Package ‘dlookr’

October 13, 2022

Type Package

Title Tools for Data Diagnosis, Exploration, Transformation

Version 0.6.0

Date 2022-06-06

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), showtext (>= 0.9-4), sysfonts (>= 0.7.1), ggplot2 (>= 3.0.0), grid, gridExtra, hrbrthemes (>= 0.8.0), htmltools, kableExtra, knitr (>= 1.22), methods, mice, pagedown (>= 0.15), partykit, purrr, reactive, rlang, rmarkdown, shiny, tibble, tidy, tidyselect, utils

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

Author Choonghyun Ryu [aut, cre]

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

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

VignetteBuilder knitr
RoxygenNote 7.1.2
NeedsCompilation no
Repository CRAN
Date/Publication 2022-06-07 04:30:16 UTC

R topics documented:

dlookr-package ............................................ 4
binning ......................................................... 5
binning_by ..................................................... 7
binning_rgr ................................................... 9
compare_category ........................................... 11
compare_numeric ........................................... 14
correlate ...................................................... 17
cramer ......................................................... 20
describe ......................................................... 21
describe.tbl_dbi ............................................. 24
diagnose ......................................................... 27
diagnose.tbl_dbi ............................................. 29
diagnose_category ......................................... 31
diagnose_category.tbl_dbi ................................ 34
diagnose_numeric ........................................... 37
diagnose_numeric.tbl_dbi ................................ 39
diagnose_outlier ........................................... 41
diagnose_outlier.tbl_dbi .................................. 43
diagnose_paged_report .................................... 45
diagnose_paged_report.tbl_dbi ........................... 49
diagnose_report ............................................. 52
diagnose_report.tbl_dbi .................................... 54
diagnose_sparese ............................................ 56
diagnose_web_report ....................................... 58
diagnose_web_report.tbl_dbi .............................. 61
dlookr_orange_paged ....................................... 64
dlookr_templ_html ........................................... 64
eda_paged_report ........................................... 65
eda_paged_report.tbl_dbi .................................. 68
eda_report ..................................................... 71
eda_report.tbl_dbi .......................................... 73
eda_web_report .............................................. 77
eda_web_report.tbl_dbi .................................... 79
entropy ......................................................... 82
extract ......................................................... 83
find_class ...................................................... 84
find_na ........................................................ 85
find_outliers ................................................ 86
find_skewness ............................................... 87
get_class ....................................................... 88
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_column_info</td>
<td>89</td>
</tr>
<tr>
<td>get_os</td>
<td>90</td>
</tr>
<tr>
<td>get_percentile</td>
<td>91</td>
</tr>
<tr>
<td>get_transform</td>
<td>92</td>
</tr>
<tr>
<td>heartfailure</td>
<td>93</td>
</tr>
<tr>
<td>import_google_font</td>
<td>94</td>
</tr>
<tr>
<td>imputate_na</td>
<td>95</td>
</tr>
<tr>
<td>imputate_outlier</td>
<td>97</td>
</tr>
<tr>
<td>jobchange</td>
<td>99</td>
</tr>
<tr>
<td>jsd</td>
<td>100</td>
</tr>
<tr>
<td>kld</td>
<td>101</td>
</tr>
<tr>
<td>kurtosis</td>
<td>102</td>
</tr>
<tr>
<td>normality</td>
<td>102</td>
</tr>
<tr>
<td>normality.tbl_dbi</td>
<td>104</td>
</tr>
<tr>
<td>overview</td>
<td>107</td>
</tr>
<tr>
<td>performance_bin</td>
<td>108</td>
</tr>
<tr>
<td>plot.bins</td>
<td>111</td>
</tr>
<tr>
<td>plot.compare_category</td>
<td>112</td>
</tr>
<tr>
<td>plot.compare_numeric</td>
<td>114</td>
</tr>
<tr>
<td>plot.correlate</td>
<td>115</td>
</tr>
<tr>
<td>plot.imputation</td>
<td>117</td>
</tr>
<tr>
<td>plot.infogain_bins</td>
<td>118</td>
</tr>
<tr>
<td>plot.optimal_bins</td>
<td>119</td>
</tr>
<tr>
<td>plot.overview</td>
<td>121</td>
</tr>
<tr>
<td>plot.performance_bin</td>
<td>122</td>
</tr>
<tr>
<td>plot.pps</td>
<td>123</td>
</tr>
<tr>
<td>plot.relate</td>
<td>125</td>
</tr>
<tr>
<td>plot.transform</td>
<td>127</td>
</tr>
<tr>
<td>plot.univar_category</td>
<td>128</td>
</tr>
<tr>
<td>plot.univar_numeric</td>
<td>129</td>
</tr>
<tr>
<td>plot_bar_category</td>
<td>131</td>
</tr>
<tr>
<td>plot_box_numeric</td>
<td>134</td>
</tr>
<tr>
<td>plot_hist_numeric</td>
<td>136</td>
</tr>
<tr>
<td>plot_na_hclust</td>
<td>138</td>
</tr>
<tr>
<td>plot_na_intersect</td>
<td>139</td>
</tr>
<tr>
<td>plot_na_pareto</td>
<td>141</td>
</tr>
<tr>
<td>plot_normality</td>
<td>142</td>
</tr>
<tr>
<td>plot_normality.tbl_dbi</td>
<td>146</td>
</tr>
<tr>
<td>plot_outlier</td>
<td>148</td>
</tr>
<tr>
<td>plot_outlier.target_df</td>
<td>151</td>
</tr>
<tr>
<td>plot_outlier.tbl_dbi</td>
<td>152</td>
</tr>
<tr>
<td>plot_qq_numeric</td>
<td>155</td>
</tr>
<tr>
<td>pps</td>
<td>157</td>
</tr>
<tr>
<td>print.relate</td>
<td>160</td>
</tr>
<tr>
<td>relate</td>
<td>161</td>
</tr>
<tr>
<td>skewness</td>
<td>164</td>
</tr>
<tr>
<td>summary.bins</td>
<td>165</td>
</tr>
<tr>
<td>summary.compare_category</td>
<td>166</td>
</tr>
</tbody>
</table>
dlookr-package

dlookr: Tools for Data Diagnosis, Exploration, Transformation

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>
Binning the Numeric Data

Description

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

Usage

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

Arguments

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

Details

This function is useful when used with the mutate/transmute function of the dplyr package. See vignette("transformation") for an introduction to these concepts.

See Also

Useful links:
- [https://github.com/choonghyunryu/dlookr/](https://github.com/choonghyunryu/dlookr/)
- [https://choonghyunryu.github.io/dlookr/](https://choonghyunryu.github.io/dlookr/)
- Report bugs at [https://github.com/choonghyunryu/dlookr/issues](https://github.com/choonghyunryu/dlookr/issues)
Value

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

- class : "bins"
- type : binning type, "quantile", "equal", "pretty", "kmeans", "bclust".
- breaks : breaks for binning. 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.

See Also

binning_by, print.bins, summary.bins, plot.bins.

Examples

# Generate data for the example
heartfailure2 <- heartfailure
heartfailure2[sample(seq(1:dim(heartfailure2)[2]), 20), "platelets"] <- NA

# Binning the platelets 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
equalize(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()` function finds intervals for numerical variables using optical binning. Optimal binning categorizes a numeric characteristic into bins for further usage in scoring modeling.

**Usage**

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

**Arguments**

- `.data` a data frame.
- `y` 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.
- `x` character. name of continuous characteristic variable. At least 5 different values. and Inf is not allowed.
\begin{itemize}
  \item \texttt{p} \hspace{1cm} \text{numeric. percentage of records per bin. Default 5\% (0.05). This parameter only accepts values greater that 0.00 (0\%) and lower than 0.50 (50\%).}
  \item \texttt{ordered} \hspace{1cm} \text{logical. whether to build an ordered factor or not.}
  \item \texttt{labels} \hspace{1cm} \text{character. the label names to use for each of the bins.}
\end{itemize}

\textbf{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.

\textbf{Value}

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

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

\textbf{attributes of "optimal_bins" class}

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

\begin{itemize}
  \item \texttt{class : "optimal_bins".}
  \item \texttt{levels : character. factor or ordered factor levels}
  \item \texttt{type : character. binning method}
  \item \texttt{breaks : numeric. breaks for binning}
  \item \texttt{raw : numeric. before the binned the raw data}
  \item \texttt{ivtable : data.frame. information value table}
  \item \texttt{iv : numeric. information value}
  \item \texttt{target : integer. binary response variable}
\end{itemize}

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

\textbf{See Also}

\texttt{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)

# 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)

---

**binning_rgr**

*Binning by recursive information gain ratio maximization*

**Description**

The binning_rgr() finding intervals for numerical variable using recursive information gain ratio maximization.

**Usage**

`binning_rgr(.data, y, x, min_perc_bins = 0.1, max_n_bins = 5, ordered = TRUE)`
Arguments

.data          a data frame.
y             character. name of binary response variable. The variable must character of factor.
x             character. name of continuous characteristic variable. At least 5 different values. and Inf is not allowed.
min_perc_bins numeric. minimum percentange of rows for each split or segment (controls the sample size), 0.1 (or 10 percent) as default.
max_n_bins    integer. maximum number of bins or segments to split the input variable, 5 bins as default.
ordered       logical. whether to build an ordered factor or not.

Details

This function can be usefully used when developing a model that predicts y.

Value

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

• class : "infogain_bins".
• type : binning type, "infogain".
• 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.
• target : integer. binary response variable.
• x_var : character. name of x variable.
• y_var : character. name of y variable.

See Also

binning, binning_by, plot.infogain_bins.

Examples

library(dplyr)

# binning by recursive information gain ratio maximization using character
bin <- binning_rgr(heartfailure, "death_event", "creatinine")

# binning by recursive information gain ratio maximization using name
bin <- binning_rgr(heartfailure, death_event, creatinine)
bin

# summary optimal_bins class
### compare_category

**Compare categorical variables**

**Description**

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

**Usage**

```r
compare_category(.data, 
```

```r
## S3 method for class 'data.frame'
```

```r
compare_category(.data, 
```

**Arguments**

- `.data` is a data.frame or a `tbl_df`.
- `...` is 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 returns a `compare_category` class 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**

*Compare numerical variables*

**Description**

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

**Usage**

```r
compare_numeric(.data, ...)
```

```
## S3 method for class 'data.frame'
compare_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**

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.
compare_numeric

- 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

Description

The correlate() compute the correlation coefficient for numerical or categorical data.

Usage

correlate(.data, ...)

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

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

## S3 method for class 'tbl_dbi'
correlate(  
  .data,
  ...,  
  method = c("pearson", "kendall", "spearman", "cramer", "theil"),  
  in_database = FALSE,
  collect_size = Inf
)

Arguments

.data a data.frame or a grouped_df or 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, correlate() 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.
method       a character string indicating which correlation coefficient (or covariance) is to
             be computed. One of "pearson" (default), "kendall", or "spearman": can be
             abbreviated. For numerical variables, one of "pearson" (default), "kendall", or
             "spearman": can be used as an abbreviation. For categorical variables, "cramer" and
             "theil" can be used. "cramer" computes Cramer’s V statistic, "theil" com-
             putes Theil’s U statistic.

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.

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 for numerical variable. And support categorical variable
with theil’s U correlation coefficient and Cramer’s V correlation coefficient.

Correlation coefficient information

It returns data.frame with the following variables.: 

- var1 : names of numerical variable
- var2 : name of the corresponding numeric variable
- coef_corr : Correlation coefficient

When method = "cramer", data.frame with the following variables is returned.

- var1 : names of numerical variable
- var2 : name of the corresponding numeric variable
- chisq : the value the chi-squared test statistic
- df : the degrees of freedom of the approximate chi-squared distribution of the test statistic
- pval : the p-value for the test
- coef_corr : theil’s U correlation coefficient (Uncertainty Coefficient).

See Also

cor, summary.correlate, plot.correlate.

Examples

# Correlation coefficients of all numerical variables
tab_corr <- correlate(heartfailure)
tab_corr
mat_corr <- summary(tab_corr)
mat_corr

plot(tab_corr)

# Select the variable to compute
correlate(heartfailure, "creatinine", "sodium")

# Non-parametric correlation coefficient by kendall method
correlate(heartfailure, creatinine, method = "kendall")

# theil's U correlation coefficient (Uncertainty Coefficient)
tab_corr <- correlate(heartfailure, anaemia, hblood_pressure, method = "theil")
tab_corr

summary(tab_corr)
plot(tab_corr)

# Using dplyr::grouped_dt
library(dplyr)

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

# Using pipes -------------------------------
# Correlation coefficients of all numerical variables
heartfailure %>%
correlate()

# 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))

# Using pipes & dplyr -------------------------
# Compute the correlation coefficient of 'creatinine' 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)

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()

# Using pipes & dplyr ------------------------
# Compute the correlation coefficient of creatinine variable by 'hblood_pressure'
# and 'death_event' variables.
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  group_by(hblood_pressure, death_event) %>%
  correlate(creatinine)

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

---

cramer

<table>
<thead>
<tr>
<th>Cramer’s V statistic</th>
</tr>
</thead>
</table>

Description

Computes the Cramer’s V statistic and Chisquare p value between two categorical variables in data.frame.

Usage

cramer(dfm, x, y)
describe

Arguments

- `dfm` data.frame. probability distributions.
- `x` character. name of categorical or discrete variable.
- `y` character. name of another categorical or discrete variable.

Value

data.frame. It has the following variables:

- `var1`: character. first variable name.
- `var2`: character. second variable name.
- `df`: integer. degree of freedom.
- `pval`: numeric. p value of Chisquare test.

See Also

theil.

Examples

cramer(mtcars, "gear", "carb")

---

### describe

*Compute descriptive statistic*

**Description**

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

**Usage**

describe(.data, ...)

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

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

Arguments

.d.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
c("mean", "sd", "se_mean", "IQR", "skewness", "kurtosis", "quantiles")

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 (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).

all.combinations logical. When used with group_by(), this argument expresses all combinations
of group combinations. If the argument value is TRUE, cases that do not exist
as actual data are also included in the output.

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.

From version 0.5.5, the 'variable' column in the "descriptive statistic information" tibble object has
been changed to 'described_variables'. This is because there are cases where 'variable' is included
in the variable name of the data. There is probably no case where 'described_variables' is included
in the variable name of the data.

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")
heartfailure2 %>%
  filter(!(hblood_pressure %in% "Yes" | !death_event %in% "Yes") %>%
  group_by(hblood_pressure, death_event) %>%
  describe(all.combinations = TRUE)

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**

```r
## S3 method for class 'tbl_dbi'
describe(
  .data,
  ...,  # one or more unquoted expressions separated by commas. You can treat variable
  statistics = NULL,  # positive values select variables; negative values to drop variables.
  quantiles = NULL,  # If the first expression is negative, describe() will automatically start
  all.combinations = FALSE,  # with all variables. These arguments are automatically quoted
  in_database = FALSE,  # and evaluated in a context where column names represent column positions.
  collect_size = Inf  # They support unquoting and splicing.
)
```

**Arguments**

- `.data` a tbl_dbi.
- `...` character: the name of the descriptive statistic to calculate. The defaults is `c("mean", "sd", "se_mean", "IQR", "skewness", "kurtosis", "quantiles")`
**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 c(0, .01, .05, 0.1, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 0.95, 0.99, 1).

**all.combinations**
logical. When used with group_by(), this argument expresses all combinations of group combinations. If the argument value is TRUE, cases that do not exist as actual data are also included in the output.

**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.
See vignette("EDA") 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.

From version 0.5.5, the 'variable' column in the "descriptive statistic information" tibble object has been changed to 'described_variables'. This is because there are cases where 'variable' is included in the variable name of the data. There is probably no case where 'described_variables' is included in the variable name of the data.

**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")

# Using all.combinations = TRUE
con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
  filter(!(smoking %in% "Yes" | !death_event %in% "Yes") %>%
    group_by(smoking, death_event) %>%
    describe(all.combinations = TRUE) %>%
    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") %>%
The diagnose() produces information for diagnosing the quality of the variables of data.frame or tbl_df.

Usage

```r
diagnose(.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. If the first expression is negative, diagnose() 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**

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

# Select the variable to diagnose
diagnose(jobchange, gender, experience, training_hours)
# Select the variable to drop
diagnose(jobchange, -gender, -experience, -training_hours)
# Select the variable to correct
diagnose(jobchange, "gender", "experience", "training_hours")
# Select the positions
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)
# Negative values to drop variables
jobchange %>%
  diagnose(-8, -9, -10)
```
# Using pipes & dplyr -------------------------
# Diagnosis of missing variables
jobchange %>%
diagnose() %>%
filter(missing_count > 0)

---

diagnose.tbl_dbi

Diagnose data quality of variables in the DBMS

Description

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

Usage

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

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, `diagnose()` 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.
- `in_database` a logical. 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.
- `collect_size` a integer. The number of data samples from the DBMS to R. Applies only if `in_database = FALSE`.

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. unique_count / 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  

Diagnose data quality of categorical variables

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

.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, diagnose_category() 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.
**Top**
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**

```r
# 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)

# Negative values to drop 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")
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

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

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, diagnose_category() 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.
- `top` : an integer. Specifies the upper top rank to extract. Default is 10.
- `type` : 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 top 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".
- `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.
- `collect_size` : a integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.
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)

# Negative values to drop 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(alphabet = 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
# default of type argument is "rank"
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_numeric

diagnose_category(top = 2, type = "n")

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

**diagnose_numeric**  **Diagnose data quality of numerical variables**

**Description**

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

**Usage**

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

```r
# S3 method for class 'data.frame'
diagnose_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. If the first expression is negative, diagnose_numeric() 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 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("diagonosis") 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)
# Negative values to drop variables
heartfailure %>%
  diagnose_numeric(-1, -5)

# Using pipes & dplyr -------------------------
# List of variables containing outliers
heartfailure %>%
  diagnose_numeric() %>%
diagnose_numeric.tbl_dbi

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

- **.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, diagnose_numeric() 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.

- **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**
  - an integer. The number of data samples from the DBMS to R. Applies only if in_database = FALSE.

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("diagonosis") 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)

# Negative values to drop 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

Diagnose outlier of numerical variables

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

.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, diagnose_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.
Details

The scope of the diagnosis is to provide 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("diagnosis") 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 %>%
```r
# Positions values select variables
heartfailure %>%
diagnose_outlier(5)

# Negative values to drop 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

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

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, `diagnose_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.
- `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.
Details

The scope of the diagnosis is to provide 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_db, diagnose_category.tbl_db, diagnose_numeric.tbl_db.

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)

# Negative values to drop 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,
base_family = NULL,
...}

Arguments

.data a data.frame or a tbl_df.
... arguments to be passed to methods.
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 subtitle. 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.

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.

base_family

character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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

The base family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.

See Also

diagnose_paged_report.tbl_dbi.

Examples

```r
if (FALSE) {
  # 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")
}
```
**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**

```r
## 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.
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 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. 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

```r
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)

---

**diagnose_report**

*Reporting the information of data diagnosis*

**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**

- `.data` a data.frame or a `tbl_df`.
- `output_file` name of generated file. Default is NULL.
- `output_dir` name of directory to generate report file. Default is `tempdir()`.
- `...` arguments to be passed to methods.
- `font_family` character. Font family name for figure in pdf.
- `browse` 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**

```r
if (FALSE) {
  # 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")
}
```

---

**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**

```r
## 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

```r
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**

Diagnosis of level combinations of categorical variables
**Description**

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

**Usage**

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

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

**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, `diagnose_sparese()` 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.
- `type` 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.
- `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.
- `limit` 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 parse 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,
base_family = NULL,
...)

Arguments

.data
   a data.frame or a tbl_df.
...
   arguments to be passed to methods.
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".
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 file on top left.
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.
is_tbl_dbi
   logical. whether .data is a tbl_dbi object.

base_family
   character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
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

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

See Also
diagnose_web_report.tbl_dbi.

Examples

```r
if (FALSE) {
  # 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)

---
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

```r
# 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

.data a tbl_dbi.
.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".
.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

```r
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


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,
  base_family = NULL,
  ...
)
Arguments

.data a data.frame or a tbl_df.

... arguments to be passed to methods.

target character. target variable.

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 "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.

base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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

The base family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.

See Also

`eda_paged_report.tbl_dbi`.

Examples

```r
if (FALSE) {
  # 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")
}

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**

```r
## 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

- `.data` a `tbl_dbi`.
- `target` character. target variable.
- `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 "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 a 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

```r
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
```
eda_report

```r
con_sqlite %>%
tbl("TB_HEARTFAILURE") %>%
edapr darf_report(target = "death_event")

# create pdf file. file name is EDA.pdf, and collect size is 250
con_sqlite %>%
tbl("TB_HEARTFAILURE") %>%
edapr darf_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

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

```r
## 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.data</code></td>
<td>a data.frame or a <code>tbl_df</code>.</td>
</tr>
<tr>
<td><code>...</code></td>
<td>arguments to be passed to methods.</td>
</tr>
<tr>
<td><code>target</code></td>
<td>target variable.</td>
</tr>
<tr>
<td><code>output_format</code></td>
<td>character. report output type. Choose either &quot;pdf&quot; and &quot;html&quot;. &quot;pdf&quot; create pdf file by knitr::knit(). &quot;html&quot; create html file by rmarkdown::render().</td>
</tr>
<tr>
<td><code>output_file</code></td>
<td>character. name of generated file. default is NULL.</td>
</tr>
</tbody>
</table>

---

### eda_report

The `eda_report()` function reports the information of exploratory data analysis for objects inheriting from `data.frame`.

#### Usage

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

#### Arguments

- `.data`: A `data.frame` or a `tbl_df`.
- `...`: Additional arguments to be passed to methods.
- `target`: Target variable.
- `output_file`: Name of the generated file. Default is `NULL`.
output_dir character. name of directory to generate report file. default is tempdir().
font_family character. font family name for figure in pdf.
browse 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)
# Reporting the information of EDA for table of the DBMS

**Description**

The `eda_report()` function reports the information of Exploratory data analysis for objects inheriting from the DBMS table through `tbl_dbi`. It allows for generating reports in various formats such as PDF and HTML. Here are some examples of how it can be used:

- To create a PDF file named `EDA_heartfailure.pdf`:
  ```r
  eda_report(heartfailure, "death_event", output_file = "EDA_heartfailure.pdf")
  ```

- To create a PDF file named `EDA_heartfailure.pdf` without browsing:
  ```r
  eda_report(heartfailure, "death_event", output_dir = ".",
             output_file = "EDA_heartfailure.pdf", browse = FALSE)
  ```

- To create an HTML file named `EDA_heartfailure.html`:
  ```r
  eda_report(heartfailure, "death_event", output_format = "html")
  ```

- To create an HTML file named `EDA_heartfailure.html` with a specific target variable:
  ```r
  eda_report(heartfailure, sodium, output_format = "html", output_file = "EDA_heartfailure.html")
  ```

- To create a PDF file named `EDA2.pdf` for a specific target variable:
  ```r
  eda_report(heartfailure, "sodium", output_file = "EDA2.pdf")
  ```

- To create an HTML file named `EDA_Report.html` for a specific target variable:
  ```r
  eda_report(heartfailure, "sodium", output_format = "html")
  ```

- To create an HTML file named `EDA2.html` for a specific target variable:
  ```r
  eda_report(heartfailure, sodium, output_format = "html", output_file = "EDA2.html")
  ```

These functions provide a flexible way to generate comprehensive exploratory data analysis reports for database table objects.
Usage

```r
## 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_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

```r
DBI::dbDisconnect(con_sqlite)
```

---

**Description**

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

**Usage**

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

```r
## 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,
  base_family = NULL,
  ...)
```

**Arguments**

- `.data` a `data.frame` or a `tbl_df`.
- `...` arguments to be passed to methods.
- `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".
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 file on top left.
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.
is_tbl_dbi logical. whether .data is a tbl_dbi object.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

See Also

eda_web_report.tbl_dbi.

Examples

if (FALSE) {
  # 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)
}

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

```r
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
```
entropy

Calculate the entropy

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)
**Description**

The `extract()` function extracts a binned variable from "bins" or "optimal_bins" class objects.

**Usage**

```r
extract(x)
```

## S3 method for class 'bins'

```r
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**

```r
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

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

Finding variables including missing values

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

imputate_na, find_outliers.
find_outliers

Finding variables including outliers

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.
**find skewness**

Examples

```r
## 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**

*Finding skewed variables*

Description

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

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>a data.frame or a tbl_df.</td>
</tr>
<tr>
<td>index</td>
<td>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.</td>
</tr>
<tr>
<td>value</td>
<td>logical. If TRUE, returns the skewness value in the individual variable.</td>
</tr>
<tr>
<td>thres</td>
<td>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.</td>
</tr>
</tbody>
</table>

Value

Information on variables including skewness.

See Also

`find_na`, `find_outliers`. 
**get_class**

**Extracting a class of variables**

**Description**

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

**Usage**

```r
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

```r
## 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)
```

### Description

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

### Usage

```r
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  

Finding Users Machine’s OS

Description

Get the operating system that users machines.

Usage

get_os()

Value

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

Examples

get_os()
Description

Find the percentile of the value specified in numeric vector.

Usage

```r
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

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

quantile(carat)

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

## End(Not run)
```
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. \( \log(x + 1 - \text{min}(x)) \). 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

Value

numeric. transformed numeric vector.

See Also

plot_normality.
Examples

```r
## 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

```r
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(creatine 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


import_google_font

Import Google Fonts

Description

Import google font to be used when drawing charts.

Usage

import_google_font(family)

Arguments

family character. font family name

Details

When attaching the dlookr package, use "Roboto Condensed" and "Noto Sans Korean" among Google fonts. And also loads "Liberation Sans Narrow" and "NanumSquare" included in the package for offline environment.

If you want to use anything other than the 4 fonts that are loaded with the dlookr package, load the desired Google fonts with import_google_font().

dlookr recommends the following google fonts, both sans and condensed: "IBM Plex Sans Condensed", "Encode Sans Condensed", "Barlow Condensed", "Saira Condensed", "Titillium Web", "Oswald", "PT Sans Narrow"

Korean fonts: "Nanum Gothic", "Gothic A1"
Description

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

Usage

imputate_na(.data, xvar, yvar, method, seed, print_flag, no_attrs)

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_attrs` 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))


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

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

**Arguments**

- `.data`: a data.frame or a `tbl_df`.
- `xvar`: variable name to replace missing value.
- `method`: method of missing values imputation.
- `no_attrs`: logical. If TRUE, return numerical variable or categorical variable. else If FALSE, imputation class.
- `cap_ntiles`: numeric. Only used when method is "capping". Specifies the value of percentiles replaced by the values of lower outliers and upper outliers. The default is `c(0.05, 0.95)`.

**Details**

`imputate_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_attrs is FALSE then return imputation class, else no_attrs 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 lower outliers with 5 percentile.
      - You can change this criterion with the cap_ntiles argument.
- outlier_pos : position of outliers in predictor.
- outliers : outliers. outliers corresponding to outlier_pos.
- type : "outliers". type of imputation.

See Also

impute_outlier.

Examples

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

# Replace the outliers of the sodium variable with capping.
imputate_outlier(heartfailure, sodium, method = "capping")
imputate_outlier(heartfailure, sodium, method = "capping",
cap_ntiles = c(0.1, 0.9))

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

# The mean before and after the imputation of the sodium variable
heartfailure %>%
  mutate(sodium_imp = imputate_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 <- imputate_outlier(heartfailure, sodium)
sodium
summary(sodium)

plot(sodium)
Description

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

Usage

```r
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` development index of the city (scaled).
- `gender` gender of candidate.
- `relevant_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_chng` 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
jensen-shannon divergence

Description

Computes the Jensen-Shannon divergence between two probability distributions.

Usage

\[
\text{jse}(p, q, \text{base} = \text{c("log", "log2", "log10")}, \text{margin} = \text{FALSE})
\]

Arguments

- \(p\): numeric. probability distributions.
- \(q\): numeric. probability distributions.
- \(\text{base}\): character. log bases. "log", "log2", "log10". default is "log"
- \(\text{margin}\): 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 \(p\).
\[
\text{event} \leftarrow \text{c}(115, 76, 61, 39, 55, 10, 1)
\]
\[
\text{no_event} \leftarrow \text{c}(3, 3, 7, 10, 28, 44, 117)
\]
\[
\text{p} \leftarrow \text{event} / \text{sum(event)}
\]
\[
\text{q} \leftarrow \text{no_event} / \text{sum(no_event)}
\]
\[
\text{jse}(\text{p}, \text{q})
\]
\[
\text{jse}(\text{p}, \text{q}, \text{base} = \text{"log2"})
\]
\[
\text{jse}(\text{p}, \text{q}, \text{margin} = \text{TRUE})
\]
**kld**

**Kullback-Leibler Divergence**

**Description**

Computes the Kullback-Leibler divergence between two probability distributions.

**Usage**

```r
kld(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. Kullback-Leibler divergence of probability distributions `p` and `q`.

**See Also**

`jsd`.

**Examples**

```r
# 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  \textit{Kurtosis of the data}

**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**
umeric. calculated kurtosis

**See Also**
skewness.

**Examples**
set.seed(123)
kurtosis(rnorm(100))

normality \textit{Performs the Shapiro-Wilk test of normality}

**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)
normality

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, normality() 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.
sample the number of samples to perform the test.

See vignette("EDA") 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 Roystion(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)
```r
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

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

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, `normality()` 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.
- `sample` the number of samples to perform the test.
- `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`. See vignette("EDA") 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 Roystion(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`. 
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)) %>%
```r
library(dplyr)
library(GGally)

# Apply the data manipulation functions
overview <- function(.data) {
  .data <- data.frame(.data)
  .data <- tibble::rownames_to_column(.data, name = "division")
  .data <- .data %>%
    group_by(smoking, death_event) %>%
    normality(log_sodium) %>%
    filter(p_value > 0.01)
  # Disconnect DBMS
  DBI::dbDisconnect(con_sqlite)

  return(.data)
}

# Description of overview
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

```r
ov <- overview(jobchange)
ov
summary(ov)
plot(ov)
```

---

**performance_bin**

*Diagnose Performance Binned Variable*

**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. Value 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. 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.
performance_bin

- chisq_test: data.frame. Table of significance tests. Name is as follows.
  - Bin A: character. First bins.
  - Bin B: character. Second bins.
  - 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)
summary(perf)

plot(perf)

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

Arguments

x

an object of class "bins", usually, a result of a call to binning().

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.

base_family

character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

...

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

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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 platelets 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)
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,
base_family = NULL,
...
)

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.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

... 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.

• 0: always parallel to the axis [default],
• 1: always horizontal to the axis,
Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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, base_family = NULL, ...)

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 hrbrthemes package.

base_family  
character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

...  
arguments to be passed to methods, such as graphical parameters (see par). However, it does not support.

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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.correlate**

Visualize Information for an "correlate" Object

**Description**

Visualize by attribute of ‘correlate’ class. The plot of correlation matrix is a tile plot.

**Usage**

## S3 method for class 'correlate'
plot(x, typographic = TRUE, base_family = NULL, ...)

**Arguments**

- **x**: an object of class "correlate", usually, a result of a call to correlate().
- **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.
- **base_family**: character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- **...**: arguments to be passed to methods, such as graphical parameters (see par).

**Details**

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().
See Also

correlate, summary.correlate.

Examples

library(dplyr)

# correlate type is generic

# Draw a correlation matrix plot by category of Species.

# S3 method for correlate class

# S3 method for correlate class by 'tbl_dbi'

# connect DBMS

# Disconnect DBMS
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

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

Arguments

- `x`: an object of class "imputation", usually, a result of a call to `imputate_na()` or `imputate_outlier()`.
- `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.
- `base_family`: character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- `...`: 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.

Details

The `base_family` is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.

See Also

`imputate_na`, `imputate_outlier`, `summary.imputation`.

Examples

```r
# 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.infogain_bins**

**Visualize Distribution for an "infogain_bins" Object**

**Description**

It generates plots for understand distribution and distribution by target variable using infogain_bins.

**Usage**

```r
## S3 method for class 'infogain_bins'
plot(x, type = c("bar", "cross"), typographic = TRUE, base_family = NULL, ...)
```

**Arguments**

- `x` an object of class "infogain_bins", usually, a result of a call to binning_rgr().
- `type` character. options for visualization. Distribution("bar"), Relative Frequency by target ("cross").
- `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.
- `base_family` character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- `...` further arguments to be passed from or to other methods.
plot.optimal_bins

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

See Also

binning_rgr, summary.bins

Examples

# binning by recursive information gain ratio maximization using character
bin <- binning_rgr(heartfailure, "death_event", "creatinine")

# binning by recursive information gain ratio maximization using name
bin <- binning_rgr(heartfailure, death_event, creatinine)
bin

# 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 = "cross")

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

plot.optimal_bins

Visualization 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"),

**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.

- **base_family**
  - character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

- **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.

**Details**

The `base_family` is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.

**See Also**

- `binning_by`, `summary.optimal_bins`

**Examples**

```r
# 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

Visualize Information for an "overview" Object

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"),
  typographic = TRUE,
  base_family = NULL,
  ...
)

Arguments

x : an object of class "overview", usually, a result of a call to overview().
order_type : character. method of order of bars(variables).
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.
base_family : character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
...

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().
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, base_family = NULL, ...)
```

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.
- `base_family` character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- `...` further arguments to be passed from or to other methods.

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

See Also

`overview`, `summary.overview`.

Examples

```
ov <- overview(jobchange)

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")
```
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)

# Binning 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)

plot(perf, typographic = FALSE)

plot(perf, typographic = FALSE)

plot(perf)

plot(perf, typographic = FALSE)

plot(perf, typographic = FALSE)

plot(perf, typographic = FALSE)
Arguments

- **x**: an object of class "pps", usually, a result of a call to pps().
- **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.
- **base_family**: character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- **...**: arguments to be passed to methods, such as graphical parameters (see par).

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

See Also

pps, summary.pps.

Examples

```r
library(dplyr)

# pps type is generic
pps_generic <- pps(iris)
pps_generic

# visualize pps class
plot(pps_generic)

# pps type is target_by
pps_cat <- iris %>%
  target_by(Species) %>%
  pps()

# plot pps class
plot(pps_cat)

# If the target variable is a numerical variable
pps_num <- iris %>%
  target_by(Petal.Length) %>%
  pps()

# plot pps class
```

### plot.relate

#### Description

Visualize four kinds of plots by attribute of relate class.

#### Usage

```r
## S3 method for class 'relate'
plot(
  x,
  model = FALSE,
  hex_thres = 1000,
  pal = c("#FFFFB2", "#FED976", "#FEB24C", "#FD8D3C", "#FC4E2A", "#E31A1C", "#B10026"),
  typographic = TRUE,
  base_family = NULL,
  ...
)
```

#### 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.
- **base_family**: character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- **...**: 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.
Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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, base_family = NULL, ...)

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 hrbrthemes package.
base_family
character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
...
arguments to be passed to methods, such as graphical parameters (see par).

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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**

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

**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.
- `base_family`: character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- `...`: arguments to be passed to methods, such as graphical parameters (see par). However, it does not support all parameters.

**Details**

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().
plot.univar_numeric

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**

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.
- **base_family**: character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
- **...**: arguments to be passed to methods, such as graphical parameters (see par). However, it does not support.

Details

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

See Also

univar_numeric, print.univar_numeric, summary.univar_numeric.

Examples

```r
# 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.
```
*plot_bar_category*

```r
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)
```

---

**plot_bar_category**  
*Plot bar chart of categorical variables*

**Description**

The `plot_bar_category()` function visualizes the distribution of categorical data by level or relationship to specific numerical data by level.

**Usage**

```r
plot_bar_category(.data, ...)

## S3 method for class 'data.frame'
plot_bar_category(
  .data,
  ...
)
```
```r
plot_bar_category(
  top = 10,
  add_character = TRUE,
  title = "Frequency by levels of category",
  each = FALSE,
  typographic = TRUE,
  base_family = NULL
)
```

```r
## 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,
  base_family = NULL
)
```

**Arguments**

- `top` an integer. Specifies the upper top rank to extract. Default is 10.
- `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.
- `title` 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.
- `base_family` character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

**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.
The base family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.

Examples

```r
# 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 groupd_df --------------------------
heartfailure2 %>%
  group_by(death_event) %>%
  plot_bar_category(top = 5)

heartfailure2 %>%
  group_by(death_event) %>%
  plot_bar_category(each = TRUE, top = 5)
```
Description

The `plot_box_numeric()` function visualizes the box plot of numeric data or relationship to specific categorical data.

Usage

`plot_box_numeric(.data, ...)`

```r
## S3 method for class 'data.frame'
plot_box_numeric(
  .data,
  ..., 
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE,
  base_family = NULL
)
```

```r
## S3 method for class 'grouped_df'
plot_box_numeric(
  .data,
  ..., 
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE,
  base_family = NULL
)
```

Arguments

- `.data` : 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, `plot_box_numeric()` 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.
- `title` : 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.
plot_box_numeric

Arguments

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.

base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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.

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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 group_dif ------------------------------
heartfailure %>%
  group_by(smoking) %>%
  plot_box_numeric()

heartfailure %>%
  group_by(smoking) %>%
  plot_box_numeric(each = TRUE)
plot_hist_numeric  

Plot histogram of numerical variables

Description

The `plot_hist_numeric()` function visualizes the histogram of numeric data or relationship to specific categorical data.

Usage

```r
plot_hist_numeric(.data, ...)
```

## S3 method for class 'data.frame'

```r
plot_hist_numeric(
  .data,
  ..., 
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE,
  base_family = NULL
)
```

## S3 method for class 'grouped_df'

```r
plot_hist_numeric(
  .data,
  ..., 
  title = "Distribution by numerical variables",
  each = FALSE,
  typographic = TRUE,
  base_family = NULL
)
```

Arguments

- `data`  
  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, `plot_hist_numeric()` 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.

- `title`  
  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.
plot_hist_numeric

**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.

**base_family** character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

**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 \((2 * \text{IQR}(x) / \text{length}(x)^{1/3})\)

The **base_family** is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

**Examples**

```r
# 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

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,
  base_family = NULL
)

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.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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.

The base family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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")

# Non 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**

```r
plot_na_intersect(
  x,  
  only_na = TRUE,  
  n_intersacts = NULL,  
  n_vars = NULL,  
  main = NULL,  
  typographic = TRUE,  
  base_family = NULL
)
```

**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_intersacts`: 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.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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.

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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)

# Non typographic elements
plot_na_intersect(jobchange2, typographic = FALSE)
**Description**

Visualize pareto chart for variables with missing value.

**Usage**

```r
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,
  base_family = NULL
)
```

**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.
- `base_family` character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

**Details**

The `base_family` is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.
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)

# Non 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.
plot_normality

Usage

plot_normality(.data, ...)

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

## S3 method for class 'grouped_df'
plot_normality(
  .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. If the first expression is negative, plot_normality() 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.

left character. Specifies the data transformation method to draw the histogram in the lower left corner. The default is "log".

right character. Specifies the data transformation method to draw the histogram in the lower right corner. The default is "sqrt".

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.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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")
}

# Non 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")
}
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

```r
## 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, 
  base_family = NULL
)
```

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_normality()` 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.
- `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.
- `left` character. Specifies the data transformation method to draw the histogram in the lower left corner. The default is "log".
- `right` character. Specifies the data transformation method to draw the histogram in the lower right corner. The default is "sqrt".
- `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.

base_family

character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

Details

The scope of the visualization is to provide 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-Cox 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

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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

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,
  base_family = NULL
)

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, 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 "steelblue".

typographic logical. Whether to apply focuses on typographic elements to ggplot2 visualization.

base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details) 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.

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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)

# Negative values to drop variables
heartfailure %>%
  plot_outlier(-1, -5)

# Using pipes & dplyr -------------------------
# Visualization of numerical variables with a ratio of
# outliers greater than 5%

```r
heartfailure %>%
  plot_outlier(heartfailure %>%
    diagnose_outlier() %>%
    filter(outliers_ratio > 5) %>%
    select(variables) %>%
    pull())
```

## Description

The `plot_outlier()` function visualizes 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, base_family = NULL)
```

## Arguments

- `.data`:
  - a `target_df` reference
  - `target_by`.

- `...`:
  - one or more unquoted expressions separated by commas.
  - You can treat variable names as 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.

- `typographic`:
  - logical. Whether to apply focuses on typographic elements to ggplot2 visualization.

- `base_family`:
  - character. The name of the base font family to use for the visualization. If not specified, the font defined in `dlookr` is applied.
  - 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 to provide 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.

The `base_family` is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with `import_google_font()`.
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(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

```r
## S3 method for class 'tbl_dbi'
plot_outlier(
  .data,
  ..., 
  col = "steelblue",
  in_database = FALSE,
  collect_size = Inf,
  typographic = TRUE,
  base_family = NULL
)
```

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.
- `base_family` character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

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.
The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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)

# Negative values to drop 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**

*Plot Q-Q plot of numerical variables*

### Description

The `plot_qq_numeric()` function visualizes the Q-Q plot of numeric data or relationship to specific categorical data.

### Usage

```r
plot_qq_numeric(.data, ...)

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

## S3 method for class 'grouped_df'
plot_qq_numeric(
  .data,
  ...
)
```

## Arguments

- `.data`: Data frame to be plotted.
- `...`: Additional arguments to be passed to `ggplot2`.
- `col_point`: Color for points (default: "steelblue").
- `col_line`: Color for line (default: "black").
- `title`: Title of the plot (default: "Q-Q plot by numerical variables").
- `each`: Whether to display each variable separately (default: FALSE).
- `typographic`: Whether to use typographic font (default: TRUE).
- `base_family`: Base family for font (default: NULL).

### Examples

```r
# Example usage
plot_qq_numeric(.data)
```
title = "Q-Q plot by numerical variables",
   each = FALSE,
   typographic = TRUE,
   base_family = NULL
)

Arguments

.data 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, plot_qq_numeric() 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_point character. a color of points in Q-Q plot.
col_line character. a color of line in Q-Q plot.
title 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.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

Details

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.

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

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 group_df ------------------------------
heartfailure %>%
  group_by(smoking) %>%
  plot_qq_numeric()

heartfailure %>%
  group_by(smoking) %>%
  plot_qq_numeric(each = TRUE)

ppps

## Compute Predictive Power Score

### Description
The pps() compute PPS(Predictive Power Score) for exploratory data analysis.

### Usage
pps(.data, ...)

## S3 method for class 'data.frame'
pps(.data, ..., cv_folds = 5, do_parallel = FALSE, n_cores = -1)

## S3 method for class 'target_df'
pps(.data, ..., cv_folds = 5, do_parallel = FALSE, n_cores = -1)

### Arguments
- `.data` a target_df or data.frame.
- `...` 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.
- `cv_folds` integer. number of cross-validation folds.
- `do_parallel` logical. whether to perform score calls in parallel.
- `n_cores` integer. number of cores to use, defaults to maximum cores - 1.
Details

The PPS is an asymmetric, data-type-agnostic score that can detect linear or non-linear relationships between two variables. The score ranges from 0 (no predictive power) to 1 (perfect predictive power).

Value

An object of the class as pps. Attributes of pps class is as follows.

- type : type of pps
- target : name of target variable
- predictor : name of predictor

Information of Predictive Power Score

The information of PPS is as follows.

- x : the name of the predictor variable
- y : the name of the target variable
- result_type : text showing how to interpret the resulting score
- pps : the predictive power score
- metric : the evaluation metric used to compute the PPS
- baseline_score : the score of a naive model on the evaluation metric
- model_score : the score of the predictive model on the evaluation metric
- cv_folds : how many cross-validation folds were used
- seed : the seed that was set
- algorithm : text showing what algorithm was used
- model_type : text showing whether classification or regression was used

References

- RIP correlation. Introducing the Predictive Power Score - by Florian Wetschoreck
  – https://towardsdatascience.com/rip-correlation-introducing-the-predictive-power-score-3d90808b9598

See Also

print.relate, plot.relate.

Examples

library(dplyr)

# pps type is generic
pps_generic <- pps(iris)
pps_generic
# summary pps class
mat <- summary(pps_generic)
mat

# visualize pps class
plot(pps_generic)

# pps type is target_by =====================================
##-----------------------------------------------------------
# If the target variable is a categorical variable
categ <- target_by(iris, Species)

# compute all variables
pps_cat <- pps(categ)
pps_cat

# compute Petal.Length and Petal.Width variable
pps_cat <- pps(categ, Petal.Length, Petal.Width)
pps_cat

# Using dplyr
pps_cat <- iris %>%
  target_by(Species) %>%
  pps()
pps_cat

# Using parallel process
# pps_cat <- iris %>%
#  target_by(Species) %>%
#  pps(do_parallel = TRUE)
#
# pps_cat

# summary pps class
tab <- summary(pps_cat)
tab

# visualize pps class
plot(pps_cat)

##-----------------------------------------------------------
# If the target variable is a numerical variable
num <- target_by(iris, Petal.Length)

pps_num <- pps(num)
pps_num

# summary pps class
tab <- summary(pps_num)
tab
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)
um_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 as 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)
um_num
summary(num_num)
plot(num_num)

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

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**

```r
## S3 method for class 'bins'
summary(object, ...)

## S3 method for class 'bins'
print(x, ...)
```

**Arguments**

- `object` an object of "bins" and "optimal_bins", usually, a result of a call to binning().
- `...` further arguments passed to or from other methods.
- `x` an object of class "bins" and "optimal_bins", usually, a result of a call to binning().

**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 platelets 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**

```r
## 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

```r
# 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.correlate**  
*Summary Correlation Coefficient*

**Description**

Summary method for "correlate" class.

**Usage**

```r
## S3 method for class 'correlate'
summary(object, ...)
```
Arguments

object  an object of class "correlate", usually, a result of a call to correlate().
... further arguments passed to or from other methods.

Details

summary.correlate compares the correlation coefficient by variables.

See Also

correlate, plot.correlate.

Examples

library(dplyr)

# Correlation type is "generic" -----------------------------------------
# Correlation coefficients of all numerical variables
corr_tab <- correlate(heartfailure)
corr_tab

# summary correlate class
mat <- summary(corr_tab)
mat

# Select the variable to compute
corr_tab <- correlate(heartfailure, creatinine, sodium)
corr_tab

# summary correlate class
mat <- summary(corr_tab)
mat

# Correlation type is "group" ------------------------------------------
##---------------------------------------------------------------
# If the target variable is a categorical variable
# Using dplyr
corr_tab <- heartfailure %>%
  group_by(smoking, death_event) %>%
correlate()
corr_tab

# summary correlate class
mat <- summary(corr_tab)
mat

corr_tab <- heartfailure %>%
  group_by(smoking, death_event) %>%
correlate(creatinine) %>%
filter(abs(coef_corr) >= 0.2)
corr_tab

# summary correlate class
mat <- summary(corr_tab)
mat

# 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
corr_tab <- con_sqlite %>%
  tbl("TB_HEARTFAILURE") %>%
correlate()

# summary correlate class
mat <- summary(corr_tab)
mat

# Disconnect DBMS
DBI::dbDisconnect(con_sqlite)

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 impute_na() or impute_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, impute_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")
summary(platelets)
plot(platelets)

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

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

## 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

```r
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 models.

Usage

```r
## 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

- 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

```r
# 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)
summary(perf)

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

plot(perf)
```

---

### summary.pps

**Summarizing Predictive Power Score**

**Description**

Print and summary method for "pps" class.

**Usage**

```r
## S3 method for class 'pps'
summary(object, ...)
```
Arguments

object: an object of class "pps", usually, a result of a call to pps().

... further arguments passed to or from other methods.

Details

summary.pps compares the PPS by variables.

See Also

pps, plot.pps.

Examples

library(dplyr)

# pps type is generic
pps_generic <- pps(iris)
pps_generic

# summary pps class
mat <- summary(pps_generic)
mat

# pps type is target_by
##-----------------------------------------------------------
# If the target variable is a categorical variable
# Using dplyr
pps_cat <- iris %>%
  target_by(Species) %>%
  pps()
pps_cat

# summary pps class
tab <- summary(pps_cat)
tab

##-----------------------------------------------------------
# If the target variable is a numerical variable
num <- target_by(iris, Petal.Length)
pps_num <- pps(num)
pps_num

# summary pps class
tab <- summary(pps_num)
tab
Summary of transformation information

### Description

Print and summary method for "transform" class.

### Usage

```r
## 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

```r
# 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

```r
## 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

```r
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**

```r
## 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

```r
# Calculates the all categorical variables
all_var <- univar_numeric(heartfailure)

# Print univar_numeric class object
all_var

# Calculates the platelets, sodium variable
```
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)

# Non typographic
plot(num_cat, typographic = FALSE)
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

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

Arguments

- `.data`: a tbl_dbi.
- `target`: target variable.
- `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

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

```r
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)
```

---

**theil**

*Theil’s U statistic*
**Theil**

**Description**
Computes the Theil’s U statistic between two categorical variables in data.frame.

**Usage**
```
theil(dfm, x, y)
```

**Arguments**
- `dfm`: data.frame. probability distributions.
- `x`: character. name of categorical or discrete variable.
- `y`: character. name of another categorical or discrete variable.

**Value**
data.frame. It has the following variables:
- `var1`: character. first variable name.
- `var2`: character. second variable name.
- `coef_corr`: numeric. Theil’s U statistic.

**See Also**
cramer.

**Examples**
```
theil(mtcars, "gear", "carb")
```

---

**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")  
)
```
transfom

Arguments

- x: numeric vector for transformation.

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. \((x - \text{min}) / (\text{max} - \text{min})\)
  
  - 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": \(x^2\) 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

```r
# Standardization ------------------------------
creatinine_minmax <- transform(heartfailure$creatinine, method = "minmax")
summary(creatinine_minmax)
plot(creatinine_minmax)

# Resolving Skewness --------------------------
creatinine_log <- transform(heartfailure$creatinine, method = "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

---

### Description

The eda_paged_report() paged report the information for data transformation.

### Usage

```r
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,
  base_family = NULL,
  ...
)
```

### Arguments

- `.data` a data.frame or a `tbl_df`.
- `target` character. target variable.
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 subtitle. default is "tomato1".
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.
base_family  character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)
...  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
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,
)
transformation_report

```r
output_dir = tempdir(),
font_family = NULL,
browse = TRUE
```

**Arguments**

- `.data` a data.frame or a `tbl_df`.
- `target` 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.
- `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.
- `browse` 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

```r
if (FALSE) {
# 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")
}
```

---

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

```r
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,
base_family = NULL,
...
)

Arguments

.data a data.frame or a tbl_df.
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 file on top left.
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.
base_family character. The name of the base font family to use for the visualization. If not specified, the font defined in dlookr is applied. (See details)

... 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

The base_family is selected from "Roboto Condensed", "Liberation Sans Narrow", "NanumSquare", "Noto Sans Korean". If you want to use a different font, use it after loading the Google font with import_google_font().

Examples

```r
if (FALSE) {
  # 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)
}
```

### univar_category

#### Statistic of univariate categorical variables

**Description**

The univar_category() calculates statistic of categorical variables that is frequency table.

**Usage**

```r
univar_category(.data, ...)
```

```r
## S3 method for class 'data.frame'
univar_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

`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 chi-squared 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

```r
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 an 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

Statist 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()`.
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 : standard 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 
service(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, 93
  jobchange, 99
binning, 5, 8, 10, 83, 111, 165
binning_by, 6, 7, 10, 83, 110, 120, 123, 175, 178
binning_rgr, 9, 119
compare_category, 11, 113
compare_numeric, 14, 114
cor, 18
correlate, 15, 17, 116, 172
cramer, 20, 188
describe, 21
describe.data.frame, 26, 103
describe.tbl_dbi, 23, 24, 105
diagnose, 27
diagnose.data.frame, 30, 32, 38, 42
diagnose.tbl_dbi, 28, 29, 35, 40, 44
diagnose_category, 31
diagnose_category.data.frame, 28, 35, 38, 42
diagnose_category.tbl_dbi, 30, 32, 34, 35, 40, 44
diagnose_numeric, 37
diagnose_numeric.data.frame, 23, 28, 32, 40, 42, 103
diagnose_numeric.tbl_dbi, 26, 30, 35, 38, 39, 44, 105
diagnose_outlier, 41
diagnose_outlier.data.frame, 32, 38, 44, 150
diagnose_outlier.tbl_dbi, 35, 40, 42, 43, 154
diagnose_paged_report, 45
diagnose_paged_report.data.frame, 51
diagnose_paged_report.tbl_dbi, 48, 49
diagnose_report, 52
diagnose_report.data.frame, 56
diagnose_report.tbl_dbi, 54
diagnose_sparese, 56
diagnose_web_report, 58
diagnose_web_report.data.frame, 63
diagnose_web_report.tbl_dbi, 60, 61
dlookr (dlookr-package), 4
dlookr-package, 4
dlookr_blue_paged
  (dlookr_orange_paged), 64
dlookr_orange_paged, 64
dlookr_templ_html, 64
eda_paged_report, 65
eda_paged_report.data.frame, 70
eda_paged_report.tbl_dbi, 67, 68
eda_report, 71
eda_report.data.frame, 75
eda_report.tbl_dbi, 73
eda_web_report, 77
eda_web_report.data.frame, 81
eda_web_report.tbl_dbi, 79, 79
entropy, 82
extract, 83
find_class, 84, 88
find_na, 85, 86, 87
find_outliers, 85, 86, 87
find_skewness, 87, 164
get_class, 84, 88
get_column_info, 89
get_os, 90
get_percentile, 91
get_transform, 92
group_by, 22, 25, 103, 105
grouped_df, 17, 18, 22, 132, 134, 136, 156, 184, 186
heartfailure, 93
html_paged, 64
import_google_font, 94
impute_na, 85, 95, 98, 117, 174
impute_outlier, 86, 96, 97, 117, 174
jobchange, 99
jkd, 100, 101
kurtosis, 102, 164
normality, 102
normality.data.frame, 105
normality.tbl_dbi, 103, 104
overview, 107, 122, 176
performance_bin, 108, 123, 178
plot.bins, 6, 111
plot.compare_category, 12, 112, 167
plot.compare_numeric, 15, 114, 170
plot.correlate, 18, 115, 172
plot.imputation, 117
plot.infogain_bins, 10, 118
plot.normality, 102
plot.normality.data.frame, 103, 147
plot.normality.tbl_dbi, 144, 146
plot.outlier, 148
plot.outlier.data.frame, 144, 152, 154
plot.univar_category, 128, 181, 197
plot.univar_numeric, 129, 183, 199
plot_bar_category, 131
plot_box_numeric, 134
plot_hist_numeric, 136
plot_na_hclust, 138
plot_na_intersect, 139
plot_na_pareto, 141
plot_normality, 92, 142
plot_normality.data.frame, 103, 147
plot_normality.tbl_dbi, 144, 146
plot_outlier, 148
plot_outlier.data.frame, 144, 152, 154
plot_outlier_target_df, 151
plot_outlier.tbl_dbi, 147, 150, 152
plot_qq_numeric, 155
print.bins, 6, 111
print.bins(summary.bins), 165
print.compare_category, 12, 113
print.compare_category
(summary.compare_category), 166
print.compare_numeric, 15, 114
print.compare_numeric
(summary.compare_numeric), 168
print.relate, 126, 158, 160, 163
print.univar_category, 129, 197
print.univar_category
(summary.univar_category), 181
print.univar_numeric, 130, 199
print.univar_numeric
(summary.univar_numeric), 182
relate, 126, 161, 185, 186
shapiro.test, 103, 105
skewness, 102, 164
summary.bins, 6, 111, 119, 165
summary.compare_category, 12, 113, 166
summary.compare_numeric, 15, 114, 168
summary.correlate, 18, 116, 171
summary.imputation, 117, 173, 174
summary.normality, 120, 174, 178
summary.overview, 108, 122, 176
summary.performance_bin, 110, 123, 177
summary.pps, 124, 178
summary.transform, 127, 180, 189
summary.univar_category, 129, 181, 197
summary.univar_numeric, 130, 182, 199
target_by, 151, 184
target_by.data.frame, 186
target_by.tbl_dbi, 186
tbl_df, 11, 14, 15, 22, 27, 31, 37, 41, 46, 53, 57, 59, 66, 71, 77, 85–87, 95, 97, 103, 107, 132, 134, 136, 143, 149, 156, 184, 190, 193, 195, 196, 198, 199
theil, 21, 187
transform, 127, 180, 188
transformationpaged_report, 190
transformation_report, 192
transformation_web_report, 194
univar_category, 129, 196
univar_numeric, 130, 198