Package ‘manymodelr’

February 12, 2020

Title  Build and Tune Several Models

Version  0.2.4

Description  Frequently one needs a convenient way to build and tune several models in one go. The goal is to provide a number of convenience functions useful in machine learning applications. It provides the ability to build, tune and obtain predictions of several models in one function. The models are built using ‘caret’ functions with easier to read syntax.


License  GPL-2

Encoding  UTF-8

LazyData  true

RoxygenNote  7.0.2

Imports  dplyr(>= 0.7.8), caret(>= 6.0-81), reshape2(>= 1.4.3), plyr(>= 1.8.4), magrittr(>= 1.5), Metrics(>= 0.1.4), tibble(>= 2.0.1), e1071(>= 1.7-0.1), purrr(>= 0.3.2), ggplot2(>= 3.2.0)

URL  https://github.com/Nelson-Gon/manymodelr

BugReports  https://github.com/Nelson-Gon/manymodelr/issues

Suggests  knitr, rmarkdown, testthat

VignetteBuilder  knitr

NeedsCompilation  no

Author  Nelson Gonzabato [aut, cre]

Maintainer  Nelson Gonzabato <gonzabato@hotmail.com>

Repository  CRAN

Date/Publication  2020-02-12 11:10:02 UTC
**R topics documented:**

- `add_model_predictions` .......................... 2
- `add_model_residuals` ......................... 3
- `agg_by_group` ................................. 4
- `extract_model_info` ............................. 4
- `fit_model` ...................... 5
- `get_data_Stats` ................................. 6
- `get_exponent` ................................. 7
- `get_mode` .................................. 7
- `get_this` ................................ 8
- `get_var_corr` ................................. 9
- `get_var_corr_` ............................... 10
- `modeleR` ................................... 11
- `multi_model_1` ............................... 12
- `multi_model_2` ............................... 13
- `na_replace` ................................ 14
- `na_replace_grouped` ....................... 14
- `plot_corr` .................................. 14
- `rowdiff` .................................. 15
- `row_mean_na` ................................. 17
- `select_col` ................................ 18
- `select_percentile` ......................... 19

---

**add_model_predictions**

*Add predictions to the data set. A dplyr compatible way to add predictions to a data set.*

**Description**

Add predictions to the data set. A dplyr compatible way to add predictions to a data set.

**Usage**

`add_model_predictions(model, old_data, new_data)`

**Arguments**

- `model` : A model object from ‘fit_model’
- `old_data` : The data set to which predicted values will be added.
- `new_data` : The data set to use for predicting.

**Value**

A data.frame object with a new column for predicted values
add_model_residuals

See Also

fit_model extract_model_info

Examples

iris1 <- iris[1:50,]
iris2 <- iris[51:100,]
lm_model <- fit_model(iris1,"Sepal.Length","Sepal.Width","lm")
head(add_model_predictions(lm_model,iris1,iris2))

add_model_residuals  Add model residuals

Description

A dplyr compatible convenience function to add residuals to a data set

Usage

add_model_residuals(model, old_data)

Arguments

model  A model object from 'fit_model'
old_data  The data set to which predicted values will be added.

Value

A data.frame object with residuals added.

Examples

iris1 <- iris[1:50,]
iris2 <- iris[51:100,]
lm_model <- fit_model(iris1,"Sepal.Length","Sepal.Width","lm")
head(add_model_residuals(lm_model,iris1))
**agg_by_group**  
*A convenient way to perform grouped operations*

**Description**

This function performs operations by grouping the data.

**Usage**

```r
agg_by_group(df, my_formula, func, ...)
```

**Arguments**

- `df`: The data set for which correlations are required
- `my_formula`: A formula such as A~B where B is the grouping variable (normally a factor). See examples below
- `func`: The kind of operation e.g sum, mean, min, max, manymodelr::get_mode
- `...`: Other arguments to 'aggregate' see ?aggregate for details

**Value**

A grouped data.frame object with results of the chosen operation.

**Examples**

```r
agg_by_group(airquality,.~Month,sum)
```

---

**extract_model_info**  
*Extract important model attributes*

**Description**

Provides a convenient way to extract any kind of model information from common model objects.

**Usage**

```r
extract_model_info(model_object, what, ...)
```

**Arguments**

- `model_object`: A model object for example a linear model object, generalized linear model object, analysis of variance object.
- `what`: character. The attribute you would like to obtain for instance p_value
- `...`: For ‘glm’ and ‘lm’ “aic”, additional arguments to ‘AIC’ from ‘stats’. 

---
Details

This provides a convenient way to extract model information for any kind of model. For linear models, one can extract such attributes as coefficients, p value("p_value"), standard error("std_err"), estimate, t value("t_value"), residuals, aic and other known attributes. For analysis of variance (aov), other attributes like sum squared(ssq), mean squared error(msq), degrees of freedom(df), p_value.

Examples

```r
# perform analysis of variance
aov_mod <- fit_model(iris, "Sepal.Length","Petal.Length + Species","aov")
extract_model_info(aov_mod, "ssq")
extract_model_info(aov_mod, c("ssq","p_value","aic"),k=5)
# select multiple
extract_model_info(aov_mod, c("ssq","p_value"))

# linear regression
lm_model <- fit_model(iris, "Sepal.Length","Petal.Length","lm")
extract_model_info(lm_model, "p_value")
extract_model_info(lm_model,"r2")
extract_model_info(lm_model,"aic",k=3)

## glm
glm_model <- fit_model(iris, "Sepal.Length","Petal.Length","lm")
extract_model_info(glm_model,"p_value")
```

Description

Fit and predict in a single function.

Usage

`fit_model(df, yname, xname, modeltype, ...)`

Arguments

- `df`: The data for which analysis is required
- `yname`: The dependent variable
- `xname`: The independent variable. Supports formulae x1+x2+...
- `modeltype`: Currently one of lm, glm and aov. Other models may work with inaccuracies
- `...`: Additional arguments to the modeltype

Examples

```r
fit_model(iris,"Sepal.Length","Species","aov")
```
get_data_Stats  
A pipe friendly way to get summary stats for exploratory data analysis

Description

A pipe friendly way to get summary stats for exploratory data analysis

Usage

get_data_Stats(x, func, exclude = NULL, na.rm = TRUE, na_action = NULL)
get_stats(x, func, exclude = NULL, na.rm = TRUE, na_action = NULL)

Arguments

x The data for which stats are required
func The nature of function to apply
exclude What kind of data should be excluded? Defaults to NULL. Currently only supports removing non-numeric data i.e exclude="non_numeric"
na.rm Logical. Should NAs be removed. Defaults to TRUE.
na_action If na.rm is set to TRUE, this uses na_replace to replace missing values. See ?na_replace for details.

Details

A convenient wrapper especially useful for get_mode

Value

A data.frame object showing the requested stats

Examples

g get_data_Stats(airquality,mean,"non_numeric",na.rm = TRUE,na_action = "mean")
g get_stats(airquality,mean,"non_numeric",na.rm = TRUE,na_action = "mean")
**get_exponent**  
*Get the exponent of any number or numbers*

**Description**  
Get the exponent of any number or numbers

**Usage**  
get_exponent(y, x)

**Arguments**  
- **y**: The number for which an exponent is required  
- **x**: The power to which y is raised

**Details**  
Depends on the expo and expo1 functions in expo

**Value**  
A data.frame object showing the value, power and result

**Examples**  
```r  
df<-data.frame(A=c(1123,25657,3987))  
get_exponent(df,3)  
get_exponent(1:5, 6)```

---

**get_mode**  
*A convenience function that returns the mode*

**Description**  
A convenience function that returns the mode

**Usage**  
get_mode(x, na.rm = TRUE)

**Arguments**  
- **x**: The dataframe or vector for which the mode is required.  
- **na.rm**: Logical. Should NAs be removed from analysis? Defaults to TRUE.
get_this

Details
Useful when used together with get_stats in a pipe fashion. These functions are for exploratory
data analysis The smallest number is returned if there is a tie in values The function is currently
slow for greater than 300,000 rows. It may take up to a minute. Getting the mode for character/
factor columns is also currently unsupported. It may work with inaccuracies. By default, NAs are
discarded.

Value
a data.frame or vector showing the mode of the variable(s)

Examples

test<-c(1,2,3,3,3,4,5)
test2<-c(455,7878,908981,NA,456,455,7878,7878,NA)
go_mode(test)
go_mode(test2)
## Not run:
mtcars %>%
go_data_Stats(go_mode)
go_data_Stats(mtcars,go_mode)
## End(Not run)

get_this

Helper function to easily access elements

Description
Helper function to easily access elements

Usage
get_this(what, where)

Arguments

what What do you want to extract from the ‘data.frame’ or ‘list’? No quotes. See
texamples below.

where Where do you want to get it from? Currently only supports ‘list’s and ‘data.frame’objects.

Details
This is a helper function useful if you would like to extract data from the output of ‘multi_model_1’.

Examples

my_list<-list(list(A=520),list(B=456))
go_this(A,my_list)
get_var_corr

Get correlations between variables

Description

This function returns the correlations between different variables.

Usage

```r
get_var_corr(
  df,
  comparison_var,
  other_vars = NULL,
  method = "pearson",
  drop_columns = c("factor", "character"),
  ...
)
```

Arguments

- `df`: The data set for which correlations are required
- `comparison_var`: The variable to compare to
- `other_vars`: Variables for which correlation with `comparison_var` is required. If not supplied, all variables will be used.
- `method`: The method used to perform the correlation test as defined in `cor.test`. Defaults to `pearson`.
- `drop_columns`: A character vector specifying column classes to drop. Defaults to `c("factor","character")`
- `...`: Other arguments to `cor.test` see `?cor.test` for details

Value

A data.frame object containing correlations between `comparison_var` and each of `other_vars`

Examples

```r
# Get correlations between all variables
get_var_corr(mtcars,"mpg")
# Use only a few variables
get_var_corr(mtcars,"mpg", other_vars = c("disp","drat"), method = "kendall", exact=FALSE)
```
get_var_corr_ Get correlations for combinations

Description

Get correlations for combinations

Usage

get_var_corr_(
  df,
  subset_cols = NULL,
  drop_columns = c("character", "factor"),
  ...
)

Arguments

df A 'data.frame' object for which correlations are required in combinations.
subset_cols A 'list' of length 2. The values in the list correspond to the comparison and other_Var arguments in 'get_var_corr'. See examples below.
drop_columns A character vector specifying column classes to drop. Defaults to c("factor","character")
... Other arguments to 'get_var_corr'

Details

This function extends get_var_corr by providing an opportunity to get correlations for combinations of variables. It is currently slow and may take up to a minute depending on system specifications.

Value

A data.frame object with combinations.

Examples

get_var_corr_(mtcars,method="pearson")
#use only a subset of the data.
get_var_corr_(mtcars,
  subset_cols = list(c("mpg","vs"),
                    c("disp","wt")),
  method="spearman",exact=FALSE)
**modeleR**

Perform several kinds of models in one function

**Description**

Perform several kinds of models in one function

**Usage**

`modeleR(df, yname, xname, modeltype, na.rm = FALSE, new_data, ...)`

**Arguments**

- `df` The data for which analysis is required
- `yname` The dependent variable
- `xname` The independent variable. Supports formulae `x1+x2+...`
- `modeltype` Currently one of `lm`, `glm` and `aov`. Other models may work with inaccuracies
- `na.rm` Logical. Should missing values be removed from analysis?
- `new_data` A data.frame object for which new predictions are to be made
- `...` Additional arguments to the modeltype

**Details**

This function provides a friendly way to perform any kind of model in one line. The model uses the inbuilt R functions `aov` and `lm` to make the predictions. If the target is missing in the new data frame, the function will (currently) make an empty column and fill this with predictions.

**Value**

A list containing summary stats and a data.frame object of some stats.

**References**


multi_model_1

Simultaneously train and predict on new data.

Description
This function provides a convenient way to train several model types. It allows a user to predict on new data and depending on the metrics, the user is able to decide which model predictions to finally use. The models are built based on Max Kuhn’s models in the caret package.

Usage
multi_model_1(
  df,
  yname,
  xname,
  method,
  metric,
  control,
  ..., 
  newdata,
  valid = FALSE
)

Arguments
- df: The data holding the training dataset
- yname: The outcome variable
- xname: The predictor variable(s)
- method: A vector containing methods to be used as defined in the caret package
- metric: One of several metrics. Accuracy, RMSE, MAE, etc
- control: See caret ?trainControl for details.
- ...: Other arguments to caret’s train function
- newdata: A data set to validate the model or for which predictions are required
- valid: Logical. Are you performing validation or dealing with new data? Defaults to False. Predictions added to new data

Details
Most of the details of the parameters can be found in the caret package documentation. This function is meant to help in exploratory analysis to make an informed choice of the best models.

Value
A list containing two objects. A tibble containing a summary of the metrics per model, a tibble containing predicted values and information concerning the model.
References


Examples

```r
library(caret)
train_set<-createDataPartition(iris$Species,p=0.8,list=FALSE)
valid_set<-iris[-train_set,]
train_set<-iris[train_set,]
ctrl<-trainControl(method="cv",number=5)
set.seed(233)
m<-multi_model_1(train_set,"Species",".",c("knn","rpart"),
"Accuracy",ctrl,newdata =valid_set,valid=TRUE)
m$Predictions
m$Metrics
m$modelInfo
```

---

multi_model_2  

Fit and predict in one function

Description

Fit and predict in one function

Usage

```r
multi_model_2(old_data, new_data, yname, xname, modeltype, ...)
```

Arguments

- `old_data`  
The data set to which predicted values will be added.
- `new_data`  
The data set to use for predicting.
- `yname`  
The dependent variable
- `xname`  
The independent variable. Supports formulae x1+x2+...
- `modeltype`  
Currently one of lm, glm and aov. Other models may work with inaccuracies
- `...`  
Additional arguments to the modeltype

Examples

```r
# fit a linear model and get predictions
multi_model_2(iris[1:50,],iris[50:99,],"Sepal.Length","Petal.Length","lm")
# multilinear
multi_model_2(iris[1:50,],iris[50:99,],"Sepal.Length",
 "Petal.Length + Sepal.Width","lm")
# glm
multi_model_2(iris[1:50,],iris[50:99,],"Sepal.Length","Petal.Length","glm")
```
Description

Replace missing values

Usage

na_replace(df, how = NULL, value = NULL)

Arguments

df

The data set (data.frame or vector) for which replacements are required

how

How should missing values be replaced? One of ffill, samples, value or any other known method e.g. mean, median, max, min. The default is NULL meaning no imputation is done. For character vectors, the use of `get_mode` is also supported. No implementation for class factor (yet).

value

If how is set to value, this allows the user to provide a specific fill value for the NAs.

Details

This function currently does not support grouping although this may be achieved with some inaccuracies using grouping functions from other packages.

Value

A data.frame object with missing values replaced.

Examples

na_replace(airquality, how = "value", value = "Missing")

na_replace_grouped

Replace NAs by group

Description

A convenient way to replace NAs by group.

Usage

na_replace_grouped(df, group_by_cols = NULL, ...)
Arguments

- df: A data.frame object for which grouped NA replacement is desired.
- group_by_cols: The column(s) used to use for the grouping.
- ...: Other arguments to `na_replace`

Value

A `data.frame` object with `NA`'s replaced.

Examples

test2 <- data.frame(A=c("A","A","A","B","B","B"),
                    B=c(NA,5,2,2,NA,2))
head(na_replace_grouped(test2,"A",
                        how="value","Replaced"))
test <- data.frame(groups=c("A","A","A","B","B","B"),
                    values = c(1,NA,2,1,NA,NA))
na_replace_grouped(test,"groups",how="mean")

plot_corr

Plot a correlations matrix

Description

This function plots the results produced by `get_var_corr_`.

Usage

plot_corr(
  df,
  x = "Comparison_Var",
  y = "Other_Var",
  xlabel = "Comparison_Variable",
  ylabel = "Other_Variable",
  title = "Correlations Plot",
  plot_style = "circles",
  title_just = 0.5,
  round_values = TRUE,
  round_which = "Correlation",
  colour_by = "Correlation",
  decimals = 2,
  show_which = "corr",
  size = 12.6,
  value_angle = 360,
  shape = 16,
  value_size = 3.5,
value_col = "black",
width = 1.1,
custom_cols = c("indianred2", "green2", "gray34"),
legend_labels = waiver(),
legend_title = NULL,
signif_cutoff = 0.05,
signif_size = 7,
signif_col = "gray13",
...}

Arguments

df The data to be plotted. A ‘data.frame‘ object produced by ‘get_var_corr_‘
x Value for the x axis. Defaults to "Comparison_Var"
y Values for the y axis. Defaults to "Other_Var."
xlabel label for the x axis
ylabel label for the y axis
title plot title.
plot_style One of squares and circles(currently).
title_just Justification of the title. Defaults to 0.5, title is centered.
round_values Logical. Should values be rounded off? Defaults to TRUE.
round_which Character. The column name to be rounded off.
colour_by The column to use for coloring. Defaults to "Correlation". Colour strength thus indicates the strength of correlations.
decimals Numeric. To how many decimal places should the rounding be done? Defaults to 2.
show_which Character. One of either corr or signif to control whether to show the correlation values or significance stars of the correlations. This is case sensitive and defaults to corr ie correlation values are shown.
size Size of the circles for plot_style set to circles
value_angle What angle should the text be?
shape Values for the shape if plot_style is circles
value_size Size of the text.
value_col What colour should the text in the squares/circles be?
width width value for plot_style set to squares.
custom_cols A vector(length 2) of colors to use for the plot. The first colour specifies the lower end of the correlations. The second specifies the higher end.
legend_labels Text to use for the legend labels. Defaults to the default labels produced by the plot method.
legend_title Title to use for the legend.
rowdiff

Get row differences between values

Description

This function returns the differences between rows depending on the user’s choice.

Usage

rowdiff(
  df,
  direction = "forward",
  exclude = NULL,
  na.rm = FALSE,
  na.action = NULL,
  ...
)
Arguments

df          The data set for which differences are required

direction   One of forward and reverse. The default is forward meaning the differences are
calculated in such a way that the difference between the current value and the
next is returned

exclude     A character vector specifying what classes should be removed. See examples
below

na.rm       Logical. Should missing values be removed? The missing values referred to are
those introduced during the calculation i.e. when subtracting a row with itself.
Defaults to FALSE.

na.action   If na.rm is TRUE, how should missing values be replaced? Depending on the
value as set out in ‘na_replace’, the value can be replaced as per the user’s re-
quirement.

...         Other arguments to ‘na_replace’.

Value

A data.frame object of row differences

See Also

na_replace

Examples

# Remove factor columns
rowdiff(iris, exclude = "factor", direction = "reverse")
rowdiff(iris[1:5,], exclude="factor",
       na.rm = TRUE, na_action = "get_mode",
       direction = "reverse")

row_mean_na  Replacing all NAs with mean values of a given row

Description

Replacing all NAs with mean values of a given row

Usage

row_mean_na(data, func, observations, na.rm = FALSE, exc)
Arguments

- **data**: is the data you for which the mean is needed
- **func**: describes the function to use. Currently only supports the mean(others may work with some inaccuracies)
- **observations**: takes on column names for which manipulations are required
- **na.rm**: Logical. Should NAs be removed from analysis?
- **exc**: the column to exclude from analysis. Useful for removing factor columns

Value

Returns a data.frame object showing columns with NAs and their replacement if na.rm=T

Description

A convenient selector gadget

Usage

```r
select_col(df, x, ...)
```

Arguments

- **df**: The data set from which to select a column
- **x**: The name of a column to select(no quotes)
- **...**: Other columns to select

Details

A friendly way to select a column or several columns. Mainly for non-pipe usage It is recommended to use known select functions to do pipe manipulations. Otherwise convert to tibble

Value

Returns a dataframe with selected columns

Examples

```r
select_col(iris,Petal.Length,Sepal.Length,Species,Petal.Width)
# A pipe friendly example
## Not run:
library(dplyr)
as_tibble(iris) %>%
select_col(Species)
## End(Not run)
```
Description

Get the row corresponding to a given percentile

Usage

select_percentile(df, percentile, get_all = TRUE, descend = FALSE)

Arguments

df A `data.frame` object for which a percentile is required. Other data structures are not yet supported.
percentile The percentile required eg 10 percentile
get_all Logical. Should all variables be used? Currently only supports using all variables.
descend Logical. Should the data be arranged in descending order? Defaults to FALSE.

Details

Returns the value corresponding to a percentile. Returns mean values if the position of the percentile is whole number. Values are sorted in ascending order. You can change this by setting descend to TRUE.

Value

A dataframe showing the row corresponding to the required percentile.

Examples

select_percentile(iris, 5)
Index

add_model_predictions, 2
add_model_residuals, 3
agg_by_group, 4
extract_model_info, 3, 4
fit_model, 3, 5
get_data_Stats, 6
get_exponent, 7
get_mode, 7
get_stats(get_data_Stats), 6
get_this, 8
get_var_corr, 9
get_var_corr_, 10
modeleR, 11
multi_model_1, 12
multi_model_2, 13
na_replace, 14, 18
na_replace_grouped, 14
plot_corr, 15
row_mean_na, 18
rowdiff, 17
select_col, 19
select_percentile, 20