Package 'treeheatr'

May 8, 2026

Type Package

Title Heatmap-Integrated Decision Tree Visualizations

Version 0.2.3

Maintainer Trang Le <grixor@gmail.com>

Description Creates interpretable decision tree visualizations with the data represented as a heatmap at the tree's leaf nodes. 'treeheatr' utilizes the customizable 'ggparty' package for drawing decision trees.

License MIT + file LICENSE

URL https://trangdata.github.io/treeheatr/index.html, https://trangdata.github.io/treeheatr-manuscript/, https://github.com/trangdata/treeheatr

BugReports https://github.com/trangdata/treeheatr/issues

Depends R (>= 3.5.0)

Imports cluster, dplyr, ggnewscale, ggparty, ggplot2, grid, gtable, partykit, seriation, stats, tidyr, yardstick

Suggests forcats, knitr, rmarkdown, rpart, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Trang Le [aut, cre] (https://trang.page/), Jason Moore [aut] (http://www.epistasisblog.org/), University of Pennsylvania [cph]

Repository CRAN

Date/Publication 2026-01-29 22:20:02 UTC

Contents

`align_plots` *Align decision tree and heatmap:*

Description

Align decision tree and heatmap:

Usage

```
align_plots(
  dheat,
  dtree,
  heat_rel_height,
  show = c("heat-tree", "heat-only", "tree-only")
)
```

Arguments

dheat	ggplot2 grob object of the heatmap.
dtree	ggplot2 grob object of the decision tree
heat_rel_height	Relative height of heatmap compared to whole figure (with tree).
show	Character string indicating which components of the decision tree-heatmap should be drawn. Can be 'heat-tree', 'heat-only' or 'tree-only'.

Value

A gtable/grob object of the decision tree (top) and heatmap (bottom).

---

clust_feat_func *Performs clustering or features.*

Description

Performs clustering or features.

Usage

```
clust_feat_func(dat, clust_vec, clust_feats = TRUE)
```

Arguments

dat	Dataframe of the original dataset. Samples may be reordered.
clust_vec	Character vector of variable names to be applied clustering on. Can include class labels.
clust_feats	if TRUE clusters displayed features (passed through 'clust_vec') using the the Gower metric based on the values of all samples and returns the ordered features. When 'clust_samps = FALSE' and 'clust_feats = FALSE', no clustering is performed.

Value

Character vector of reordered features when 'clust_feats == TRUE'.

clust_samp_func *Performs clustering of samples.*

Description

Performs clustering of samples.

Usage

```
clust_samp_func(leaf_node = NULL, dat, clust_vec, clust_samps = TRUE)
```

Arguments

leaf_node	Integer value indicating terminal node id.
dat	Dataframe of the original dataset. Samples may be reordered.
clust_vec	Character vector of variable names to be applied clustering on. Can include class labels.
clust_samps	Logical. If TRUE, hierarchical clustering would be performed among samples within each leaf node.

Value

Dataframe of reordered original dataset when clust_samps == TRUE.

compute_tree *Compute decision tree from data set*

Description

Compute decision tree from data set

Usage

```
compute_tree(
  x,
  data_test = NULL,
  target_lab = NULL,
  task = c("classification", "regression"),
  feat_types = NULL,
  label_map = NULL,
  clust_samps = TRUE,
  clust_target = TRUE,
  custom_layout = NULL,
  lev_fac = 1.3,
  panel_space = 0.001
)
```

Arguments

x	Dataframe or a ‘party‘ or ‘partynode‘ object representing a custom tree. If a dataframe is supplied, conditional inference tree is computed. If a custom tree is supplied, it must follow the partykit syntax: https://cran.r-project.org/web/packages/partykit/vignettes/par
data_test	Tidy test dataset. Required if ‘x‘ is a ‘partynode‘ object. If NULL, heatmap displays (training) data ‘x‘.
target_lab	Name of the column in data that contains target/label information.
task	Character string indicating the type of problem, either ’classification’ (categorical outcome) or ’regression’ (continuous outcome).
feat_types	Named vector indicating the type of each features, e.g., c(sex = ’factor’, age = ’numeric’). If feature types are not supplied, infer from column type.
label_map	Named vector of the meaning of the target values, e.g., c(‘0‘ = ’Edible’, ‘1‘ = ’Poisonous’).
clust_samps	Logical. If TRUE, hierarchical clustering would be performed among samples within each leaf node.
clust_target	Logical. If TRUE, target/label is included in hierarchical clustering of samples within each leaf node and might yield a more interpretable heatmap.
custom_layout	Dataframe with 3 columns: id, x and y for manually input custom layout.
lev_fac	Relative weight of child node positions according to their levels, commonly ranges from 1 to 1.5. 1 for parent node perfectly in the middle of child nodes.
panel_space	Spacing between facets relative to viewport, recommended to range from 0.001 to 0.01.

Value

A list of results from ‘partykit::ctree‘ or provided custom tree, including fit, estimates, smart layout and terminal data.

Examples

```
fit_tree <- compute_tree(penguins, target_lab = "species")
fit_tree$fit
fit_tree$layout
dplyr::select(fit_tree$term_dat, -contains("nodedata"))
```

---

diabetes *Diabetes patient records.*

Description

http://archive.ics.uci.edu/ml/datasets/diabetes https://www.kaggle.com/uciml/pima-indians-diabetes-database

Usage

```
diabetes
```

Format

A data frame with 768 observations and 9 variables: `Pregnancies`, `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI`, `DiabetesPedigreeFunction`, `Age` and `Outcome`.

`draw_heat` *Draws the heatmap.*

Description

Draws the heatmap to be placed below the decision tree.

Usage

```
draw_heat(
  dat,
  fit,
  feat_types = NULL,
  target_cols = NULL,
  target_lab_disp = fit$target_lab,
  trans_type = c("percentize", "normalize", "scale", "none"),
  clust_feats = TRUE,
  feats = NULL,
  show_all_feats = FALSE,
  p_thres = 0.05,
  cont_legend = "none",
  cate_legend = "none",
  cont_cols = ggplot2::scale_fill_viridis_c,
  cate_cols = ggplot2::scale_fill_viridis_d,
  panel_space = 0.001,
  target_space = 0.05,
  target_pos = "top"
)
```

Arguments

`dat`	Dataframe with samples from original dataset ordered according to the clustering within each leaf node.
`fit`	party object, e.g., as output from partykit::ctree()
`feat_types`	Named vector indicating the type of each features, e.g., c(sex = 'factor', age = 'numeric'). If feature types are not supplied, infer from column type.
`target_cols`	Character vectors representing the hex values of different level colors for targets, defaults to viridis option B.

target_lab_disp
: Character string for displaying the label of target label. If not provided, use 'target_lab'.

trans_type
: Character string of 'normalize', 'scale' or 'none'. If 'scale', subtract the mean and divide by the standard deviation. If 'normalize', i.e., max-min normalize, subtract the min and divide by the max. If 'none', no transformation is applied. More information on what transformation to choose can be acquired here: https://cran.rstudio.com/package=heatmaply/vignettes/heatmaply.html#data-transformation-scaling-normalize-and-percentize

clust_feats
: Logical. If TRUE, performs cluster on the features.

feats
: Character vector of feature names to be displayed in the heatmap. If NULL, display features of which P values are less than 'p_thres'.

show_all_feats
: Logical. If TRUE, show all features regardless of 'p_thres'.

p_thres
: Numeric value indicating the p-value threshold of feature importance. Feature with p-values computed from the decision tree below this value will be displayed on the heatmap.

cont_legend
: Function determining the options for legend of continuous variables, defaults to FALSE. If TRUE, use 'guide_colorbar(barwidth = 10, barheight = 0.5, title = NULL)'. Any other ['guides()'](https://ggplot2.tidyverse.org/reference/guides.html) functions would also work.

cate_legend
: Function determining the options for legend of categorical variables, defaults to FALSE. If TRUE, use 'guide_legend(title = NULL)'. Any other ['guides()'](https://ggplot2.tidyverse.org/ functions would also work.

cont_cols
: Function determining color scale for continuous variable, defaults to 'scale_fill_viridis_c(guide = cont_legend)'.

cate_cols
: Function determining color scale for nominal categorical variable, defaults to 'scale_fill_viridis_d(begin = 0.3, end = 0.9)'.

panel_space
: Spacing between facets relative to viewport, recommended to range from 0.001 to 0.01.

target_space
: Numeric value indicating spacing between the target label and the rest of the features

target_pos
: Character string specifying the position of the target label on heatmap, can be 'top', 'bottom' or 'none'.

Value

A ggplot2 grob object of the heatmap.

Examples

```
x <- compute_tree(penguins, target_lab = "species")
draw_heat(x$dat, x$fit)
```

`draw_tree` *Draws the conditional decision tree.*

Description

Draws the conditional decision tree output from partykit::ctree(), utilizing ggparty geoms: geom_edge, geom_edge_label, geom_node_label.

Usage

```
draw_tree(
  dat,
  fit,
  term_dat,
  layout,
  target_cols = NULL,
  title = NULL,
  tree_space_top = 0.05,
  tree_space_bottom = 0.05,
  print_eval = FALSE,
  metrics = NULL,
  x_eval = 0,
  y_eval = 0.9,
  task = c("classification", "regression"),
 par_node_vars = list(label.padding = unit(0.15, "lines"), line_list = list(aes(label =
    splitvar)), line_gpar = list(list(size = 9)), ids = "inner"),
 terminal_vars = list(label.padding = unit(0.25, "lines"), size = 3, col = "white"),
  edge_vars = list(color = "grey70", linewidth = 0.5),
  edge_text_vars = list(color = "grey30", size = 3, mapping = aes(label =
    paste(breaks_label, "*NA")))
)
```

Arguments

`dat`	Dataframe with samples from original dataset ordered according to the clustering within each leaf node.
`fit`	party object, e.g., as output from partykit::ctree()
`term_dat`	Dataframe for terminal nodes, must include these columns: id, x, y and y_hat.
`layout`	Dataframe of layout of all nodes, must include these columns: id, x, y and y_hat.
`target_cols`	Character vectors representing the hex values of different level colors for targets, defaults to viridis option B.
`title`	Character string for plot title.
`tree_space_top`	Numeric value to pass to expand for top margin of tree.
`tree_space_bottom`	Numeric value to pass to expand for bottom margin of tree.

print_eval	Logical. If TRUE, print evaluation of the tree performance.
metrics	A set of metric functions to evaluate decision tree, defaults to common metrics for classification/regression problems. Can be defined with 'yardstick::metric_set'.
x_eval	Numeric value indicating x position to print performance statistics.
y_eval	Numeric value indicating y position to print performance statistics.
task	Character string indicating the type of problem, either 'classification' (categorical outcome) or 'regression' (continuous outcome).
par_node_vars	Named list containing arguments to be passed to the 'geom_node_label()' call for non-terminal nodes.
terminal_vars	Named list containing arguments to be passed to the 'geom_node_label()' call for terminal nodes.
edge_vars	Named list containing arguments to be passed to the 'geom_edge()' call for tree edges.
edge_text_vars	Named list containing arguments to be passed to the 'geom_edge_label()' call for tree edge annotations.

Value

A ggplot2 grob object of the decision tree.

Examples

```
x <- compute_tree(penguins, target_lab = "species")
draw_tree(x$dat, x$fit, x$term_dat, x$layout)
```

---

eval_tree *Print decision tree performance according to different metrics.*

Description

Print decision tree performance according to different metrics.

Usage

```
eval_tree(
  dat,
  target_lab = colnames(dat)[1],
  task = c("classification", "regression"),
  metrics = NULL
)
```

Arguments

dat	Dataframe with truths (column 'target_lab') and estimates (column 'y_hat') of samples from original dataset.
target_lab	Name of the column in data that contains target/label information.
task	Character string indicating the type of problem, either 'classification' (categorical outcome) or 'regression' (continuous outcome).
metrics	A set of metric functions to evaluate decision tree, defaults to common metrics for classification/regression problems. Can be defined with 'yardstick::metric_set'.

Value

Character string of the decision tree evaluation.

Examples

```
eval_tree(compute_tree(penguins, target_lab = "species")$dat)
```

---

galaxy *Galaxy dataset for regression.*

Description

Fetched from PMLB.

Usage

```
galaxy
```

Format

An object of class `data.frame` with 323 rows and 5 columns.

Details

#' @format A data frame with 323 observations and 5 variables: `eastwest`, `northsouth`, `angle`, `radialposition` and `target` (velocity).

https://www.openml.org/d/690

get_cols *Get color functions from character vectors*

Description

Get color functions from character vectors

Usage

```
get_cols(my_cols, task, guide = "none")
```

Arguments

my_cols	Character vectors of different hex values
task	Character string indicating the type of problem, either 'classification' (categorical outcome) or 'regression' (continuous outcome).
guide	A function used to create a guide or its name. Inherit from [‘ggplot2::guides()‘](https://ggplot2.tidyverse.o

get_disp_feats *Select the important features to be displayed.*

Description

Select features with p-value (computed from decision tree) < ‘p_thres‘ or all features if ‘show_all_feats == TRUE‘.

Usage

```
get_disp_feats(fit, feat_names, show_all_feats, p_thres)
```

Arguments

fit	constparty object of the decision tree.
feat_names	Character vector specifying the feature names in dat.
show_all_feats	Logical. If TRUE, show all features regardless of ‘p_thres‘.
p_thres	Numeric value indicating the p-value threshold of feature importance. Feature with p-values computed from the decision tree below this value will be displayed on the heatmap.

Value

A character vector of feature names.

get_fit *Get the fitted tree depending on the input 'x'.*

Description

If 'x' is a data.frame object, computes conditional tree from partkit::ctree(). If 'x' is a partynode object specifying the customized tree, fit 'x' on 'data_test'. If 'x' is a party (or constparty) object specifying the precomputed tree, simply coerce 'x' to have class constparty.

Usage

```
get_fit(x, ...)

## Default S3 method:
get_fit(x, ...)

## S3 method for class 'partynode'
get_fit(x, data_test, target_lab, ...)

## S3 method for class 'party'
get_fit(x, data_test, target_lab, task, ...)

## S3 method for class 'data.frame'
get_fit(x, data_test, target_lab, ...)
```

Arguments

x	Dataframe or a 'party' or 'partynode' object representing a custom tree. If a dataframe is supplied, conditional inference tree is computed. If a custom tree is supplied, it must follow the partykit syntax: https://cran.r-project.org/web/packages/partykit/vignettes/par
...	Further arguments passed to each method.
data_test	Tidy test dataset. Required if 'x' is a 'partynode' object. If NULL, heatmap displays (training) data 'x'.
target_lab	Name of the column in data that contains target/label information.
task	Character string indicating the type of problem, either 'classification' (categorical outcome) or 'regression' (continuous outcome).

Value

Fitted object as a list with prepped 'data_test' if available.

heat_tree *Draws and aligns decision tree and heatmap.*

Description

heat_tree() alias.

Usage

```
heat_tree(
  x,
  target_lab = NULL,
  data_test = NULL,
  task = c("classification", "regression"),
  feat_types = NULL,
  label_map = NULL,
  target_cols = NULL,
  target_legend = FALSE,
  clust_samps = TRUE,
  clust_target = TRUE,
  custom_layout = NULL,
  show = "heat-tree",
  heat_rel_height = 0.2,
  lev_fac = 1.3,
  panel_space = 0.001,
  print_eval = (!is.null(data_test)),
  ...
)

treeheatr(
  x,
  target_lab = NULL,
  data_test = NULL,
  task = c("classification", "regression"),
  feat_types = NULL,
  label_map = NULL,
  target_cols = NULL,
  target_legend = FALSE,
  clust_samps = TRUE,
  clust_target = TRUE,
  custom_layout = NULL,
  show = "heat-tree",
  heat_rel_height = 0.2,
  lev_fac = 1.3,
  panel_space = 0.001,
  print_eval = (!is.null(data_test)),
  ...
```

```
)
```

Arguments

x	Dataframe or a ‘party‘ or ‘partynode‘ object representing a custom tree. If a dataframe is supplied, conditional inference tree is computed. If a custom tree is supplied, it must follow the partykit syntax: https://cran.r-project.org/web/packages/partykit/vignettes/par
target_lab	Name of the column in data that contains target/label information.
data_test	Tidy test dataset. Required if ‘x‘ is a ‘partynode‘ object. If NULL, heatmap displays (training) data ‘x‘.
task	Character string indicating the type of problem, either ’classification’ (categorical outcome) or ’regression’ (continuous outcome).
feat_types	Named vector indicating the type of each features, e.g., c(sex = ’factor’, age = ’numeric’). If feature types are not supplied, infer from column type.
label_map	Named vector of the meaning of the target values, e.g., c(‘0‘ = ’Edible’, ‘1‘ = ’Poisonous’).
target_cols	Character vectors representing the hex values of different level colors for targets, defaults to viridis option B.
target_legend	Logical. If TRUE, target legend is drawn.
clust_samps	Logical. If TRUE, hierarchical clustering would be performed among samples within each leaf node.
clust_target	Logical. If TRUE, target/label is included in hierarchical clustering of samples within each leaf node and might yield a more interpretable heatmap.
custom_layout	Dataframe with 3 columns: id, x and y for manually input custom layout.
show	Character string indicating which components of the decision tree-heatmap should be drawn. Can be ’heat-tree’, ’heat-only’ or ’tree-only’.
heat_rel_height	Relative height of heatmap compared to whole figure (with tree).
lev_fac	Relative weight of child node positions according to their levels, commonly ranges from 1 to 1.5. 1 for parent node perfectly in the middle of child nodes.
panel_space	Spacing between facets relative to viewport, recommended to range from 0.001 to 0.01.
print_eval	Logical. If TRUE, print evaluation of the tree performance. Defaults to TRUE when ‘data_test‘ is supplied.
...	Further arguments passed to ‘draw_tree()‘ and/or ‘draw_heat()‘.

Value

A gtable/grob object of the decision tree (top) and heatmap (bottom).

Examples

```
heat_tree(penguins, target_lab = "species")
heat_tree(
  x = galaxy[1:100, ],
  target_lab = "target",
  task = "regression",
  terminal_vars = NULL,
  tree_space_bottom = 0
)

treeheatr(penguins, target_lab = "species")

treeheatr(
  x = galaxy[1:100, ],
  target_lab = "target",
  task = "regression",
  terminal_vars = NULL,
  tree_space_bottom = 0
)
```

---

penguins *Data of three different species of penguins.*

Description

Collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.

Usage

```
penguins
```

Format

A data frame with 344 observations and 7 variables: `species`, `island`, `culmen_length_mm`, `culmen_depth_mm`, `flipper_length_mm`, `body_mass_g` and `sex`.

Gorman KB, Williams TD, Fraser WR (2014). Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis). PLoS ONE 9(3): e90081. doi:10.1371/journal.pone.0090081

Details

Fetched from https://github.com/allisonhorst/penguins.

position_nodes *Creates smart node layout.*

Description

Create node layout using a bottom-up approach (literally) and overwrites ggparty-precomputed positions in plot_data.

Usage

```
position_nodes(plot_data, terminal_data, custom_layout, lev_fac, panel_space)
```

Arguments

plot_data	Dataframe output of 'ggparty:::get_plot_data()'.
terminal_data	Dataframe of terminal node information including id and raw terminal node size.
custom_layout	Dataframe with 3 columns: id, x and y for manually input custom layout.
lev_fac	Relative weight of child node positions according to their levels, commonly ranges from 1 to 1.5. 1 for parent node perfectly in the middle of child nodes.
panel_space	Spacing between facets relative to viewport, recommended to range from 0.001 to 0.01.

Value

Dataframe with 3 columns: id, x and y of smart layout combined with custom_layout.

prediction_df *Apply the predicted tree on either new test data or training data.*

Description

Select features with p-value (computed from decision tree) < 'p_thres' or all features if 'show_all_feats == TRUE'.

Usage

```
prediction_df(fit, task, clust_samps, clust_target)
```

Arguments

fit	constparty object of the decision tree.
task	Character string indicating the type of problem, either 'classification' (categorical outcome) or 'regression' (continuous outcome).
clust_samps	Logical. If TRUE, hierarchical clustering would be performed among samples within each leaf node.
clust_target	Logical. If TRUE, target/label is included in hierarchical clustering of samples within each leaf node and might yield a more interpretable heatmap.

Value

A dataframe of prediction values with scaled columns and clustered samples.

`prepare_feats` *Prepares the feature dataframes for tiles.*

Description

If R does not recognize a categorical feature (input from user) as factor, converts to factor.

Usage

```
prepare_feats(dat, disp_feats, feat_types, clust_feats, trans_type)
```

Arguments

`dat`	Dataframe with samples from original dataset ordered according to the clustering within each leaf node.
`disp_feats`	Character vector specifying features to be displayed.
`feat_types`	Named vector indicating the type of each features, e.g., c(sex = 'factor', age = 'numeric'). If feature types are not supplied, infer from column type.
`clust_feats`	Logical. If TRUE, performs cluster on the features.
`trans_type`	Character string of 'normalize', 'scale' or 'none'. If 'scale', subtract the mean and divide by the standard deviation. If 'normalize', i.e., max-min normalize, subtract the min and divide by the max. If 'none', no transformation is applied. More information on what transformation to choose can be acquired here: https://cran.rstudio.com/package=heatmaply/vignettes/heatmaply.html#data-transformation-scaling-normalize-and-percentize

Value

A list of two dataframes (continuous and categorical) from the original dataset.

`prep_data` ——————————————————————— *Prepare dataset*

Description

——————————————————————— Prepare dataset

Usage

```
prep_data(data, target_lab, task, feat_types = NULL)
```

Arguments

data	Original data frame with features to be converted to correct types.
target_lab	Name of the column in data that contains target/label information.
task	Character string indicating the type of problem, either 'classification' (categorical outcome) or 'regression' (continuous outcome).
feat_types	Named vector indicating the type of each features, e.g., c(sex = 'factor', age = 'numeric'). If feature types are not supplied, infer from column type.

Value

List of dataframes (training + test) with proper feature types and target name.

---

scale_norm *Performs transformation on continuous variables.*

Description

Performs transformation on continuous variables for the heatmap color scales.

Usage

```
scale_norm(x, trans_type = c("percentize", "normalize", "scale", "none"))
```

Arguments

x	Numeric vector.
trans_type	Character string of 'normalize', 'scale' or 'none'. If 'scale', subtract the mean and divide by the standard deviation. If 'normalize', i.e., max-min normalize, subtract the min and divide by the max. If 'none', no transformation is applied. More information on what transformation to choose can be acquired here: https://cran.rstudio.com/package=heatmaply/vignettes/heatmaply.html#data-transformation-scaling-normalize-and-percentize

Value

Numeric vector of the transformed 'x'.

Examples

```
scale_norm(1:5)
scale_norm(1:5, "normalize")
```

term_node_pos *Determines terminal node position.*

Description

Create node layout using a bottom-up approach (literally) and overwrites ggparty-precomputed positions in plot_data.

Usage

```
term_node_pos(plot_data, dat)
```

Arguments

plot_data	Dataframe output of 'ggparty:::get_plot_data()'.
dat	Dataframe of prediction values with scaled columns and clustered samples.

Value

Dataframe with terminal node information.

test_covid *External test dataset. Medical information of Wuhan patients collected between 2020-01-10 and 2020-02-18.*

Description

External test dataset. Medical information of Wuhan patients collected between 2020-01-10 and 2020-02-18.

Usage

```
test_covid
```

Format

A data frame with 110 observations and 7 XGBoost-selected variables: PATIENT_ID, Lactate dehydrogenase, High sensitivity C-reactive protein, (%)lymphocyte, Admission time, Discharge time and outcome.

An interpretable mortality prediction model for COVID-19 patients. Yan et al. https://doi.org/10.1038/s42256-020-0180-7 https://github.com/HAIRLAB/Pre_Surv_COVID_19

`train_covid` *Training dataset. Medical information of Wuhan patients collected between 2020-01-10 and 2020-02-18. Containing NAs.*

Description

Training dataset. Medical information of Wuhan patients collected between 2020-01-10 and 2020-02-18. Containing NAs.

Usage

```
train_covid
```

Format

A data frame with 375 observations and 77 variables.

An interpretable mortality prediction model for COVID-19 patients. Yan et al. https://doi.org/10.1038/s42256-020-0180-7 https://github.com/HAIRLAB/Pre_Surv_COVID_19

`wine` *Results of a chemical analysis of wines grown in a specific area of Italy.*

Description

Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample.

Usage

```
wine
```

Format

A data frame with 178 observations and 14 variables: `Alcohol`, `Malic`, `Ash`, `Alcalinity`, `Magnesium`, `Phenols`, `Flavanoids`, `Nonflavanoids`, `Proanthocyanins`, `Color`, `Hue`, `Dilution`, `Proline` and `Type` (target).

Details

Import with data(wine, package = 'rattle'). Dependent variable: Type. https://rdrr.io/cran/rattle.data/man/wine.html http://archive.ics.uci.edu/ml/datasets/wine

wine_quality_red *Red variant of the Portuguese "Vinho Verde" wine.*

Description

Fetched from PMLB. Physicochemical and quality of wine.

Usage

```
wine_quality_red
```

Format

A data frame with 1599 observations and 12 variables: `fixed.acidity`, `volatile.acidity`, `citric.acid`, `residual.sugar`, `chlorides`, `free.sulfur.dioxide`, `total.sulfur.dioxide`, `density`, `pH`, `sulphates`, `alcohol` and `target` (quality).

http://archive.ics.uci.edu/ml/datasets/Wine+Quality

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Index