Package 'DataExplorer'

January 26, 2017

Title Data Explorer

Version 0.4.0

Description Data exploration process for data analysis and model building, so that users could focus on understanding data and extracting insights. The package automatically scans through each variable and does data profiling. Typical graphical techniques will be performed for both discrete and continuous features.

Depends R (>= 3.3.0)

Imports data.table (>= 1.10.0), reshape2 (>= 1.4.2), ggplot2 (>= 2.2.0), scales (>= 0.4.1), gridExtra (>= 2.2.1), rmarkdown (>= 1.2), networkD3 (>= 0.2.13), stats, utils

Suggests testthat, covr, knitr, jsonlite, nycflights13

SystemRequirements pandoc (>= 1.12.3) - http://pandoc.org

License GPL-2

LazyData true

URL https://github.com/boxuancui/DataExplorer

BugReports https://github.com/boxuancui/DataExplorer/issues

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Boxuan Cui [aut, cre]

Maintainer Boxuan Cui <boxuancui@gmail.com>

Repository CRAN

Date/Publication 2017-01-26 12:03:11

R topics documented:

`DataExplorer-package` *Data Explorer*

Description

Simplify and automate EDA process and report generation.

Details

Data exploration process for data analysis and model building, so that users could focus on understanding data and extracting insights. The package automatically scans through each variable and does data profiling. Typical graphical techniques will be performed for both discrete and continuous features.

See Also

GenerateReport, PlotMissing, BarDiscrete, HistogramContinuous, PlotStr, CollapseCategory, SetNaTo

---

`BarDiscrete` *Create bar charts for discrete features*

Description

This function creates frequency bar charts for each discrete feature.

Usage

```
BarDiscrete(data, na.rm = TRUE, maxcat = 50, order_bar = TRUE)
```

Arguments

data	input data to be plotted, in either data.frame or data.table format.
na.rm	logical, indicating if missing values should be removed for each feature. The default is TRUE.
maxcat	maximum categories allowed for each feature. The default is 50. More information in 'Details' section.
order_bar	logical, indicating if bars should be ordered.

Details

If a discrete feature contains more categories than maxcat specifies, it will not be passed to the plotting function.

Examples

```
# load packages
library(ggplot2)
library(data.table)

# load diamonds dataset from ggplot2
data("diamonds")

# plot bar charts for diamonds dataset
BarDiscrete(diamonds)
BarDiscrete(diamonds, maxcat = 5)
```

CollapseCategory *Collapse categories for discrete features*

Description

Sometimes discrete features have sparse categories. This function will collapse the sparse categories for a discrete feature based on a given threshold.

Usage

```
CollapseCategory(data, feature, threshold, measure, update = FALSE,
  category_name = "OTHER")
```

Arguments

data	input data, in either data.frame or data.table format.
feature	name of the discrete feature to be collapsed.
threshold	the bottom x% categories to be collapsed, e.g., if set to 20%, categories with cumulative frequency of the bottom 20% will be collapsed.
measure	name of variable to be treated as additional measure to frequency.

update	logical, indicating if the data should be modified. Setting to TRUE will modify the input data directly, and **will only work with data.table**. The default is FALSE.
category_name	name of the bucket to group selected categories if update is set to TRUE. The default is "OTHER".

Details

If a continuous feature is passed to the argument feature, it will be force set to character-class.

Value

If update is set to FALSE, returns categories with cumulative frequency less than the input threshold. The output class will match the class of input data.

Examples

```
# load packages
library(data.table)

# generate data
data <- data.table("a" = as.factor(round(rnorm(500, 10, 5))), "b" = rexp(500, 1:500))

# view cumulative frequency without collpasing categories
CollapseCategory(data, "a", 0.2)

# view cumulative frequency based on another measure
CollapseCategory(data, "a", 0.2, measure = "b")

# collapse bottom 20% categories based on cumulative frequency
CollapseCategory(data, "a", 0.2, update = TRUE)
BarDiscrete(data)
```

CorrelationContinuous *Create correlation heatmap for continuous features*

Description

This function creates a correlation heatmap for all continuous features.

Usage

```
CorrelationContinuous(data, ...)
```

Arguments

data	input data to be plotted, in either data.frame or data.table format.
...	other arguments to be passed to cor.

Examples

```
# correlation of features from mtcars dataset
data(mtcars)
CorrelationContinuous(mtcars)
```

CorrelationDiscrete *Create correlation heatmap for discrete features*

Description

This function creates a correlation heatmap for all discrete categories.

Usage

```
CorrelationDiscrete(data, maxcat = 20, ...)
```

Arguments

data	input data to be plotted, in either data.frame or data.table format.
maxcat	maximum categories allowed for each feature. The default is 20.
...	other arguments to be passed to cor.

Details

The function first transposes all discrete categories into columns with binary outcomes, then calculates the correlation matrix (see cor) and plots it.

Examples

```
# correlation of discrete categories from diamonds dataset
library(ggplot2)
data(diamonds)
CorrelationDiscrete(diamonds)
```

DensityContinuous *Visualize density estimates for continuous features*

Description

This function visualizes density estimates for each continuous feature.

Usage

```
DensityContinuous(data, ...)
```

Arguments

data	input data to be plotted, in either data.frame or data.table format.
...	other arguments to be passed to geom_density.

Examples

```
# load library
library(data.table)

# plot using iris data
DensityContinuous(iris)

# plot using random data
set.seed(1)
data <- cbind(sapply(1:9, function(x) {
          runif(500, min = sample(100, 1), max = sample(1000, 1))
        }))
DensityContinuous(data)
```

DropVar *Drop selected variables*

Description

Quickly drop variables by either name or column position.

Usage

```
DropVar(data, ind)
```

Arguments

data	input data, in data.table format only.
ind	a vector of either names or column positions of the variables to be dropped.

Details

This function will only work with data.table object as input. Consider setting your input to data.table first then assign the original class back after applying the function.

Examples

```
# load packages
library(data.table)

# generate data
dt <- data.table(sapply(setNames(letters, letters), function(x) {assign(x, rnorm(100))}))
dt2 <- copy(dt)
```

```
# drop variables by name
names(dt)
DropVar(dt, letters[2:25])
names(dt)

# drop variables by column position
names(dt2)
DropVar(dt2, seq(2, 25))
names(dt2)

# work with non-data.table objects
iris_df <- data.table(iris)
DropVar(iris_df, "Species")
class(iris_df) <- "data.frame"
```

GenerateReport *GenerateReport Function*

Description

This function generates the report of data profiling.

Usage

```
GenerateReport(input_data, output_file = "report.html",
  output_dir = getwd(), ...)
```

Arguments

input_data	data source to be profiled, in either data.frame or data.table format.
output_file	output file name. The default is "report.html".
output_dir	output directory for report. The default is user's current directory.
...	other arguments to be passed to render.

Examples

```
## Not run:
# load library
library(ggplot2)
library(data.table)

# load data
data(diamonds)
diamonds2 <- data.table(diamonds)

# manually set some missing values
for (j in 5:ncol(diamonds2)) {
  set(diamonds2,
```

```
      i = sample.int(nrow(diamonds2), sample.int(nrow(diamonds2), 1)),
      j,
      value = NA_integer_)}

# generate report for diamonds dataset
GenerateReport(diamonds2,
               output_file = "report.html",
               output_dir = getwd(),
               html_document(toc = TRUE, toc_depth = 6, theme = "flatly"))

## End(Not run)
```

`HistogramContinuous` *Create histogram for continuous features*

Description

This function creates histogram for each continuous feature.

Usage

```
HistogramContinuous(data, ...)
```

Arguments

`data`	input data to be plotted, in either data.frame or data.table format.
`...`	other arguments to be passed to geom_histogram.

Examples

```
# load library
library(data.table)

# plot using iris data
HistogramContinuous(iris)

# plot using random data
set.seed(1)
data <- cbind(sapply(1:9, function(x) {rnorm(10000, sd = 30 * x)}))
HistogramContinuous(data, breaks = seq(-400, 400, length = 10))
```

PlotMissing *Plot missing values*

Description

This function returns and plots frequency of missing values for each feature.

Usage

```
PlotMissing(data)
```

Arguments

data	input data to be profiled, in either data.frame or data.table format.

Details

The returned object is suppressed by invisible to prevent unwanted text in GenerateReport.

Value

missing value information, such as frequency, percentage and suggested action.

Examples

```
# load packages
library(data.table)

# manipulate data
data <- data.table(iris)
for (j in 1:4) set(data, i=sample(150, j * 30), j, value = NA_integer_)

# plot and assign missing value information
plot_data <- PlotMissing(data)
plot_data
```

PlotStr *Visualize data structure*

Description

Visualize data structures in D3 network graph

Usage

```
PlotStr(data, type = c("diagonal", "radial"), max_level,
  print_network = TRUE, ...)
```

Arguments

data	input data
type	type of network diagram. Defaults to diagonalNetwork.
max_level	integer threshold of nested level to be visualized. Minimum 1 nested level and defaults to all.
print_network	logical indicating if network graph should be plotted. Defaults to TRUE.
...	other arguments to be passed to plotting functions. See diagonalNetwork and radialNetwork.

Value

input data structure in nested list. Could be transformed to json format with most JSON packages.

Examples

```
## Visualize structure of iris dataset
PlotStr(iris)

## Visualize object with radial network
PlotStr(rep(list(rep(list(mtcars), 6)), 4), type = "r")

## Generate complicated data object
obj <- list(
  "a" = list(iris, airquality, list(mtcars = mtcars, USArrests = USArrests)),
  "b" = list(list(ts(1:10, frequency = 4))),
  "c" = lm(rnorm(5) ~ seq(5)),
  "d" = lapply(1:5, function(x) return(as.function(function(y) y + 1)))
)
## Visualize data object with diagnal network
PlotStr(obj, type = "d")
## Visualize only top 2 nested levels
PlotStr(obj, type = "d", max_level = 2)
```

SetNaTo *Set all missing values to indicated value*

Description

Quickly set all missing values to indicated value.

Usage

```
SetNaTo(data, value)
```

Arguments

data	input data, in data.table format only.
value	a single value or a list of two values to be set to. See 'Details'.

Details

This function will only work with data.table object as input. Consider setting your input to data.table first then assign the original class back after applying the function.

The class of `value` will determine what type of columns to be set, e.g., if `value` is 0, then missing values for continuous features will be set. When supplying a list of two values, only one numeric and one non-numeric is allowed.

Examples

```
# Load packages
library(data.table)

# Generate missing values in iris data
dt <- data.table(iris)
for (j in 1:4) set(dt, i = sample.int(150, j * 30), j, value = NA_integer_)
set(dt, i = sample.int(150, 25), 5L, value = NA_character_)

# Set all missing values to 0L and unknown
dt2 <- copy(dt)
SetNaTo(dt2, list(0L, "unknown"))

# Set missing numerical values to 0L
dt3 <- copy(dt)
SetNaTo(dt3, 0L)

# Set missing discrete values to unknown
dt4 <- copy(dt)
SetNaTo(dt4, "unknown")
```

SplitColType *Split data into discrete and continuous*

Description

This function splits the input data into two data.table objects: discrete and continuous. A feature is continuous if `is.numeric` returns `TRUE`.

Usage

```
SplitColType(data)
```

Arguments

`data`	input data to be split, in either data.frame or data.table format.

Details

Features with all missing values will be dropped from the output data, but will be counted towards the column count.

The elements in the output list will have the same class as the input data.

Value

`discrete` all discrete features

`continous` all continuous features

`num_discrete` number of discrete features

`num_continuous` number of continuous features

`num_all_missing` number of features with no observations (all values are missing)

Examples

```
output <- SplitColType(iris)
output$discrete
output$continuous
output$num_discrete
output$num_continuous
output$num_all_missing
```

Index