Package 'bartMan'

April 15, 2024

Type Package

Title Create Visualisations for BART Models

Version 0.1.0

Maintainer Alan Inglis <alan.inglis@mu.ie>

Description Investigating and visualising Bayesian Additive Regression Tree (BART) (Chipman, H. A., George, E. I., & McCulloch, R. E. 2010) <doi:10.1214/09-AOAS285> model fits. We construct conventional plots to analyze a model's performance and stability as well as create new tree-based plots to analyze variable importance, interaction, and tree structure. We employ Value Suppressing Uncertainty Palettes (VSUP) to construct heatmaps that display variable importance and interactions jointly using colour scale to represent posterior uncertainty. Our visualisations are designed to work with the most popular BART R packages available, namely 'BART' Rodney Sparapani and Charles Spanbauer and Robert McCulloch 2021 <doi:10.18637/jss.v097.i01>, 'dbarts' (Vincent Dorie 2023) <https://CRAN.R-project.org/package=dbarts>, and 'bartMachine' (Adam Kapelner and Justin Bleich 2016) <doi:10.18637/jss.v070.i04>.

License GPL (>= 2)

Imports colorspace, cowplot, DendSer, dplyr, ggiraph, ggnewscale, ggplot2, ggraph, grid, grDevices, gtable, igraph, patchwork, purrr, rlang, rrapply, scales, stats, tidybayes, tidygraph, tidyr, tidytreatment, utils, tibble, BART, bartMachine, dbarts, rJava, cli

Suggests ROCR, ggridges, ggforce, lvplot

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Alan Inglis [aut, cre], Andrew Parnell [aut], Catherine Hurley [aut], Claus Wilke [ctb] (Developer of VSUP script)

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-04-15 15:40:07 UTC

R topics documented:

acceptRate *acceptRate*

Description

Plots the acceptance rate of trees from a BART model.

Usage

```
acceptRate(trees)
```

Arguments

trees	A data frame created by extractTreeData function. Displays a division on the plot to separate prior and post burn-in iterations.

Value

A ggplot object plot of acceptance rate.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
# Load the dbarts package to access the bart function
library(dbarts)
# Get Data
df <- na.omit(airquality)
# Create Simple dbarts Model For Regression:
set.seed(1701)
dbartModel <- bart(df[2:6], df[,1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)
# Tree Data
trees_data <- extractTreeData(model = dbartModel, data = df)
acceptRate(trees = trees_data)}
```

bartClassifDiag *bartClassifDiag*

Description

Displays a selection of diagnostic plots for a BART model.

Usage

```
bartClassifDiag(
  model,
  data,
  response,
  threshold = "Youden",
  pNorm = FALSE,
  showInterval = TRUE,
  combineFactors = FALSE
)
```

Arguments

model	a model created from either the BART, dbarts, or bartMachine package.
data	A dataframe
response	The name of the response for the fit.
threshold	A dashed line on some plots to indicate a chosen threshold value. by default the Youden index is shown.
pNorm	apply pnorm to the y-hat data
showInterval	LOGICAL if TRUE then show 5% and 95% quantile intervals.
combineFactors	Whether or not to combine dummy variables (if present) in display.

Value

A selection of diagnostic plots

---

bartDiag *bartDiag*

Description

Displays a selection of diagnostic plots for a BART model.

Usage

```
bartDiag(
  model,
  data,
  response,
  burnIn = 0,
  threshold = "Youden",
  pNorm = FALSE,
  showInterval = TRUE,
  combineFactors = FALSE
)
```

Arguments

model	a model created from either the BART, modelarts, or bartMachine package.
data	A dataframe used to build the model.
response	The name of the response for the fit.
burnIn	Trace plot will only show iterations above selected burn in value.
threshold	A dashed line on some plots to indicate a chosen threshold value (classification only). by default the Youden index is shown.
pNorm	apply pnorm to the y-hat data (classification only).
showInterval	LOGICAL if TRUE then show 5% and 95% quantile intervals on ROC an PC curves (classification only).
combineFactors	Whether or not to combine dummy variables (if present) in display.

Value

A selection of diagnostic plots.

Examples

```
# For Regression
# Generate Friedman data
fData <- function(n = 200, sigma = 1.0, seed = 1701, nvar = 5) {
  set.seed(seed)
  x <- matrix(runif(n * nvar), n, nvar)
  colnames(x) <- paste0("x", 1:nvar)
  Ey <- 10 * sin(pi * x[, 1] * x[, 2]) + 20 * (x[, 3] - 0.5)^2 + 10 * x[, 4] + 5 * x[, 5]
  y <- rnorm(n, Ey, sigma)
  data <- as.data.frame(cbind(x, y))
  return(data)
}
f_data <- fData(nvar = 10)
x <- f_data[, 1:10]
y <- f_data$y

# Create dbarts model
library(dbarts)
set.seed(1701)
dbartModel <- bart(x, y, ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

bartDiag(model = dbartModel, response = "y", burnIn = 100, data = f_data)


# For Classification
data(iris)
iris2 <- iris[51:150, ]
iris2$Species <- factor(iris2$Species)

# Create dbarts model
dbartModel <- bart(iris2[, 1:4], iris2[, 5], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)
```

```
bartDiag(model = dbartModel, data = iris2, response = iris2$Species)
```

`bartRegrDiag` *bartRegrDiag*

Description

Displays a selection of diagnostic plots for a BART model.

Usage

```
bartRegrDiag(model, response, burnIn = 0, data, combineFactors = FALSE)
```

Arguments

`model`	a model created from either the BART, modelarts, or bartMachine package.
`response`	The name of the response for the fit.
`burnIn`	Trace plot will only show iterations above selected burn in value.
`data`	A dataframe used to build the model.
`combineFactors`	Whether or not to combine dummy variables (if present) in display.

Value

A selection of diagnostic plots

`clusterTrees` *Cluster Trees by Variable*

Description

Reorders a list of tree structures based on the clustering of variables within each tree.

Usage

```
clusterTrees(tree_list)
```

Arguments

`tree_list`	A list of trees, where each tree is expected to have a 'var' column.

Value

A list of trees reordered based on the clustering of variables.

combineDummy *Update Dummy Variable Names*

Description

This function updates the ‘var‘ column in the ‘structure‘ component of the ‘trees‘ list, replacing dummy variable names derived from factor variables with their original factor variable names.

Usage

```
combineDummy(trees)
```

Arguments

trees	A list containing at least two components: ‘data‘ and ‘structure‘. ‘data‘ should be a dataframe, and ‘structure‘ a dataframe that includes a ‘var‘ column.

Details

The function first identifies factor variables in ‘trees$data‘, then checks each entry in ‘trees$structure$var‘ for matches with these factor variables. If a match is found, indicating a dummy variable, the entry is replaced with the original factor variable name.

Value

The modified ‘trees‘ list with updated ‘var‘ column entries in ‘trees$structure‘.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
# Load the dbarts package to access the bart function
 library(dbarts)
 # Create Simple dbarts Model with Dummies
 set.seed(1701)
 dbartModel <- bart(iris[2:5], iris[,1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)
 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = iris)
 combined_trees <- combineDummy(trees = trees_data)
}
```

`extractTreeData` *extractTreeData*

Description

Creates a list of all tree attributes for a model created by either the BART, dbarts or bartMachine packages.

Usage

```
extractTreeData(model, data)
```

Arguments

`model`	Model created from either the BART, dbarts or bartMachine packages.
`data`	a data frame used to build the BART model.

Value

A list containing the extracted and processed tree data. This list includes:

1. **Tree Data Frame**: A data frame containing tree attributes.
2. **Variable Name**: The names of the variables used in building the model.
3. **nMCMC**: The total number of iterations (posterior draws) after burn-in.
4. **nTree**: The total number of trees grown in the sum-of-trees model.
5. **nVar**: The total number of covariates used in the model.

The object created by the 'extractTreeData' function encompasses these elements, facilitating detailed analysis and visualisation of BART model components.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
# Load the dbarts package to access the bart function
library(dbarts)
# Get Data
df <- na.omit(airquality)
# Create Simple dbarts Model For Regression:
set.seed(1701)
dbartModel <- bart(df[2:6], df[,1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)
# Tree Data
trees_data <- extractTreeData(model = dbartModel, data = df)
}
```

getChildren *Generate Child and Parent Node Relationships*

Description

Populates 'childLeft', 'childRight', and 'parent' columns in the dataset to establish parent-child relationships between nodes based on tree structure.

Usage

```
getChildren(data)
```

Arguments

data	A data frame with tree structure, including 'iteration', 'treeNum', 'node', and 'depth' columns, along with a 'terminal' indicator.

Value

The modified data frame with 'childLeft', 'childRight', and 'parent' columns added, detailing the tree's parent-child node relationships.

Examples

```
data("tree_data_example")
# Create Terminal Column
tree_data_example <- transform(tree_data_example,
                               terminal = ifelse(is.na(var),
                               TRUE,
                               FALSE))
# Get depths
depthList <- lapply(split(tree_data_example, ~treeNum + iteration),
                    function(x) cbind(x, depth = node_depth(x)-1))
# Turn into data frame
tree_data_example <- dplyr::bind_rows(depthList, .id = "list_id")
# Add node number sequntially
tree_data_example$node <- with(tree_data_example,
                               ave(seq_along(iteration),
                               list(iteration, treeNum),
                               FUN = seq_along))
# get children
getChildren(data = tree_data_example)
```

`getObservations` *Get Observations Falling into Each Node*

Description

This function determines which observations from a given dataset fall into which nodes of a tree, based on a tree structure defined in ‘treeData‘. The treeData object must include ‘iteration‘, ‘treeNum‘, ‘var‘, and ‘splitValue‘ columns.

Usage

```
getObservations(data, treeData)
```

Arguments

`data`	A data frame used to build BART model.
`treeData`	A data frame representing the tree structure, including the necessary columns ‘iteration‘, ‘treeNum‘, ‘var‘, and ‘splitValue‘.

Value

A modified version of ‘treeData‘ that includes two new columns: ‘obsNode‘ and ‘noObs‘. ‘obsNode‘ lists the observations falling into each node, and ‘noObs‘ provides the count of observations for each node.

Examples

```
data("tree_data_example")
# Create Terminal Column
tree_data_example <- transform(tree_data_example,
                               terminal = ifelse(is.na(var),
                               TRUE,
                                FALSE))
# Create Split Value Column
tree_data_example <- transform(tree_data_example,
                                splitValue = ifelse(terminal == FALSE,
                                value,
                                NA_integer_))
# get the observations
getObservations(data = input_data, treeData = tree_data_example)
```

get_stump_colour_for_legend *Determines the stump color for a legend based on its mean value*

Description

This function is internal and is used to compute the color of a stump for the purpose of legend display, based on the mean value relative to specified limits.

Usage

```
get_stump_colour_for_legend(lims, mean_value, palette)
```

Arguments

lims	A numeric vector of length 2 specifying the limits within which the mean value falls.
mean_value	The mean value for which the color needs to be determined.
palette	A character vector of colors representing the palette from which the color is selected.

Value

A character string specifying the color corresponding to the mean value.

guide_colourfan *Colourfan guide*

Description

Colourfan guide

Usage

```
guide_colourfan(
  title = waiver(),
  title.x.position = "top",
  title.y.position = "right",
  title.theme = NULL,
  title.hjust = 0.5,
  title.vjust = NULL,
  label = TRUE,
  label.theme = NULL,
  barwidth = NULL,
  barheight = NULL,
```

```
  nbin = 32,
  reverse = FALSE,
  order = 0,
  available_aes = c("colour", "color", "fill"),
  ...
)

guide_colorfan(
  title = waiver(),
  title.x.position = "top",
  title.y.position = "right",
  title.theme = NULL,
  title.hjust = 0.5,
  title.vjust = NULL,
  label = TRUE,
  label.theme = NULL,
  barwidth = NULL,
  barheight = NULL,
  nbin = 32,
  reverse = FALSE,
  order = 0,
  available_aes = c("colour", "color", "fill"),
  ...
)
```

Arguments

`title`	Title
`title.x.position`	Title x position
`title.y.position`	Title y position
`title.theme`	Title theme
`title.hjust`	Title hjust
`title.vjust`	Title vjust
`label`	Label
`label.theme`	Label theme
`barwidth`	Barwidth
`barheight`	Barheight
`nbin`	Number of bins
`reverse`	Reverse
`order`	order
`available_aes`	Available aesthetics
`...`	Extra paramters

Value

A ‘grob‘ object representing a color fan. This ‘grob‘ can be added to a grid-based plot or a ggplot2 object to visualize a range of colors in a fan-like structure. Each segment of the fan corresponds to a color specified in the ‘colours‘ parameter, allowing for an intuitive representation of color gradients or palettes.

---

input_data *input_data*

Description

Small example of Friedman data following the formula:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e$$

Usage

```
input_data
```

Format

A data frame with 10 rows and 6 columns:

x1 Covariate

x2 Covariate

x3 Covariate

x4 Covariate

x5 Covariate

y Response ...

---

localProcedure *localProcedure*

Description

A variable selection approach performed by permuting the response.

Usage

```
localProcedure(
  model,
  data,
  response,
  numRep = 10,
  numTreesRep = NULL,
  alpha = 0.5,
  shift = FALSE
)
```

Arguments

model	Model created from either the BART, dbarts or bartMachine packages.
data	A data frame containing variables in the model.
response	The name of the response for the fit.
numRep	The number of replicates to perform for the BART null model's variable inclusion proportions.
numTreesRep	The number of trees to be used in the replicates. As suggested by Chipman (2009), a small number of trees is recommended (~20) to force important variables to used in the model. If NULL, then the number of trees from the true model is used.
alpha	The cut-off level for the thresholds.
shift	Whether to shift the inclusion proportion points by the difference in distance between the quantile and the value of the inclusion proportion point.

Value

A variable selection plot using the local procedure method.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
# Load the dbarts package to access the bart function
library(dbarts)

# Get Data
df <- na.omit(airquality)
# Create Simple dbarts Model For Regression:
set.seed(1701)
dbartModel <- bart(df[2:6], df[,1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)
localProcedure(model = dbartModel,
               data = df,
               numRep = 5,
               numTreesRep = 5,
               alpha = 0.5,
               shift = FALSE)
}
```

---

mdsBart *mdsBart*

Description

Multi-dimensional Scaling Plot of proximity matrix from a BART model.

Usage

```
mdsBart(
  trees,
  data,
  target,
  response,
  plotType = "rows",
  showGroup = TRUE,
  level = 0.95
)
```

Arguments

trees	A data frame created by ‘extractTreeData‘ function.
data	a dataframe used in building the model.
target	A target proximity matrix to
response	The name of the response for the fit.
plotType	Type of plot to show. Either ’interactive’ - showing interactive confidence ellipses. ’point’ - a point plot showing the average position of a observation. ’rows’ - displaying the average position of a observation number instead of points. ’all’ - show all observations (not averaged).
showGroup	Logical. Show confidence ellipses.
level	The confidence level to show. Default is 95% confidence level.

Value

For this function, the MDS coordinates are calculated for each iteration. Procrustes method is then applied to align each of the coordinates to a target set of coordinates. The returning result is then a clustered average of each point.

Examples

```
if (requireNamespace("dbarts", quietly = TRUE)) {
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6],
   df[, 1],
   ntree = 5,
   keeptrees = TRUE,
   nskip = 10,
   ndpost = 10
 )
 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
```

```
 # Cretae Porximity Matrix
 bmProx <- proximityMatrix(
   trees = trees_data,
   reorder = TRUE,
   normalize = TRUE,
   iter = 1
 )
 # MDS plot
 mdsBart(
   trees = trees_data, data = df, target = bmProx,
   plotType = "interactive", level = 0.25, response = "Ozone"
 )
}
```

---

node_depth *Calculate Node Depths in a Tree Data Frame*

Description

Computes the depth of each node in a given tree data frame, assuming a binary tree structure. Requires the tree data frame to contain a logical column ‘terminal‘ indicating terminal nodes.

Usage

```
node_depth(tree)
```

Arguments

tree	A data frame representing a tree, must contain a ‘terminal‘ column.

Value

A vector of depths corresponding to each node in the tree.

Examples

```
data("tree_data_example")
# Create Terminal Column
tree_data_example <- transform(tree_data_example, terminal = ifelse(is.na(var), TRUE, FALSE))
# Get depths
depthList <- lapply(split(tree_data_example, ~treeNum + iteration),
                    function(x) cbind(x, depth = node_depth(x)-1))
# Turn into data frame
tree_data_example <- dplyr::bind_rows(depthList, .id = "list_id")
```

pal_vsup *Variance suppressing uncertainty palette*

Description

Returns a palette function that turns ‘v‘ (value) and ‘u‘ (uncertainty) (both between 0 and 1) into colors.

Usage

```
pal_vsup(
  values,
  unc_levels = 4,
  max_light = 0.9,
  max_desat = 0,
  pow_light = 0.8,
  pow_desat = 1
)
```

Arguments

values	Color values to be used at minimum uncertainty. Needs to be a vector of length ‘2^unc_levels‘.
unc_levels	Number of discrete uncertainty levels. The number of discrete colors at each level doubles.
max_light	Maximum amount of lightening
max_desat	Maximum amount of desaturation
pow_light	Power exponent of lightening
pow_desat	Power exponent of desaturation

Value

A function that takes two parameters, ‘v‘ (value) and ‘u‘ (uncertainty), both expected to be in the range of 0 to 1, and returns a color. This color is determined by the specified ‘values‘ colors at minimum uncertainty, and modified according to the given ‘v‘ and ‘u‘ parameters to represent uncertainty by adjusting lightness and saturation. The resulting function is useful for creating color palettes that can encode both value and uncertainty in visualizations.

permVimp *permVimp*

Description

A variable selection approach which creates a null model by permuting the response, rebuilding the model, and calculating the inclusion proportion (IP) on the null model. The final result displayed is the original model's IP minus the null IP.

Usage

```
permVimp(model, data, response, numTreesPerm = NULL, plotType = "barplot")
```

Arguments

model	Model created from either the BART, dbarts or bartMachine packages.
data	A data frame containing variables in the model.
response	The name of the response for the fit.
numTreesPerm	The number of trees to be used in the null model. As suggested by Chipman (2009), a small number of trees is recommended (~20) to force important variables to used in the model. If NULL, then the number of trees from the true model is used.
plotType	Either a bar plot ('barplot') or a point plot ('point')

Value

A variable selection plot.

Examples

```
if (requireNamespace("dbarts", quietly = TRUE)) {
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6],
   df[, 1],
   ntree = 5,
   keeptrees = TRUE,
   nskip = 10,
   ndpost = 10
 )
 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 permVimp(model = dbartModel, data = df, response = 'Ozone', numTreesPerm = 2, plotType = 'point')
}
```

permVint *permVint*

Description

A variable interaction evaluation which creates a null model by permuting the response, rebuilding the model, and calculating the inclusion proportion (IP) of adjacent splits on the null model. The final result displayed is the original model's IP minus the null IP.

Usage

```
permVint(model, data, trees, response, numTreesPerm = NULL, top = NULL)
```

Arguments

model	Model created from either the BART, dbarts or bartMachine packages.
data	A data frame containing variables in the model.
trees	A data frame created by extractTreeData function.
response	The name of the response for the fit.
numTreesPerm	The number of trees to be used in the null model. As suggested by Chipman (2009), a small number of trees is recommended (~20) to force important variables to used in the model. If NULL, then the number of trees from the true model is used.
top	Display only the top X interactions.

Value

A variable interaction plot. Note that for a dbarts fit, due to the internal workings of dbarts, the null model is hard-coded to 20 trees, a burn-in of 100, and 1000 iterations. Both a BART and bartMachine null model will extract the identical parameters from the original model.

plotProximity *plotProximity*

Description

Plot a proximity matrix

Usage

```
plotProximity(
  matrix,
  pal = rev(colorspace::sequential_hcl(palette = "Blues 2", n = 100)),
  limit = NULL
)
```

Arguments

matrix	A matrix of proximities created by the proximityMatrix function
pal	A vector of colours to show proximity scores, for use with scale_fill_gradientn.
limit	Specifies the fit range for the color map for proximity scores.

Value

A plot of proximity values.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6],
   df[, 1],
   ntree = 5,
   keeptrees = TRUE,
   nskip = 10,
   ndpost = 10)
 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 # Create Proximity Matrix
 mProx <- proximityMatrix(trees = trees_data, reorder = TRUE, normalize = TRUE, iter = 1)
 # Plot
 plotProximity(matrix = mProx)
}
```

---

plotSingleTree *plotSingleTree*

Description

Plots individual trees.

Usage

```
plotSingleTree(trees, iter = 1, treeNo = 1, plotType = "icicle")
```

Arguments

trees	A data frame created by extractTreeData function
iter	The MCMC iteration or chain to plot.
treeNo	The tree number to plot.
plotType	What type of plot to display. either dendrogram or icicle.

Value

A plot of an individual tree

Examples

```
if (requireNamespace("dbarts", quietly = TRUE)) {
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6],
   df[, 1],
   ntree = 5,
   keeptrees = TRUE,
   nskip = 10,
   ndpost = 10
 )
 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 plotSingleTree(trees = trees_data, iter = 1, treeNo = 1)
}
```

---

plotTrees *Plot Trees with Customisations*

Description

This function plots trees from a list of tidygraph objects. It allows for various customisations such as fill colour based on node response or value, node size adjustments, and color palettes.

Usage

```
plotTrees(
  trees,
  iter = NULL,
  treeNo = NULL,
  fillBy = NULL,
  sizeNodes = FALSE,
  removeStump = FALSE,
  selectedVars = NULL,
  pal = rev(colorRampPalette(c("steelblue", "#f7fcfd", "orange"))(5)),
  center_Mu = TRUE,
  cluster = NULL
)
```

Arguments

trees	A data frame of trees.
iter	An integer specifying the iteration number of trees to be included in the output. If NULL, trees from all iterations are included.
treeNo	An integer specifying the number of the tree to include in the output. If NULL, all trees are included.
fillBy	A character string specifying the attribute to color nodes by. Options are 'response' for coloring nodes based on their mean response values or 'mu' for coloring nodes based on their predicted value, or NULL for no specific fill attribute.
sizeNodes	A logical value indicating whether to adjust node sizes. If TRUE, node sizes are adjusted; if FALSE, all nodes are given the same size.
removeStump	A logical value. If TRUE, then stumps are removed from plot.
selectedVars	A vector of selected variables to display. Either a character vector of names or the variables column number.
pal	A colour palette for node colouring. Palette is used when 'fillBy' is specified for gradient colouring.
center_Mu	A logical value indicating whether to center the color scale for the 'mu' attribute around zero. Applicable only when 'fillBy' is set to "mu".
cluster	A character string that specifies the criterion for reordering trees in the output. Currently supports "depth" for ordering by the maximum depth of nodes, and "var" for a clustering based on variables. If NULL, no reordering is performed.

Value

A ggplot object representing the plotted trees with the specified customisations.

Examples

```
if (requireNamespace("dbarts", quietly = TRUE)) {
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6],
   df[, 1],
   ntree = 5,
   keeptrees = TRUE,
   nskip = 10,
   ndpost = 10
 )
 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 plotTrees(trees = trees_data, fillBy = 'response', sizeNodes = TRUE)
}
```

print.hideHelper1 *print.hideHelper*

Description

This function hides parts from the print out but are still accessible via indexing.

Usage

```
## S3 method for class 'hideHelper1'
print(x, ...)
```

Arguments

x	A data frame of trees
...	Extra parameters

Value

No return value; this function is called for its side effect of printing a formatted summary of the tree data frame. It displays parts of the data frame, such as the tree structure and various counts (like number of MCMC iterations, number of trees, and number of variables), while keeping the complete data accessible via indexing.

proximityMatrix *proximityMatrix*

Description

Creates a matrix of proximity values.

Usage

```
proximityMatrix(trees, nRows, normalize = TRUE, reorder = TRUE, iter = NULL)
```

Arguments

trees	A list of tree attributes created by ‘extractTreeData‘ function.
nRows	Number of rows to consider.
normalize	Default is TRUE. Divide the total number of pairs of observations by the number of trees.
reorder	Default is TRUE. Whether to sort the matrix so high values are pushed to top left.
iter	Which iteration to use, if NULL the proximity matrix is calculated over all iterations.

Value

A matrix containing proximity values.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 # Create Proximity Matrix
 mProx <- proximityMatrix(trees = trees_data, reorder = TRUE, normalize = TRUE, iter = 1)
}
```

RangeBivariate *Constructor for bivariate range object*

Description

Constructor for bivariate range object

Usage

```
bivariate_range()
```

Format

An object of class `RangeBivariate` (inherits from `Range`, `ggproto`, `gg`) of length 2.

ScaleBivariate *Constructor for bivariate scale object*

Description

Constructor for bivariate scale object

Usage

```
bivariate_scale(
  aesthetics,
  palette,
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  rescaler = scales::rescale,
  oob = scales::censor,
  expand = waiver(),
  na.value = NA_real_,
  trans = "identity",
  guide = "none",
  super = ScaleBivariate,
  scale_name = "bivariate_scale"
)
```

Arguments

aesthetics	The names of the aesthetics that this scale works with.
palette	A palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values (e.g., `scales::area_pal()`).
name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
breaks	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.
labels	One of: • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object • A character vector giving labels (must be same length as `breaks`) • An expression vector (must be the same length as breaks). See ?plotmath for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
limits	Data frame with two columns of length two each defining the limits for the two data dimensions.
rescaler	Either one rescaling function applied to both data dimensions or list of two rescaling functions, one for each data dimension.

oob	One of: • Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation. • The default (scales::censor()) replaces out of bounds values with NA. • scales::squish() for squishing out of bounds values into range. • scales::squish_infinite() for squishing infinite values into range.
expand	For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function expansion() to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.
na.value	Missing values will be replaced with this value.
trans	Either one transformation applied to both data dimensions or list of two transformations, one for each data dimension. Transformations can be given as either the name of a transformation object or the object itself. See ['ggplot2::continuous_scale()'] for details.
guide	A function used to create a guide or its name. See guides() for more information.
super	The super class to use for the constructed scale
scale_name	The name of the scale that should be used for error messages associated with this scale.

Format

An object of class ScaleBivariate (inherits from Scale, ggproto, gg) of length 15.

`sort_trees_by_depthMax` *Sort Trees by Maximum Depth*

Description

Sort Trees by Maximum Depth

Usage

```
sort_trees_by_depthMax(tree_list)
```

Arguments

tree_list	List of 'tbl_graph' trees.

Value

Sorted list of 'tbl_graph' trees by decreasing maximum depth.

splitDensity *splitDensity*

Description

Density plots of the split value for each variable.

Usage

```
splitDensity(
  trees,
  data,
  bandWidth = NULL,
  panelScale = NULL,
  scaleFactor = NULL,
  display = "histogram"
)
```

Arguments

trees	A list of trees created using the trees function.
data	Data frame containing variables from the model.
bandWidth	Bandwidth used for density calculation. If not provided, is estimated from the data.
panelScale	If TRUE, the default, relative scaling is calculated separately for each panel. If FALSE, relative scaling is calculated globally. @param scaleFactor A scaling factor to scale the height of the ridgelines relative to the spacing between them. A value of 1 indicates that the maximum point of any ridgeline touches the baseline right above, assuming even spacing between baselines.
scaleFactor	A numerical value to scale the plot.
display	Choose how to display the plot. Either histogram, facet wrap, ridges or display both the split value and density of the predictor by using dataSplit.

Value

A faceted group of density plots

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
```

```
dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

# Tree Data
trees_data <- extractTreeData(model = dbartModel, data = df)
splitDensity(trees = trees_data, data = df, display = 'ridge')
}
```

---

terminalFunction *Generate Terminal Node Indicator*

Description

Adds a boolean ‘terminal‘ column to the dataset indicating whether each node is terminal.

Usage

```
terminalFunction(data)
```

Arguments

data	A data frame containing tree structure information with at least ‘treeNum‘, ‘iteration‘, and ‘depth‘ columns.

Value

The modified data frame with an additional ‘terminal‘ column.

---

train_bivariate *Train range for bivariate scale*

Description

Train range for bivariate scale

Usage

```
train_bivariate(new, existing = NULL)
```

Arguments

new	New data on which to train.
existing	Existing range

Value

A tibble containing two columns, ‘range1‘ and ‘range2‘, each representing the trained continuous range based on the new and existing data. This function is used to update or define the scales of a bivariate analysis by considering both new input data and any existing range specifications.

`treeBarPlot` *Plot Frequency of Tree Structures*

Description

Generates a bar plot showing the frequency of different tree structures represented in a list of tree graphs. Optionally, it can filter to show only the top N trees and handle stump trees specially.

Usage

```
treeBarPlot(trees, iter = NULL, topTrees = NULL, removeStump = FALSE)
```

Arguments

`trees`	A list of tree graphs to display
`iter`	Optional; specifies the iteration to display.
`topTrees`	Optional; the number of top tree structures to display. If NULL, displays all.
`removeStump`	Logical; if TRUE, trees with no edges (stumps) are excluded from the display

Details

This function processes a list of tree structures to compute the frequency of each unique structure, represented by a bar plot. It has options to exclude stump trees (trees with no edges) and to limit the plot to the top N most frequent structures.

Value

A 'ggplot' object representing the bar plot of tree frequencies.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 plot <- treeBarPlot(trees = trees_data, topTrees = 3, removeStump = TRUE)
}
```

treeDepth *treeDepth*

Description

A plot of tree depth over iterations.

Usage

```
treeDepth(trees)
```

Arguments

trees	A list of tree attributes created using the extractTreeData function.

Value

A plot of average tree depths over iteration

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 treeDepth(trees = trees_data)
}
```

treeList *Generate a List of Tree Structures from BART Model Output*

Description

This function takes a dataframe of trees, which is output from a BART model, and organizes it into a list of tree structures. It allows for filtering based on iteration number, tree number, and optionally reordering based on the maximum depth of nodes or variables.

Usage

```
treeList(trees, iter = NULL, treeNo = NULL)
```

Arguments

trees	A dataframe that contains the tree structures generated by a BART model. Expected columns include iteration, treeNum, parent, node, obsNode,
iter	An integer specifying the iteration number of trees to be included in the output. If NULL, trees from all iterations are included.
treeNo	An integer specifying the number of the tree to include in the output. If NULL, all trees are included.

Value

A list of tidygraph objects, each representing the structure of a tree. Each tidygraph object includes node and edge information necessary for visualisation.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 library(ggplot2)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 trees_list <- treeList(trees_data)
}
```

---

treeNodes *treeNodes*

Description

A plot of number of nodes over iterations.

Usage

```
treeNodes(trees)
```

Arguments

trees	A list of tree attributes created using the extractTreeData function.

Value

A plot of tree number of nodes over iterations.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 treeNodes(trees = trees_data)
}
```

---

tree_dataframe *Transform tree data into a structured dataframe*

Description

This function takes raw data and a tree structure, then processes it to form a detailed and structured dataframe. The data is transformed to indicate terminal nodes, calculate leaf values, and determine split values. It then assigns labels, calculates node depth, and establishes hierarchical relationships within the tree. Additional metadata about the tree, such as maximum depth, parent and child node relationships, and observation nodes are also included. The final dataframe is organized and enriched with necessary attributes for further analysis.

Usage

```
tree_dataframe(data, trees, response = NULL)
```

Arguments

`data`	A dataframe containing the raw data used for building the tree.
`trees`	A dataframe representing the initial tree structure, including variables and values for splits.
`response`	Optional character of the name of the response variable in your BART model. Including the response will remove it from the list elements 'Variable names' and 'nVar'.

Value

A list containing a detailed dataframe of the tree structure ('structure') with added information such as node depth, parent and child nodes, and observational data, along with meta-information about the tree like variable names ('varNames'), number of MCMC iterations ('nMCMC'), number of trees ('nTree'), and number of variables ('nVar').

Examples

```
data("input_data")
data("tree_data_example")
my_trees <- tree_dataframe(data = input_data, trees = tree_data_example, response = "y")
```

`tree_data_example` *tree_data_example*

Description

Small example of tree data, like that obtained when using ‘extractTreeData()‘ function.

Usage

```
tree_data_example
```

Format

A data frame with 14 rows and 4 columns representing the structure of trees:

var Variable name used for splitting.

value The value in a node (i.e., either the split value or leaf value).

iteration Iteration Number.

treeNum Tree Number in the iteration. ...

`vimpBart` *vimpBart*

Description

A matrix with nMCMC rows with each variable as a column. Each row represents an MCMC iteration. For each variable, the total count of the number of times that variable is used in a tree is given.

Usage

```
vimpBart(trees, type = "prop")
```

Arguments

`trees`	A data frame created by ‘extractTreeData‘ function.
`type`	What value to return. Either the raw count ’val’, the proportion ’prop’, the column means of the proportions ’propMean’, or the median of the proportions ’propMedian’.

Value

A matrix of importance values

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 vimpBart(trees_data, type = 'prop')
 }
```

---

vimpPlot *vimpPlot*

Description

Plot the variable importance for a BART model with the 25 quantile.

Usage

```
vimpPlot(trees, type = "prop", plotType = "barplot", metric = "median")
```

Arguments

trees	A data frame created by ‘extractTreeData‘ function.
type	What value to return. Either the raw count ’count’ or the proportions ’prop’ averaged over iterations.
plotType	Which type of plot to return. Either a barplot ’barplot’ with the quantiles shown as a line, a point plot with the quantiles shown as a gradient ’point’, or a letter-value plot ’lvp’.
metric	Whether to show the ’mean’ or ’median’ importance values. Note, this has no effect when using plotType = ’lvp’.

Value

A plot of variable importance.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
 vimpPlot(trees = trees_data, plotType = 'point')
}
```

`vintPlot` *vintPlot*

Description

Plot the pair-wise variable interactions inclusion porportions for a BART model with the 25

Usage

```
vintPlot(trees, plotType = "barplot", top = NULL)
```

Arguments

`trees`	A data frame created by ‘extractTreeData‘ function.
`plotType`	Which type of plot to return. Either a barplot ’barplot’ with the quantiles shown as a line, a point plot with the quantiles shown as a gradient ’point’, or a letter-value plot ’lvp’.
`top`	Display only the top X metrics (does not apply to the letter-value plot).

Value

A plot of variable importance.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
```

```
dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

# Tree Data
trees_data <- extractTreeData(model = dbartModel, data = df)
vintPlot(trees = trees_data, top = 5)
}
```

---

viviBart *viviBart*

Description

Returns a list containing a dataframe of variable importance summaries and a dataframe of variable interaction summaries.

Usage

```
viviBart(trees, out = "vivi")
```

Arguments

trees	A data frame created by 'extractTreeData' function.
out	Choose to either output just the variable importance ('vimp'), the variable interaction ('vint'), or both ('vivi') (default).

Value

A list of dataframes of VIVI summaries.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
# Load the dbarts package to access the bart function
library(dbarts)
# Get Data
df <- na.omit(airquality)
# Create Simple dbarts Model For Regression:
set.seed(1701)
dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

# Tree Data
trees_data <- extractTreeData(model = dbartModel, data = df)
viviBart(trees = trees_data, out = 'vivi')
}
```

viviBartMatrix *viviBartMatrix*

Description

Returns a matrix or list of matrices. If type = 'standard' a matrix filled with vivi values is returned. If type = 'vsup' two matrices are returned. One with the actual values and another matrix of uncertainty values. If type = 'quantiles', three matrices are returned. One for the 25

Usage

```
viviBartMatrix(
  trees,
  type = "standard",
  metric = "propMean",
  metricError = "CV",
  reorder = FALSE
)
```

Arguments

trees	A data frame created by 'extractTreeData' function.
type	Which type of matrix to return. Either 'standard', 'vsup', 'quantiles'
metric	Which metric to use to fill the actual values matrix. Either 'propMean' or 'count'.
metricError	Which metric to use to fill the uncertainty matrix. Either 'SD', 'CV' or 'SE'.
reorder	LOGICAL. If TRUE then the matrix is reordered so high values are pushed to the top left.

Value

A heatmap plot showing variable importance on the diagonal and variable interaction on the off-diagonal.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)
```

```
# VSUP Matrix
vsupMat <- viviBartMatrix(trees = trees_data,
                          type = 'vsup',
                          metric = 'propMean',
                           metricError = 'CV')
}
```

---

viviBartPlot *viviBartPlot*

Description

Plots a Heatmap showing variable importance on the diagonal and variable interaction on the off-diagonal with uncertainty included.

Usage

```
viviBartPlot(
  matrix,
  intPal = NULL,
  impPal = NULL,
  intLims = NULL,
  impLims = NULL,
  uncIntLims = NULL,
  uncImpLims = NULL,
  unc_levels = 4,
  max_desat = 0.6,
  pow_desat = 0.2,
  max_light = 0.6,
  pow_light = 1,
  angle = 0,
  border = FALSE,
  label = NULL
)
```

Arguments

matrix	Matrices, such as that returned by viviBartMatrix, of values to be plotted.
intPal	A vector of colours to show interactions, for use with scale_fill_gradientn. Palette number has to be 2^x/2
impPal	A vector of colours to show importance, for use with scale_fill_gradientn. Palette number has to be 2^x/2
intLims	Specifies the fit range for the color map for interaction strength.
impLims	Specifies the fit range for the color map for importance.

uncIntLims	Specifies the fit range for the color map for interaction strength uncertainties.
uncImpLims	Specifies the fit range for the color map for importance uncertainties.
unc_levels	The number of uncertainty levels
max_desat	The maximum desaturation level.
pow_desat	The power of desaturation level.
max_light	The maximum light level.
pow_light	The power of light level.
angle	The angle to rotate the x-axis labels. Defaults to zero.
border	Logical. If TRUE then draw a black border around the diagonal elements.
label	legend label for the uncertainty measure.

Value

Either a heatmap, VSUP, or quantile heatmap plot.

Examples

```
if(requireNamespace("dbarts", quietly = TRUE)){
 # Load the dbarts package to access the bart function
 library(dbarts)
 # Get Data
 df <- na.omit(airquality)
 # Create Simple dbarts Model For Regression:
 set.seed(1701)
 dbartModel <- bart(df[2:6], df[, 1], ntree = 5, keeptrees = TRUE, nskip = 10, ndpost = 10)

 # Tree Data
 trees_data <- extractTreeData(model = dbartModel, data = df)

 # VSUP Matrix
 vsupMat <- viviBartMatrix(trees = trees_data,
                           type = 'vsup',
                           metric = 'propMean',
                           metricError = 'CV')
 # Plot
 viviBartPlot(vsupMat, label = 'CV')
 }
```

Index