_color</code>	character. color of title. default is "white".
<code>subtitle_color</code>	character. color of subtitle. default is "tomato1".
<code>cover_img</code>	character. name of cover image.
<code>create_date</code>	Date or POSIXct, character. The date on which the report is generated. The default value is the result of <code>Sys.time()</code> .
<code>logo_img</code>	character. name of logo image file on top right.

theme character. name of theme for report. support "orange" and "blue". default is "orange".

sample_percent numeric. Sample percent of data for performing Diagnosis. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.

... arguments to be passed to methods.

Details

Generate transformation reports automatically. You can choose to output to pdf and html files. This is useful for Binning a data frame with a large number of variables than data with a small number of variables.

Create an PDF through the Chrome DevTools Protocol. If you want to create PDF, Google Chrome or Microsoft Edge (or Chromium on Linux) must be installed prior to using this function. If not installed, you must use `output_format = "html"`.

Reported information

The transformation process will report the following information:

- Overview
 - Data Structures
 - Job Informations
- Imputation
 - Missing Values
 - Outliers
- Resolving Skewness
- Binning
- Optimal Binning

Examples

```
## Not run:
# create pdf file. file name is Transformation_Paged_Report.pdf
# transformation_paged_report(heartfailure, sample_percent = 80)

# create pdf file. file name is Transformation_heartfailure. and change cover image
# cover <- file.path(system.file(package = "dlookr"), "report", "cover2.jpg")
# transformation_paged_report(heartfailure, cover_img = cover, title_color = "gray",
#   output_file = "Transformation_heartfailure")

# create pdf file. file name is ./Transformation.pdf and not browse
# cover <- file.path(system.file(package = "dlookr"), "report", "cover1.jpg")
# transformation_paged_report(heartfailure, output_dir = ".", cover_img = cover,
#   flag_content_missing = FALSE, output_file = "Transformation.pdf", browse = FALSE)

# create pdf file. file name is Transformation_Paged_Report.html
# transformation_paged_report(heartfailure, target = "death_event", output_format = "html")
```

```
## End(Not run)
```

transformation_report *Reporting the information of transformation*

Description

The `transformation_report()` report the information of transform numerical variables for object inheriting from `data.frame`.

Usage

```
transformation_report(
  .data,
  target = NULL,
  output_format = c("pdf", "html"),
  output_file = NULL,
  output_dir = tempdir(),
  font_family = NULL,
  browse = TRUE
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>target</code>	target variable. If the target variable is not specified, the method of using the target variable information is not performed when the missing value is imputed. and Optimal binning is not performed if the target variable is not a binary class.
<code>output_format</code>	report output type. Choose either "pdf" and "html". "pdf" create pdf file by <code>knitr::knit()</code> . "html" create html file by <code>rmarkdown::render()</code> .
<code>output_file</code>	name of generated file. default is NULL.
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>font_family</code>	character. font family name for figure in pdf.
<code>browse</code>	logical. choose whether to output the report results to the browser.

Details

Generate transformation reports automatically. You can choose to output to pdf and html files. This is useful for Binning a data frame with a large number of variables than data with a small number of variables. For pdf output, Korean Gothic font must be installed in Korean operating system.

Reported information

The transformation process will report the following information:

- Imputation
 - Missing Values
 - * * Variable names including missing value
 - Outliers
 - * * Variable names including outliers
- Resolving Skewness
 - Skewed variables information
 - * * Variable names with an absolute value of skewness greater than or equal to 0.5
- Binning
 - Numerical Variables for Binning
 - Binning
 - * Numeric variable names
 - Optimal Binning
 - * Numeric variable names

See vignette("transformation") for an introduction to these concepts.

Examples

```
## Not run:
# reporting the Binning information -----
# create pdf file. file name is Transformation_Report.pdf & No target variable
transformation_report(heartfailure)

# create pdf file. file name is Transformation_Report.pdf
transformation_report(heartfailure, death_event)

# create pdf file. file name is Transformation_heartfailure.pdf
transformation_report(heartfailure, "death_event",
                      output_file = "Transformation_heartfailure.pdf")

# create html file. file name is Transformation_Report.html
transformation_report(heartfailure, "death_event", output_format = "html")

# create html file. file name is Transformation_heartfailure.html
transformation_report(heartfailure, death_event, output_format = "html",
                      output_file = "Transformation_heartfailure.html")

## End(Not run)
```

 transformation_web_report

Reporting the information of transformation with html

Description

The `transformation_web_report()` report the information of transform numerical variables for object inheriting from `data.frame`.

Usage

```
transformation_web_report(
  .data,
  target = NULL,
  output_file = NULL,
  output_dir = tempdir(),
  browse = TRUE,
  title = "Transformation",
  subtitle = deparse(substitute(.data)),
  author = "dlookr",
  title_color = "gray",
  logo_img = NULL,
  create_date = Sys.time(),
  theme = c("orange", "blue"),
  sample_percent = 100,
  ...
)
```

Arguments

<code>.data</code>	a <code>data.frame</code> or a <code>tbl_df</code> .
<code>target</code>	character. target variable.
<code>output_file</code>	name of generated file. default is <code>NULL</code> .
<code>output_dir</code>	name of directory to generate report file. default is <code>tempdir()</code> .
<code>browse</code>	logical. choose whether to output the report results to the browser.
<code>title</code>	character. title of report. default is "EDA Report".
<code>subtitle</code>	character. subtitle of report. default is name of data.
<code>author</code>	character. author of report. default is "dlookr".
<code>title_color</code>	character. color of title. default is "gray".
<code>logo_img</code>	character. name of logo image file on top left.
<code>create_date</code>	Date or <code>POSIXct</code> , character. The date on which the report is generated. The default value is the result of <code>Sys.time()</code> .
<code>theme</code>	character. name of theme for report. support "orange" and "blue". default is "orange".

sample_percent numeric. Sample percent of data for performing EDA. It has a value between (0, 100]. 100 means all data, and 5 means 5% of sample data. This is useful for data with a large number of observations.

... arguments to be passed to methods.

Details

Generate transformation reports automatically. This is useful for Binning a data frame with a large number of variables than data with a small number of variables.

Reported information

The transformation process will report the following information:

- Overview
 - Data Structures
 - Data Types
 - Job Informations
- Imputation
 - Missing Values
 - Outliers
- Resolving Skewness
- Binning
- Optimal Binning

Examples

```
## Not run:
# create html file. file name is Transformation_Report.html
# transformation_web_report(heartfailure)

# file name is Transformation.html. and change logo image
# logo <- file.path(system.file(package = "dlookr"), "report", "R_logo_html.svg")
# transformation_web_report(heartfailure, logo_img = logo, title_color = "black",
#   output_file = "Transformation.html")

# file name is ./Transformation.html, "blue" theme and not browse
# transformation_web_report(heartfailure, output_dir = ".", target = "death_event",
#   author = "Choonghyun Ryu", output_file = "Transformation.html",
#   theme = "blue", browse = FALSE)

## End(Not run)
```

univar_category	<i>Statistic of univariate categorical variables</i>
-----------------	--

Description

The `univar_category()` calculates statistic of categorical variables that is frequency table

Usage

```
univar_category(.data, ...)

## S3 method for class 'data.frame'
univar_category(.data, ...)
```

Arguments

<code>.data</code>	a data.frame or a tbl_df .
<code>...</code>	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

`univar_category()` calculates the frequency table of categorical variables. If a specific variable name is not specified, frequency tables for all categorical variables included in the data are calculated. The `univar_category` class returned by `univar_category()` is useful because it can draw chisquare tests and bar plots as well as frequency tables of individual variables. and return `univar_category` class that based list object.

Value

An object of the class as individual variables based list. The information to examine the relationship between categorical variables is as follows each components.

- `variable` : factor. The level of the variable. 'variable' is the name of the variable.
- `n` : integer. frequency by variable.
- `rate` : double. relative frequency.

Attributes of return object

Attributes of `compare_category` class is as follows.

- `variables` : character. List of variables selected for calculate frequency.

See Also

[summary.univar_category](#), [print.univar_category](#), [plot.univar_category](#).

Examples

```
library(dplyr)

# Calculates the all categorical variables
all_var <- univar_category(heartfailure)

# Print univar_category class object
all_var

# Calculates the only smoking variable
all_var %>%
  "["(names(all_var) %in% "smoking")

smoking <- univar_category(heartfailure, smoking)

# Print univar_category class object
smoking

# Filtering the case of smoking included NA
smoking %>%
  "["(1) %>%
  filter(!is.na(smoking))

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Summary by returned object
stat

# plot all variables
# plot(all_var)

# plot smoking
# plot(smoking)

# plot all variables by prompt
# plot(all_var, prompt = TRUE)
```

univar_numeric

Statistic of univariate numerical variables

Description

The `univar_numeric()` calculates statistic of numerical variables that is frequency table

Usage

```
univar_numeric(.data, ...)

## S3 method for class 'data.frame'
univar_numeric(.data, ...)
```

Arguments

.data	a data.frame or a tbl_df .
...	one or more unquoted expressions separated by commas. You can treat variable names like they are positions. Positive values select variables; negative values to drop variables. These arguments are automatically quoted and evaluated in a context where column names represent column positions. They support unquoting and splicing.

Details

`univar_numeric()` calculates the popular statistics of numerical variables. If a specific variable name is not specified, statistics for all categorical numerical included in the data are calculated. The statistics obtained by `univar_numeric()` are part of those obtained by `describe()`. Therefore, it is recommended to use `describe()` to simply calculate statistics. However, if you want to visualize the distribution of individual variables, you should use `univar_numeric()`.

Value

An object of the class as individual variables based list. A component named "statistics" is a tibble object with the following statistics.:

- variable : factor. The level of the variable. 'variable' is the name of the variable.
- n : number of observations excluding missing values
- na : number of missing values
- mean : arithmetic average
- sd : standard deviation
- se_mean : standrd error mean. sd/\sqrt{n}
- IQR : interquartile range (Q3-Q1)
- skewness : skewness
- kurtosis : kurtosis
- median : median. 50% percentile

Attributes of return object

Attributes of `compare_category` class is as follows.

- raw : a data.frame or a [tbl_df](#). Data containing variables to be compared. Save it for visualization with `plot.univar_numeric()`.
- variables : character. List of variables selected for calculate statistics.

See Also

[summary.univar_numeric](#), [print.univar_numeric](#), [plot.univar_numeric](#).

Examples

```
# Calculates the all categorical variables
all_var <- univar_numeric(heartfailure)

# Print univar_numeric class object
all_var

# Calculates the platelets, sodium variable
univar_numeric(heartfailure, platelets, sodium)

# Summary the all case : Return a invisible copy of an object.
stat <- summary(all_var)

# Summary by returned object
stat

# Statistics of numerical variables normalized by Min-Max method
summary(all_var, stand = "minmax")

# Statistics of numerical variables standardized by Z-score method
summary(all_var, stand = "zscore")

# one plot with all variables
# plot(all_var)

# one plot with all normalized variables by Min-Max method
# plot(all_var, stand = "minmax")

# one plot with all variables
# plot(all_var, stand = "none")

# one plot with all robust standardized variables
# plot(all_var, viz = "boxplot")

# one plot with all standardized variables by Z-score method
# plot(all_var, viz = "boxplot", stand = "zscore")

# individual boxplot by variables
# plot(all_var, indiv = TRUE, "boxplot")

# individual histogram by variables
# plot(all_var, indiv = TRUE, "hist")

# individual histogram by robust standardized variable
# plot(all_var, indiv = TRUE, "hist", stand = "robust")

# plot all variables by prompt
# plot(all_var, indiv = TRUE, "hist", prompt = TRUE)
```


Index

* datasets

- heartfailure, 90
- jobchange, 95

- binning, 5, 8, 80, 107, 153
- binning_by, 6, 7, 80, 106, 112, 114, 161, 164

- compare_category, 9, 108
- compare_numeric, 12, 110
- cor, 16, 18
- correlate, 14, 15
- correlate.data.frame, 18
- correlate.tbl_dbi, 16, 17

- describe, 20
- describe.data.frame, 24, 99
- describe.tbl_dbi, 21, 22, 101
- diagnose, 25
- diagnose.data.frame, 28, 30, 36, 40
- diagnose.tbl_dbi, 26, 27, 33, 38, 42
- diagnose_category, 29
- diagnose_category.data.frame, 26, 33, 36, 40
- diagnose_category.tbl_dbi, 28, 30, 32, 33, 38, 42
- diagnose_numeric, 35
- diagnose_numeric.data.frame, 21, 26, 30, 38, 40, 99
- diagnose_numeric.tbl_dbi, 24, 28, 33, 36, 37, 42, 101
- diagnose_outlier, 39
- diagnose_outlier.data.frame, 30, 36, 42, 141
- diagnose_outlier.tbl_dbi, 33, 38, 40, 41, 145
- diagnose_paged_report, 43
- diagnose_paged_report.data.frame, 49
- diagnose_paged_report.tbl_dbi, 45, 46
- diagnose_report, 49
- diagnose_report.data.frame, 53
- diagnose_report.tbl_dbi, 51
- diagnose_sparese, 54
- diagnose_web_report, 55
- diagnose_web_report.data.frame, 60
- diagnose_web_report.tbl_dbi, 57, 58
- dlookr (dlookr-package), 4
- dlookr-package, 4
- dlookr_blue_paged (dlookr_orange_paged), 61
- dlookr_orange_paged, 61
- dlookr_templ_html, 61

- eda_paged_report, 62
- eda_paged_report.data.frame, 67
- eda_paged_report.tbl_dbi, 64, 64
- eda_report, 68
- eda_report.data.frame, 72
- eda_report.tbl_dbi, 70
- eda_web_report, 73
- eda_web_report.data.frame, 78
- eda_web_report.tbl_dbi, 75, 76
- entropy, 79
- extract, 79

- find_class, 80, 85
- find_na, 82, 83, 84
- find_outliers, 82, 83, 84
- find_skewness, 84, 152

- get_class, 81, 85
- get_column_info, 86
- get_os, 87
- get_percentile, 87
- get_transform, 88
- group_by, 21, 23, 99, 101
- grouped_df, 16, 18, 20, 121, 123, 129, 147, 169, 171

- heartfailure, 90
- html_paged, 61

- import_liberation, [91](#)
- imputate_na, [82](#), [91](#), [94](#), [111](#), [160](#)
- imputate_outlier, [83](#), [92](#), [93](#), [111](#), [160](#)
- jobchange, [95](#)
- jsd, [96](#), [97](#)
- kld, [96](#), [97](#)
- kurtosis, [98](#), [152](#)
- normality, [98](#)
- normality.data.frame, [101](#)
- normality.tbl_dbi, [99](#), [100](#)
- overview, [103](#), [113](#), [162](#)
- performance_bin, [104](#), [114](#), [164](#)
- plot.bins, [6](#), [107](#)
- plot.compare_category, [10](#), [108](#), [155](#)
- plot.compare_numeric, [14](#), [109](#), [158](#)
- plot.imputation, [110](#)
- plot.optimal_bins, [8](#), [112](#), [114](#), [161](#)
- plot.overview, [104](#), [113](#), [162](#)
- plot.performance_bin, [106](#), [114](#), [164](#)
- plot.relate, [115](#), [148](#), [151](#)
- plot.transform, [117](#), [165](#), [173](#)
- plot.univar_category, [118](#), [166](#), [181](#)
- plot.univar_numeric, [119](#), [168](#), [183](#)
- plot_bar_category, [120](#)
- plot_box_numeric, [123](#)
- plot_correlate, [124](#)
- plot_correlate.data.frame, [127](#)
- plot_correlate.tbl_dbi, [125](#), [126](#)
- plot_hist_numeric, [128](#)
- plot_na_hclust, [130](#)
- plot_na_intersect, [131](#)
- plot_na_pareto, [133](#)
- plot_normality, [89](#), [134](#)
- plot_normality.data.frame, [99](#), [139](#)
- plot_normality.tbl_dbi, [101](#), [136](#), [137](#)
- plot_outlier, [140](#)
- plot_outlier.data.frame, [125](#), [136](#), [143](#), [145](#)
- plot_outlier.target_df, [142](#)
- plot_outlier.tbl_dbi, [127](#), [139](#), [141](#), [144](#)
- plot_qq_numeric, [146](#)
- print.bins, [6](#), [107](#)
- print.bins (summary.bins), [153](#)
- print.compare_category, [10](#), [108](#)
- print.compare_category
 (summary.compare_category), [154](#)
- print.compare_numeric, [14](#), [110](#)
- print.compare_numeric
 (summary.compare_numeric), [156](#)
- print.relate, [116](#), [148](#), [151](#)
- print.univar_category, [118](#), [181](#)
- print.univar_category
 (summary.univar_category), [165](#)
- print.univar_numeric, [120](#), [183](#)
- print.univar_numeric
 (summary.univar_numeric), [167](#)
- relate, [116](#), [149](#), [169](#), [171](#)
- shapiro.test, [99](#), [101](#)
- skewness, [98](#), [152](#)
- summary.bins, [6](#), [107](#), [153](#)
- summary.compare_category, [10](#), [108](#), [154](#)
- summary.compare_numeric, [14](#), [110](#), [156](#)
- summary.imputation, [111](#), [159](#), [160](#)
- summary.optimal_bins, [112](#), [160](#), [164](#)
- summary.overview, [104](#), [113](#), [162](#)
- summary.performance_bin, [106](#), [114](#), [163](#)
- summary.transform, [117](#), [164](#), [173](#)
- summary.univar_category, [118](#), [165](#), [181](#)
- summary.univar_numeric, [120](#), [167](#), [183](#)
- target_by, [142](#), [168](#)
- target_by.data.frame, [171](#)
- target_by.tbl_dbi, [170](#)
- tbl_df, [10](#), [12](#), [13](#), [15](#), [20](#), [25](#), [29](#), [35](#), [39](#), [44](#),
 [50](#), [54](#), [56](#), [62](#), [68](#), [74](#), [82–84](#), [91](#), [94](#),
 [99](#), [103](#), [121](#), [123](#), [125](#), [129](#), [135](#),
 [140](#), [147](#), [169](#), [174](#), [176](#), [178](#), [180](#),
 [182](#)
- transform, [117](#), [165](#), [172](#)
- transformation_paged_report, [173](#)
- transformation_report, [176](#)
- transformation_web_report, [178](#)
- univar_category, [118](#), [180](#)
- univar_numeric, [120](#), [181](#)