Package 'TockyRandomForest'

January 25, 2025

Version 0.1.1

Date 2025-01-24

Title Random Forest Machine Learning Methods for Flow Cytometric Fluorescent Timer Data

Author Masahiro Ono [aut, cre]

Maintainer Masahiro Ono <monotockylab@gmail.com>

Description This package provides Random Forest-based machine learning methods for analysing flow cytometric data from Fluorescent Timer reporters. Specifically, the package provides random forest-based methods for classifying experimental groups and thereby identifying group-specific feature cells. It is optimised to Timer-of-Cell-Kinetics-and-Activity (Tocky), which enables the analysis of temporal dynamics of individual cells in vivo.

Depends R (>= 4.2.0), utils, stats, graphics, grDevices, methods

Imports dplyr, randomForest, pROC, ROCR, ggplot2, rlang, dbscan, TockyPrep, tidyr, caret, gridExtra, clue

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/MonoTockyLab/TockyRandomForest, https://MonoTockyLab.github.io/TockyRandomForest

BugReports https://github.com/MonoTockyLab/TockyRandomForest/issues

License Apache-2.0

Encoding UTF-8

RoxygenNote 7.3.2

Contents

ClusteringFeatureCells *Cluster Feature Cells*

Description

This function performs density-based clustering on a subset of important data points

Usage

```
ClusteringFeatureCells(
  res_tockyrf,
  percentile = 0.75,
  eps = 4,
  minPts = 2,
  colors = NULL,
  xlim = NULL,
  ylim = NULL,
  test = TRUE
)
```

Arguments

res_tockyrf	A list object output from TockyKmeansRF.
percentile	A numeric value defining the cutoff for filtering data based on importance scores.
eps	The epsilon parameter for DBSCAN, controlling the maximum distance between points in a cluster.
minPts	The minimum number of points required to form a dense region in DBSCAN.
colors	Optional. A vector to define the colour code for clusters.
xlim	Optional. The x range of Timer Blue-Red plot
ylim	Optional. The y range of Timer Blue-Red plot
test	Logical. The default is TRUE, which enables analysis of test data (recommended). Optionally, test = FALSE allows analysis of training data.

Value

Prints plots directly and may return statistical test results if needed.

Examples

```
## Not run:
  clusteringFeatureCells(TockyData, result, percentile = 0.5)

## End(Not run)
```

`partitionTockyPrepData` *Split a TockyPrepData object into training and testing sets.*

Description

Split a TockyPrepData object into training and testing sets.

Usage

```
partitionTockyPrepData(x, p = 0.7)
```

Arguments

`x`	A TockyPrepData object.
`p`	A numeric value between 0 and 1 specifying the proportion of data to use for the training set. Defaults to 0.7.

Value

A list containing the following elements: * ‘trainData‘: A TockyPrepData object containing the training data. * ‘testData‘: A TockyPrepData object containing the testing data.

Examples

```
## Not run:
out <- partitionTockyPrepData(my_data)
train_data <- out$trainData
test_data <- out$testData

## End(Not run)
```

`plotClustersMFI` *Generate a boxplot of MFI (median fluorescence intensity) for each cluster identified by TockyKmeans.*

Description

Generate a boxplot of MFI (median fluorescence intensity) for each cluster identified by TockyK-means.

Usage

```
plotClustersMFI(res_tockyrf, group = NULL, select = FALSE, variables = NULL)
```

Arguments

res_tockyrf	A list object output from 'TockyKmeansRF'
group	A character vector specifying the group(s) to use for analysis. If NULL, all groups are used.
select	A logical indicating whether to allow interactive selection of variables for analysis.
variables	A character vector specifying the variables to analyze. Only used if 'select' is TRUE.

Value

A list containing the following elements: * 'plot': A ggplot object of the boxplot. * 'data': A data frame containing the summarized data used to create the plot.

Examples

```
## Not run:
plotMFIcluster(res_tockyrf)

## End(Not run)
```

---

plotImportanceScores *Plot Importance Scores on Test Data*

Description

This function plots the importance scores in test data, showing the Angle versus Intensity coloured by the feature importance derived from a RandomForest model, alongside a colour bar representing the importance score scale.

Usage

```
plotImportanceScores(
  res_tockyrf,
  percentile = 0.9,
  test = TRUE,
  plot_mode = "Angle"
)
```

Arguments

res_tockyrf	A list object output from TockyKmeansRF.
percentile	Numeric, percentile threshold for importance scores (between 0 and 1).
test	Logical. If TRUE, Importance Score plot using test data used for creating the TockyKmeansRF model will be used. If FALSE, the training data will be used instead.
plot_mode	Either "raw_Timer" for Blue vs Red plots, or "Angle" for Angle vs Intensity plots.

Value

A plot of Angle vs Intensity coloured by importance and a colour bar indicating the importance scores.

Examples

```
## Not run:
  # Assuming 'res_tockyrf' is already available from using TockyKmeansRF
  plotImportanceScores(res_tockyrf)

## End(Not run)
```

---

plotTockyKmeansRF *Plot ROC Curve from TockyKmeansRF Results*

Description

This function generates a Receiver Operating Characteristic (ROC) curve for the results obtained from a TockyKmeansRF function, specifically designed for two-group comparisons. It calculates the area under the curve (AUC) and its confidence interval, and returns a ggplot object depicting the ROC curve with confidence bands.

Usage

```
plotTockyKmeansRF(
  res_tockyrf,
  expr_group = NULL,
  ctrl_group = NULL,
  ROC = TRUE
)
```

Arguments

res_tockyrf	The output object from 'TockyKmeansRF'
expr_group	The name of the experimental group within 'sampledef'.
ctrl_group	The name of the control group within 'sampledef'.
ROC	Logical. If TRUE, ROC curve is produced. If FALSE, Precision-Recall curve is generated instead.

Value

A ggplot object representing the ROC curve with the area shaded for the 95 interval around the curve.

Examples

```
## Not run:
p <- plotTockyKmeansRF(res_tockyrf)

## End(Not run)
```

TockyKmeansRF *Train and Evaluate TockyKmeansRF Model*

Description

This function integrates kmeans clustering and Random Forest classification to analyze flow cytometric Fluorescent Timer data. It applies K-means clustering to both training and test datasets as data frame to create clusters, matches these clusters across datasets, and then uses Random Forest to predict outcomes based on the relative proportions of cells in each cluster.

Usage

```
TockyKmeansRF(
  trainData,
  testData,
  num_cluster = 4,
  verbose = TRUE,
  iter.max = 10,
  nstart = 1,
  mtry = NULL,
  ntree = 100
)
```

Arguments

trainData	Training dataset as a TockyPrepData
testData	Test dataset as a TockyPrepData.
num_cluster	The number of clusters (metaclusters) to generate via k-means.
verbose	Logical indicating whether to print progress messages and outputs. Default is TRUE.
iter.max	the maximum number of iterations allowed. To be passed to kmeans.
nstart	the number of random sets to be used in each clustering. To be passed to kmeans.
mtry	The number of variables randomly sampled as candidates at each split when building a tree within the Random Forest. To be passed to randomForest.
ntree	The number of trees to grow in the Random Forest. The default value is set to 100. To be passed to randomForest.

Value

A list containing key Tocky data and the Random Forest model and its performance data.

Examples

```
## Not run:
result <- TockyKmeansRF(trainData, testData, num_cluster = 4, verbose = TRUE)

## End(Not run)
```

`TockyRFClusterOptimization` *Evaluate ROC Curves with Confidence Intervals and Calculate AUC for TockyKmeansRF*

Description

This function applies the TockyKmeansRF model to assess the performance of clustering with a specified range of cluster numbers over repeats. It evaluates the model's performance by calculating ROC curves, along with their confidence intervals, and computes the AUC for each curve across multiple runs.

Usage

```
TockyRFClusterOptimization(
  trainData,
  testData,
  num_cluster_vec = 4:9,
  k = 1,
  ctrl_group = NULL,
  expr_group = NULL,
  iter.max = 10,
  nstart = 1,
  mtry = NULL,
  ntree = 100
)
```

Arguments

`trainData`	Training dataset as a TockyPrepData object.
`testData`	Test dataset as a TockyPrepData object.
`num_cluster_vec`	A numeric vector of cluster numbers to evaluate.
`k`	Integer, number of iterations to estimate confidence intervals.
`ctrl_group`	The name of the control group within 'sampledef'.
`expr_group`	The name of the experimental group within 'sampledef'.
`iter.max`	the maximum number of iterations allowed. To be passed to kmeans.
`nstart`	the number of random sets to be used in each clustering. To be passed to kmeans.
`mtry`	The number of variables randomly sampled as candidates at each split when building a tree within the Random Forest. To be passed to randomForest.
`ntree`	The number of trees to grow in the Random Forest. The default value is set to 100. To be passed to randomForest.

Value

A list containing three elements: 'roc_results' a list of ROC curve objects for each cluster number, 'auc_values' a list of AUC values with their respective confidence intervals for each cluster configuration, and 'roc_plots' a list of ggplot objects each depicting the ROC curve with confidence intervals for the range of specified clusters.

Examples

```
## Not run:
result <- TockyRFClusterOptimization(trainData, testData, num_cluster_vec = 4:9, k = 50)

## End(Not run)
```

---

violinPlotFeatureCells

Generate Plots for Analysing Cluster Abundance

Description

This function processes clustering results, plots each cluster, and overlays each cluster's convex hull. It is adaptable to any number of cell_cluster_id.

Usage

```
violinPlotFeatureCells(res_tockyrf, p_adjust_method = "BH", ncol = 3)
```

Arguments

res_tockyrf	A list object output from 'TockyKmeansRF', which has been further processed using 'ClusteringFeatureCells'.
p_adjust_method	A method for p-value adjustment in multiple testing using Mann Whitney. clusteringFeatureCells cen be used.
ncol	Number of columns in output figure panel.

Examples

```
## Not run:
  data <- data.frame(Angle = runif(100), Intensity = runif(100))
  cell_cluster_id <- dbscan(data, eps = 0.1, minPts = 5)$cluster
  plotHullsGating(data, cell_cluster_id)

## End(Not run)
```

Index