Package 'dtGAP'

February 18, 2026

Type Package

Title Supervised Generalized Association Plots Based on Decision Trees

Version 0.0.2

Date 2026-02-13

Description Enhances decision tree visualization by incorporating Generalized Association Plots (GAP) through matrix-based visualizations including confusion matrix maps, decision tree matrix maps, and predicted class membership maps based on supervised correlation and distance metrics.

License MIT + file LICENSE

URL https://github.com/hanmingwu1103/dtGAP, https://CRAN.R-project.org/package=dtGAP

BugReports https://github.com/hanmingwu1103/dtGAP/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports C50, caret, circlize, ComplexHeatmap, dplyr, ggparty, grDevices, grid, magrittr, partykit, RColorBrewer, rlang, rpart, seriation, stats, stringr, utils, yardstick

Suggests InteractiveComplexHeatmap, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Chia-Yu Chang [aut], Chun-houh Chen [aut], Han-Ming Wu [cre, aut]

Maintainer Han-Ming Wu <wuhm@g.nccu.edu.tw>

Repository CRAN

Date/Publication 2026-02-18 18:10:02 UTC

Contents

`add_data_type` *Assigns a train/test indicator to a combined dataset*

Description

Assigns a train/test indicator to a combined dataset

Usage

```
add_data_type(
  data_train = NULL,
  data_test = NULL,
  data_all = NULL,
  test_size = 0.3,
  seed = 42
)
```

Arguments

data_train	A data frame of training observations (or NULL).
data_test	A data frame of testing observations (or NULL).
data_all	A data frame of all observations (or NULL).
test_size	Numeric in (0,1). Proportion for testing (default 0.3).
seed	Integer. Random seed for splitting (default 42).

Value

A data frame with a data_type factor column.

---

compare_dtGAP *Compare Multiple Decision Tree Models Side-by-Side*

Description

Runs the dtGAP pipeline for each specified model and composes the results side-by-side on a single wide page. Shared data preparation is performed once; each model gets its own tree + heatmap panel.

Usage

```
compare_dtGAP(
  models = c("rpart", "party"),
  data_train = NULL,
  data_test = NULL,
  data_all = NULL,
  target_lab = NULL,
  show = c("all", "train", "test"),
  test_size = 0.3,
  task = c("classification", "regression"),
  total_w = 594,
  total_h = 210,
  ...
)
```

Arguments

models	Character vector of length >= 2. Models to compare. Each must be one of "rpart", "party", "C50", or "caret".
data_train	Data frame. Training data.
data_test	Data frame. Test data.
data_all	Data frame. Full dataset (alternative to separate train/test).
target_lab	Character. Name of the target column.

show	Character. Which subset to show: "all", "train", or "test".
test_size	Numeric. Proportion for test split (default 0.3).
task	Character. "classification" or "regression".
total_w	Numeric. Total page width in mm (default 594, 2x A4 width).
total_h	Numeric. Total page height in mm (default 210).
...	Additional visual parameters passed to each dtGAP panel (e.g. trans_type, col_proximity, print_eval).

Value

Draws the side-by-side comparison to the current graphics device. Called for its side effect; returns invisibly.

Examples

```
compare_dtGAP(
  models = c("rpart", "party"),
  data_all = Psychosis_Disorder,
  target_lab = "UNIQID",
  show = "all",
  trans_type = "none",
  print_eval = FALSE
)
```

---

compute_tree *Compute Decision Tree Data for Plotting and Analysis*

Description

Builds and processes a decision tree model object to prepare data for plotting, including layout positions and terminal node summaries. need to run util.R first

Usage

```
compute_tree(
  fit = NULL,
  model = c("rpart", "party", "C50", "caret", "cforest"),
  show = c("all", "train", "test"),
  data = NULL,
  target_lab = NULL,
  task = c("classification", "regression"),
  custom_layout = NULL,
  panel_space = 0.001
)
```

Arguments

fit	A fitted decision party tree object.
model	Character. Which implementation to use: one of "rpart", "party", "C50", or "caret".
show	Character. Which subset to return: "all", "train" or "test" .
data	A data.frame containing the features and target for prediction.
target_lab	Character. Name of the target column.
task	Character. Task type: "classification" or "regression".
custom_layout	Optional data.frame with custom node positions (columns: id, x, y).
panel_space	Numeric. Vertical spacing between panels in layout.

Value

A list with components:

- fit: the original fitted model
- dat: data.frame of observations with node assignments and predictions
- plot_data: data.frame of nodes with plotting variables and probabilities
- layout: data.frame of node x/y positions

Examples

```
library(rpart)
library(partykit)
library(ggparty)
library(dplyr)
data <- add_data_type(
  data_all = Psychosis_Disorder
)
data <- prepare_features(
  data,
  target_lab = "UNIQID",
  task = "classification"
)
fit <- train_tree(
  data = data, target_lab = "UNIQID",
  model = "rpart"
)$fit
tree_res <- compute_tree(
  fit,
  model = "rpart", show = "all",
  data = data, target_lab = "UNIQID",
  task = "classification"
)
tree_res$dat
tree_res$plot_data
```

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_all *Draw Full Visualization: Decision Tree with Heatmap and Evaluation*

Description

This function creates a full-page layout consisting of a decision tree plot, a heatmap, and optional evaluation results. It is designed for use in reporting classification or clustering trees with additional visual indicators.

Usage

```
draw_all(
  prepare_tree,
  heat,
  total_w = 297,
  total_h = 210,
  layout,
  x_eval_start = 15,
  y_eval_start = NULL,
  eval_text = 7,
  eval_res = NULL,
  print_eval = TRUE,
  show_col_prox = TRUE,
  show_row_prox = TRUE
)
```

Arguments

prepare_tree	A list returned from a tree preparation function, containing plot_data and branches for tree structure.
heat	A grob object representing the heatmap visualization. Usually generated with grid.grabExpr(draw(...))
total_w	Total width of the drawing in mm. Default is 297 (A4 landscape width).
total_h	Total height of the drawing in mm. Default is 210 (A4 landscape height).
layout	A list specifying layout parameters: tree_w, tree_h, margin, and offset_h.
x_eval_start	X-axis starting position (in mm) for evaluation text. Default is 15.
y_eval_start	Y-axis starting position (in mm) for evaluation text. If NULL, it will be computed automatically.
eval_text	Font size for the evaluation text. Default is 6.
eval_res	A list with evaluation result text from eval_tree() (including data_info, train_metrics, and test_metrics).
print_eval	Logical, whether to show evaluation results. Default is TRUE.
show_col_prox	Logical, whether to show column proximity.
show_row_prox	Logical, whether to show row proximity.

Value

Draws the full visualization to the current graphics device. Called for its side effect; returns invisible(NULL).

Examples

```
# See dtGAP() for a full end-to-end example
# that internally calls draw_all().
```

---

dtGAP *Decision Tree Generalized Association Plots (dtGAP)*

Description

The dtGAP function enhances decision tree visualization by incorporating the strengths of Generalized Association Plots (GAP). While decision trees are valued for their interpretability, they often overlook deeper data structures. In contrast, GAP is effective for revealing complex associations but is typically limited to unsupervised settings. dtGAP bridges this gap by introducing matrix-based visualizations—such as the confusion matrix map, decision tree matrix map, and predicted class membership map—based on supervised correlation and distance metrics. This offers a more comprehensive and interpretable representation of decision-making processes in tree-based models.

Usage

```
dtGAP(
  x = NULL,
  target_lab = NULL,
  show = c("all", "train", "test"),
  model = c("rpart", "party", "C50", "caret"),
  control = NULL,
  fit = NULL,
  user_var_imp = NULL,
  data_train = NULL,
  data_test = NULL,
  data_all = NULL,
  test_size = 0.3,
  task = c("classification", "regression"),
  trans_type = c("normalize", "scale", "percentize", "none"),
  col_proximity = c("pearson", "spearman", "kendall"),
  linkage_method = c("CT", "SG", "CP"),
  seriate_method = "TSP",
  cRGAR_w = 5,
  select_vars = NULL,
  sort_by_data_type = TRUE,
  custom_layout = NULL,
  panel_space = 0.001,
  margin = 20,
  total_w = 297,
  total_h = 210,
  tree_p = 0.3,
  include_var_imp = TRUE,
  col_var_imp = "orange",
  var_imp_bar_width = 0.8,
  var_imp_fontsize = 5,
  split_var_bg = "darkgreen",
  split_var_fontsize = 5,
  Col_Prox_palette = "RdBu",
  Col_Prox_n_colors = 11,
  label_map = NULL,
  label_map_colors = NULL,
  type_palette = "Dark2",
  label_palette = "OrRd",
  n_label_color = 9,
  pred_ha_gap = unit(1, "mm"),
  prop_palette = gray,
  n_prop_colors = 11,
  Row_Prox_palette = "Spectral",
  Row_Prox_n_colors = 11,
  row_border = TRUE,
  row_gap = unit(1, "mm"),
  sorted_dat_palette = "Blues",
```

```
  sorted_dat_n_colors = 9,
  show_row_names = TRUE,
  row_names_gp = gpar(fontsize = 5),
  show_row_prox = TRUE,
  show_col_prox = TRUE,
  raw_value_col = NULL,
  lgd_direction = c("vertical", "horizontal"),
  x_eval_start = 15,
  y_eval_start = NULL,
  eval_text = 7,
  print_eval = TRUE,
  simple_metrics = FALSE,
  interactive = FALSE
)
```

Arguments

x	Character. Name or label of the dataset.
target_lab	Character. Name of the target column. Required.
show	Character. Which subset to return: "all", "train" or "test" .
model	Character. Which implementation to use: one of "rpart", "party", "C50", or "caret". Ignored when `fit` is provided.
control	List or control object. Optional control parameters passed to the chosen tree function. Ignored when `fit` is provided.
fit	Optional pre-built tree model object. Supported classes: `party`, `rpart`, `C5.0`, or `train` (caret). When supplied, `model` and `control` are ignored; the model type is auto-detected.
user_var_imp	Optional named numeric vector of variable importance scores. Only used when `fit` is provided. If NULL, importance is extracted automatically (or the importance barplot is suppressed if extraction fails).
data_train	Data frame. Training data. Required if show == "train" or when splitting from all.
data_test	Data frame. Test data. Required if show == "test" or when splitting from all.
data_all	Data frame. Full dataset. If provided and show == "all", used directly; otherwise split into train/test.
test_size	Numeric. Proportion of data to assign to testing set when splitting data_all (default 0.3).
task	Character. Type of task: "classification" or "regression".
trans_type	Character. One of "percentize","normalize","scale","none" passed to scale_norm().
col_proximity	Character. Correlation method: "pearson","spearman","kendall".
linkage_method	Character. Linkage for supervised distance: "CT","SG","CP".
seriate_method	Character. Seriation method for distance objects; see `seriation::list_seriation_methods("dist")` for all supported options. Default: `"TSP"`.
cRGAR_w	Integer. Window size for RGAR calculation.

`select_vars` Character vector or NULL. If provided, only these variables are displayed in the heatmap panels. The tree is always fit on ALL variables; this parameter is display-only. Names must match feature column names.

`sort_by_data_type` Logical. If TRUE, preserves data_type grouping within nodes.

`custom_layout` Optional data.frame with custom node positions (columns: id, x, y).

`panel_space` Numeric. Vertical spacing between panels in layout.

`margin` Numeric. Margin around the drawing area (mm).

`total_w` Numeric. Total width of page (mm).

`total_h` Numeric. Total height of page (mm).

`tree_p` Numeric. Proportion of total width allocated to the tree panel.

`include_var_imp` Logical; include importance barplot if TRUE (default TRUE).

`col_var_imp` Color for importance bars (default "orange").

`var_imp_bar_width` Numeric width of bars (default 0.8).

`var_imp_fontsize` Font size for importance text (default 5).

`split_var_bg` Background color for split variable names (default "darkgreen").

`split_var_fontsize` Font size for split variable names (default 5).

`Col_Prox_palette` RColorBrewer palette for correlation heatmap (default "RdBu").

`Col_Prox_n_colors` Number of colors in correlation scale (default 11).

`label_map` Optional named vector to map raw labels to new labels.

`label_map_colors` Optional named vector of colors for mapped labels.

`type_palette` RColorBrewer palette for data_type (default "Dark2").

`label_palette` Function or vector of colors for true and predicted value (default OrRd).

`n_label_color` Number of colors for label palette (default 9).

`pred_ha_gap` Unit for gap between annotations (default `unit(1, "mm")`).

`prop_palette` Function or vector of colors for probability gradient (default gray).

`n_prop_colors` Number of colors for probability palette (default 11).

`Row_Prox_palette` RColorBrewer palette name for row proximity color scale (default "Spectral").

`Row_Prox_n_colors` Number of discrete colors for row proximity (default 11).

`row_border` Logical; draw cell borders if TRUE (default TRUE).

`row_gap` Unit object for gap between annotation blocks (default `unit(1, "mm")`).

`sorted_dat_palette`
: RColorBrewer palette for heatmap values (default "Blues").

`sorted_dat_n_colors`
: Number of colors for heatmap (default 9).

`show_row_names`
: Logical. Whether to display row names in the heatmap (default TRUE).

`row_names_gp`
: gpar settings for row name font (default fontsize=5).

`show_row_prox`
: Logical, whether to show row proximity.

`show_col_prox`
: Logical, whether to show column proximity.

`raw_value_col`
: User-defined colors for raw data values.

`lgd_direction`
: Character. Layout direction of packed legends, either "vertical" or "horizontal".

`x_eval_start`
: X-axis starting position (in mm) for evaluation text. Default is 15.

`y_eval_start`
: Y-axis starting position (in mm) for evaluation text. If NULL, it will be computed automatically.

`eval_text`
: Font size for the evaluation text. Default is 7.

`print_eval`
: Logical, whether to show evaluation results. Default is TRUE.

`simple_metrics`
: Logical. If TRUE, use simple metric summary instead of full confusion matrix. Default is FALSE.

`interactive`
: Logical. If TRUE, launches an interactive Shiny app via `InteractiveComplexHeatmap::htShiny()` showing the heatmap panels with hover/click/zoom. The tree panel is omitted in interactive mode (limitation of InteractiveComplexHeatmap). Requires the **InteractiveComplexHeatmap** package (from Bioconductor). Default is FALSE.

Value

Draws the full dtGAP visualization (decision tree + heatmap + evaluation) to the current graphics device. Called for its side effect; returns invisibly.

Examples

```
# Case 1: test_covid
dtGAP(
  data_train = train_covid,
  data_test = test_covid,
  target_lab = "Outcome", show = "test",
  label_map = c("0" = "Survival", "1" = "Death"),
  label_map_colors = c(
    "Survival" = "#50046d", "Death" = "#fcc47f"
  ),
  raw_value_col = colorRampPalette(
    c("#33286b", "#26828e", "#75d054", "#fae51f")
  )(9)
)
# Case 2: Psychosis_Disorder
dtGAP(
  data_all = Psychosis_Disorder,
  model = "party", show = "all",
  trans_type = "none", target_lab = "UNIQID"
```

```
)
```

`eval_tree` *Evaluate Tree Model Predictions and Metrics*

Description

Generates summary information and confusion matrix metrics for training and/or test subsets based on a fitted decision tree and sorted matrix results.

Usage

```
eval_tree(
  x = NULL,
  fit = NULL,
  task = c("classification", "regression"),
  tree_res = NULL,
  target_lab = NULL,
  sorted_dat = NULL,
  show = c("all", "train", "test"),
  model = c("rpart", "party", "C50", "caret", "cforest"),
  col_proximity = c("pearson", "spearman", "kendall"),
  linkage_method = c("CT", "SG", "CP"),
  seriate_method = "TSP",
  simple_metrics = FALSE
)
```

Arguments

`x`	Character. Name or label of the dataset.
`fit`	A fitted partykit tree object used to extract split variables.
`task`	Character. Type of task: "classification" or "regression".
`tree_res`	List. Output from `compute_tree()`
`target_lab`	Character. Name of the target column in `tree_res$dat`.
`sorted_dat`	List. Output from `sorted_mat()`.
`show`	Character. "train","test", or "all" to select subset before sorting.
`model`	Character. Identifier for the model method (e.g., "rpart").
`col_proximity`	Character. Correlation method: "pearson","spearman","kendall".
`linkage_method`	Character. Linkage for supervised distance: "CT","SG","CP".
`seriate_method`	Character. Seriation method for distance objects; see `seriation::list_seriation_methods("dist")` for all supported options. Default: `"TSP"`.
`simple_metrics`	Logical. If TRUE, use simple metric summary instead of full confusion matrix (default FALSE).

Value

A list with elements:

`data_info`	Character summary of dataset name, sizes, methods, and scores.
`train_metrics`	Character output of the train confusion matrix (if applicable).
`test_metrics`	Character output of the test confusion matrix (if applicable).

Examples

```
library(rpart)
library(partykit)
library(ggparty)
library(dplyr)
library(seriation)
data_all <- add_data_type(
  data_train = train_covid, data_test = test_covid
)
data <- prepare_features(
  data_all,
  target_lab = "Outcome",
  task = "classification"
)
train_tree <- train_tree(
  data_train = train_covid,
  target_lab = "Outcome", model = "rpart"
)
fit <- train_tree$fit
var_imp <- train_tree$var_imp
tree_res <- compute_tree(
  fit,
  model = "rpart", show = "test",
  data = data, target_lab = "Outcome",
  task = "classification"
)
sorted_dat <- sorted_mat(
  tree_res,
  target_lab = "Outcome", show = "test"
)
# Case 1: Pass the dataset name
eval_tree(
  x = "covid", fit = fit,
  task = "classification",
  tree_res = tree_res,
  target_lab = "Outcome",
  sorted_dat = sorted_dat,
  show = "test", model = "rpart"
)
```

galaxy *Galaxy dataset for regression.*

Description

Fetched from PMLB.

Usage

```
galaxy
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `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

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.

prepare_features *Prepare Features for Modeling*

Description

Converts target variable for classification tasks and coerces logical/character columns to factors.

Usage

```
prepare_features(
  data,
  target_lab = NULL,
  task = c("classification", "regression")
)
```

Arguments

data	Data frame or tibble. Input dataset (train or test).
target_lab	Character. Name of the target column. Required for classification.
task	Character. Type of task: "classification" or "regression".

Value

A tibble with processed feature types.

prepare_tree *Prepare Tree Plot Data for Visualization*

Description

This function processes a tree model's output and prepares node and segment data for visualization using ggplot2 or other plotting tools. It supports various tree model formats such as rpart, party, C50, and caret.

Usage

```
prepare_tree(tree_res, model = c("rpart", "party", "C50", "caret", "cforest"))
```

Arguments

tree_res	A list object containing tree plotting information, including a plot_data data frame.
model	A string indicating the tree model used. Options are "rpart", "party", "C50", or "caret".

Value

A list with two elements:

plot_data A data frame of node-level information with labels for visualization.

branches A data frame of edge (branch) coordinates for connecting parent and child nodes.

Examples

```
library(rpart)
library(partykit)
library(ggparty)
library(dplyr)
library(seriation)
data_all <- add_data_type(
  data_train = train_covid, data_test = test_covid
)
data <- prepare_features(
  data_all,
  target_lab = "Outcome",
  task = "classification"
)
train_tree <- train_tree(
  data_train = train_covid,
  target_lab = "Outcome", model = "rpart"
)
fit <- train_tree$fit
var_imp <- train_tree$var_imp
tree_res <- compute_tree(
  fit,
  model = "rpart", show = "test",
  data = data, target_lab = "Outcome",
  task = "classification"
)
prepare_tree(tree_res, model = "rpart")
```

Psychosis_Disorder *Psychosis Disorder Data*

Description

Ratings of positive and negative symptoms in psychosis disorders, based on Andreasen's Scale for Assessment of Positive Symptoms (SAPS) and Scale for Assessment of Negative Symptoms (SANS).

Usage

```
Psychosis_Disorder
```

Format

A data frame with 95 observations and 51 variables:

UNIQID Factor indicating disorder type.

AH1, AH2, AH3, AH4, AH5, AH6 Hallucinations subscale (SAPS).

DL1, DL2, DL3, DL4, DL5, DL6, DL7, DL8, DL9, DL10, DL11, DL12 Delusions subscale (SAPS).

BE1, BE2, BE3, BE4 Behavior subscale (SAPS).

TH1, TH2, TH3, TH4, TH5, TH6, TH7, TH8 Thought disorder subscale (SAPS).

NA1, NA2, NA3, NA4, NA5, NA6, NA7 Expression subscale (SANS).

NB1, NB2, NB3, NB4 Speech subscale (SANS).

NC1, NC2, NC3 Hygiene subscale (SANS).

ND1, ND2, ND3, ND4 Activity subscale (SANS).

NE1, NE2 Inattentiveness subscale (SANS).

Details

This data set comprises 95 subjects, of whom 69 were diagnosed with schizophrenia and 26 with bipolar disorder. All symptoms were rated on a six-point scale (0–5).

---

rf_dtGAP *Visualize a Single Tree from a Conditional Random Forest*

Description

Fits a `partykit::cforest` and visualizes one of its individual trees using the full dtGAP pipeline (decision tree + heatmap + evaluation).

Usage

```
rf_dtGAP(
  x = NULL,
  target_lab = NULL,
  show = c("all", "train", "test"),
  tree_index = 1L,
  ntree = 500L,
  mtry = NULL,
  rf_control = NULL,
  data_train = NULL,
  data_test = NULL,
  data_all = NULL,
  test_size = 0.3,
  task = c("classification", "regression"),
  trans_type = c("normalize", "scale", "percentize", "none"),
  col_proximity = c("pearson", "spearman", "kendall"),
```

```
  linkage_method = c("CT", "SG", "CP"),
  seriate_method = "TSP",
  cRGAR_w = 5,
  sort_by_data_type = TRUE,
  custom_layout = NULL,
  panel_space = 0.001,
  margin = 20,
  total_w = 297,
  total_h = 210,
  tree_p = 0.3,
  include_var_imp = TRUE,
  col_var_imp = "orange",
  var_imp_bar_width = 0.8,
  var_imp_fontsize = 5,
  split_var_bg = "darkgreen",
  split_var_fontsize = 5,
  Col_Prox_palette = "RdBu",
  Col_Prox_n_colors = 11,
  label_map = NULL,
  label_map_colors = NULL,
  type_palette = "Dark2",
  label_palette = "OrRd",
  n_label_color = 9,
  pred_ha_gap = unit(1, "mm"),
  prop_palette = gray,
  n_prop_colors = 11,
  Row_Prox_palette = "Spectral",
  Row_Prox_n_colors = 11,
  row_border = TRUE,
  row_gap = unit(1, "mm"),
  sorted_dat_palette = "Blues",
  sorted_dat_n_colors = 9,
  show_row_names = TRUE,
  row_names_gp = gpar(fontsize = 5),
  show_row_prox = TRUE,
  show_col_prox = TRUE,
  raw_value_col = NULL,
  lgd_direction = c("vertical", "horizontal"),
  x_eval_start = 15,
  y_eval_start = NULL,
  eval_text = 7,
  print_eval = TRUE,
  simple_metrics = FALSE
)
```

Arguments

x	Character. Name or label of the dataset.

target_lab	Character. Name of the target column.
show	Character. Which subset to show: "all", "train", or "test".
tree_index	Integer. Which tree to extract (1-based). Default is 1.
ntree	Integer. Number of trees in the forest (default 500).
mtry	Integer or NULL. Number of variables randomly sampled at each split. If NULL, uses the cforest default.
rf_control	A ctree_control object or NULL.
data_train	Data frame. Training data.
data_test	Data frame. Test data.
data_all	Data frame. Full dataset.
test_size	Numeric. Proportion for test split (default 0.3).
task	Character. "classification" or "regression".
trans_type	Character. Transformation type.
col_proximity	Character. Correlation method.
linkage_method	Character. Linkage method.
seriate_method	Character. Seriation method.
cRGAR_w	Integer. Window size for RGAR.
sort_by_data_type	Logical. Preserve data_type grouping.
custom_layout	Optional custom node positions.
panel_space	Numeric. Vertical spacing.
margin	Numeric. Margin in mm.
total_w	Numeric. Page width in mm.
total_h	Numeric. Page height in mm.
tree_p	Numeric. Tree panel proportion.
include_var_imp	Logical. Show importance barplot.
col_var_imp	Color for importance bars.
var_imp_bar_width	Numeric. Bar width.
var_imp_fontsize	Numeric. Font size for importance.
split_var_bg	Background for split variable names.
split_var_fontsize	Font size for split variable names.
Col_Prox_palette	Palette for correlation heatmap.
Col_Prox_n_colors	Number of correlation colors.
label_map	Named vector for label mapping.

label_map_colors
: Named vector of mapped label colors.

type_palette
: Palette for data_type.

label_palette
: Palette for labels.

n_label_color
: Number of label colors.

pred_ha_gap
: Gap between annotations.

prop_palette
: Probability gradient palette.

n_prop_colors
: Number of probability colors.

Row_Prox_palette
: Palette for row proximity.

Row_Prox_n_colors
: Number of row proximity colors.

row_border
: Draw cell borders.

row_gap
: Gap between annotation blocks.

sorted_dat_palette
: Palette for heatmap.

sorted_dat_n_colors
: Number of heatmap colors.

show_row_names
: Show row names.

row_names_gp
: Font settings for row names.

show_row_prox
: Show row proximity.

show_col_prox
: Show column proximity.

raw_value_col
: Colors for raw data values.

lgd_direction
: Legend direction.

x_eval_start
: Eval text x position.

y_eval_start
: Eval text y position.

eval_text
: Eval text font size.

print_eval
: Show evaluation results.

simple_metrics
: Use simple metrics.

Value

Draws the dtGAP visualization for the selected tree to the current graphics device. Called for its side effect; returns invisibly.

Examples

```
rf_dtGAP(
  data_train = train_covid,
  data_test = test_covid,
  target_lab = "Outcome",
  show = "test",
  tree_index = 1,
```

```
  ntree = 50,
  print_eval = FALSE
)
```

`rf_summary` *Random Forest Ensemble Summary*

Description

Fits a `partykit::cforest` and displays a multi-panel summary: variable importance barplot, OOB error curve, and optionally a representative tree (the tree with highest prediction agreement with the full ensemble).

Usage

```
rf_summary(
  x = NULL,
  target_lab = NULL,
  data_train = NULL,
  data_test = NULL,
  data_all = NULL,
  test_size = 0.3,
  task = c("classification", "regression"),
  ntree = 500L,
  mtry = NULL,
  rf_control = NULL,
  show_var_imp = TRUE,
  show_rep_tree = TRUE,
  top_n_vars = 15L,
  total_w = 297,
  total_h = 210
)
```

Arguments

`x`	Character. Dataset name/label. If NULL, inferred from data arguments.
`target_lab`	Character. Name of the target column.
`data_train`	Data frame. Training data.
`data_test`	Data frame. Test data.
`data_all`	Data frame. Full dataset.
`test_size`	Numeric. Proportion for test split (default 0.3).
`task`	Character. `"classification"` or `"regression"`.
`ntree`	Integer. Number of trees (default 500).
`mtry`	Integer or NULL. Variables per split.

rf_control	A ctree_control object or NULL.
show_var_imp	Logical. Show variable importance barplot (default TRUE).
show_rep_tree	Logical. Show representative tree info (default TRUE).
top_n_vars	Integer. How many top variables to show (default 15).
total_w	Numeric. Page width in mm (default 297).
total_h	Numeric. Page height in mm (default 210).

Value

A list (invisible) with:

forest	The fitted cforest object.
var_imp	Named numeric vector of variable importance.
rep_tree_index	Index of the representative tree.

Examples

```
rf_summary(
  data_train = train_covid,
  data_test = test_covid,
  target_lab = "Outcome",
  ntree = 50
)
```

---

save_dtGAP *Save dtGAP Visualization to File*

Description

Exports the dtGAP plot to PNG, PDF, or SVG format.

Usage

```
save_dtGAP(
  file,
  format = NULL,
  width = 297,
  height = 210,
  dpi = 300,
  bg = "white",
  ...
)
```

Arguments

file	Character. Output file path. The format is inferred from the file extension unless format is specified explicitly.
format	Character or NULL. One of "png", "pdf", or "svg". If NULL (default), inferred from file extension.
width	Numeric. Page width in mm (default 297, A4 landscape).
height	Numeric. Page height in mm (default 210, A4 landscape).
dpi	Numeric. Resolution for PNG output (default 300). Ignored for PDF and SVG.
bg	Character. Background color (default "white").
...	Additional arguments passed to dtGAP().

Value

Invisible file path of the created file.

Examples

```
save_dtGAP(
  file = tempfile(fileext = ".png"),
  data_train = train_covid,
  data_test = test_covid,
  target_lab = "Outcome",
  show = "test",
  print_eval = FALSE
)
```

---

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. One of "percentize","normalize","scale","none" passed to scale_norm().

Value

Numeric vector of the transformed x.

References

https://github.com/trangdata/treeheatr/blob/85be4a61e35a62285c95b553f03729721bb18a0b/R/utils.R

Examples

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

`sorted_mat` *Sort Feature Matrix by Tree and Correlation Structure*

Description

Orders samples and features based on tree-derived node grouping and correlation-based seriation.

Usage

```
sorted_mat(
  tree_res = NULL,
  target_lab = NULL,
  show = c("all", "train", "test"),
  trans_type = c("normalize", "scale", "percentize", "none"),
  col_proximity = c("pearson", "spearman", "kendall"),
  linkage_method = c("CT", "SG", "CP"),
  seriate_method = "TSP",
  w = 5,
  sort_by_data_type = TRUE
)
```

Arguments

`tree_res`	A list returned by compute_tree(), containing fit, dat, and plot_data.
`target_lab`	Character. Name of the target column to exclude from features.
`show`	Character. "train","test", or "all" to select subset before sorting.
`trans_type`	Character. One of "percentize","normalize","scale","none" passed to scale_norm().
`col_proximity`	Character. Correlation method: "pearson","spearman","kendall".
`linkage_method`	Character. Linkage for supervised distance: "CT","SG","CP".
`seriate_method`	Character. Seriation method for distance objects; see `seriation::list_seriation_methods("dist")` for all supported options. Default: `"TSP"`.
`w`	Integer. Window size for RGAR calculation.
`sort_by_data_type`	Logical. If TRUE, preserves data_type grouping within nodes.

Value

A list with:

- sorted_row_names, sorted_col_names
- row_pro_mat_sorted, col_pro_mat_sorted
- cRGAR_score
- sorted_test_matrix
- node_ids
- dat_sorted

Examples

```
library(rpart)
library(partykit)
library(ggparty)
library(dplyr)
library(seriation)
data <- add_data_type(
  data_all = Psychosis_Disorder
)
data <- prepare_features(
  data,
  target_lab = "UNIQID",
  task = "classification"
)
fit <- train_tree(
  data = data, target_lab = "UNIQID",
  model = "rpart"
)$fit
tree_res <- compute_tree(
  fit,
  model = "rpart", show = "all",
  data = data, target_lab = "UNIQID",
  task = "classification"
)
sorted_dat <- sorted_mat(
  tree_res,
  target_lab = "UNIQID",
  show = "all", trans_type = "none",
  seriate_method = "GW_average",
  sort_by_data_type = FALSE
)
sorted_dat$row_pro_mat_sorted
sorted_dat$col_pro_mat_sorted
sorted_dat$cRGAR_score
```

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

`train_rf` *Fit a Conditional Random Forest*

Description

Fits a conditional random forest using `partykit::cforest()` and returns the forest object along with variable importance scores.

Usage

```
train_rf(
  data_train,
  target_lab,
  task = c("classification", "regression"),
  ntree = 500L,
  mtry = NULL,
  control = NULL
)
```

Arguments

`data_train`	Data frame. Training data.
`target_lab`	Character. Name of the target column.
`task`	Character. `"classification"` or `"regression"`.
`ntree`	Integer. Number of trees (default 500).
`mtry`	Integer or NULL. Number of variables randomly sampled at each split. If NULL, uses the `cforest` default.
`control`	A `ctree_control` object or NULL.

Value

A list with elements:

`forest`	The fitted `cforest` object.
`var_imp`	A named numeric vector of relative variable importance (scaled to sum to 1 and rounded to two decimals).
`ntree`	Integer. Number of trees in the forest.

Examples

```
data(train_covid)
rf_res <- train_rf(train_covid, target_lab = "Outcome", ntree = 50)
rf_res$var_imp
```

`train_tree` *Fit a Decision Tree Model*

Description

Fits a decision tree to training data using one of several supported tree implementations (rpart, party, C50, or via caret) and returns a standardized party object along with variable importance scores.

Usage

```
train_tree(
  data_train = NULL,
  data = NULL,
  target_lab = NULL,
  model = c("rpart", "party", "C50", "caret"),
  task = c("classification", "regression"),
  control = NULL
)
```

Arguments

`data_train`	Data frame. Explicit training set. If NULL, will be subset from `data` by `data_type == 'train'`.
`data`	Data frame. Combined dataset with a `data_type` column when `data_train` is NULL.
`target_lab`	Character. Name of the target column to predict.
`model`	Character. Which implementation to use: one of "rpart", "party", "C50", or "caret".
`task`	Character. Type of task: "classification" or "regression".
`control`	List or control object. Optional control parameters passed to the chosen tree function.

Value

A list with elements:

`fit`	A party object representing the fitted tree.
`var_imp`	A named numeric vector of relative variable importance (scaled to sum to 1 and rounded to two decimals).

Examples

```
library(partykit)
library(C50)
library(caret)
```

```
data(train_covid)
train_tree(data_train = train_covid, target_lab = "Outcome", model = "rpart")
train_tree(data_train = train_covid, target_lab = "Outcome", model = "C50")
train_tree(data_train = train_covid, target_lab = "Outcome", model = "caret")

data(Psychosis_Disorder)
data <- add_data_type(data_all = Psychosis_Disorder)
data <- prepare_features(data, target_lab = "UNIQID", task = "classification")
train_tree(
  data = data, target_lab = "UNIQID", model = "party",
  control = ctree_control(minbucket = 15)
)
```

`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