Package ‘HVT’

May 7, 2024

Type Package

Date 2024-05-02

Title Constructing Hierarchical Voronoi Tessellations and Overlay Heatmaps for Data Analysis

Version 24.5.1

Description Facilitates building topology preserving maps for rich multivariate data. Credits to Mu Sigma for their continuous support throughout the development of the package.

License Apache License 2.0

Encoding UTF-8

Imports MASS, deldir, grDevices, splancs, conf.design, stats, dplyr, purrr, magrittr, polyclip, ggplot2, tidyr, scales, cluster, reshape2,data.table, ggforce, plyr, rlang, gganimate, gifski, markovchain, methods

Depends R (>= 3.6.0)

BugReports https://github.com/Mu-Sigma/HVT/issues

URL https://github.com/Mu-Sigma/HVT

RoxygenNote 7.3.1

Suggests rmarkdown, testthat, geozoo, plotly, DT, patchwork, sp, Hmisc, gridExtra, gtable, htmlwidgets, installr, skimr, tibble, devtools, tidyverse, DataExplorer, htmltools, corrplot,knitr,kableExtra

VignetteBuilder knitr

NeedsCompilation no

Author Zubin Dowlaty [aut], Mu Sigma, Inc. [cre]

Maintainer ``Mu Sigma, Inc.'' <ird.experiencelab@mu-sigma.com>

Repository CRAN

Date/Publication 2024-05-02 02:20:00 UTC

Collate 'Add_boundary_points.R' 'Corrected_Tessellations.R' 'Transform_Coordinates.R' 'ScaleMat.R' 'DelaunayInfo.R' 'Delete_Outpoints.R' 'VQ_codebookSplit.R' 'diagPlot.R' 'diagSuggestion.R' 'displayTable.R' 'edaPlots.R' 'getCellId.R' 'getCentroids.R' 'getCentroids_for_opti.R' 'getOptimalCentroids.R' 'getTransitionProbability.R' 'global.R' 'hvq.R' 'madPlot.R' 'plotAnimatedFlowmap.R' 'plotHVT.R' 'plotModelDiagnostics.R' 'plotNovelCells.R' 'plotQuantErrorHistogram.R' 'plotStateTransition.R' 'reconcileTransitionProbability.R' 'removeNovelty.R' 'scoreHVT.R' 'scoreLayeredHVT.R' 'trainHVT.R'

R topics documented:

`displayTable` *Table for displaying summary*

Description

This is the main function for displaying summary from model training and scoring

Usage

```
displayTable(
  data,
  columnName,
  value,
  tableType = "summary",
  scroll = TRUE,
  limit = 100
)
```

Arguments

`data`	List. A listed object from trainHVT or scoreHVT
`columnName`	Character. Name of the column that needs highlighting.
`value`	Numeric. The value above will be highlighted in red or green.
`tableType`	Character. Type of table to generate ('summary', 'compression')
`scroll`	Logical. A value to have a scroll or not in the table.
`limit`	Numeric. A value to indicate how many rows to display. Applicable for summary tableType.

Value

A consolidated table of results

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

See Also

`trainHVT`

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
displayTable(data = hvt.results[[3]]$compression_summary,
columnName = 'percentOfCellsBelowQuantizationErrorThreshold',
value = 0.8, tableType = "compression")

displayTable(data =hvt.results[[3]][['summary']], columnName= 'Quant.Error',
 value = 0.1, tableType = "summary")
```

`edaPlots` *plots for data analysis*

Description

This is the main function that provides exploratory data analysis plots

Usage

```
edaPlots(df, time_series = FALSE, time_column)
```

Arguments

`df`	Dataframe. A data frame object.
`time_series`	Logical. A value to indicate whether the dataset is time_series or not.
`time_column`	Character. The name of the time column in the data frame.

Value

Five objects which include time series plots, data distribution plots, box plots, correlation plot and a descriptive statistics table.

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
edaPlots(dataset, time_series = TRUE, time_column = 'date')
```

`getCellId` *Cell ID*

Description

Function to generate cell ID based on 1D sammons projection

Usage

```
getCellId(hvt.results, seed = 123)
```

Arguments

`hvt.results`	List. A list of hvt.results obtained from the trainHVT function.
`seed`	Numeric. Random Seed

Details

To generate cell id for the multivariate data, the data is being projected from n-dimensions to 1-dimension and the cell id is being assigned by ordering these values and finding the corresponding indexes. The output Cell id gets appended to the HVT model.

Value

Object containing Cell.ID mappings for the given hvt.results list.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>

`getOptimalCentroids` *getOptimalCentroids*

Description

Get Optimal Centroids

Usage

```
getOptimalCentroids(
  x,
  iter.max,
  algorithm,
  n_cells,
  seed = 100,
  function_to_calculate_distance_metric,
  function_to_calculate_error_metric = c("mean", "max"),
  quant.err,
  distance_metric = "L1_Norm",
  quant_method = c("kmeans", "kmedoids"),
  ...
)
```

Arguments

x	Data Frame. A dataframe of multivariate data. Each row corresponds to an observation, and each column corresponds to a variable. Missing values are not accepted.
algorithm	String. The type of algorithm used for quantization. Available algorithms are Hartigan and Wong, "Lloyd", "Forgy", "MacQueen". (default is "Hartigan-Wong")
n_cells	Numeric. Indicating the number of nodes per hierarchy.
seed	Numeric. Random Seed.
function_to_calculate_distance_metric	Function. The function is to find 'L1_Norm" or "L2_Norm" distances. L1_Norm is selected by default.
function_to_calculate_error_metric	Character. The error metric can be "mean" or "max". mean is selected by default
quant.err	Numeric. The quantization error for the algorithm.
distance_metric	Character. The distance metric to calculate inter point distance. It can be 'L1_Norm" or "L2_Norm". L1_Norm is selected by default.
quant_method	Character. The quant_method can be "kmeans" or "kmedoids". kmeans is selected by default

Details

The raw data is first scaled and this scaled data is supplied as input to the vector quantization algorithm. Vector quantization technique uses a parameter called quantization error. This parameter acts as a threshold and determines the number of levels in the hierarchy. It means that, if there are 'n' number of levels in the hierarchy, then all the clusters formed till this level will have quantization error equal or greater than the threshold quantization error. The user can define the number of clusters in the first level of hierarchy and then each cluster in first level is sub-divided into the same number of clusters as there are in the first level. This process continues and each group is divided into smaller clusters as long as the threshold quantization error is met. The output of this technique will be hierarchically arranged vector quantized data.

Value

values	List. A list showing observations assigned to a cluster.
maxQE	List. A list corresponding to maximum QE values for each cell.
meanQE	List. A list corresponding to mean QE values for each cell.
centers	List. A list of quantization error for all levels and nodes.
nsize	List. A list corresponding to number of observations in respective groups.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>

`getTransitionProbability` *Creating Transition Probabilities list*

Description

This is the main function to create transition probabilities list. The transition probability table quantifies the likelihood of transitioning from one state to another. States: The table includes the current states and the possible next states. Probabilities: For each current state, it lists the probability of transitioning to each of the next possible states.

Usage

```
getTransitionProbability(df, cellid_column, time_column)
```

Arguments

`df`	Data frame. The input data frame should contain two columns, cell ID from scoreHVT function and time stamp of that dataset.
`cellid_column`	Character. Name of the column containing cell IDs.
`time_column`	Character. Name of the column containing time stamps.

Value

Prints and stores a nested list of data frames with transition probabilities.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>

See Also

`trainHVT`
`scoreHVT`

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
```

```
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)
table <- getTransitionProbability(dataset, cellid_column = "cell_id",time_column = "time_sta
```

`hvq` *hvq*

Description

Hierarchical Vector Quantization

Usage

```
hvq(
  x,
  min_compression_perc = NA,
  n_cells = NA,
  depth = 3,
  quant.err = 10,
  seed = 300,
  algorithm = "Hartigan-Wong",
  distance_metric = c("L1_Norm", "L2_Norm"),
  error_metric = c("mean", "max"),
  quant_method = c("kmeans", "kmedoids")
)
```

Arguments

`x`	Data Frame. A dataframe of multivariate data. Each row corresponds to an observation, and each column corresponds to a variable. Missing values are not accepted.
`min_compression_perc`	Numeric. An integer indicating the minimum percent compression rate to be achieved for the dataset
`n_cells`	Numeric. Indicating the number of nodes per hierarchy.
`depth`	Numeric. Indicating the hierarchy depth (or) the depth of the tree (1 = no hierarchy, 2 = 2 levels, etc..)
`quant.err`	Numeric. The quantization error for the algorithm.
`seed`	Numeric. Random Seed.
`algorithm`	String. The type of algorithm used for quantization. Available algorithms are Hartigan and Wong, "Lloyd", "Forgy", "MacQueen". (default is "Hartigan-Wong")
`distance_metric`	character. The distance metric can be 'L1_Norm" or "L2_Norm". L1_Norm is selected by default.

`error_metric`	character. The error metric can be "mean" or "max". mean is selected by default
`quant_method`	character. The quant_method can be "kmeans" or "kmedoids". kmeans is selected by default

Details

The raw data is first scaled and this scaled data is supplied as input to the vector quantization algorithm. Vector quantization technique uses a parameter called quantization error. This parameter acts as a threshold and determines the number of levels in the hierarchy. It means that, if there are 'n' number of levels in the hierarchy, then all the clusters formed till this level will have quantization error equal or greater than the threshold quantization error. The user can define the number of clusters in the first level of hierarchy and then each cluster in first level is sub-divided into the same number of clusters as there are in the first level. This process continues and each group is divided into smaller clusters as long as the threshold quantization error is met. The output of this technique will be hierarchically arranged vector quantized data.

Value

`clusters`	List. A list showing each ID assigned to a cluster.
`nodes.clust`	List. A list corresponding to nodes' details.
`idnodes`	List. A list of ID and segments similar to `nodes.clust` with additional columns for nodes ID.
`error.quant`	List. A list of quantization error for all levels and nodes.
`plt.clust`	List. A list of logical values indicating if the quantization error was met.
`summary`	Summary. Output table with summary.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>

See Also

`plotHVT`

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
dataset_hvt <- dataset[,-c(1)]
hvqOutput = hvq(dataset_hvt, n_cells = 5, depth = 2, quant.err = 0.2,
distance_metric='L1_Norm',error_metric='mean',quant_method="kmeans")
```

madPlot *Mean Absolute Deviation Plot*

Description

Function to create Mean Absolute Deviation Plot

Usage

```
madPlot(hvt.scoring, ...)
```

Arguments

`hvt.scoring`	List. A list of hvt.scoring obtained from the scoreHVT function.
`...`	The ellipsis is passed to it as additional argument. (Used internally)

Details

This function plots percentage anomalies vs mean absolute deviation for test data. The plot helps in deciding an optimal MAD value for the use case.

Value

Mean Absolute Deviation Plot

`mad_plot`	ggplot plot. A plot with percentage anomalies on y axis and mean absolute deviation values on xaxis.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>

See Also

scoreHVT

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                     DAX = EuStockMarkets[, "DAX"],
                     SMI = EuStockMarkets[, "SMI"],
                     CAC = EuStockMarkets[, "CAC"],
                     FTSE = EuStockMarkets[, "FTSE"])
#adding this step especially for this function
rownames(EuStockMarkets) <- dataset$date
train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]
hvt_summary <- list()
```

```
hvt_summary<- trainHVT(train,n_cells = 15, depth = 1, quant.err = 0.2,
                       distance_metric = "L1_Norm", error_metric = "mean",
                       projection.scale = 10, normalize = TRUE,seed = 123,
                       quant_method = "kmeans")
score_var <- scoreHVT(test, hvt_summary, child.level = 2, mad.threshold = 0.2)
madPlot(hvt.scoring=score_var)
```

`plotAnimatedFlowmap` *Generating flow maps and animations based on transition probabilities*

Description

This is the main function for generating flow maps and animations based on transition probabilities including self states and excluding self states. Flow maps are a type of data visualization used to represent the transition probability of different states. Animations are the gifs used to represent the movement of data through the cells.

Usage

```
plotAnimatedFlowmap(
  hvt_model_output,
  transition_probability_df,
  df,
  animation = NULL,
  flow_map = NULL,
  fps_time = 1,
  fps_state = 1,
  time_duration = 2,
  state_duration = 2,
  cellid_column,
  time_column
)
```

Arguments

`hvt_model_output`	List. Output from a trainHVT function.
`transition_probability_df`	List. Output from getTransitionProbability function
`df`	Data frame. The input dataframe should contain two columns, cell ID from scoreHVT function and time stamp of that dataset.
`animation`	Character. Type of animation ('state_based', 'time_based', 'All' or NULL)
`flow_map`	Character. Type of flow map ('self_state', 'without_self_state', 'All' or NULL)
`fps_time`	Numeric. A numeric value for the frames per second of the time transition gif. (Must be a numeric value and a factor of 100). Default value is 1.

`fps_state`	Numeric. A numeric value for the frames per second of the state transition gif. (Must be a numeric value and a factor of 100). Default value is 1.
`time_duration`	Numeric. A numeric value for the duration of the time transition gif. Default value is 2.
`state_duration`	Numeric. A numeric value for the duration of the state transition gif. Default value is 2.
`cellid_column`	Character. Name of the column containing cell IDs.
`time_column`	Character. Name of the column containing time stamps

Value

A list of flow maps and animation gifs.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>, Vishwavani <vishwavani@mu-sigma.com>

See Also

`trainHVT`
`scoreHVT`
`getTransitionProbability`

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])

hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")

scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)

table <- getTransitionProbability(dataset, cellid_column = "cell_id",time_column = "time_sta
plots <- plotAnimatedFlowmap(hvt_model_output = hvt.results, transition_probability_df = tab
df = dataset, animation = 'All', flow_map = 'All',fps_time = 1,fps_state =  1,time_duration
state_duration = 2,cellid_column = "cell_id", time_column = "time_stamp")
```

plotHVT — *Plot the hierarchical tessellations.*

Description

This is the main plotting function to construct hierarchical voronoi tessellations in 1D,2D or Interactive surface plot.

Usage

```
plotHVT(
  hvt.results,
  line.width = 0.5,
  color.vec = "black",
  pch1 = 21,
  centroid.size = 1.5,
  title = NULL,
  maxDepth = NULL,
  child.level,
  hmap.cols,
  quant.error.hmap = NULL,
  cell_id = FALSE,
  n_cells.hmap = NULL,
  label.size = 0.5,
  sepration_width = 7,
  layer_opacity = c(0.5, 0.75, 0.99),
  dim_size = 1000,
  plot.type = "2Dhvt"
)
```

Arguments

`hvt.results`	(2DProj/2Dhvt/2Dheatmap/surface_plot) List. A list containing the output of `trainHVT` function which has the details of the tessellations to be plotted.
`line.width`	(2Dhvt/2Dheatmap) Numeric Vector. A vector indicating the line widths of the tessellation boundaries for each level.
`color.vec`	(2Dhvt/2Dheatmap) Vector. A vector indicating the colors of the boundaries of the tessellations at each level.
`pch1`	(2Dhvt/2Dheatmap) Numeric. Symbol of the centroids of the tessellations (parent levels). Default value is 21.
`centroid.size`	(2Dhvt/2Dheatmap) Numeric. Size of centroids of first level tessellations.
`title`	(2Dhvt) Character. Set a title for the plot. (default = NULL)
`maxDepth`	(2Dhvt) Numeric. An integer indicating the number of levels. (default = NULL)

`child.level`	(2Dheatmap/surface_plot) Numeric. Indicating the level for which the heat map is to be plotted.
`hmap.cols`	(2Dheatmap/surface_plot) Numeric or Character. The column number or column name from the dataset indicating the variables for which the heat map is to be plotted.
`quant.error.hmap`	(2Dheatmap) Numeric. A number indicating the quantization error threshold.
`cell_id`	(2Dhvt) Logical. To indicate whether the plot should have Cell IDs or not for the first layer. (default = FALSE)
`n_cells.hmap`	(2Dheatmap/surface_plot) Numeric. An integer indicating the number of cells/clusters per hierarchy (level)
`label.size`	(2Dheatmap) Numeric. The size by which the tessellation labels should be scaled. (default = 0.5)
`sepration_width`	(surface_plot) Numeric. An integer indicating the width between two levels
`layer_opacity`	(surface_plot) Numeric. A vector indicating the opacity of each layer/ level
`dim_size`	(surface_plot) Numeric. An integer indicating the dimension size used to create the matrix for the plot
`plot.type`	Character. An option to indicate which type of plot should be generated. Accepted entries are '1D','2Dproj','2Dhvt','2Dheatmap'and 'surface_plot'. Default value is '2Dhvt'.

Value

plot object containing the visualizations of reduced dimension(1D/2D) for the given dataset.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>

See Also

trainHVT

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")

#change the 'plot.type' argument to '2Dproj' or '2DHVT' to visualize respective plots.
plotHVT(hvt.results, plot.type='1D')

#change the 'plot.type' argument to 'surface_plot' to visualize the Interactive surface plot
plotHVT(hvt.results,child.level = 1,
hmap.cols = "DAX", plot.type = '2Dheatmap')
```

plotModelDiagnostics

Make the diagnostic plots for hierarchical voronoi tessellations

Description

This is the main function that generates diagnostic plots for hierarchical voronoi tessellations models and scoring.

Usage

```
plotModelDiagnostics(model_obj)
```

Arguments

`model_obj`	List. A list obtained from the trainHVT function or scoreHVT function

Value

For trainHVT, Minimum Intra-DataPoint Distance Plot, Minimum Intra-Centroid Distance Plot Mean Absolute Deviation Plot, Distribution of Number of Observations in Cells, for Training Data and Mean Absolute Deviation Plot for Validation Data are plotted. For scoreHVT Mean Absolute Deviation Plot for Training Data and Validation Data are plotted

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>

See Also

plotHVT

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans",diagnose = TRUE,
                        hvt_validation = TRUE)
plotModelDiagnostics(hvt.results)
```

plotNovelCells *Plot the identified outlier cell(s) in the voronoi tessellation map.*

Description

This is the main plotting function to construct hierarchical voronoi tessellations and highlight the outlier cells

Usage

```
plotNovelCells(
  plot.cells,
  hvt.map,
  line.width = c(0.6),
  color.vec = c("#141B41"),
  pch = 21,
  centroid.size = 0.5,
  title = NULL,
  maxDepth = 1
)
```

Arguments

`plot.cells`	Vector. A vector indicating the cells to be highlighted in the map
`hvt.map`	List. A list containing the output of `trainHVT` function which has the details of the tessellations to be plotted
`line.width`	Numeric Vector. A vector indicating the line widths of the tessellation boundaries for each level
`color.vec`	Vector. A vector indicating the colors of the boundaries of the tessellations at each level
`pch`	Numeric. Symbol of the centroids of the tessellations (parent levels) Default value is 21.
`centroid.size`	Numeric. Size of centroids of first level tessellations. Default value is 0.5
`title`	String. Set a title for the plot. (default = NULL)
`maxDepth`	Numeric. An integer indicating the number of levels. (default = NULL)

Value

Returns a ggplot object containing hierarchical voronoi tessellation plot highlighting the outlier cells

Author(s)

Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>

See Also

`trainHVT`
`plotHVT`

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")
#selected 55,58 are for demo purpose
plotNovelCells(c(55,58),hvt.results)
```

`plotQuantErrorHistogram` *Make the quantization error plots for training and scoring.*

Description

This is the function that produces histograms displaying the distribution of Quantization Error (QE) values for both train and test datasets, highlighting mean values with dashed lines for quick evaluation.

Usage

```
plotQuantErrorHistogram(hvt.results, hvt.scoring)
```

Arguments

`hvt.results`	List. A list of hvt.results obtained from the trainHVT function.
`hvt.scoring`	List. A list of hvt.scoring obtained from the scoreHVT function.

Value

Returns the ggplot object containing the quantization error distribution plots for the given HVT results of training and scoring

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>

See Also

`plotHVT`

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
#Split in train and test
train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]

hvt.results<- trainHVT(train,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE, quant_method = "kmeans")
scoring <- scoreHVT(test, hvt.results)
plotQuantErrorHistogram(hvt.results, scoring)
```

`plotStateTransition` *Creating State Transition Plot*

Description

This is the main function to create a state transition plot from a data frame. A state transition plot is a type of data visualization used to represent the changes or transitions in states over time for a given system. State refers to a particular condition or status of a cell at a specific point in time. Transition refers to the change of state for a cell from one condition to another over time.

Usage

```
plotStateTransition(
  df,
  sample_size = NULL,
  line_plot = NULL,
  cellid_column,
  time_column
)
```

Arguments

`df`	Data frame. The Input data frame should contain two columns. Cell ID from scoreHVT function and time stamp of that dataset.
`sample_size`	Numeric. An integer indicating the fraction of the data frame to visualize in the plot. Default value is 0.2
`line_plot`	Logical. A logical value indicating to create a line plot. Default value is NULL.
`cellid_column`	Character. Name of the column containing cell IDs.
`time_column`	Character. Name of the column containing time stamps.

Value

A plotly object representing the state transition plot for the given data frame.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>

See Also

trainHVT
scoreHVT

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])

hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)

plotStateTransition(dataset, sample_size = 1, cellid_column = "cell_id",time_column = "time_
```

`reconcileTransitionProbability` *Reconciliation of Transition Probability*

Description

This is the main function for creating reconciliation plots and tables which helps in comparing the transition probabilities calculated manually and from markovchain function

Usage

```
reconcileTransitionProbability(
  df,
  hmap_type = NULL,
  cellid_column,
  time_column
)
```

Arguments

`df`	Data frame. The input data frame should contain two columns, cell ID from scoreHVT function and timestamp of that dataset.
`hmap_type`	Character. ('self_state', 'without_self_state', or 'All')
`cellid_column`	Character. Name of the column containing cell IDs.
`time_column`	Character. Name of the column containing timestamps

Value

A list of plotly heatmap objects and tables representing the transition probability heatmaps.

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>, Vishwavani <vishwavani@mu-sigma.com>

See Also

`trainHVT`
`scoreHVT`

Examples

```
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])

hvt.results<- trainHVT(dataset,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results)
cell_id <- scoring$scoredPredictedData$Cell.ID
time_stamp <- dataset$date
dataset <- data.frame(cell_id, time_stamp)

reconcileTransitionProbability(dataset, hmap_type = "All",
cellid_column = "cell_id", time_column = "time_stamp")
```

`removeNovelty` *Remove identified novelty cell(s)*

Description

This function is used to remove the identified novelty cells.

Usage

```
removeNovelty(outlier_cells, hvt_results)
```

Arguments

`outlier_cells`	Vector. A vector with the cell number of the identified novelty
`hvt_results`	List. A list having the results of the compressed map i.e. output of `trainHVT` function

Value

A list of two items

`[[1]]`	Dataframe of novelty cell(s)
`[[2]]`	Dataframe without the novelty cell(s) from the dataset used in model training

Author(s)

Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>

See Also

`trainHVT`
`scoreLayeredHVT`

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method="kmeans")
identified_Novelty_cells <<- c(2, 10)
output_list <- removeNovelty(identified_Novelty_cells, hvt.results)
data_with_novelty <- output_list[[1]]
data_without_novelty <- output_list[[2]]
```

`scoreHVT` *Score which cell each point in the test dataset belongs to.*

Description

This function scores each data point in the test dataset based on a trained hierarchical Voronoi tessellations model.

Usage

```
scoreHVT(
  data,
  hvt.results.model,
  child.level = 1,
  mad.threshold = 0.2,
  line.width = c(0.6, 0.4, 0.2),
  color.vec = c("navyblue", "slateblue", "lavender"),
  normalize = TRUE,
  seed = 300,
  distance_metric = "L1_Norm",
  error_metric = "max",
  yVar = NULL
)
```

Arguments

`data`	Data frame. A data frame containing the test dataset.
`hvt.results.model`	List. A list obtained from the trainHVT function
`child.level`	Numeric. A number indicating the depth for which the heat map is to be plotted.
`mad.threshold`	Numeric. A numeric value indicating the permissible Mean Absolute Deviation.
`line.width`	Vector. A vector indicating the line widths of the tessellation boundaries for each layer.
`color.vec`	Vector. A vector indicating the colors of the tessellation boundaries at each layer.
`normalize`	Logical. A logical value indicating if the dataset should be normalized. When set to TRUE, the data (testing dataset) is standardized by 'mean' and 'sd' of the training dataset referred from the trainHVT(). When set to FALSE, the data is used as such without any changes.
`seed`	Numeric. Random Seed to preserve the repeatability
`distance_metric`	Character. The distance metric can be L1_Norm(Manhattan) or L2_Norm(Eucledian). L1_Norm is selected by default. The distance metric is used to calculate the distance between an n dimensional point and centroid. The distance metric can be different from the one used during training.
`error_metric`	Character. The error metric can be mean or max. max is selected by default. max will return the max of m values and mean will take mean of m values where each value is a distance between a point and centroid of the cell.
`yVar`	Character. A character or a vector representing the name of the dependent variable(s)

Value

Dataframe containing scored data, plots and summary

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>

See Also

`trainHVT`
`plotHVT`

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date
# Split in train and test
train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]
#model training
hvt.results<- trainHVT(train,n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(test, hvt.results)
data_scored <- scoring$scoredPredictedData
```

`scoreLayeredHVT` *Score which cell and what layer each data point in the test dataset belongs to*

Description

This function that scores the cell and corresponding layer for each data point in a test dataset using three hierarchical vector quantization (HVT) models (Map A, Map B, Map C) and returns a data frame containing the scored layer output. The function incorporates the scored results from each map and merges them to provide a comprehensive result.

Usage

```
scoreLayeredHVT(
  data,
  hvt_mapA,
  hvt_mapB,
  hvt_mapC,
  mad.threshold = 0.2,
  normalize = TRUE,
  seed = 300,
  distance_metric = "L1_Norm",
  error_metric = "max",
  child.level = 1,
  yVar = NULL
)
```

Arguments

`data`	Data Frame. A data frame containing test dataset. The data frame should have all the variable(features) used for training.
`hvt_mapA`	A list of hvt.results.model obtained from trainHVT function while performing ‘trainHVT()‘ on train data
`hvt_mapB`	A list of hvt.results.model obtained from trainHVT function while performing ‘trainHVT()‘ on data with novelty(s)
`hvt_mapC`	A list of hvt.results.model obtained from trainHVT function while performing ‘trainHVT()‘ on data without novelty(s)
`mad.threshold`	Numeric. A number indicating the permissible Mean Absolute Deviation
`normalize`	Logical. A logical value indicating if the dataset should be normalized. When set to TRUE, the data (testing dataset) is standardized by ’mean’ and ’sd’ of the training dataset referred from the trainHVT(). When set to FALSE, the data is used as such without any changes. (Default value is TRUE).
`seed`	Numeric. Random Seed.
`distance_metric`	Character. The distance metric can be L1_Norm(Manhattan) or L2_Norm(Eucledian). L1_Norm is selected by default. The distance metric is used to calculate the distance between an n dimensional point and centroid. The distance metric can be different from the one used during training.
`error_metric`	Character. The error metric can be mean or max. max is selected by default. max will return the max of m values and mean will take mean of m values where each value is a distance between a point and centroid of the cell.
`child.level`	Numeric. A number indicating the level for which the heat map is to be plotted.
`yVar`	Character. A character or a vector representing the name of the dependent variable(s)

Value

Dataframe containing scored layer output

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>, Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>,Somya Shambhawi <somya.shambhawi@mu-sigma.com>

See Also

trainHVT
plotHVT

Examples

```
data("EuStockMarkets")
dataset <- data.frame(date = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$date

train <- EuStockMarkets[1:1302, ]
test <- EuStockMarkets[1303:1860, ]

###MAP-A
hvt_mapA <- trainHVT(train, n_cells = 150, depth = 1, quant.err = 0.1,
                    distance_metric = "L1_Norm", error_metric = "max",
                    normalize = TRUE,quant_method = "kmeans")

identified_Novelty_cells <- c(127,55,83,61,44,35,27,77)
output_list <- removeNovelty(identified_Novelty_cells, hvt_mapA)
data_with_novelty <- output_list[[1]]
data_with_novelty <- data_with_novelty[, -c(1,2)]

### MAP-B
hvt_mapB <- trainHVT(data_with_novelty,n_cells = 10, depth = 1, quant.err = 0.1,
                    distance_metric = "L1_Norm", error_metric = "max",
                    normalize = TRUE,quant_method = "kmeans")
data_without_novelty <- output_list[[2]]

### MAP-C
hvt_mapC <- trainHVT(data_without_novelty,n_cells = 135,
                    depth = 1, quant.err = 0.1, distance_metric = "L1_Norm",
                    error_metric = "max", quant_method = "kmeans",
                    normalize = TRUE)

##SCORE LAYERED
data_scored <- scoreLayeredHVT(test, hvt_mapA, hvt_mapB, hvt_mapC)
```

---

trainHVT *Constructing Hierarchical Voronoi Tessellations*

Description

This is the main function to construct hierarchical voronoi tessellations. This is done using hierarchical vector quantization(hvq). The data is represented in 2D coordinates and the tessellations are plotted using these coordinates as centroids. For subsequent levels, transformation is performed on the 2D coordinates to get all the points within its parent tile. Tessellations are plotted using these transformed points as centroids.

Usage

```
trainHVT(
  dataset,
  min_compression_perc = NA,
  n_cells = NA,
  depth = 1,
  quant.err = 0.2,
  projection.scale = 10,
  normalize = FALSE,
  seed = 279,
  distance_metric = c("L1_Norm", "L2_Norm"),
  error_metric = c("mean", "max"),
  quant_method = c("kmeans", "kmedoids"),
  scale_summary = NA,
  diagnose = FALSE,
  hvt_validation = FALSE,
  train_validation_split_ratio = 0.8
)
```

Arguments

`dataset`	Data frame. A data frame, with numeric columns (features) will be used for training the model.
`min_compression_perc`	Numeric. An integer, indicating the minimum compression percentage to be achieved for the dataset. It indicates the desired level of reduction in dataset size compared to its original size.
`n_cells`	Numeric. An integer, indicating the number of cells per hierarchy (level).
`depth`	Numeric. An integer, indicating the number of levels. A depth of 1 means no hierarchy (single level), while higher values indicate multiple levels (hierarchy).
`quant.err`	Numeric. A number indicating the quantization error threshold. A cell will only breakdown into further cells if the quantization error of the cell is above the defined quantization error threshold.
`projection.scale`	Numeric. A number indicating the scale factor for the tessellations to visualize the sub-tessellations well enough. It helps in adjusting the visual representation of the hierarchy to make the sub-tessellations more visible.
`normalize`	Logical. A logical value indicating if the dataset should be normalized. When set to TRUE, scales the values of all features to have a mean of 0 and a standard deviation of 1 (Z-score).
`seed`	Numeric. A Random Numeric Seed to preserve the repeatability.

`distance_metric`	Character. The distance metric can be L1_Norm(Manhattan) or L2_Norm(Eucledian). L1_Norm is selected by default. The distance metric is used to calculate the distance between an n dimensional point and centroid.
`error_metric`	Character. The error metric can be mean or max. max is selected by default. max will return the max of m values and mean will take mean of m values where each value is a distance between a point and centroid of the cell.
`quant_method`	Character. The quantization method can be kmeans or kmedoids. Kmeans uses means (centroids) as cluster centers while Kmedoids uses actual data points (medoids) as cluster centers. kmeans is selected by default.
`scale_summary`	List. A list with user-defined mean and standard deviation values for all the features in the dataset. Pass the scale summary when normalize is set to FALSE.
`diagnose`	Logical. A logical value indicating whether user wants to perform diagnostics on the model. Default value is FALSE.
`hvt_validation`	Logical. A logical value indicating whether user wants to holdout a validation set and find mean absolute deviation of the validation points from the centroid. Default value is FALSE.
`train_validation_split_ratio`	Numeric. A numeric value indicating train validation split ratio. This argument is only used when hvt_validation has been set to TRUE. Default value for the argument is 0.8

Value

A Nested list that contains the hierarchical tessellation information. This list has to be given as input argument to plot the tessellations.

`[[1]]`	A list containing information related to plotting tessellations. This information will include coordinates, boundaries, and other details necessary for visualizing the tessellations
`[[2]]`	A list containing information related to Sammon's projection coordinates of the data points in the reduced-dimensional space.
`[[3]]`	A list containing detailed information about the hierarchical vector quantized data along with a summary section containing no of points, Quantization Error and the centroids for each cell.
`[[4]]`	A list that contains all the diagnostics information of the model when diagnose is set to TRUE. Otherwise NA.
`[[5]]`	A list that contains all the information required to generates a Mean Absolute Deviation (MAD) plot, if hvt_validation is set to TRUE. Otherwise NA
`[[6]]`	A list containing detailed information about the hierarchical vector quantized data along with a summary section containing no of points, Quantization Error and the centroids for each cell which is the output of 'hvq'
`[[7]]`	model info: A list that contains model-generated timestamp, input parameters passed to the model and the validation results

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>, Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>

See Also

`plotHVT`

Examples

```
data("EuStockMarkets")
hvt.results <- trainHVT(EuStockMarkets, n_cells = 60, depth = 1, quant.err = 0.1,
                        distance_metric = "L1_Norm", error_metric = "max",
                        normalize = TRUE,quant_method="kmeans")
```

`VQ_codebookSplit` *VQ_codebookSplit*

Description

Vector Quantization by codebook split method

Usage

```
VQ_codebookSplit(dataset, quant.err = 0.5, epsilon = NULL)
```

Arguments

`dataset`	Matrix. A matrix of multivariate data. Each row corresponds to an observation, and each column corresponds to a variable. Missing values are not accepted.
`quant.err`	Numeric. The quantization error for the algorithm.
`epsilon`	Numeric. The value to offset the codebooks during the codebook split. Default is NULL, in which case the value is set to quant.err parameter.

Details

Performs Vector Quantization by codebook split method. Initially, the entire dataset is considered to be one cluster where the codebook is the mean of the cluster. The quantization criteria is checked and the codebook is split such that the new codebooks are (codebook+epsilon) and (codebook-epsilon). The observations are reassigned to these new codebooks based on the nearest neighbour condition and the means recomputed for the new clusters. This is done iteratively until all the clusters meet the quantization criteria.

Value

`clusters`	List. A list showing each ID assigned to a cluster.
`nodes.clust`	List. A list corresponding to nodes' details.
`idnodes`	List. A list of ID and segments similar to `nodes.clust` with additional columns for nodes ID.
`error.quant`	List. A list of quantization error for all levels and nodes.
`plt.clust`	List. A list of logical values indicating if the quantization error was met.
`summary`	Summary. Output table with summary.

Author(s)

Sangeet Moy Das <sangeet.das@mu-sigma.com>

See Also

`plotHVT`

Examples

```
data("iris", package = "datasets")
iris <- iris[, 1:2]

vqOutput <- VQ_codebookSplit(iris, quant.err = 0.5)
```