Package 'HVT'

February 5, 2026

Type Package

Date 2026-02-05

Title Constructing Hierarchical Voronoi Tessellations and Overlay Heatmaps for Data Analysis

Version 26.1.2

Description Facilitates building topology preserving maps for data analysis.

License Apache License 2.0

Encoding UTF-8

Imports MASS, grDevices, splancs, stats, dplyr, NbClust, purrr, magrittr, ggplot2, tidyr, scales, cluster, reshape2, FNN, Rtsne, umap, plyr, markovchain, methods, deldir, gridExtra

Depends R (>= 4.0.0)

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

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

RoxygenNote 7.3.3

Suggests knitr,rmarkdown,testthat,geozoo, plotly, rlang, DT,patchwork,sp,Hmisc,data.table,gtable, htmlwidgets,skimr,tibble,devtools,gifski,gganimate, tidyverse,DataExplorer,htmltools,corrplot,kableExtra,polyclip,conf.design, furrr, future, dendextend

VignetteBuilder knitr

NeedsCompilation no

Maintainer Zubin Dowlaty <zubin.dowlaty@mu-sigma.com>

Repository CRAN

Date/Publication 2026-02-05 17:50:03 UTC

Collate 'Add_boundary_points.R' 'Corrected_Tessellations.R' 'Transform_Coordinates.R' 'ScaleMat.R' 'DelaunayInfo.R' 'Delete_Outpoints.R' 'HVTMSM_support.R' 'HVTMSMoptimization.R' 'OptimizationResults.R' 'VQ_codebookSplit.R' 'clusterPlot.R' 'clustHVT.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'
'mcmc_plots.R' 'msm.R' 'plotAnimatedFlowmap.R'
'plotExAnteRawSeries.R' 'plotHVT.R' 'plotModelDiagnostics.R'
'plotMsmKN.R' 'plotNovelCells.R' 'plotQuantErrorHistogram.R'
'plotStateTransition.R' 'plotZscore.R'
'reconcileTransitionProbability.R' 'removeNovelty.R'
'scoreHVT.R' 'scoreLayeredHVT.R' 'summary.R' 'trainHVT.R'

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

Contents

`clustHVT` *Performing Hierarchical Clustering Analysis*

Description

This is the main function to perform hierarchical clustering analysis which determines optimal number of clusters, perform AGNES clustering and plot the 2D cluster hvt plot.

Usage

```
clustHVT(
  data,
  trainHVT_results,
  scoreHVT_results,
  clustering_method = "ward.D2",
 indices = c("kl", "ch", "hartigan", "cindex", "db", "silhouette", "ratkowsky", "ball",
  "hubert", "dindex", "ptbiserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau",
    "dunn", "sdindex", "sdbw"),
  clusters_k = "champion",
  type = "default",
  domains.column = NULL,
  highlight_labels = NULL,
  only_dendro = FALSE
)
```

Arguments

data	Data frame. A data frame intended for performing hierarchical clustering analysis.
trainHVT_results	List. A list object which is obtained as a result of trainHVT function.
scoreHVT_results	List. A list object which is obtained as a result of scoreHVT function.
clustering_method	Character. The method used for clustering in both NbClust and hclust function. Defaults to 'ward.D2'.
indices	Character. The indices used for determining the optimal number of clusters in NbClust function. By default it uses 20 different indices.
clusters_k	Character. A parameter that specifies the number of clusters for the provided data. The options include "champion," "challenger," or any integer between 1 and 20. Selecting "champion" will use the highest number of clusters recommended by the 'NbClust' function, while "challenger" will use the second-highest recommendation. If a numerical value from 1 to 20 is provided, that exact number will be used as the number of clusters.
type	Character. The type of output required. Default is 'default'. Other option is 'plot' which will return only the clustered heatmap.
domains.column	Character. A vector of cluster names for the clustered heatmap. Used only when type is 'plot'.
highlight_labels	Vector. Numeric vector specifying problematic states and their neighboring states to be highlighted in the dendrogram. This argument is intended for internal use only. The default value is NULL.
only_dendro	Logical. A logical string specifies whether to generate only the dendrogram or to include the results as well. This argument is intended for internal use only. The default value is FALSE.

Value

A list object that contains the hierarchical clustering results.

[[1]]	Summary of k suggested by all indices with plots
[[2]]	A dendrogram plot with the selected number of clusters
[[3]]	A 2D Cluster HVT Plotly visualization that colors cells according to clusters derived from AGNES clustering results. It is interactive, allowing users to view cell contents by hovering over them

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

Examples

```
data("EuStockMarkets")
dataset <- data.frame(t = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$t
hvt.results<- trainHVT(dataset[-1],n_cells = 30, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
scoring <- scoreHVT(dataset, hvt.results, analysis.plots = TRUE, names.column = dataset[,1])
centroid_data <- scoring$centroidData
hclust_data_1 <- centroid_data[,2:3]
clust.results <- clustHVT(data = hclust_data_1,
                          trainHVT_results = hvt.results,
                          scoreHVT_results = scoring,
                          clusters_k = 'champion', indices = 'hartigan')
```

---

displayTable *function for displaying table*

Description

This is the main function for displaying data in table format

Usage

```
displayTable(data, scroll = TRUE, limit = 20)
```

Arguments

data	Data frame. The dataframe to be displayed in table format.
scroll	Logical. A value to have a scroll or not in the table. Default is TRUE.
limit	Numeric. A value to indicate how many rows to display. Default is 20.

Value

A table with proper formatting for html notebook

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

Examples

```
data <- datasets::EuStockMarkets
dataset <- as.data.frame(data)
displayTable(dataset)
```

`edaPlots` *plots for data analysis*

Description

This is the main function that provides exploratory data analysis plots

Usage

```
edaPlots(
  df,
  time_column,
  output_type = "summary",
  n_cols = -1,
  grey_bars = NULL
)
```

Arguments

`df`	Dataframe. A data frame object.
`time_column`	Character. The name of the time column in the data frame. Can be given only when the data is time series
`output_type`	Character. The name of the output to be displayed. Options are 'summary', 'histogram', 'boxplot', 'timeseries' & 'correlation'. Default value is summary.
`n_cols`	Numeric. A value to indicate how many columns to be included in the output.
`grey_bars`	List. A list of timestamps where each list contains two elements: start and end period, which will be highlighted in gray in the time series plot. Default value is NULL.

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 = as.numeric(EuStockMarkets[, "DAX"]),
SMI = as.numeric(EuStockMarkets[, "SMI"]),
CAC = as.numeric(EuStockMarkets[, "CAC"]),
FTSE = as.numeric(EuStockMarkets[, "FTSE"]))
edaPlots(dataset)
edaPlots(dataset, time_column = 'date', output_type = 'timeseries', n_cols = 4)
```

---

getTransitionProbability

Creating Transition Probability Matrix

Description

This is the main function to create transition probability matrix The transition probability matrix 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,
  type = "with_self_state"
)
```

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.
type	Character. A character value indicating the type of transition probability table to create. Accepted entries are "with_self_state" and "without_self_state".

Value

Stores a data frames with transition probabilities.

Author(s)

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

Examples

```
dataset <- data.frame(t = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
hvt.results<- trainHVT(dataset[-1],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$t
dataset <- data.frame(cell_id, time_stamp)
table <- getTransitionProbability(dataset, cellid_column = "cell_id",time_column = "time_stamp")
```

HVTMSMoptimization *HVT-MSM Optimization Function*

Description

This function runs multiple iterations/experiments over the dataset across different cell counts, and computes the MAE. If a given cell configuration not results in problematic states, the baseline simulation is used; otherwise, the simulation proceeds with problematic state handling. This process helps identify the best-performing model (lowest MAE). In essence, it performs trainHVT $\rightarrow$ scoreHVT $\rightarrow$ transition probability estimation on the training data, followed by msm simulation on the ex-post (test) dataset. Note: This is not applicable for ex-ante analysis.

Usage

```
HVTMSMoptimization(
  entire_dataset,
  expost_forecasting,
  time_column,
  ncell_range = 3:5,
  k_range = 2:9,
  nn_range = 2:7,
  num_simulations = 10,
  mae_metric,
 hvt_params = list(depth = 1, quant.err = 0.2, normalize = TRUE, distance_metric =
    "L1_Norm", error_metric = "max", dim_reduction_method = "sammon"),
  parallel = TRUE,
  verbose = TRUE,
  parallel_strategy = "multisession",
  max_workers = NULL
)
```

Arguments

`entire_dataset`	Dataframe. Train dataset for model training
`expost_forecasting`	Dataframe. Test dataset for ex-post forecasting
`time_column`	Character. Name of the time column
`ncell_range`	Numeric vector. Range of cells to run experiments (default 3:5)
`k_range`	Numeric vector. Range of clusters to run experiments (default: 2:9)
`nn_range`	Numeric vector. Range of nearest neighbors to run experiments (default: 2:7)
`num_simulations`	Integer. Number of simulations (default: 10)
`mae_metric`	Character. MAE calculation method(s): "mean", "median", "mode", or "all" (default: "median")
`hvt_params`	List. Set of parameters for Model Training (refer trainHVT)
`parallel`	Character. Whether to use parallel processing (default: TRUE)
`verbose`	Character. Whether to print progress information (default: TRUE)
`parallel_strategy`	Character. Parallel processing strategy: "multisession", "multicore", etc. (default: "multisession")
`max_workers`	Maximum number of parallel workers (default: NULL for auto-detect)

Value

List containing optimization results for each MAE metric:

`[[successful_results]]`	All successful parameter combinations
`[[nclust_best_results]]`	Best combination for each cell
`[[overall_best]]`	Overall best parameter combination
`[[all_results]]`	All attempted combinations with status

Author(s)

Vishwavani <vishwavani@mu-sigma.com>, Nithya <nithya.sn@mu-sigma.com>

msm *Performing Monte Carlo Simulations of Markov Chain*

Description

This is the main function to perform Monte Carlo simulations of Markov Chain on the dynamic forecasting of HVT States of a time series dataset. It includes both ex-post and ex-ante analysis offering valuable insights into future trends while resolving state transition challenges through clustering and nearest-neighbor methods to enhance simulation accuracy.

Usage

```
msm(
  state_time_data,
  forecast_type = "ex-post",
  initial_state,
  n_ahead_ante = 10,
  transition_probability_matrix,
  num_simulations = 500,
  trainHVT_results,
  scoreHVT_results,
  actual_data = NULL,
  raw_dataset,
  k = NULL,
  handle_problematic_states = TRUE,
  n_nearest_neighbor = NULL,
  show_simulation = TRUE,
  mae_metric = "median",
  time_column,
  precomputed_problematic_states = NULL,
  plot_mode = c("all", "mae-only")
)
```

Arguments

state_time_data	Dataframe. A dataframe containing state transitions over time(cell id and timestamp)
forecast_type	Character. A character to indicate the type of forecasting. Accepted values are "ex-post" or "ex-ante".
initial_state	Numeric. An integer indicatiog the state at t0.
n_ahead_ante	Numeric. A vector of n ahead points to be predicted further in ex-ante analyzes.
transition_probability_matrix	Dataframe. A dataframe of transition probabilities/ output of ‘getTransitionProbability‘ function

`num_simulations`
: Integer. A number indicating the total number of simulations to run. Default is 100.

`trainHVT_results`
: List.‘trainHVT‘ function output

`scoreHVT_results`
: List. ‘scoreHVT‘ function output

`actual_data`
: Dataframe. A dataframe for ex-post prediction period with the actual raw data values

`raw_dataset`
: DataFrame. A dataframe of input raw dataset from the mean and standard deviation will be calculated to scale up the predicted values

`k`
: Integer. A number of optimal clusters when handling problematic states. Default is 5.

`handle_problematic_states`
: Logical. To indicate whether to handle problematic states or not. Default is FALSE.

`n_nearest_neighbor`
: Integer. A number of nearest neighbors to consider when handling problematic states. Default is 1.

`show_simulation`
: Logical. To indicate whether to show the simulation lines in plots or not. Default is TRUE.

`mae_metric`
: Character. A character to indicate which metric to calculate Mean Absolute Error. Accepted entries are "mean", "median", or "mode". Default is "median".

`time_column`
: Character. The name of the column containing time data. Used for aligning and plotting the results.

`precomputed_problematic_states`
: Vector. An internal parameter used in HVTMSMoptimization call to pass the problematic states detected

`plot_mode`
: Character. A character to indicate what should be included in output list "mae-only" values (used in HVTMSMoptimization) and "all" (used in direct call, include plots and mae)

Value

A list object that contains the forecasting plots and MAE values.

`[[1]]`
: Simulation plots and MAE values for state and centroids plot

`[[2]]`
: Summary Table, Dendrogram plot and Clustered Heatmap when handle_problematic_states is TRUE

Author(s)

Vishwavani <vishwavani@mu-sigma.com>, Nithya <nithya.sn@mu-sigma.com>

Examples

```
dataset <- data.frame(t = as.numeric(time(EuStockMarkets)),
DAX = EuStockMarkets[, "DAX"],
SMI = EuStockMarkets[, "SMI"],
CAC = EuStockMarkets[, "CAC"],
FTSE = EuStockMarkets[, "FTSE"])
hvt.results<- trainHVT(dataset[,-1],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$t
temporal_data <- data.frame(cell_id, time_stamp)
table <- getTransitionProbability(temporal_data,
cellid_column = "cell_id",time_column = "time_stamp")
colnames(temporal_data) <- c("Cell.ID","t")
ex_post_forecasting <- dataset[1830:1860,]
ex_post <- msm(state_time_data = temporal_data,
               forecast_type = "ex-post",
               transition_probability_matrix = table,
               initial_state = 2,
               num_simulations = 2,
               scoreHVT_results = scoring,
               trainHVT_results = hvt.results,
               actual_data = ex_post_forecasting,
               raw_dataset = dataset,
               mae_metric = "median",
               handle_problematic_states = FALSE,
              show_simulation = FALSE,
              time_column = 't')
```

`OptimizationResults` *Table display of all combinations performed in 'HVTMSMoptimization'*

Description

This function helps to tabulate the 'HVTMSMoptimization' all iterations performed (successful and failed combinations)

Usage

```
OptimizationResults(msm_results, show_top_global = NULL)
```

Arguments

`msm_results`	Dataframe. The 'all_results' object of 'HVTMSMoptimization' output based on a specific mae_metric
`show_top_global`	Numeric. To tabulate top n lowest MAE results

Value

Table of all iterations will be displayed

Author(s)

Vishwavani <vishwavani@mu-sigma.com>, Nithya <nithya.sn@mu-sigma.com>

`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 = "All",
  flow_map = "All",
  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 = 10, 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_stamp")
plots <- plotAnimatedFlowmap(hvt_model_output = hvt.results, transition_probability_df = table,
df = dataset, animation = 'All', flow_map = 'All',fps_time = 1,fps_state =  1,time_duration = 2,
state_duration = 2,cellid_column = "cell_id", time_column = "time_stamp")
```

`plotExAnteRawSeries` *Ex-ante raw series forecasting*

Description

Transforms ex-ante forecasts generated on a log-difference-12 scale back to the original raw data scale

Usage

```
plotExAnteRawSeries(
  ex_ante_results,
  original_dataset,
  transformed_dataset = NULL,
  time_column,
  mae_metric = "median"
)
```

Arguments

`ex_ante_results`	List. Output from ‘msm()‘ function with ‘forecast_type = "ex-ante"‘
`original_dataset`	Dataframe. The dataset imported with all features in its original scale of measures without normalization or log difference
`transformed_dataset`	Dataframe. The dataset that is transformed for analysis including scaling and log difference
`time_column`	Character. Name of the time column in the dataset.
`mae_metric`	Character. Metric to highlight in plots ("mean", "median", or "mode"). Default is "median".

Value

A list containing:

`reverted_forecasts`	List of data frames, one per variable, with columns: time, mean, median, mode (reverted values)
`plots`	List of plotly objects, one per variable, showing historical data and reverted forecasts

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

Examples

```
## Not run:
# After running msm() for ex-ante forecasting
# Note: trainHVT_results and scoreHVT_results are needed for msm(),
# but NOT for plotExAnteRawSeries() which uses the forecast values directly
ex_ante <- msm(state_time_data = temporal_data,
               forecast_type = "ex-ante",
               transition_probability_matrix = prob_trans_matx,
               initial_state = tail(temporal_data$Cell.ID, 1),
               n_ahead_ante = ex_ante_period,
               num_simulations = 500,
               scoreHVT_results = scoring,
               trainHVT_results = hvt.results,
               raw_dataset = entire_dataset,
               time_column = "t")

# Revert to raw scale - only needs ex_ante_results and original_dataset
raw_forecasts <- plotExAnteRawSeries(
  ex_ante_results = ex_ante,
  original_dataset = entire_dataset_original,  # Pre-transformation raw data
  transformed_dataset = entire_dataset,        # Post-transformation data
  time_column = "t",
  mae_metric = "median"
)

# Access reverted forecasts
raw_forecasts$reverted_forecasts$CPI_Food

# Access plots
raw_forecasts$plots$CPI_Food

## End(Not run)
```

---

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",
  centroid.size = 0.6,
  centroid.color = "black",
```

```
  child.level = 1,
  hmap.cols,
  separation_width = 7,
  layer_opacity = c(0.5, 0.75, 0.99),
  dim_size = 1000,
  plot.type = "2Dhvt",
  quant.error.hmap = NULL,
  cell_id = FALSE,
  cell_id_position = "bottom",
  cell_id_size = 2.6,
  centroids = TRUE
)
```

Arguments

hvt.results	(1D/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.
centroid.size	(2Dhvt/2Dheatmap) Numeric Vector. A vector indicating the size of centroids for each level.
centroid.color	(2Dhvt/2Dheatmap) Numeric Vector. A vector indicating the color of centroids for each level.
child.level	(2Dheatmap/surface_plot) Numeric. Indicating the level for which the plot should be displayed
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.
separation_width	(surface_plot) Numeric. An integer indicating the width between hierarchical levels in surface plot
layer_opacity	(surface_plot) Numeric. A vector indicating the opacity of each hierarchical levels in surface plot
dim_size	(surface_plot) Numeric. An integer controls the resolution or granularity of the 3D surface grid
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'.
quant.error.hmap	(2Dheatmap) Numeric. A number representing the quantization error threshold to be highlighted in the heatmap. When a value is provided, it will emphasize cells with quantization errors equal or less than the specified threshold, indicating that these cells cannot be further subdivided in the next depth layer. The default value is NULL, meaning all cells will be colored in the heatmap across various depths.

cell_id	(2Dhvt/2Dheatmap) Logical. A logical indicating whether the cell IDs should be displayed. Default is FALSE
cell_id_position	(2Dhvt/2Dheatmap) Character. A character indicating the position of the cell IDs. Accepted entries are 'top' , 'bottom', 'left', 'right', and 'center'. Default is 'bottom'
cell_id_size	(2Dhvt/2Dheatmap) Numeric. A numeric vector indicating the size of the cell IDs. Default is 2.6
centroids	(2Dhvt/2Dheatmap) Logical. A logical indicating whether the centroid points should be displayed. Default value is TRUE.

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>, Vishwavani <vishwavani@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 = 10, 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)
```

---

plotMsmKN *plotting MAE across Number of cells*

Description

This function helps to visualize the 'HVTMSMoptimization' function's output. It creates a plotly object where the lowest MAE combination achieved for each cell will be plotted

Usage

```
plotMsmKN(optimization_results)
```

Arguments

optimization_results	List. The output of 'HVTMSMoptimization' function specific for a mae_metric

Value

List containing visualization results for the selected MAE metric:

[[plot]]	A Plotly object of cell-level lowest MAE combination
[[table]]	A table of plotted data

Author(s)

Vishwavani <vishwavani@mu-sigma.com>, Nithya <nithya.sn@mu-sigma.com>

`plotNovelCells` *Plot the identified outlier cells 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,
  v_intercept = NULL,
  time_periods = NULL
)
```

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.
v_intercept	Numeric. A numeric value indicating the time stamp to draw a vertical line on the plot.
time_periods	List. A list of vectors, each containing start and end times for highlighting time periods.

Value

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

Author(s)

PonAnuReka Seenivasan <ponanureka.s@mu-sigma.com>, 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"])

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_stamp")
```

---

plotZscore *Plots of z scores*

Description

This is the main function to plot the z scores against cell ids.

Usage

```
plotZscore(
  data,
  cell_range = NULL,
  segment_size = 2,
  reference_lines = c(-1.65, 1.65)
)
```

Arguments

data	Data frame. A data frame of cell id and features.
cell_range	Vector. A numeric vector of cell id range for which the plot should be displayed. Default is NULL, which plots all the cells.
segment_size	Integer. A numeric value to indicate the size of the bars in the plot. Default is 2.
reference_lines	Vector. A numeric vector of confidence interval values for the reference lines in the plot. Default is c(-1.65, 1.65).

Value

A grid of plots of z score against cell id of teh given features.

Author(s)

Vishwavani <vishwavani@mu-sigma.com>

Examples

```
data("EuStockMarkets")
dataset <- data.frame(t = as.numeric(time(EuStockMarkets)),
                      DAX = EuStockMarkets[, "DAX"],
                      SMI = EuStockMarkets[, "SMI"],
                      CAC = EuStockMarkets[, "CAC"],
                      FTSE = EuStockMarkets[, "FTSE"])
rownames(EuStockMarkets) <- dataset$t
hvt.results<- trainHVT(dataset[-1],n_cells = 60, depth = 1, quant.err = 0.1,
                       distance_metric = "L1_Norm", error_metric = "max",
                       normalize = TRUE,quant_method = "kmeans")
col_names <- c("Cell.ID","DAX","SMI","CAC","FTSE")
data <- dplyr::arrange(dplyr::select(hvt.results[[3]][["summary"]],col_names),Cell.ID)
data <- round(data, 2)
plotZscore(data)
```

---

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>

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(
  dataset,
  hvt.results.model,
  child.level = 1,
  mad.threshold = 0.2,
  line.width = 0.6,
  color.vec = c("navyblue", "slateblue", "lavender"),
  normalize = TRUE,
  distance_metric = "L1_Norm",
  error_metric = "max",
  yVar = NULL,
  analysis.plots = FALSE,
  names.column = NULL,
  cell_id = TRUE,
  cell_id_position = "bottom",
  centroids = TRUE,
  cell_id_size = 4,
  centroid.size = 0.8
)
```

Arguments

dataset	Data frame. A data frame which to be scored. Can have categorical columns if 'analysis.plots' are required.
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.

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)

analysis.plots
: Logical. A logical value indicating that the scored plot should be plotted or not. If TRUE, the identifier column(character column) name should be supplied in 'names.column' argument. The output will be a 2D heatmap plotly which gives info on the cell id and the observations of a cell.

names.column
: Character. A character or a vector representing the name of the identifier column/character column.

cell_id
: Logical. A logical indicating whether the cell IDs should be displayed. Default is TRUE.

cell_id_position
: Character. A character indicating the position of the cell IDs. Accepted entries are 'top', 'bottom', 'left', 'right', and 'center'. Default is 'center'.

centroids
: Logical. A logical indicating whether the centroid points should be displayed. Default value is TRUE

cell_id_size
: Numeric. A numeric value indicating the size of the cell IDs. Default is 4

centroid.size
: Numeric. A numeric value or vector indicating the size of centroids for each level. Default is 0.8

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> , Vishwavani <vishwavani@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[1:500, ]
test <- EuStockMarkets[501:800, ]
```

```
###MAP-A
hvt_mapA <- trainHVT(train, n_cells = 70, depth = 1, quant.err = 0.1,
                     distance_metric = "L1_Norm", error_metric = "max",
                     normalize = TRUE,quant_method = "kmeans")

identified_Novelty_cells <- c(7,55) #example
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 = 30,
                     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)
```

---

summary *Table for displaying summary*

Description

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

Usage

```
summary(data, limit = 20, scroll = TRUE)
```

Arguments

data	List. A listed object from trainHVT or scoreHVT
limit	Numeric. A value to indicate how many rows to display.
scroll	Logical. A value to indicate whether to display scroll bar or not. Default value is TRUE.

Value

A consolidated table of summary for training, scoring and forecasting

Author(s)

Vishwavani <vishwavani@mu-sigma.com>, Alimpan Dey <alimpan.dey@mu-sigma.com>

Examples

```
data <- datasets::EuStockMarkets
dataset <- as.data.frame(data)
#model training
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", dim_reduction_method = 'sammon')
summary(data =  hvt.results)
```

---

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,
  normalize = FALSE,
  distance_metric = "L1_Norm",
  error_metric = "max",
  quant_method = "kmeans",
  scale_summary = NA,
  diagnose = FALSE,
  hvt_validation = FALSE,
  train_validation_split_ratio = 0.8,
  dim_reduction_method = "sammon",
  tsne_theta = 0.2,
  tsne_eta = 200,
  tsne_perplexity = 30,
  tsne_verbose = TRUE,
  tsne_max_iter = 500,
  umap_n_neighbors = 60,
  umap_n_components = 2,
  umap_min_dist = 0.1
)
```

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.
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).
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.
dim_reduction_method	Character.The dim_reduction_method can be one of "tsne", "umap", "sammon".
tsne_theta	Numeric.The tsne_theta is only used when dim_reduction_method is set to "tsne". Default value is 0.5 and common values are between 0.2 and 0.5.
tsne_eta	Numeric.The tsne_eta are used only when dim_reduction method is set to "tsne". Default value is 200.

`tsne_perplexity`
: Numeric.The tsne_perplexity is only used when dim_reduction_method is set to "tsne". Default value is 30 and common values are between between 30 and 50.

`tsne_verbose`
: Logical. A logical value which indicates the t-SNE algorithm to print detailed information about its progress to the console.

`tsne_max_iter`
: Numeric.The tsne_max_iter is used only when dim_reduction_method is set to "tsne". Default value is 1000.More iterations can improve results but increase computation time.

`umap_n_neighbors`
: Integer.The umap_n_neighbors is used only when dim_reduction_method is set to "umap". Default value is 15.Controls the balance between local and global structure in data.

`umap_n_components`
: Integer.The umap_n_components is used only when dim_reduction_method is set to "umap". Default value is 2.Indicates the number of dimensions for embedding.

`umap_min_dist`
: Numeric.The umap_map_dist is used only when dim_reduction_method is set to "umap". Default value is 0.1.Controls how tightly UMAP packs points together.

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 , the validation results and the dimensionality reduction evaluation metrics table.

Author(s)

Shubhra Prakash <shubhra.prakash@mu-sigma.com>, Sangeet Moy Das <sangeet.das@mu-sigma.com>, Shantanu Vaidya <shantanu.vaidya@mu-sigma.com>,Bidesh Ghosh <bidesh.gosh@mu-sigma.com>,Alimpan Dey <alimpan.dey@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")
```

Index