

# Package ‘explore’

May 8, 2026

**Type** Package

**Title** Simplifies Exploratory Data Analysis

**Version** 1.4.0

**Description** Interactive data exploration with one line of code, automated reporting or use an easy to remember set of tidy functions for low code exploratory data analysis.

**License** MIT + file LICENSE

**URL** <https://rolkra.github.io/explore/>,  
<https://github.com/rolkra/explore>

**BugReports** <https://github.com/rolkra/explore/issues>

**Depends** R (>= 3.5.0)

**Imports** cli, dplyr (>= 1.1.0), DT (>= 0.3.0), forcats (>= 1.0.0), ggplot2 (>= 3.4.0), grDevices, gridExtra, magrittr, palmerpenguins, plotly, rlang (>= 1.1.0), rmarkdown, rpart, rpart.plot, shiny, stats, stringr, tibble

**Suggests** knitr, MASS, randomForest, xgboost, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Roland Krasser [aut, cre]

**Maintainer** Roland Krasser <roland.krasser@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-12-11 06:10:31 UTC

## Contents

abtest	4
abtest_shiny	5
abtest_targetnum	5
abtest_targetpct	6
add_var_id	7
add_var_random_01	8
add_var_random_cat	8
add_var_random_dbl	9
add_var_random_int	10
add_var_random_moon	11
add_var_random_starsign	12
balance_target	13
check_vec_low_variance	13
clean_var	14
count_pct	15
create_data_abtest	15
create_data_app	16
create_data_buy	17
create_data_churn	18
create_data_empty	19
create_data_esoteric	20
create_data_newsletter	21
create_data_person	21
create_data_random	22
create_data_unfair	23
create_notebook_explore	24
cut_vec_num_avg	24
data_dict_md	25
decrypt	26
describe	26
describe_all	27
describe_cat	28
describe_num	28
describe_tbl	29
drop_obs_if	30
drop_obs_with_na	30
drop_var_by_names	31
drop_var_low_variance	31
drop_var_not_numeric	32
drop_var_no_variance	32
drop_var_with_na	33
encrypt	33
explain_forest	34
explain_logreg	35
explain_tree	35
explain_xgboost	37

explore	38
explore_all	39
explore_bar	40
explore_col	41
explore_cor	42
explore_count	43
explore_density	44
explore_shiny	45
explore_targetpct	46
explore_tbl	47
format_num_auto	47
format_num_kMB	48
format_num_space	48
format_target	49
format_type	49
get_color	50
get_type	50
get_var_buckets	51
guess_cat_num	52
interact	52
log_info_if	53
mix_color	53
plot_legend_targetpct	54
plot_text	54
plot_var_info	55
predict_target	55
replace_na_with	56
report	57
rescale01	57
show_color	58
simplify_text	58
target_explore_cat	59
target_explore_num	60
total_fig_height	61
use_data_beer	62
use_data_diamonds	62
use_data_iris	63
use_data_mpg	63
use_data_mtcars	64
use_data_penguins	64
use_data_starwars	65
use_data_titanic	65
use_data_wordle	66
weight_target	66
yyymm_calc	67

---

abtest	<i>A/B testing</i>
--------	--------------------

---

## Description

A/B testing

## Usage

```
abtest(data, expr, n, target, sign_level = 0.05, color = "grey")
```

## Arguments

data	A dataset. If no data is provided, a shiny app is launched
expr	Logical expression, that return in a FALSE/TRUE
n	A Variable for number of observations (count data)
target	Target variable
sign_level	Significance Level (typical 0.01/0.05/0.10)
color	Fill color of bar/violin-plot

## Value

Plot that shows if difference is significant

## Examples

```
## Using chi2-test or t-test depending on target type
data <- create_data_buy(obs = 100)
abtest(data, female_ind == 1, target = buy) # chi2 test
abtest(data, city_ind == 1, target = age) # t test

## If small number of observations, Fisher's Exact test
## is used for a binary target (if <= 5 observations in a subgroup)
data <- create_data_buy(obs = 25, seed = 1)
abtest(data, female_ind == 1, target = buy) # Fisher's Exact test
```

---

abtest_shiny	<i>A/B testing interactive</i>
--------------	--------------------------------

---

**Description**

Launches a shiny app to A/B test

**Usage**

```
abtest_shiny(  
  size_a = 100,  
  size_b = 100,  
  success_a = 10,  
  success_b = 20,  
  success_unit = "percent",  
  sign_level = 0.05  
)
```

**Arguments**

size_a	Size of Group A
size_b	Size of Group B
success_a	Success of Group A
success_b	Success of Group B
success_unit	"count"   "percent"
sign_level	Significance Level (typical 0.01/0.05/0.10)

**Examples**

```
# Only run examples in interactive R sessions  
if (interactive()) {  
  abtest_shiny()  
}
```

---

abtest_targetnum	<i>A/B testing comparing two mean</i>
------------------	---------------------------------------

---

**Description**

A/B testing comparing two mean

**Usage**

```
abtest_targetnum(data, expr, target, sign_level = 0.05, color = "grey")
```

**Arguments**

data	A dataset
expr	Expression, that results in a FALSE/TRUE
target	Target variable (must be numeric)
sign_level	Significance Level (typical 0.01/0.05/0.10)
color	fill color

**Value**

Plot that shows if difference is significant

**Examples**

```
data <- create_data_buy(obs = 100)
abtest(data, city_ind == 1, target = age)
```

---

abtest_targetpct	<i>A/B testing comparing percent per group</i>
------------------	--

---

**Description**

A/B testing comparing percent per group

**Usage**

```
abtest_targetpct(
  data,
  expr,
  n,
  target,
  sign_level = 0.05,
  group_label,
  ab_label = FALSE,
  color = "grey"
)
```

**Arguments**

data	A dataset
expr	Expression, that results in a FALSE/TRUE
n	A Variable for number of observations (count data)
target	Target variable (must be 0/1 or FALSE/TRUE)
sign_level	Significance Level (typical 0.01/0.05/0.10)
group_label	Label of groups (default = expr)
ab_label	Label Groups as A and B (default = FALSE)
color	color of bar

**Value**

Plot that shows if difference is significant

**Examples**

```
data <- create_data_buy(obs = 100)
abtest(data, female_ind == 1, target = buy)
abtest(data, age >= 40, target = buy)
```

---

add_var_id	<i>Add a variable id at first column in dataset</i>
------------	---

---

**Description**

Add a variable id at first column in dataset

**Usage**

```
add_var_id(data, name = "id", overwrite = FALSE)
```

**Arguments**

data	A dataset
name	Name of new variable (as string)
overwrite	Can new id variable overwrite an existing variable in dataset?

**Value**

Data set containing new id variable

**Examples**

```
library(magrittr)
iris %>% add_var_id() %>% head()
iris %>% add_var_id(name = "iris_nr") %>% head()
```

---

add\_var\_random\_01      *Add a random 0/1 variable to dataset*

---

### Description

Add a random 0/1 variable to dataset

### Usage

```
add_var_random_01(  
  data,  
  name = "random_01",  
  prob = c(0.5, 0.5),  
  overwrite = TRUE,  
  seed  
)
```

### Arguments

data	A dataset
name	Name of new variable (as string)
prob	Vector of probabilities
overwrite	Can new random variable overwrite an existing variable in dataset?
seed	Seed for random number generation (integer)

### Value

Dataset containing new random variable

### Examples

```
library(magrittr)  
iris %>% add_var_random_01() %>% head()  
iris %>% add_var_random_01(name = "my_var") %>% head()
```

---

add\_var\_random\_cat      *Add a random categorical variable to dataset*

---

### Description

Add a random categorical variable to dataset

**Usage**

```
add_var_random_cat(  
  data,  
  name = "random_cat",  
  cat = LETTERS[1:6],  
  prob,  
  overwrite = TRUE,  
  seed  
)
```

**Arguments**

data	A dataset
name	Name of new variable (as string)
cat	Vector of categories
prob	Vector of probabilities
overwrite	Can new random variable overwrite an existing variable in dataset?
seed	Seed for random number generation (integer)

**Value**

Dataset containing new random variable

**Examples**

```
library(magrittr)  
iris %>% add_var_random_cat() %>% head()  
iris %>% add_var_random_cat(name = "my_cat") %>% head()  
iris %>% add_var_random_cat(cat = c("Version A", "Version B")) %>% head()  
iris %>% add_var_random_cat(cat = c(1,2,3,4,5)) %>% head()
```

---

add\_var\_random\_dbl      *Add a random double variable to dataset*

---

**Description**

Add a random double variable to dataset

**Usage**

```
add_var_random_dbl(  
  data,  
  name = "random_dbl",  
  min_val = 0,  
  max_val = 100,  
  overwrite = TRUE,  
  seed  
)
```

**Arguments**

data	A dataset
name	Name of new variable (as string)
min_val	Minimum random integers
max_val	Maximum random integers
overwrite	Can new random variable overwrite an existing variable in dataset?
seed	Seed for random number generation (integer)

**Value**

Dataset containing new random variable

**Examples**

```
library(magrittr)
iris %>% add_var_random_dbl() %>% head()
iris %>% add_var_random_dbl(name = "random_var") %>% head()
iris %>% add_var_random_dbl(min_val = 1, max_val = 10) %>% head()
```

---

add\_var\_random\_int      *Add a random integer variable to dataset*

---

**Description**

Add a random integer variable to dataset

**Usage**

```
add_var_random_int(  
  data,  
  name = "random_int",  
  min_val = 1,  
  max_val = 10,  
  overwrite = TRUE,  
  seed  
)
```

**Arguments**

data	A dataset
name	Name of new variable (as string)
min_val	Minimum random integers
max_val	Maximum random integers
overwrite	Can new random variable overwrite an existing variable in dataset?
seed	Seed for random number generation (integer)

**Value**

Dataset containing new random variable

**Examples**

```
library(magrittr)
iris %>% add_var_random_int() %>% head()
iris %>% add_var_random_int(name = "random_var") %>% head()
iris %>% add_var_random_int(min_val = 1, max_val = 10) %>% head()
```

---

add\_var\_random\_moon    *Add a random moon variable to dataset*

---

**Description**

Add a random moon variable to dataset

**Usage**

```
add_var_random_moon(data, name = "random_moon", overwrite = TRUE, seed)
```

**Arguments**

data	A dataset
name	Name of new variable (as string)
overwrite	Can new random variable overwrite an existing variable in dataset?
seed	Seed for random number generation (integer)

**Value**

Dataset containing new random variable

**Examples**

```
library(magrittr)
iris %>% add_var_random_moon() %>% head()
```

---

`add_var_random_starsign`*Add a random starsign variable to dataset*

---

### Description

Add a random starsign variable to dataset

### Usage

```
add_var_random_starsign(  
  data,  
  name = "random_starsign",  
  lang = "en",  
  overwrite = TRUE,  
  seed  
)
```

### Arguments

<code>data</code>	A dataset
<code>name</code>	Name of new variable (as string)
<code>lang</code>	Language used for starsign (en = English, de = Deutsch, es = Espanol)
<code>overwrite</code>	Can new random variable overwrite an existing variable in dataset?
<code>seed</code>	Seed for random number generation (integer)

### Value

Dataset containing new random variable

### Examples

```
library(magrittr)  
iris %>% add_var_random_starsign() %>% head()  
iris %>% add_var_random_starsign(lang = "de") %>% head()
```

---

balance_target	<i>Balance target variable</i>
----------------	--------------------------------

---

**Description**

Balances the target variable in your dataset using downsampling. Target must be 0/1, FALSE/TRUE or no/yes

**Usage**

```
balance_target(data, target, min_prop = 0.1, seed)
```

**Arguments**

data	A dataset
target	Target variable (0/1, TRUE/FALSE, yes/no)
min_prop	Minimum proportion of one of the target categories
seed	Seed for random number generator

**Value**

Data

**Examples**

```
iris$is_versicolor <- ifelse(iris$Species == "versicolor", 1, 0)
balanced <- balance_target(iris, target = is_versicolor, min_prop = 0.5)
describe(balanced, is_versicolor)
```

---

check_vec_low_variance	<i>Check vector for low variance</i>
------------------------	--------------------------------------

---

**Description**

Check vector for low variance

**Usage**

```
check_vec_low_variance(values, max_prop = 0.99)
```

**Arguments**

values	Vector of values
max_prop	Maximum proportion of values without variance

**Value**

TRUE/FALSE (low variance)

**Examples**

```
## Not run:
values <- c(1, rep(0, 1000))
check_vec_low_variance(values, max_prop = 0.9)

## End(Not run)
```

---

clean_var	<i>Clean variable</i>
-----------	-----------------------

---

**Description**

Clean variable (replace NA values, set min\_val and max\_val)

**Usage**

```
clean_var(
  data,
  var,
  na = NA,
  min_val = NA,
  max_val = NA,
  max_cat = NA,
  rescale01 = FALSE,
  simplify_text = FALSE,
  name = NA
)
```

**Arguments**

data	A dataset
var	Name of variable
na	Value that replaces NA
min_val	All values < min_val are converted to min_val (var numeric or character)
max_val	All values > max_val are converted to max_val (var numeric or character)
max_cat	Maximum number of different factor levels for categorical variable (if more, .OTHER is added)
rescale01	IF TRUE, value is rescaled between 0 and 1 (var must be numeric)
simplify_text	If TRUE, a character variable is simplified (trim, upper, ...)
name	New name of variable (as string)

**Value**

Dataset

**Examples**

```
library(magrittr)
iris %>% clean_var(Sepal.Width, max_val = 3.5, name = "sepal_width") %>% head()
iris %>% clean_var(Sepal.Width, rescale01 = TRUE) %>% head()
```

---

count_pct	<i>Adds percentage to dplyr::count()</i>
-----------	--

---

**Description**

Adds variables total and pct (percentage) to dplyr::count()

**Usage**

```
count_pct(data, ...)
```

**Arguments**

data	A dataset
...	Other parameters passed to count()

**Value**

Dataset

**Examples**

```
count_pct(iris, Species)
```

---

create_data_abtest	<i>Create data of A/B testing</i>
--------------------	-----------------------------------

---

**Description**

Data that can be used for unit-testing or teaching

**Usage**

```
create_data_abtest(  
  n_a = 100,  
  n_b = 100,  
  success_a = 10,  
  success_b = 5,  
  success_unit = "count",  
  count = TRUE  
)
```

**Arguments**

n_a	Total size of group A
n_b	Total size of group B
success_a	Success in group A
success_b	Success in group B
success_unit	Unit ("count" "percent")
count	Create as count-data (FALSE TRUE)

**Value**

A dataset as tibble

**Examples**

```
library(dplyr)  
create_data_abtest() %>% abtest()  
create_data_abtest(  
  n_a = 100,  
  n_b = 100,  
  success_a = 20,  
  success_b = 30,  
  success_unit = "count"  
) %>% abtest()
```

---

create\_data\_app

*Create data app*

---

**Description**

Artificial data that can be used for unit-testing or teaching

**Usage**

```
create_data_app(obs = 1000, add_id = FALSE, seed = 123)
```

**Arguments**

obs	Number of observations
add_id	Add an id-variable to data?
seed	Seed for randomization (integer)

**Value**

A dataset as tibble

**Examples**

```
create_data_app()
```

---

create_data_buy	<i>Create data buy</i>
-----------------	------------------------

---

**Description**

Artificial data that can be used for unit-testing or teaching

**Usage**

```
create_data_buy(  
  obs = 1000,  
  target_name = "buy",  
  factorise_target = FALSE,  
  target1_prob = 0.5,  
  add_extreme = TRUE,  
  flip_gender = FALSE,  
  add_id = FALSE,  
  seed = 123  
)
```

**Arguments**

obs	Number of observations
target_name	Variable name of target
factorise_target	Should target variable be factorised? (from 0/1 to factor no/yes)?
target1_prob	Probability that target = 1
add_extreme	Add an observation with extreme values?
flip_gender	Should Male/Female be flipped in data?
add_id	Add an id-variable to data?
seed	Seed for randomization

## Details

Variables in dataset:

- id = Identifier
- period = Year & Month (YYYYMM)
- city\_ind = Indicating if customer is residing in a city (1 = yes, 0 = no)
- female\_ind = Gender of customer is female (1 = yes, 0 = no)
- fixedvoice\_ind = Customer has a fixed voice product (1 = yes, 0 = no)
- fixeddata\_ind = Customer has a fixed data product (1 = yes, 0 = no)
- fixedtv\_ind = Customer has a fixed TV product (1 = yes, 0 = no)
- mobilevoice\_ind = Customer has a mobile voice product (1 = yes, 0 = no)
- mobiledata\_prd = Customer has a mobile data product (NO/MOBILE STICK/BUSINESS)
- bbi\_speed\_ind = Customer has a Broadband Internet (BBI) with extra speed
- bbi\_usg\_gb = Broadband Internet (BBI) usage in Gigabyte (GB) last month
- hh\_single = Expected to be a Single Household (1 = yes, 0 = no)

Target in dataset:

- buy (may be renamed) = Did customer buy a new product in next month? (1 = yes, 0 = no)

## Value

A dataset as tibble

## Examples

```
create_data_buy()
```

---

```
create_data_churn      Create data churn
```

---

## Description

Artificial data that can be used for unit-testing or teaching

## Usage

```
create_data_churn(  
  obs = 1000,  
  target_name = "churn",  
  factorise_target = FALSE,  
  target1_prob = 0.4,  
  add_id = FALSE,  
  seed = 123  
)
```

**Arguments**

obs	Number of observations
target_name	Variable name of target
factorise_target	Should target variable be factorised?
target1_prob	Probability that target = 1
add_id	Add an id-variable to data?
seed	Seed for randomization (integer)

**Value**

A dataset as tibble

**Examples**

```
create_data_churn()
```

---

create_data_empty	<i>Create an empty dataset</i>
-------------------	--------------------------------

---

**Description**

Create an empty dataset

**Usage**

```
create_data_empty(obs = 1000, add_id = FALSE)
```

**Arguments**

obs	Number of observations
add_id	Add an id

**Value**

Dataset as tibble

**Examples**

```
create_data_empty(obs = 100)  
create_data_empty(obs = 100, add_id = TRUE)
```

---

create\_data\_esoteric *Create data esoteric*

---

### Description

Random data that can be used for unit-testing or teaching

### Usage

```
create_data_esoteric(obs = 1000, add_id = FALSE, seed = 123)
```

### Arguments

obs	Number of observations
add_id	Add an id-variable to data?
seed	Seed for randomization

### Details

Variables in dataset:

- id = Identifier
- starsign = random starsign
- chinese = random chinese zodiac
- moon = random moon phase
- blood = random blood type
- fingers\_crossed = random fingers crossed (1 = yes, 0 = no)
- success = random success (1 = yes, 0 = no)

### Value

A dataset as tibble

### Examples

```
create_data_esoteric(obs = 100)
```

---

create\_data\_newsletter  
*Create data newsletter*

---

**Description**

Artificial data that can be used for unit-testing or teaching (fairness & AI bias)

**Usage**

```
create_data_newsletter(obs = 1000, add_id = FALSE, seed = 123)
```

**Arguments**

obs	Number of observations
add_id	Add an id-variable to data?
seed	Seed for randomization (integer)

**Value**

A dataset as tibble

**Examples**

```
create_data_newsletter()
```

---

create\_data\_person      *Create data person*

---

**Description**

Artificial data that can be used for unit-testing or teaching

**Usage**

```
create_data_person(obs = 1000, add_id = FALSE, seed = 123)
```

**Arguments**

obs	Number of observations
add_id	Add an id
seed	Seed for randomization (integer)

**Value**

A dataset as tibble

## Examples

```
create_data_person()
```

---

```
create_data_random
```

*Create data random*

---

## Description

Random data that can be used for unit-testing or teaching

## Usage

```
create_data_random(  
  obs = 1000,  
  vars = 10,  
  target_name = "target_ind",  
  factorise_target = FALSE,  
  target1_prob = 0.5,  
  add_id = TRUE,  
  seed = 123  
)
```

## Arguments

obs	Number of observations
vars	Number of variables
target_name	Variable name of target
factorise_target	Should target variable be factorised? (from 0/1 to facotr no/yes)?
target1_prob	Probability that target = 1
add_id	Add an id-variable to data?
seed	Seed for randomization

## Details

Variables in dataset:

- id = Identifier
- var\_X = variable containing values between 0 and 100

Target in dataset:

- target\_ind (may be renamed) = random values (1 = yes, 0 = no)

## Value

A dataset as tibble

**Examples**

```
create_data_random(obs = 100, vars = 5)
```

---

```
create_data_unfair      Create data unfair
```

---

**Description**

Artificial data that can be used for unit-testing or teaching (fairness & AI bias)

**Usage**

```
create_data_unfair(  
  obs = 1000,  
  target_name = "target_ind",  
  factorise_target = FALSE,  
  target1_prob = 0.25,  
  add_id = FALSE,  
  seed = 123  
)
```

**Arguments**

obs	Number of observations
target_name	Variable name of target
factorise_target	Should target variable be factorised?
target1_prob	Probability that target = 1
add_id	Add an id-variable to data?
seed	Seed for randomization (integer)

**Value**

A dataset as tibble

**Examples**

```
create_data_unfair()
```

---

`create_notebook_explore`*Generate a notebook*

---

**Description**

Generate an RMarkdown Notebook template for a report. You must provide a output-directory (parameter `output_dir`). The default file-name is "notebook-explore.Rmd" (may overwrite existing file with same name)

**Usage**

```
create_notebook_explore(output_file = "notebook-explore.Rmd", output_dir)
```

**Arguments**

<code>output_file</code>	Filename of the html report
<code>output_dir</code>	Directory where to save the html report

**Examples**

```
create_notebook_explore(output_file = "explore.Rmd", output_dir = tempdir())
```

---

`cut_vec_num_avg`*Cut a variable*

---

**Description**

Cut a variable

**Usage**

```
cut_vec_num_avg(values, bins = 8)
```

**Arguments**

<code>values</code>	Variable
<code>bins</code>	Number of bins

**Value**

Data frame

---

data_dict_md	<i>Create a data dictionary Markdown file</i>
--------------	---

---

## Description

Create a data dictionary Markdown file

## Usage

```
data_dict_md(  
  data,  
  title = "",  
  description = NA,  
  output_file = "data_dict.md",  
  output_dir  
)
```

## Arguments

data	A dataframe (data dictionary for all variables)
title	Title of the data dictionary
description	Detailed description of variables in data (dataframe with columns 'variable' and 'description')
output_file	Output filename for Markdown file
output_dir	Directory where the Markdown file is saved

## Value

Create Markdown file

## Examples

```
# Data dictionary of a dataframe  
data_dict_md(iris,  
             title = "iris flower data set",  
             output_dir = tempdir())  
  
# Data dictionary of a dataframe with additional description of variables  
description <- data.frame(  
  variable = c("Species"),  
  description = c("Species of Iris flower"))  
data_dict_md(iris,  
             title = "iris flower data set",  
             description = description,  
             output_dir = tempdir())
```

---

decrypt	<i>decrypt text</i>
---------	---------------------

---

**Description**

decrypt text

**Usage**

```
decrypt(text, codeletters = c(toupper(letters), letters, 0:9), shift = 18)
```

**Arguments**

text	A text (character)
codeletters	A string of letters that are used for decryption
shift	Number of elements shifted

**Value**

Decrypted text

**Examples**

```
decrypt("zw336 E693v")
```

---

describe	<i>Describe a dataset or variable</i>
----------	---------------------------------------

---

**Description**

Describe a dataset or variable (depending on input parameters)

**Usage**

```
describe(data, var, n, target, out = "text", ...)
```

**Arguments**

data	A dataset
var	A variable of the dataset
n	Weights variable for count-data
target	Target variable (0/1 or FALSE/TRUE)
out	Output format ("text" "list") of variable description
...	Further arguments

**Value**

Description as table, text or list

**Examples**

```
# Load package
library(magrittr)

# Describe a dataset
iris %>% describe()

# Describe a variable
iris %>% describe(Species)
iris %>% describe(Sepal.Length)
```

---

describe_all	<i>Describe all variables of a dataset</i>
--------------	--

---

**Description**

Describe all variables of a dataset

**Usage**

```
describe_all(data, out = "large")
```

**Arguments**

data	A dataset
out	Output format ("small" "large")

**Value**

Dataset (tibble)

**Examples**

```
describe_all(iris)
```

---

describe\_cat                    *Describe categorical variable*

---

**Description**

Describe categorical variable

**Usage**

```
describe_cat(data, var, n, max_cat = 10, out = "text", margin = 0)
```

**Arguments**

data	A dataset
var	Variable or variable name
n	Weights variable for count-data
max_cat	Maximum number of categories displayed
out	Output format ("text" "list" "tibble" "df")
margin	Left margin for text output (number of spaces)

**Value**

Description as text or list

**Examples**

```
describe_cat(iris, Species)
```

---

describe\_num                    *Describe numerical variable*

---

**Description**

Describe numerical variable

**Usage**

```
describe_num(data, var, n, out = "text", margin = 0)
```

**Arguments**

data	A dataset
var	Variable or variable name
n	Weights variable for count-data
out	Output format ("text" "list")
margin	Left margin for text output (number of spaces)

**Value**

Description as text or list

**Examples**

```
describe_num(iris, Sepal.Length)
```

---

describe_tbl	<i>Describe table</i>
--------------	-----------------------

---

**Description**

Describe table (e.g. number of rows and columns of dataset)

**Usage**

```
describe_tbl(data, n, target, out = "text")
```

**Arguments**

data	A dataset
n	Weights variable for count-data
target	Target variable (binary)
out	Output format ("text" "list")

**Value**

Description as text or list

**Examples**

```
describe_tbl(iris)

iris[1,1] <- NA
describe_tbl(iris)
```

---

drop_obs_if	<i>Drop all observations where expression is true</i>
-------------	---

---

**Description**

Drop all observations where expression is true

**Usage**

```
drop_obs_if(data, expr)
```

**Arguments**

data	Data frame
expr	Expression

**Value**

Data frame

**Examples**

```
drop_obs_if(iris, Species == "setosa")
drop_obs_if(iris, Sepal.Length < 5 | Sepal.Length >7)
```

---

drop_obs_with_na	<i>Drop all observations with NA-values</i>
------------------	---

---

**Description**

Drop all observations with NA-values

**Usage**

```
drop_obs_with_na(data)
```

**Arguments**

data	Data frame
------	------------

**Value**

Data frame

**Examples**

```
data <- data.frame(a = 1:10, b = rep("A",10))
data[1,1] <- NA
drop_obs_with_na(data)
```

---

drop\_var\_by\_names      *Drop variables by name*

---

**Description**

Drop variables by name

**Usage**

```
drop_var_by_names(data, var_names)
```

**Arguments**

data	Data frame
var_names	Vector of variable names (as string)

**Value**

Data frame

**Examples**

```
drop_var_by_names(iris, "Species")  
drop_var_by_names(iris, c("Sepal.Length", "Sepal.Width"))
```

---

drop\_var\_low\_variance      *Drop all variables with low variance*

---

**Description**

Drop all variables with low variance

**Usage**

```
drop_var_low_variance(data, max_prop = 0.99)
```

**Arguments**

data	Data frame
max_prop	Maximum proportion of values without variance

**Value**

Data frame

**Examples**

```
data <- data.frame(a = 1:100, b = c(0, rep(1, 99)))
drop_var_low_variance(data, max_prop = 0.9)
```

---

drop\_var\_not\_numeric *Drop all not numeric variables*

---

**Description**

Drop all not numeric variables

**Usage**

```
drop_var_not_numeric(data)
```

**Arguments**

data                    Data frame

**Value**

Data frame

**Examples**

```
data <- data.frame(a = 1:10, b = rep("A",10))
drop_var_not_numeric(data)
```

---

drop\_var\_no\_variance *Drop all variables with no variance*

---

**Description**

Drop all variables with no variance

**Usage**

```
drop_var_no_variance(data)
```

**Arguments**

data                    Data frame

**Value**

Data frame

**Examples**

```
data <- data.frame(a = 1:10, b = rep(1,10))
drop_var_no_variance(data)
```

---

drop_var_with_na	<i>Drop all variables with NA-values</i>
------------------	--

---

**Description**

Drop all variables with NA-values

**Usage**

```
drop_var_with_na(data)
```

**Arguments**

data	Data frame
------	------------

**Value**

Data frame

**Examples**

```
data <- data.frame(a = 1:10, b = rep(NA,10))
drop_var_with_na(data)
```

---

encrypt	<i>encrypt text</i>
---------	---------------------

---

**Description**

encrypt text

**Usage**

```
encrypt(text, codeletters = c(toupper(letters), letters, 0:9), shift = 18)
```

**Arguments**

text	A text (character)
codeletters	A string of letters that are used for encryption
shift	Number of elements shifted

**Value**

Encrypted text

**Examples**

```
encrypt("hello world")
```

---

explain_forest	<i>Explain a target using Random Forest.</i>
----------------	--

---

**Description**

Explain a target using Random Forest.

**Usage**

```
explain_forest(data, target, ntree = 50, out = "plot", ...)
```

**Arguments**

data	A dataset
target	Target variable (binary)
ntree	Number of trees used for Random Forest
out	Output of the function: "plot"   "model"   "importance"   all"
...	Further arguments

**Value**

Plot of importance (if out = "plot")

**Examples**

```
data <- create_data_buy()
explain_forest(data, target = buy)
```

---

explain_logreg	<i>Explain a binary target using a logistic regression (glm). Model chosen by AIC in a Stepwise Algorithm (MASS::stepAIC()).</i>
----------------	--

---

**Description**

Explain a binary target using a logistic regression (glm). Model chosen by AIC in a Stepwise Algorithm (MASS::stepAIC()).

**Usage**

```
explain_logreg(data, target, out = "tibble", ...)
```

**Arguments**

data	A dataset
target	Target variable (binary)
out	Output of the function: "tibble"   "model"
...	Further arguments

**Value**

Dataset with results (term, estimate, std.error, z.value, p.value)

**Examples**

```
data <- iris
data$is_versicolor <- ifelse(iris$Species == "versicolor", 1, 0)
data$Species <- NULL
explain_logreg(data, target = is_versicolor)
```

---

explain_tree	<i>Explain a target using a simple decision tree (classification or regression)</i>
--------------	---

---

**Description**

Explain a target using a simple decision tree (classification or regression)

**Usage**

```

explain_tree(
  data,
  target,
  n,
  max_cat = 10,
  max_target_cat = 5,
  maxdepth = 3,
  minsplit = 20,
  cp = 0,
  weights = NA,
  size = 0.7,
  out = "plot",
  ...
)

```

**Arguments**

<code>data</code>	A dataset
<code>target</code>	Target variable
<code>n</code>	weights variable (for count data)
<code>max_cat</code>	Drop categorical variables with higher number of levels
<code>max_target_cat</code>	Maximum number of categories to be plotted for target (except NA)
<code>maxdepth</code>	Set the maximum depth of any node of the final tree, with the root node counted as depth 0. Maximum value 30 (as node numbers have to be stored in 32-bit signed integer types).
<code>minsplit</code>	the minimum number of observations that must exist in a node in order for a split to be attempted.
<code>cp</code>	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of <code>cp</code> is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by <code>cp</code> at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially, the user informs the program that any split which does not improve the fit by <code>cp</code> will likely be pruned off by cross-validation, and that hence the program need not pursue it.
<code>weights</code>	optional case weights.
<code>size</code>	Text size of plot
<code>out</code>	Output of function: "plot"   "model"
<code>...</code>	Further arguments

**Value**

Plot or additional the model (if `out = "model"`)

**Examples**

```
data <- iris
data$is_versicolor <- ifelse(iris$Species == "versicolor", 1, 0)
data$Species <- NULL
explain_tree(data, target = is_versicolor)
```

---

explain_xgboost	<i>Explain a binary target using xgboost</i>
-----------------	--

---

**Description**

Based on the hyperparameters defined in the setup parameter, XGBoost hyperparameter-tuning is carried out using cross-validation. The best model is chosen and returned. As default, the function returns the feature-importance plot. To get the all outputs, use parameter out = "all"

**Usage**

```
explain_xgboost(
  data,
  target,
  log = TRUE,
  nthread = 1,
  setup = list(cv_nfold = 2, max_nrounds = 1000, early_stopping_rounds = 50, grid_xgboost
    = list(eta = c(0.3, 0.1, 0.01), max_depth = c(3, 5), gamma = 0, colsample_bytree =
      0.8, subsample = 0.8, min_child_weight = 1, scale_pos_weight = 1)),
  out = "plot"
)
```

**Arguments**

data	Data frame, must contain variable defined in target, but should not contain any customer-IDs or date/period columns
target	Target variable (must be binary 0/1, FALSE/TRUE, no/yes)
log	Log?
nthread	Number of threads used for training
setup	Setup of model
out	Output of the function: "plot"   "model"   "importance"   all"

**Value**

Plot of importance (if out = "plot")

**Examples**

```
data <- use_data_iris()
data$is_versicolor <- ifelse(data$Species == "versicolor", 1, 0)
data$Species <- NULL
explain_xgboost(data, target = is_versicolor, log = FALSE)
```

---

explore

*Explore a dataset or variable*

---

### Description

Explore a dataset or variable

### Usage

```
explore(  
  data,  
  var,  
  var2,  
  n,  
  target,  
  targetpct,  
  split,  
  min_val = NA,  
  max_val = NA,  
  auto_scale = TRUE,  
  na = NA,  
  ...  
)
```

### Arguments

data	A dataset
var	A variable
var2	A variable for checking correlation
n	A Variable for number of observations (count data)
target	Target variable (0/1 or FALSE/TRUE)
targetpct	Plot variable as target% (FALSE/TRUE)
split	Alternative to targetpct (split = !targetpct)
min_val	All values < min_val are converted to min_val
max_val	All values > max_val are converted to max_val
auto_scale	Use 0.2 and 0.98 quantile for min_val and max_val (if min_val and max_val are not defined)
na	Value to replace NA
...	Further arguments (like flip = TRUE/FALSE)

### Value

Plot object

**Examples**

```
## Launch Shiny app (in interactive R sessions)
if (interactive()) {
  explore(iris)
}

## Explore grafically

# Load library
library(magrittr)

# Explore a variable
iris %>% explore(Species)
iris %>% explore(Sepal.Length)
iris %>% explore(Sepal.Length, min_val = 4, max_val = 7)

# Explore a variable with a target
iris$is_virginica <- ifelse(iris$Species == "virginica", 1, 0)
iris %>% explore(Species, target = is_virginica)
iris %>% explore(Sepal.Length, target = is_virginica)

# Explore correlation between two variables
iris %>% explore(Species, Petal.Length)
iris %>% explore(Sepal.Length, Petal.Length)

# Explore correlation between two variables and split by target
iris %>% explore(Sepal.Length, Petal.Length, target = is_virginica)
```

---

explore\_all

*Explore all variables*

---

**Description**

Explore all variables of a dataset (create plots)

**Usage**

```
explore_all(
  data,
  n,
  target,
  ncol = 2,
  targetpct,
  color = c("#ADD8E6", "#7BB8DA"),
  split = TRUE
)
```

**Arguments**

data	A dataset
n	Weights variable (only for count data)
target	Target variable (0/1 or FALSE/TRUE)
ncol	Layout of plots (number of columns)
targetpct	Plot variable as target% (FALSE/TRUE)
color	Force a default color (if possible)
split	Split by target (TRUE FALSE)

**Value**

Plot

**Examples**

```
explore_all(iris)

iris$is_virginica <- ifelse(iris$Species == "virginica", 1, 0)
explore_all(iris, target = is_virginica)
```

---

explore\_bar

*Explore categorical variable using bar charts*

---

**Description**

Create a barplot to explore a categorical variable. If a target is selected, the barplot is created for all levels of the target.

**Usage**

```
explore_bar(
  data,
  var,
  target,
  flip = NA,
  title = "",
  numeric = NA,
  max_cat = 30,
  max_target_cat = 5,
  color = c("#ADD8E6", "#7BB8DA"),
  legend_position = "right",
  label,
  label_size = 2.7,
  ...
)
```

**Arguments**

data	A dataset
var	variable
target	target (can have more than 2 levels)
flip	Should plot be flipped? (change of x and y)
title	Title of the plot (if empty var name)
numeric	Display variable as numeric (not category)
max_cat	Maximum number of categories to be plotted
max_target_cat	Maximum number of categories to be plotted for target (except NA)
color	Color for bar
legend_position	Position of the legend ("bottom" "top" "none")
label	Show labels? (if empty, automatic)
label_size	Size of labels
...	Further arguments

**Value**

Plot object (bar chart)

---

explore_col	<i>Explore data without aggregation (label + value)</i>
-------------	---

---

**Description**

Label and Value are in the data. Create a bar plot where the heights of the bars represent the values for each label.

**Usage**

```
explore_col(
  data,
  var_label,
  var_value,
  title = NA,
  subtitle = "",
  numeric = FALSE,
  max_cat = 30,
  na = 0,
  flip = NA,
  color = "#ADD8E6"
)
```

**Arguments**

<code>data</code>	A dataset (categories + frequency)
<code>var_label</code>	Variable containing the label
<code>var_value</code>	Variable containing the value
<code>title</code>	Title of the plot
<code>subtitle</code>	Subtitle of the plot
<code>numeric</code>	Display variable as numeric (not category)
<code>max_cat</code>	Maximum number of categories to be plotted
<code>na</code>	Value to use for NA
<code>flip</code>	Flip plot? (for categorical variables)
<code>color</code>	Color for bar

**Value**

Plot object

**Examples**

```
library(magrittr)
data <- data.frame(label = LETTERS[1:5], value = c(1.5,2,1.2,3,2.6))
data %>% explore_col(label, value)
```

---

`explore_cor`

*Explore the correlation between two variables*

---

**Description**

Explore the correlation between two variables

**Usage**

```
explore_cor(
  data,
  x,
  y,
  target,
  bins = 8,
  min_val = NA,
  max_val = NA,
  auto_scale = TRUE,
  title = NA,
  color = c("#ADD8E6", "#7BB8DA"),
  ...
)
```

**Arguments**

data	A dataset
x	Variable on x axis
y	Variable on y axis
target	Target variable (categorical)
bins	Number of bins
min_val	All values < min_val are converted to min_val
max_val	All values > max_val are converted to max_val
auto_scale	Use 0.2 and 0.98 quantile for min_val and max_val (if min_val and max_val are not defined)
title	Title of the plot
color	Color of the plot
...	Further arguments

**Value**

Plot

**Examples**

```
explore_cor(iris, x = Sepal.Length, y = Sepal.Width)
```

---

explore_count	<i>Explore count data (categories + frequency)</i>
---------------	--

---

**Description**

Create a plot to explore count data (categories + frequency) Variable named 'n' is auto detected as Frequency

**Usage**

```
explore_count(
  data,
  cat,
  n,
  target,
  pct = FALSE,
  split = TRUE,
  title = NA,
  numeric = FALSE,
  max_cat = 30,
  max_target_cat = 5,
  color = c("#ADD8E6", "#7BB8DA"),
  flip = NA
)
```

**Arguments**

data	A dataset (categories + frequency)
cat	Numerical variable
n	Number of observations (frequency)
target	Target variable
pct	Show as percent?
split	Split by target (FALSE/TRUE)
title	Title of the plot
numeric	Display variable as numeric (not category)
max_cat	Maximum number of categories to be plotted
max_target_cat	Maximum number of categories to be plotted for target (except NA)
color	Color for bar
flip	Flip plot? (for categorical variables)

**Value**

Plot object

**Examples**

```
library(dplyr)
iris %>%
  count(Species) %>%
  explore_count(Species)
```

---

explore_density	<i>Explore density of variable</i>
-----------------	------------------------------------

---

**Description**

Create a density plot to explore numerical variable

**Usage**

```
explore_density(
  data,
  var,
  target,
  title = "",
  min_val = NA,
  max_val = NA,
  color = c("#ADD8E6", "#7BB8DA"),
  auto_scale = TRUE,
  max_target_cat = 5,
  ...
)
```

**Arguments**

data	A dataset
var	Variable
target	Target variable (0/1 or FALSE/TRUE)
title	Title of the plot (if empty var name)
min_val	All values < min_val are converted to min_val
max_val	All values > max_val are converted to max_val
color	Color of plot
auto_scale	Use 0.02 and 0.98 percent quantile for min_val and max_val (if min_val and max_val are not defined)
max_target_cat	Maximum number of levels of target shown in the plot (except NA).
...	Further arguments

**Value**

Plot object (density plot)

**Examples**

```
explore_density(iris, Sepal.Length)
iris$is_virginica <- ifelse(iris$Species == "virginica", 1, 0)
explore_density(iris, Sepal.Length, target = is_virginica)
```

---

explore\_shiny

*Explore dataset interactive*

---

**Description**

Launches a shiny app to explore a dataset

**Usage**

```
explore_shiny(data, target, color = c("#ADD8E6", "#7BB8DA"))
```

**Arguments**

data	A dataset
target	Target variable (0/1 or FALSE/TRUE)
color	Color for plots (vector)

**Examples**

```
# Only run examples in interactive R sessions
if (interactive()) {
  explore_shiny(iris)
}
```

---

explore\_targetpct      *Explore variable + binary target (values 0/1)*

---

### Description

Create a plot to explore relation between a variable and a binary target as target percent. The target variable is chosen automatically if possible (name starts with 'target')

### Usage

```
explore_targetpct(
  data,
  var,
  target = NULL,
  title = NA,
  min_val = NA,
  max_val = NA,
  auto_scale = TRUE,
  na = NA,
  flip = NA,
  ...
)
```

### Arguments

data	A dataset
var	Numerical variable
target	Target variable (0/1 or FALSE/TRUE)
title	Title of the plot
min_val	All values < min_val are converted to min_val
max_val	All values > max_val are converted to max_val
auto_scale	Use 0.2 and 0.98 quantile for min_val and max_val (if min_val and max_val are not defined)
na	Value to replace NA
flip	Flip plot? (for categorical variables)
...	Further arguments

### Value

Plot object

### Examples

```
iris$target01 <- ifelse(iris$Species == "versicolor",1,0)
explore_targetpct(iris)
```

---

explore_tbl	<i>Explore table</i>
-------------	----------------------

---

**Description**

Explore a table. Plots variable types, variables with no variance and variables with NA

**Usage**

```
explore_tbl(data, n)
```

**Arguments**

data	A dataset
n	Weight variable for count data

**Examples**

```
explore_tbl(iris)
```

---

format_num_auto	<i>Format number as character string (auto)</i>
-----------------	---

---

**Description**

Formats a number depending on the value as number with space, scientific or big number as k (1 000), M (1 000 000) or B (1 000 000 000)

**Usage**

```
format_num_auto(number = 0, digits = 1)
```

**Arguments**

number	A number (integer or real)
digits	Number of digits

**Value**

Formatted number as text

**Examples**

```
format_num_kMB(5500, digits = 2)
```

format\_num\_kMB      *Format number as character string (kMB)*

---

**Description**

Formats a big number as k (1 000), M (1 000 000) or B (1 000 000 000)

**Usage**

```
format_num_kMB(number = 0, digits = 1)
```

**Arguments**

number	A number (integer or real)
digits	Number of digits

**Value**

Formatted number as text

**Examples**

```
format_num_kMB(5500, digits = 2)
```

---

format\_num\_space      *Format number as character string (space as big.mark)*

---

**Description**

Formats a big number using space as big.mark (1000 = 1 000)

**Usage**

```
format_num_space(number = 0, digits = 1)
```

**Arguments**

number	A number (integer or real)
digits	Number of digits

**Value**

Formatted number as text

**Examples**

```
format_num_space(5500, digits = 2)
```

---

format_target	<i>Format target</i>
---------------	----------------------

---

**Description**

Formats a target as a 0/1 variable. If target is numeric, 1 = above average.

**Usage**

```
format_target(target)
```

**Arguments**

target            Variable as vector

**Value**

Formatted target

**Examples**

```
iris$is_virginica <- ifelse(iris$Species == "virginica", "yes", "no")
iris$target <- format_target(iris$is_virginica)
table(iris$target)
```

---

format_type	<i>Format type description</i>
-------------	--------------------------------

---

**Description**

Format type description of variable to 3 letters (intdblglchrdat)

**Usage**

```
format_type(type)
```

**Arguments**

type            Type description ("integer", "double", "logical", "character", "date")

**Value**

Formatted type description (intdblglchrdat)

**Examples**

```
format_type(typeof(iris$Species))
```

---

get_color	<i>Get predefined colors</i>
-----------	------------------------------

---

**Description**

Get predefined colors

**Usage**

```
get_color(name, fill = FALSE, fill_color = "#DDDDDD", fill_n = 10)
```

**Arguments**

name	Name of color/color-vector
fill	Fill color vector?
fill_color	Color to use to fill color vector
fill_n	Number of color codes to return

**Value**

Vector of color-codes

**Examples**

```
get_color("mario")

get_color("mario")
show_color(get_color("mario"))
show_color(get_color("mario", fill = TRUE, fill_n = 10))

col <- get_color("mario")
explore(iris, Sepal.Length, target = Species,
        color = col)
explore(iris, Sepal.Length, target = Species,
        color = c(col["peach"], col["bowser"], col["donkeykong"]))
```

---

get_type	<i>Return type of variable</i>
----------	--------------------------------

---

**Description**

Return value of typeof, except if variable contains hide, then return "other"

**Usage**

```
get_type(var)
```

**Arguments**

var                    A vector (dataframe column)

**Value**

Value of typeof or "other"

**Examples**

```
get_type(iris$Species)
```

---

get_var_buckets	<i>Put variables into "buckets" to create a set of plots instead one large plot</i>
-----------------	---

---

**Description**

Put variables into "buckets" to create a set of plots instead one large plot

**Usage**

```
get_var_buckets(data, bucket_size = 100, var_name_target = NA, var_name_n = NA)
```

**Arguments**

data                    A dataset

bucket\_size            Maximum number of variables in one bucket

var\_name\_target        Name of the target variable (if defined)

var\_name\_n             Name of the weight (n) variable (if defined)

**Value**

Buckets as a list

**Examples**

```
get_var_buckets(iris)
get_var_buckets(iris, bucket_size = 2)
get_var_buckets(iris, bucket_size = 2, var_name_target = "Species")
```

---

guess_cat_num	<i>Return if variable is categorical or numerical</i>
---------------	---

---

**Description**

Guess if variable is categorical or numerical based on name, type and values of variable

**Usage**

```
guess_cat_num(var, descr)
```

**Arguments**

var	A vector (dataframe column)
descr	A description of the variable (optional)

**Value**

"cat" (categorical), "num" (numerical) or "oth" (other)

**Examples**

```
guess_cat_num(iris$Species)
```

---

interact	<i>Make a explore-plot interactive</i>
----------	--

---

**Description**

Make a explore-plot interactive

**Usage**

```
interact(obj, lower_title = TRUE, hide_geom_text = TRUE)
```

**Arguments**

obj	A object (e.g. ggplot2-object)
lower_title	Lowering the title in ggplot2-object(FALSE/TRUE)
hide_geom_text	Hiding geom_text in ggplot2-object (FALSE/TRUE)

**Value**

Plot object

**Examples**

```
library(dplyr)
if (interactive()) {
  iris %>% explore(Sepal.Length, target = Species) %>% interact()
}
```

---

log_info_if	<i>Log conditional</i>
-------------	------------------------

---

**Description**

Log conditional

**Usage**

```
log_info_if(log = TRUE, text = "log")
```

**Arguments**

log	log (TRUE FALSE)
text	text string to be logged

**Value**

prints log on screen (if log == TRUE).

---

mix_color	<i>Mix colors</i>
-----------	-------------------

---

**Description**

Mix colors

**Usage**

```
mix_color(color1, color2 = NA, n = 5)
```

**Arguments**

color1	Color 1
color2	Color 2
n	Number of different colors that should be generated

**Value**

Vector of color-codes

**Examples**

```

mix_color("blue", n = 10)
mix_color("gold", "red", n = 4)

```

---

plot\_legend\_targetpct *Plots a legend that can be used for explore\_all with a binary target*

---

**Description**

Plots a legend that can be used for explore\_all with a binary target

**Usage**

```
plot_legend_targetpct(border = TRUE)
```

**Arguments**

border            Draw a border?

**Value**

Base plot

**Examples**

```
plot_legend_targetpct(border = TRUE)
```

---

plot\_text            *Plot a text*

---

**Description**

Plots a text (base plot) and let you choose text-size and color

**Usage**

```
plot_text(text = "hello world", size = 1.2, color = "black", ggplot = FALSE)
```

**Arguments**

text            Text as string  
size            Text-size  
color           Text-color  
ggplot          return a ggplot-object? (or base plot)

**Value**

Plot

**Examples**

```
plot_text("hello", size = 2, color = "red")
```

---

plot_var_info	<i>Plot a variable info</i>
---------------	-----------------------------

---

**Description**

Creates a ggplot with the variable-name as title and a text

**Usage**

```
plot_var_info(data, var, info = "")
```

**Arguments**

data	A dataset
var	Variable
info	Text to plot

**Value**

Plot (ggplot)

---

predict_target	<i>Predict target using a trained model.</i>
----------------	--

---

**Description**

Predict target using a trained model.

**Usage**

```
predict_target(data, model, name = "prediction")
```

**Arguments**

data	A dataset (data.frame or tbl)
model	A model created with explain_*() function
name	Prefix of variable-name for prediction

**Value**

data containing predicted probabilities for target values

**Examples**

```
data_train <- create_data_buy(seed = 1)
data_test <- create_data_buy(seed = 2)
model <- explain_tree(data_train, target = buy, out = "model")
data <- predict_target(data = data_test, model = model)
describe(data)
```

---

replace_na_with	<i>Replace NA</i>
-----------------	-------------------

---

**Description**

Replace NA values of a variable in a dataframe

**Usage**

```
replace_na_with(data, var_name, with)
```

**Arguments**

data	A dataframe
var_name	Name of variable where NAs are replaced
with	Value instead of NA

**Value**

Updated dataframe

**Examples**

```
data <- data.frame(nr = c(1,2,3,NA,NA))
replace_na_with(data, "nr", 0)
```

---

report	<i>Generate a report of all variables</i>
--------	---

---

**Description**

Generate a report of all variables. If target is defined, the relation to the target is reported.

**Usage**

```
report(data, n, target, targetpct, split, color, output_file, output_dir)
```

**Arguments**

data	A dataset
n	Weights variable for count data
target	Target variable (0/1 or FALSE/TRUE)
targetpct	Plot variable as target% (FALSE/TRUE)
split	Alternative to targetpct (split = !targetpct)
color	User defined colors for plots (vector)
output_file	Filename of the html report
output_dir	Directory where to save the html report

**Examples**

```
if (rmarkdown::pandoc_available("1.12.3")) {  
  report(iris, output_dir = tempdir())  
}
```

---

rescale01	<i>Rescales a numeric variable into values between 0 and 1</i>
-----------	--

---

**Description**

Rescales a numeric variable into values between 0 and 1.

**Usage**

```
rescale01(x)
```

**Arguments**

x	numeric vector (to be rescaled)
---	---------------------------------

**Value**

vector with values between 0 and 1

**Examples**

```
rescale01(0:10)
```

---

show_color	<i>Show color vector as ggplot</i>
------------	------------------------------------

---

**Description**

Show color vector as ggplot

**Usage**

```
show_color(color)
```

**Arguments**

color	Vector of colors
-------	------------------

**Value**

ggplot

**Examples**

```
show_color("gold")
show_color(c("blue", "red", "green"))
```

---

simplify_text	<i>Simplifies a text string</i>
---------------	---------------------------------

---

**Description**

A text string is converted into a simplified version by trimming, converting to upper case, replacing german Umlaute, dropping special characters like comma and semicolon and replacing multiple spaces with one space.

**Usage**

```
simplify_text(text)
```

**Arguments**

text	text string
------	-------------

**Value**

text string

**Examples**

```
simplify_text(" Hello World !, ")
```

---

target\_explore\_cat      *Explore categorical variable + target*

---

**Description**

Create a plot to explore relation between categorical variable and a binary target

**Usage**

```
target_explore_cat(
  data,
  var,
  target = "target_ind",
  min_val = NA,
  max_val = NA,
  flip = TRUE,
  num2char = TRUE,
  title = NA,
  auto_scale = TRUE,
  na = NA,
  max_cat = 25,
  color = c("#ECEFF1", "#CFD8DC", "#B0BEC5", "#90A4AE"),
  legend_position = "bottom"
)
```

**Arguments**

data	A dataset
var	Categorical variable
target	Target variable (0/1 or FALSE/TRUE)
min_val	All values < min_val are converted to min_val
max_val	All values > max_val are converted to max_val
flip	Should plot be flipped? (change of x and y)
num2char	If TRUE, numeric values in variable are converted into character
title	Title of plot
auto_scale	Not used, just for compatibility
na	Value to replace NA

max_cat	Maximum numbers of categories to be plotted
color	Color vector (4 colors)
legend_position	Position of legend ("right" "bottom" "non")

**Value**

Plot object

---

target\_explore\_num     *Explore Numerical variable + target*

---

**Description**

Create a plot to explore relation between numerical variable and a binary target

**Usage**

```
target_explore_num(
  data,
  var,
  target = "target_ind",
  min_val = NA,
  max_val = NA,
  bins = 10,
  flip = TRUE,
  title = NA,
  auto_scale = TRUE,
  na = NA,
  color = c("#ECEFF1", "#CFD8DC", "#B0BEC5", "#90A4AE"),
  legend_position = "bottom"
)
```

**Arguments**

data	A dataset
var	Numerical variable
target	Target variable (0/1 or FALSE/TRUE)
min_val	All values < min_val are converted to min_val
max_val	All values > max_val are converted to max_val
bins	Nuber of bins
flip	Should plot be flipped? (change of x and y)
title	Title of plot
auto_scale	Use 0.02 and 0.98 quantile for min_val and max_val (if min_val and max_val are not defined)

na	Value to replace NA
color	Color vector (4 colors)
legend_position	Position of legend ("right" "bottom" "non")

**Value**

Plot object

---

total_fig_height	<i>Get fig.height for RMarkdown-junk using explore_all()</i>
------------------	--

---

**Description**

Get fig.height for RMarkdown-junk using explore\_all()

**Usage**

```
total_fig_height(
  data,
  var_name_n,
  var_name_target,
  nvar = NA,
  ncol = 2,
  size = 3
)
```

**Arguments**

data	A dataset
var_name_n	Weights variable for count data? (TRUE / MISSING)
var_name_target	Target variable (TRUE / MISSING)
nvar	Number of variables to plot
ncol	Number of columns (default = 2)
size	fig.height of 1 plot (default = 3)

**Value**

Number of rows

**Examples**

```
total_fig_height(iris)
total_fig_height(iris, var_name_target = "Species")
total_fig_height(nvar = 5)
```

---

use_data_beer	<i>Use the beer data set</i>
---------------	------------------------------

---

**Description**

This data set is an incomplete collection of popular beers in Austria, Germany and Switzerland. Data are collected from various websites in 2023. Some of the collected data may be incorrect.

**Usage**

```
use_data_beer()
```

**Value**

Dataset as tibble

**Examples**

```
use_data_beer()
```

---

use_data_diamonds	<i>Use the diamonds data set</i>
-------------------	----------------------------------

---

**Description**

This data set comes with the `ggplot2` package. It contains the prices and other attributes of almost 54,000 diamonds.

**Usage**

```
use_data_diamonds()
```

**Value**

Dataset

**See Also**

[ggplot2::diamonds](#)

**Examples**

```
use_data_diamonds()
```

---

use_data_iris	<i>Use the iris flower data set</i>
---------------	-------------------------------------

---

**Description**

This data set comes with base R. The data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

**Usage**

```
use_data_iris()
```

**Value**

Dataset as tibble

**Examples**

```
use_data_iris()
```

---

use_data_mpg	<i>Use the mpg data set</i>
--------------	-----------------------------

---

**Description**

This data set comes with the `ggplot2` package. It contains a subset of the fuel economy data that the EPA makes available on <https://fueleconomy.gov/>. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

**Usage**

```
use_data_mpg()
```

**Value**

Dataset

**See Also**

[ggplot2::mpg](#)

**Examples**

```
use_data_mpg()
```

use\_data\_mtcars      *Use the mtcars data set*

---

**Description**

This data set comes with base R. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

**Usage**

```
use_data_mtcars()
```

**Value**

Dataset

**Examples**

```
use_data_mtcars()
```

---

use\_data\_penguins      *Use the penguins data set*

---

**Description**

This data set comes with the palmerpenguins package. It contains measurements for penguin species, island in Palmer Archipelago, size (flipper length, body mass, bill dimensions), and sex.

**Usage**

```
use_data_penguins(short_names = FALSE)
```

**Arguments**

short\_names      Use short variable names

**Value**

Dataset

**See Also**

[palmerpenguins::penguins](#)

**Examples**

```
use_data_penguins()  
use_data_penguins(short_names = TRUE)
```

---

use_data_starwars	<i>Use the starwars data set</i>
-------------------	----------------------------------

---

**Description**

This data set comes with the dplyr package. It contains data of 87 star war characters

**Usage**

```
use_data_starwars()
```

**Value**

Dataset

**See Also**

[dplyr::starwars](#)

**Examples**

```
use_data_starwars()
```

---

use_data_titanic	<i>Use the titanic data set</i>
------------------	---------------------------------

---

**Description**

This data set comes with base R. Survival of passengers on the Titanic.

**Usage**

```
use_data_titanic(count = FALSE)
```

**Arguments**

count	use count data
-------	----------------

**Value**

Dataset

**Examples**

```
use_data_titanic(count = TRUE)  
use_data_titanic(count = FALSE)
```

---

use_data_wordle	<i>Use the wordle data set</i>
-----------------	--------------------------------

---

**Description**

This data set contains the result of a real wordle challenge (in german language) between tow players. Wordle is a game where a player guesses a five-letter word in six tries. The variable "try" reflects the success of player A and B. Other variables like "noun", "aeiou", "unique", "common" and "rare" reflect the properties of the word.

**Usage**

```
use_data_wordle()
```

**Value**

Dataset

**Examples**

```
use_data_wordle()
```

---

weight_target	<i>Weight target variable</i>
---------------	-------------------------------

---

**Description**

Create weights for the target variable in your dataset so that are equal weights for target = 0 and target = 1. Target must be 0/1, FALSE/TRUE ore no/yes

**Usage**

```
weight_target(data, target)
```

**Arguments**

data	A dataset
target	Target variable (0/1, TRUE/FALSE, yes/no)

**Value**

Weights for each observation (as a vector)

**Examples**

```
iris$is_versicolor <- ifelse(iris$Species == "versicolor", 1, 0)
weights <- weight_target(iris, target = is_versicolor)
versicolor <- iris$is_versicolor
table(versicolor, weights)
```

---

`yyymm_calc`*Calculate with periods (format yyymm)*

---

**Description**

Calculate with periods (format yyymm)

**Usage**

```
yyymm_calc(yyymm, add_month = 0, add_year = 0, diff_to = NA)
```

**Arguments**

<code>yyymm</code>	Input vector of periods (format yyymm)
<code>add_month</code>	How many months to add (can be negative too)
<code>add_year</code>	How many years to add (can be negative too)
<code>diff_to</code>	Difference between date and yyymm (format yyymm)

**Value**

Vector of periods (format yyymm) or number of months

**Examples**

```
yyymm_calc(202412, add_month = 1)
yyymm_calc(c(202411,202412,202501), add_month = -1, add_year = 1)
yyymm_calc(202410, diff_to = 202501)
yyymm_calc(c(202411,202412,202501,202502), diff_to = 202501)
```

# Index

abtest, 4  
abtest\_shiny, 5  
abtest\_targetnum, 5  
abtest\_targetpct, 6  
add\_var\_id, 7  
add\_var\_random\_01, 8  
add\_var\_random\_cat, 8  
add\_var\_random\_dbl, 9  
add\_var\_random\_int, 10  
add\_var\_random\_moon, 11  
add\_var\_random\_starsign, 12  
  
balance\_target, 13  
  
check\_vec\_low\_variance, 13  
clean\_var, 14  
count\_pct, 15  
create\_data\_abtest, 15  
create\_data\_app, 16  
create\_data\_buy, 17  
create\_data\_churn, 18  
create\_data\_empty, 19  
create\_data\_esoteric, 20  
create\_data\_newsletter, 21  
create\_data\_person, 21  
create\_data\_random, 22  
create\_data\_unfair, 23  
create\_notebook\_explore, 24  
cut\_vec\_num\_avg, 24  
  
data\_dict\_md, 25  
decrypt, 26  
describe, 26  
describe\_all, 27  
describe\_cat, 28  
describe\_num, 28  
describe\_tbl, 29  
dplyr::starwars, 35  
drop\_obs\_if, 30  
drop\_obs\_with\_na, 30  
  
drop\_var\_by\_names, 31  
drop\_var\_low\_variance, 31  
drop\_var\_no\_variance, 32  
drop\_var\_not\_numeric, 32  
drop\_var\_with\_na, 33  
  
encrypt, 33  
explain\_forest, 34  
explain\_logreg, 35  
explain\_tree, 35  
explain\_xgboost, 37  
explore, 38  
explore\_all, 39  
explore\_bar, 40  
explore\_col, 41  
explore\_cor, 42  
explore\_count, 43  
explore\_density, 44  
explore\_shiny, 45  
explore\_targetpct, 46  
explore\_tbl, 47  
  
format\_num\_auto, 47  
format\_num\_kMB, 48  
format\_num\_space, 48  
format\_target, 49  
format\_type, 49  
  
get\_color, 50  
get\_type, 50  
get\_var\_buckets, 51  
ggplot2::diamonds, 62  
ggplot2::mpg, 63  
guess\_cat\_num, 52  
  
interact, 52  
  
log\_info\_if, 53  
  
mix\_color, 53

palmerpenguins::penguins, [64](#)  
plot\_legend\_targetpct, [54](#)  
plot\_text, [54](#)  
plot\_var\_info, [55](#)  
predict\_target, [55](#)  
  
replace\_na\_with, [56](#)  
report, [57](#)  
rescale01, [57](#)  
  
show\_color, [58](#)  
simplify\_text, [58](#)  
  
target\_explore\_cat, [59](#)  
target\_explore\_num, [60](#)  
total\_fig\_height, [61](#)  
  
use\_data\_beer, [62](#)  
use\_data\_diamonds, [62](#)  
use\_data\_iris, [63](#)  
use\_data\_mpg, [63](#)  
use\_data\_mtcars, [64](#)  
use\_data\_penguins, [64](#)  
use\_data\_starwars, [65](#)  
use\_data\_titanic, [65](#)  
use\_data\_wordle, [66](#)  
  
weight\_target, [66](#)  
  
yyyymm\_calc, [67](#)