# Package ‘DataExplorer’

July 25, 2025

**Title** Automate Data Exploration and Treatment

**Version** 0.8.4

**Description** Automated data exploration process for analytic tasks and predictive modeling, so that users could focus on understanding data and extracting insights. The package scans and analyzes each variable, and visualizes them with typical graphical techniques. Common data processing methods are also available to treat and format data.

**Depends** R (>= 3.6)

**Imports** data.table (>= 1.13.4), reshape2 (>= 1.4.3), scales (>= 1.1.0), ggplot2, gridExtra, rmarkdown (>= 2.5), networkD3 (>= 0.4), stats, utils, tools, parallel

**Suggests** testthat, covr, knitr, jsonlite, nycflights13

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

**License** MIT + file LICENSE

**Language** en-US

**URL** `http://boxuancui.github.io/DataExplorer/`

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

**RoxygenNote** 7.3.2

**Encoding** UTF-8

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Boxuan Cui [aut, cre]

**Maintainer** Boxuan Cui <`boxuancui@gmail.com`>

**Repository** CRAN

**Date/Publication** 2025-07-25 02:20:02 UTC

## Contents

---

## `configure_report` *Configure report template*

### Description

This function configures the content of the to-be-generated data profiling report.

### Usage

```
configure_report(
  add_introduce = TRUE,
  add_plot_intro = TRUE,
  add_plot_str = TRUE,
  add_plot_missing = TRUE,
  add_plot_histogram = TRUE,
  add_plot_density = FALSE,
  add_plot_qq = TRUE,
  add_plot_bar = TRUE,
  add_plot_correlation = TRUE,
  add_plot_prcomp = TRUE,
  add_plot_boxplot = TRUE,
  add_plot_scatterplot = TRUE,
  introduce_args = list(),
  plot_intro_args = list(),
 plot_str_args = list(type = "diagonal", fontSize = 35, width = 1000, margin = list(left
    = 350, right = 250)),
```

```
  plot_missing_args = list(),
  plot_histogram_args = list(),
  plot_density_args = list(),
  plot_qq_args = list(sampled_rows = 1000L),
  plot_bar_args = list(),
  plot_correlation_args = list(cor_args = list(use = "pairwise.complete.obs")),
  plot_prcomp_args = list(),
  plot_boxplot_args = list(),
  plot_scatterplot_args = list(sampled_rows = 1000L),
  global_ggtheme = quote(theme_gray()),
  global_theme_config = list()
)
```

## Arguments

| | |
|---|---|
| `add_introduce` | add introduce? Default is `TRUE`. |
| `add_plot_intro` | add plot_intro? Default is `TRUE`. |
| `add_plot_str` | add plot_str? Default is `TRUE`. |
| `add_plot_missing` | add plot_missing? Default is `TRUE`. |
| `add_plot_histogram` | add plot_histogram? Default is `TRUE`. |
| `add_plot_density` | add plot_density? Default is `FALSE`. |
| `add_plot_qq` | add plot_qq? Default is `TRUE`. |
| `add_plot_bar` | add plot_bar? Default is `TRUE`. |
| `add_plot_correlation` | add plot_correlation? Default is `TRUE`. |
| `add_plot_prcomp` | add plot_prcomp? Default is `TRUE`. |
| `add_plot_boxplot` | add plot_boxplot? Default is `TRUE`. |
| `add_plot_scatterplot` | add plot_scatterplot? Default is `TRUE`. |
| `introduce_args` | arguments to be passed to introduce. Default is `list()`. |
| `plot_intro_args` | arguments to be passed to plot_intro. Default is `list()`. |
| `plot_str_args` | arguments to be passed to plot_str. Default is `list(type = "diagonal", fontSize = 35, width = 1000, margin = list(left = 350, right = 250))`. |
| `plot_missing_args` | arguments to be passed to plot_missing. Default is `list()`. |
| `plot_histogram_args` | arguments to be passed to plot_histogram. Default is `list()`. |
| `plot_density_args` | arguments to be passed to plot_density. Default is `list()`. |

| | |
|---|---|
| `plot_qq_args` | arguments to be passed to plot_qq. Default is `list(sampled_rows = 1000L)`. |
| `plot_bar_args` | arguments to be passed to plot_bar. Default is `list()`. |
| `plot_correlation_args` | arguments to be passed to plot_correlation. Default is `list("cor_args" = list("use" = "pairwise.complete.obs"))`. |
| `plot_prcomp_args` | arguments to be passed to plot_prcomp. Default is `list()`. |
| `plot_boxplot_args` | arguments to be passed to plot_boxplot. Default is `list()`. |
| `plot_scatterplot_args` | arguments to be passed to plot_scatterplot. Default is `list(sampled_rows = 1000L)`. |
| `global_ggtheme` | global setting for theme. Default is `quote(theme_gray())`. |
| `global_theme_config` | global setting for theme. Default is `list()`. |

## Note

Individual settings will overwrite global settings. For example: if `plot_intro_args` has `ggtheme` set to `theme_light()` while `global_ggtheme` is set to `theme_gray()`, `theme_light()` will be used.

When setting global themes using `global_ggtheme`, please pass an unevaluated call to the theme function, e.g., `quote(theme_light())`.

## See Also

create_report

## Examples

```
## Get default configuration
configure_report()

## Set global theme
configure_report(global_ggtheme = quote(theme_light(base_size = 20L)))
```

---

# `create_report` *Create report*

## Description

This function creates a data profiling report.

## Usage

```
create_report(
  data,
  output_format = html_document(toc = TRUE, toc_depth = 6, theme = "yeti"),
  output_file = "report.html",
  output_dir = getwd(),
  y = NULL,
  config = configure_report(),
  report_title = "Data Profiling Report",
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `output_format` | output format in render. Default is `html_document(toc = TRUE, toc_depth = 6, theme = "yeti")`. |
| `output_file` | output file name in render. Default is "report.html". |
| `output_dir` | output directory for report in render. Default is user's current directory. |
| `y` | name of response variable if any. Response variables will be passed to appropriate plotting functions automatically. |
| `config` | report configuration generated by configure_report. |
| `report_title` | report title. Default is "Data Profiling Report". |
| `...` | other arguments to be passed to render. |

## Details

`config` is a named list to be evaluated by `create_report`. Each name should exactly match a function name. By doing so, that function and corresponding content will be added to the report. If you do not want to include certain functions/content, do not add it to `config`.

configure_report generates the default template. You may customize the content using that function.

All function arguments will be passed to do.call as a list.

## Note

If both `y` and `plot_prcomp` are present, `y` will be removed from `plot_prcomp`.

If there are multiple options for the same function, all of them will be plotted. For example, `create_report(..., y = "a", config = list("plot_bar" = list("with" = "b")))` will create 3 bar charts:

- regular frequency bar chart
- bar chart aggregated by response variable "a"
- bar chart aggregated by ‘with‘ variable "b"‘

## See Also

configure_report

## Examples

```
## Not run:
# Create report
create_report(iris)
create_report(airquality, y = "Ozone")

# Load library
library(ggplot2)
library(data.table)
library(rmarkdown)

# Set some missing values
diamonds2 <- data.table(diamonds)
for (j in 5:ncol(diamonds2)) {
  set(diamonds2,
      i = sample.int(nrow(diamonds2), sample.int(nrow(diamonds2), 1)),
      j,
      value = NA_integer_)
}

# Create customized report for diamonds2 dataset
create_report(
  data = diamonds2,
  output_format = html_document(toc = TRUE, toc_depth = 6, theme = "flatly"),
  output_file = "report.html",
  output_dir = getwd(),
  y = "price",
  config = configure_report(
    add_plot_prcomp = TRUE,
    plot_qq_args = list("by" = "cut", sampled_rows = 1000L),
    plot_bar_args = list("with" = "carat"),
    plot_correlation_args = list("cor_args" = list("use" = "pairwise.complete.obs")),
    plot_boxplot_args = list("by" = "cut"),
    global_ggtheme = quote(theme_light())
  )
)

## Configure report without `configure_report`
config <- list(
  "introduce" = list(),
  "plot_intro" = list(),
  "plot_str" = list(
    "type" = "diagonal",
    "fontSize" = 35,
    "width" = 1000,
    "margin" = list("left" = 350, "right" = 250)
  ),
  "plot_missing" = list(),
```

```
  "plot_histogram" = list(),
  "plot_density" = list(),
  "plot_qq" = list(sampled_rows = 1000L),
  "plot_bar" = list(),
  "plot_correlation" = list("cor_args" = list("use" = "pairwise.complete.obs")),
  "plot_prcomp" = list(),
  "plot_boxplot" = list(),
  "plot_scatterplot" = list(sampled_rows = 1000L)
)

## End(Not run)
```

---

## `drop_columns` *Drop selected variables*

### Description

Quickly drop variables by either column names or positions.

### Usage

```
drop_columns(data, ind)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `ind` | a vector of either names or column positions of the variables to be dropped. |

### Details

**This function updates data.table object directly.** Otherwise, output data will be returned matching input object class.

### Examples

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

# Generate data
dt <- data.table(sapply(setNames(letters, letters), function(x) {assign(x, rnorm(10))}))
dt2 <- copy(dt)

# Drop variables by name
names(dt)
drop_columns(dt, letters[2L:25L])
names(dt)

# Drop variables by column position
names(dt2)
```

```
drop_columns(dt2, seq(2, 25))
names(dt2)

# Return from non-data.table input
df <- data.frame(sapply(setNames(letters, letters), function(x) {assign(x, rnorm(10))}))
drop_columns(df, letters[2L:25L])
```

---

## `dummify` *Dummify discrete features to binary columns*

### Description

Data dummification is also known as one hot encoding or feature binarization. It turns each category to a distinct column with binary (numeric) values.

### Usage

```
dummify(data, maxcat = 50L, select = NULL)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `maxcat` | maximum categories allowed for each discrete feature. Default is 50. |
| `select` | names of selected features to be dummified. Default is `NULL`. |

### Details

Continuous features will be ignored if added in `select`.

`select` features will be ignored if categories exceed `maxcat`.

### Value

dummified dataset (discrete features only) preserving original features. However, column order might be different.

### Note

This is different from model.matrix, where the latter aims to create a full rank matrix for regression-like use cases. If your intention is to create a design matrix, use model.matrix instead.

### Examples

```
## Dummify iris dataset
str(dummify(iris))

## Dummify diamonds dataset ignoring features with more than 5 categories
data("diamonds", package = "ggplot2")
str(dummify(diamonds, maxcat = 5))
str(dummify(diamonds, select = c("cut", "color")))
```

---

## `group_category` *Group categories for discrete features*

---

### Description

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

### Usage

```
group_category(
  data,
  feature,
  threshold,
  measure,
  update = FALSE,
  category_name = "OTHER",
  exclude = NULL
)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `feature` | name of the discrete feature to be collapsed. |
| `threshold` | the bottom x% categories to be grouped, e.g., if set to 20%, categories with cumulative frequency of the bottom 20% will be grouped |
| `measure` | name of feature to be used as an alternative measure. |
| `update` | logical, indicating if the data should be modified. The default is `FALSE`. Setting to `TRUE` will modify the input data.table object directly. Otherwise, input class will be returned. |
| `category_name` | name of the new category if update is set to `TRUE`. The default is "OTHER". |
| `exclude` | categories to be excluded from grouping when update is set to `TRUE`. |

### Details

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

### 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. If `update` is set to `TRUE`, updated data will be returned, and 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, 500))

# View cumulative frequency without collpasing categories
group_category(data, "a", 0.2)

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

# Group bottom 20% categories based on cumulative frequency
group_category(data, "a", 0.2, update = TRUE)
plot_bar(data)

# Exclude categories from being grouped
dt <- data.table("a" = c(rep("c1", 25), rep("c2", 10), "c3", "c4"))
group_category(dt, "a", 0.8, update = TRUE, exclude = c("c3", "c4"))
plot_bar(dt)

# Return from non-data.table input
df <- data.frame("a" = as.factor(round(rnorm(50, 10, 5))), "b" = rexp(50, 10))
group_category(df, "a", 0.2)
group_category(df, "a", 0.2, measure = "b", update = TRUE)
group_category(df, "a", 0.2, update = TRUE)
```

---

## `introduce` *Describe basic information*

---

### Description

Describe basic information for input data.

### Usage

```
introduce(data)
```

### Arguments

| | |
|---|---|
| `data` | input data |

### Value

Describe basic information in input data class:

- rows: number of rows
- columns: number of columns

- discrete_columns: number of discrete columns
- continuous_columns: number of continuous columns
- all_missing_columns: number of columns with everything missing
- total_missing_values: number of missing observations
- complete_rows: number of rows without missing values. See complete.cases.
- total_observations: total number of observations
- memory_usage: estimated memory allocation in bytes. See object.size.

### Examples

```
introduce(mtcars)
```

---

## `plot_bar` *Plot bar chart*

---

### Description

Plot bar chart for each discrete feature, based on either frequency or another continuous feature.

### Usage

```
plot_bar(
  data,
  with = NULL,
  by = NULL,
  by_position = "fill",
  maxcat = 50,
  order_bar = TRUE,
  binary_as_factor = TRUE,
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 3L,
  ncol = 3L,
  parallel = FALSE
)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `with` | name of continuous feature to be summed. Default is `NULL`, i.e., frequency. |
| `by` | discrete feature name to be broken down by. |
| `by_position` | position argument in geom_bar if `by` is supplied. Default is `"fill"`. |
| `maxcat` | maximum categories allowed for each feature. Default is 50. |

| | |
|---|---|
| `order_bar` | logical, indicating if bars should be ordered. Default is `TRUE`. |
| `binary_as_factor` | treat binary as categorical? Default is `TRUE`. |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. Default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme |
| `nrow` | number of rows per page. Default is 3. |
| `ncol` | number of columns per page. Default is 3. |
| `parallel` | enable parallel? Default is `FALSE`. |

### Details

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

### Value

invisibly return the named list of ggplot objects

### Examples

```
# Plot bar charts for diamonds dataset
library(ggplot2)
plot_bar(diamonds)
plot_bar(diamonds, maxcat = 5)

# Plot bar charts with `price`
plot_bar(diamonds, with = "price")

# Plot bar charts by `cut`
plot_bar(diamonds, by = "cut")
plot_bar(diamonds, by = "cut", by_position = "dodge")
```

---

## `plot_boxplot` *Create boxplot for continuous features*

### Description

This function creates boxplot for each continuous feature based on a selected feature.

### Usage

```
plot_boxplot(
  data,
  by,
  binary_as_factor = TRUE,
  geom_boxplot_args = list(),
  geom_jitter_args = list(),
  scale_y = "continuous",
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 3L,
  ncol = 4L,
  parallel = FALSE
)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `by` | feature name to be broken down by. If selecting a continuous feature, boxplot will be grouped by 5 equal ranges, otherwise, all existing categories for a discrete feature. |
| `binary_as_factor` | treat binary as categorical? Default is `TRUE`. |
| `geom_boxplot_args` | a list of other arguments to geom_boxplot |
| `geom_jitter_args` | a list of other arguments to geom_jitter. If empty, geom_jitter will not be added. |
| `scale_y` | scale of original y axis (before `coord_flip`). See scale_y_continuous for all options. Default is `continuous`. |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |
| `nrow` | number of rows per page |
| `ncol` | number of columns per page |
| `parallel` | enable parallel? Default is `FALSE`. |

### Value

invisibly return the named list of ggplot objects

### See Also

geom_boxplot

### Examples

```
plot_boxplot(iris, by = "Species", ncol = 2L)
plot_boxplot(iris, by = "Species", geom_boxplot_args = list("outlier.color" = "red"))

# Plot skewed data on log scale
set.seed(1)
skew <- data.frame(y = rep(c("a", "b"), 500), replicate(4L, rbeta(1000, 1, 5000)))
plot_boxplot(skew, by = "y", ncol = 2L)
plot_boxplot(skew, by = "y", scale_y = "log10", ncol = 2L)

# Plot with `geom_jitter`
plot_boxplot(iris, by = "Species", ncol = 2L,
geom_jitter_args = list(width = NULL)) # Turn on with default settings
```

---

## `plot_correlation` *Create correlation heatmap for discrete features*

### Description

This function creates a correlation heatmap for all discrete categories.

### Usage

```
plot_correlation(
  data,
  type = c("all", "discrete", "continuous"),
  maxcat = 20L,
  cor_args = list(),
  geom_text_args = list(),
  title = NULL,
  ggtheme = theme_gray(),
 theme_config = list(legend.position = "bottom", axis.text.x = element_text(angle = 90))
)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `type` | column type to be included in correlation calculation. "all" for all columns, "discrete" for discrete features, "continuous" for continuous features. |
| `maxcat` | maximum categories allowed for each discrete feature. The default is 20. |
| `cor_args` | a list of other arguments to cor |
| `geom_text_args` | a list of other arguments to geom_text |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |

## Details

For discrete features, the function first dummifies all categories, then calculates the correlation matrix (see cor) and plots it.

## Value

invisibly return the ggplot object

## Examples

```
plot_correlation(iris)
plot_correlation(iris, type = "c")
plot_correlation(airquality, cor_args = list("use" = "pairwise.complete.obs"))
```

---

# `plot_density` *Plot density estimates*

---

## Description

Plot density estimates for each continuous feature

## Usage

```
plot_density(
  data,
  binary_as_factor = TRUE,
  geom_density_args = list(),
  scale_x = "continuous",
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 4L,
  ncol = 4L,
  parallel = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `binary_as_factor` | treat binary as categorical? Default is `TRUE`. |
| `geom_density_args` | a list of other arguments to geom_density |
| `scale_x` | scale of x axis. See scale_x_continuous for all options. Default is `continuous`. |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |

| | |
|---|---|
| `theme_config` | a list of configurations to be passed to theme. |
| `nrow` | number of rows per page. Default is 4. |
| `ncol` | number of columns per page. Default is 4. |
| `parallel` | enable parallel? Default is `FALSE`. |

### Value

invisibly return the named list of ggplot objects

### See Also

geom_density plot_histogram

### Examples

```
# Plot iris data
plot_density(iris, ncol = 2L)

# Add color to density area
plot_density(iris, geom_density_args = list("fill" = "black", "alpha" = 0.6), ncol = 2L)

# Plot skewed data on log scale
set.seed(1)
skew <- data.frame(replicate(4L, rbeta(1000, 1, 5000)))
plot_density(skew, ncol = 2L)
plot_density(skew, scale_x = "log10", ncol = 2L)
```

---

## `plot_histogram` *Plot histogram*

### Description

Plot histogram for each continuous feature

### Usage

```
plot_histogram(
  data,
  binary_as_factor = TRUE,
  geom_histogram_args = list(bins = 30L),
  scale_x = "continuous",
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 4L,
  ncol = 4L,
  parallel = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `binary_as_factor` | treat binary as categorical? Default is `TRUE`. |
| `geom_histogram_args` | a list of other arguments to geom_histogram |
| `scale_x` | scale of x axis. See scale_x_continuous for all options. Default is `continuous`. |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |
| `nrow` | number of rows per page. Default is 4. |
| `ncol` | number of columns per page. Default is 4. |
| `parallel` | enable parallel? Default is `FALSE`. |

## Value

invisibly return the named list of ggplot objects

## See Also

geom_histogram plot_density

## Examples

```
# Plot iris data
plot_histogram(iris, ncol = 2L)

# Plot skewed data on log scale
set.seed(1)
skew <- data.frame(replicate(4L, rbeta(1000, 1, 5000)))
plot_histogram(skew, ncol = 2L)
plot_histogram(skew, scale_x = "log10", ncol = 2L)
```

---

# `plot_intro` *Plot introduction*

## Description

Plot basic information (from introduce) for input data.

### Usage

```
plot_intro(
  data,
  geom_label_args = list(),
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list()
)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `geom_label_args` | a list of other arguments to geom_label |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |

### Value

invisibly return the ggplot object

### See Also

introduce

### Examples

```
plot_intro(airquality)
plot_intro(iris)
```

---

## `plot_missing` *Plot missing value profile*

### Description

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

### Usage

```
plot_missing(
  data,
  group = list(Good = 0.05, OK = 0.4, Bad = 0.8, Remove = 1),
  group_color = list(Good = "#1B9E77", OK = "#E6AB02", Bad = "#D95F02", Remove =
    "#E41A1C"),
  missing_only = FALSE,
```

```
  geom_label_args = list(),
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(legend.position = c("bottom"))
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `group` | missing profile band taking a list of group name and group upper bounds. Default is `list("Good" = 0.05, "OK" = 0.4, "Bad" = 0.8, "Remove" = 1)`. |
| `group_color` | bar color list for each of the group. Default is `list("Good" = "#1B9E77", "OK" = "#E6AB02", "Bad" = "#D95F02", "Remove" = "#E41A1C")`. |
| `missing_only` | plot features with missing values only? Default is `FALSE`. |
| `geom_label_args` | a list of other arguments to geom_label |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |

## Value

invisibly return the ggplot object

## See Also

profile_missing

## Examples

```
plot_missing(airquality)
plot_missing(airquality, missing_only = TRUE)

## Customize band
plot_missing(airquality, group = list("B1" = 0, "B2" = 0.06, "B3" = 1))
plot_missing(airquality, group = list("Awesome!" = 0.2, "Oh no!" = 1),
group_color = list("Awesome!" = "green", "Oh no!" = "red"))

## Shrink geom_label size
library(ggplot2)
plot_missing(airquality, geom_label_args = list("size" = 2,
"label.padding" = unit(0.1, "lines")))
```

---

`plot_prcomp` *Visualize principal component analysis*

---

## Description

Visualize output of prcomp.

## Usage

```
plot_prcomp(
  data,
  variance_cap = 0.8,
  maxcat = 50L,
  prcomp_args = list(scale. = TRUE),
  geom_label_args = list(),
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 3L,
  ncol = 3L,
  parallel = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `variance_cap` | maximum cumulative explained variance allowed for all principal components. Default is 80%. |
| `maxcat` | maximum categories allowed for each discrete feature. The default is 50. |
| `prcomp_args` | a list of other arguments to prcomp |
| `geom_label_args` | a list of other arguments to geom_label |
| `title` | plot title starting from page 2. |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |
| `nrow` | number of rows per page |
| `ncol` | number of columns per page |
| `parallel` | enable parallel? Default is `FALSE`. |

## Details

When cumulative explained variance exceeds `variance_cap`, remaining principal components will be ignored. Set `variance_cap` to 1 for all principal components.

Discrete features containing more categories than `maxcat` specifies will be ignored.

## Value

invisibly return the named list of ggplot objects

## Note

Discrete features will be dummify-ed first before passing to prcomp.

Missing values may create issues in prcomp. Consider na.omit your input data first.

Features with zero variance are dropped.

## Examples

```
plot_prcomp(na.omit(airquality), nrow = 2L, ncol = 2L)
```

---

# `plot_qq` *Plot QQ plot*

## Description

Plot quantile-quantile for each continuous feature

## Usage

```
plot_qq(
  data,
  by = NULL,
  sampled_rows = nrow(data),
  geom_qq_args = list(),
  geom_qq_line_args = list(),
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 3L,
  ncol = 3L,
  parallel = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `by` | feature name to be broken down by. If selecting a continuous feature, it will be grouped by 5 equal ranges, otherwise, all existing categories for a discrete feature. Default is `NULL`. |
| `sampled_rows` | number of rows to sample if data has too many rows. Default is all rows, which means do not sample. |
| `geom_qq_args` | a list of other arguments to geom_qq |

| | |
|---|---|
| `geom_qq_line_args` | a list of other arguments to geom_qq_line |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. Default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme |
| `nrow` | number of rows per page. Default is 3. |
| `ncol` | number of columns per page. Default is 3. |
| `parallel` | enable parallel? Default is `FALSE`. |

### Value

invisibly return the named list of ggplot objects

### Examples

```
plot_qq(iris)
plot_qq(iris, by = "Species", ncol = 2L)

plot_qq(
  data = airquality,
  geom_qq_args = list(na.rm = TRUE),
  geom_qq_line_args = list(na.rm = TRUE)
)
```

---

## `plot_scatterplot` *Create scatterplot for all features*

### Description

This function creates scatterplot for all features fixing on a selected feature.

### Usage

```
plot_scatterplot(
  data,
  by,
  sampled_rows = nrow(data),
  geom_point_args = list(),
  geom_jitter_args = list(),
  scale_x = NULL,
  scale_y = NULL,
  title = NULL,
  ggtheme = theme_gray(),
  theme_config = list(),
  nrow = 3L,
  ncol = 3L,
  parallel = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `by` | feature name to be fixed at |
| `sampled_rows` | number of rows to sample if data has too many rows. Default is all rows, which means do not sample. |
| `geom_point_args` | a list of other arguments to geom_point |
| `geom_jitter_args` | a list of other arguments to geom_jitter. If empty, geom_jitter will not be added. |
| `scale_x` | scale of original x axis (before `coord_flip`). See scale_x_continuous for all options. Default is `NULL`. |
| `scale_y` | scale of original y axis (before `coord_flip`). See scale_y_continuous for all options. Default is `NULL`. |
| `title` | plot title |
| `ggtheme` | complete ggplot2 themes. The default is theme_gray. |
| `theme_config` | a list of configurations to be passed to theme. |
| `nrow` | number of rows per page |
| `ncol` | number of columns per page |
| `parallel` | enable parallel? Default is `FALSE`. |

## Value

invisibly return the named list of ggplot objects

## See Also

geom_point

## Examples

```
plot_scatterplot(iris, by = "Species")

# Plot skewed data on log scale
set.seed(1)
skew <- data.frame(replicate(5L, rbeta(1000, 1, 5000)))
plot_scatterplot(skew, by = "X5", ncol = 2L)
plot_scatterplot(skew, by = "X5", scale_x = "log10",
scale_y = "log10", ncol = 2L)

# Plot with `geom_jitter`
plot_scatterplot(iris, by = "Species",
geom_jitter_args = list(width = NULL)) # Turn on with default settings
plot_scatterplot(iris, by = "Species",
geom_jitter_args = list(width = 0.1, height = 0.1))

## Not run:
```

```
# Customize themes
library(ggplot2)
plot_scatterplot(
  data = mpg,
  by = "hwy",
  geom_point_args = list(size = 1L),
  theme_config = list("axis.text.x" = element_text(angle = 90)),
  ncol = 4L
)

## End(Not run)
```

---

## `plot_str` *Visualize data structure*

### Description

Visualize data structures in D3 network graph

### Usage

```
plot_str(
  data,
  type = c("diagonal", "radial"),
  max_level = NULL,
  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.

### See Also

str

### Examples

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

## Visualize object with radial network
plot_str(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
plot_str(obj, type = "d")
## Visualize only top 2 nested levels
plot_str(obj, type = "d", max_level = 2)
```

---

## `profile_missing` *Profile missing values*

### Description

Analyze missing value profile

### Usage

```
profile_missing(data)
```

### Arguments

| | |
|---|---|
| `data` | input data |

### Value

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

### See Also

plot_missing

### Examples

```
profile_missing(airquality)
```

## `set_missing` *Set all missing values to indicated value*

### Description

Quickly set all missing values to indicated value.

### Usage

```
set_missing(data, value, exclude = NULL)
```

### 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'. |
| `exclude` | column index or name to be excluded. |

### Details

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.

**This function updates data.table object directly.** Otherwise, output data will be returned matching input object class.

### 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)
set_missing(dt2, list(0L, "unknown"))

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

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

# Set missing values excluding some columns
```

```
dt5 <- copy(dt)
set_missing(dt4, 0L, 1L:2L)
set_missing(dt4, 0L, names(dt5)[3L:4L])

# Return from non-data.table input
set_missing(airquality, 999999L)
```

---

## `split_columns` *Split data into discrete and continuous parts*

### 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

```
split_columns(data, binary_as_factor = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | input data |
| `binary_as_factor` | treat binary as categorical? Default is `FALSE`. |

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

---

`update_columns` *Update variable types or values*

---

## Description

Quickly update selected variables using column names or positions.

## Usage

```
update_columns(data, ind, what)
```

## Arguments

| | |
|---|---|
| `data` | input data |
| `ind` | a vector of either names or column positions of the variables to be dropped. |
| `what` | either a function or a non-empty character string naming the function to be called. See do.call. |

## Details

**This function updates data.table object directly.** Otherwise, output data will be returned matching input object class.

## Examples

```
str(update_columns(iris, 1L, as.factor))
str(update_columns(iris, c("Sepal.Width", "Petal.Length"), "as.integer"))

## Apply log transformation to all columns
summary(airquality)
summary(update_columns(airquality, names(airquality), log))

## Force set factor to numeric
df <- data.frame("a" = as.factor(sample.int(10L)))
str(df)
str(update_columns(df, "a", function(x) as.numeric(levels(x))[x]))
```

# Index