Package ‘CytoProfile’

March 6, 2025

Title Cytokine Profiling Analysis Tool

Version 1.0

Description CytoProfile is a comprehensive tool for cytokine profiling analysis.
It supports quality control using biologically meaningful cutoffs on raw cytokine
measurements and tests for distributional symmetry to suggest appropriate transformations.
The package offers exploratory data analysis with summary statistics, enhanced boxplots, and
barplots, along with both univariate and multivariate analysis capabilities for in-depth cytokine profiling.

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/saraswatsh/CytoProfile, https://saraswatsh.github.io/CytoProfile/

Depends R (>= 3.5)

Imports mixOmics,
moments,
dplyr,
tidyr,
pROC,
plot3D,
tidyverse,
caret,
xgboost,
randomForest,
gplots,
e1071,
ggplot2,
ggrepel,
gridExtra,
reshape2

Suggests BiocManager,
testthat,
knitr,
rmarkdown,
devtools,
Ckmeans.1d.dp

NeedsCompilation no

License GPL-2

LazyData true

VignetteBuilder knitr

Contents

`cyt.anova` *ANOVA analysis on all continuous variables within the data.*

Description

This function performs an ANOVA for each continuous variable against every categorical predictor in the input data. For each valid predictor (i.e., with more than one level and no more than 10 levels), it conducts Tukey's HSD test and extracts the adjusted p-values for pairwise comparisons.

Usage

```
cyt.anova(data)
```

Arguments

`data`	A data frame or matrix containing both categorical and continuous variables. Character columns are converted to factors; all factor columns are used as predictors, while numeric columns are used as continuous outcomes.

Value

A list of adjusted p-values from Tukey's HSD tests for each combination of continuous outcome and categorical predictor. The list elements are named in the format "outcome_predictor".

Examples

```
## Not run:
data("cytodata")
cyt.anova(cytodata[, c(2:4,5:6)])

## End(Not run)
```

---

cyt.bp *Generating a PDF file to show the measured value by column of the data frame.*

Description

This function creates a PDF file containing box plots for the continuous variables in the provided data. If the number of columns in data exceeds bin.size, the function splits the plots across multiple pages.

Usage

```
cyt.bp(data, Title, bin.size = 25, mfRow = c(1, 1), yLim = NULL)
```

Arguments

data	A matrix or data frame containing the raw data to be plotted.
Title	A string representing the name of the PDF file.
bin.size	The maximum number of box plots to display on a single page.
mfRow	A numeric vector of length two specifying the layout (rows and columns) for the plots on each page.
yLim	An optional numeric vector defining the y-axis limits for the plots.

Value

A PDF file containing the box plots of the continuous variables.

Examples

```
## Not run:
  # Generating boxplots to check for outliers for raw values:
  cyt.bp(data.df[,-c(1:4)], Title = "Boxplot.byCytokine.Raw.pdf")

  # Generating boxplots to check for outliers for log2 values:
  cyt.bp(log2(data.df[,-c(1:4)]), Title = "Boxplot.byCytokine.log2.pdf")

## End(Not run)
```

`cyt.bp2` *Boxplot Function Enhanced for Specific Group Comparisons*

Description

This function generates a PDF file containing boxplots for each combination of numeric and factor variables in the provided data. It first converts any character columns to factors and checks that the data contains at least one numeric and one factor column. If the scale argument is set to "log2", all numeric columns are log2-transformed. The function then creates boxplots using ggplot2 for each numeric variable grouped by each factor variable.

Usage

```
cyt.bp2(data, Title, mfRow = c(1, 1), scale = NULL, yLim = NULL)
```

Arguments

`data` A matrix or data frame of raw data.

`Title` A string representing the title of the PDF file.

`mfRow` A numeric vector of length two specifying the layout (rows and columns) for the plots on each page. Defaults to c(1,1).

`scale` Transformation option for continuous variables. Options are NULL (default) and "log2". When set to "log2", numeric columns are transformed using the log2 function.

`yLim` An optional numeric vector defining the y-axis limits for the plots.

Value

A PDF file containing the boxplots.

Examples

```
## Not run:
data.df <- cytodata[,-c(1,4)]
cyt.bp2(data.df, Title = "boxplot2.test2.pdf", scale = "log2")

## End(Not run)
```

`cyt.dualflashplot` *Dual flash plot for group comparisons.*

Description

This function reshapes the input data and computes summary statistics (mean and variance) for each variable grouped by a specified factor column. It then calculates the SSMD (Strictly Standardized Mean Difference) and log2 fold change between two groups (group1 and group2) and categorizes the effect strength as "Strong Effect", "Moderate Effect", or "Weak Effect". A dual flash plot is generated using ggplot2 where the x-axis represents the average log2 fold change and the y-axis represents the SSMD. Additionally, the function prints the computed statistics to the console.

Usage

```
cyt.dualflashplot(
  data,
  group_var,
  group1,
  group2,
  ssmd_thresh = 1,
  log2fc_thresh = 1,
  top_labels = 15
)
```

Arguments

`data`	A data frame containing the input data.
`group_var`	A string specifying the name of the grouping column in the data.
`group1`	A string representing the name of the first group for comparison.
`group2`	A string representing the name of the second group for comparison.
`ssmd_thresh`	A numeric threshold for the SSMD value used to determine significance. Default is 1.
`log2fc_thresh`	A numeric threshold for the log2 fold change used to determine significance. Default is 1.
`top_labels`	An integer specifying the number of top variables (based on absolute SSMD) to label in the plot. Default is 15.

Value

A ggplot object representing the dual flash plot for the comparisons between group1 and group2.

Examples

```
## Not run:
data.df <- cytodata[,-c(1,3:4)]
dfp <- cyt.dualflashplot(data.df, group_var = "Group", group1 = "T2D", group2 = "ND",
ssmd_thresh = -0.2, log2fc_thresh = 1, top_labels = 10)

## End(Not run)
```

`cyt.errbp` *Error-bar plot for comparison.*

Description

This function draws an error-bar plot for comparing groups to a baseline group. It creates a barplot of the central tendency (mean or median) and overlays error bars representing the spread (e.g., standard deviation, MAD, or standard error). Optionally, p-value and effect size labels (based on SSMD) are added, either as symbols or numeric values.

Usage

```
cyt.errbp(
  data,
  pLab = TRUE,
  esLab = TRUE,
  classSymbol = TRUE,
  xlab = "",
  ylab = "",
  main = ""
)
```

Arguments

`data` A data frame containing the following columns for each group:

- `name`: Group names.
- `center`: Mean or median values.
- `spread`: Standard deviation, MAD, or standard error.
- `p.value`: P-value for the comparison.
- `effect.size`: Effect size based on SSMD.

Note: The first row of `center.df` must correspond to the baseline group.

`pLab` Logical. Whether to label the p-values on the plot. Default is `TRUE`.

`esLab` Logical. Whether to label the effect sizes on the plot. Default is `TRUE`.

`classSymbol` Logical. Whether to use symbolic notation for significance and effect size. Default is `TRUE`.

`xlab` Character. Label for the x-axis.

`ylab` Character. Label for the y-axis.

`main` Character. Title of the graph.

Value

An error-bar plot is produced.

Examples

```
## Not run:
cytokine.mat <- cytodata[, -c(1:4)] # Extracting all cytokines to be stored in one object
cytokineNames <- colnames(cytokine.mat) # Extracting the cytokine names
nCytokine <- length(cytokineNames) # Obtaining the total number of cytokines
results <- cyt.skku(cytodata[,-c(1,4)], printResLog = TRUE,
           group.cols = c("Group", "Treatment")) # Extracting values
pdf( "barErrorPlot.pdf" )
par(mfrow=c(2,2), mar=c(8.1,  4.1, 4.1, 2.1) )
for( k in 1:nCytokine ) {
center.df <- data.frame( "name"=rownames(results[,,k]), results[,,k] )
cyt.errbp(center.df, pLab=FALSE, esLab=FALSE, classSymbol=TRUE,
ylab="Concentration in log2 scale",  main=cytokineNames[k] )
}
dev.off()

## End(Not run)
```

cyt.heatmap *Heat Map*

Description

This function creates a heatmap using the numeric columns from the provided data frame. If requested via the scale parameter, the function applies a log2 transformation to the data (with non-positive values replaced by NA). Optionally, if an annotation column is specified and exists in data, the function attempts to generate a color annotation (although the annotation is not passed to heatmap.2 in the current implementation). The heatmap is saved as a file, with the format determined by the file extension in title.

Usage

```
cyt.heatmap(data, scale = NULL, annotation_col_name = NULL, title)
```

Arguments

data	A data frame containing the input data. Only numeric columns will be used to generate the heatmap.
scale	Character. An optional scaling option. If set to "log2", the numeric data will be log2-transformed (with non-positive values set to NA). Default is NULL.
annotation_col_name	Character. An optional column name from data to be used for generating annotation colors. Default is NULL.
title	Character. The title of the heatmap and the file name for saving the plot. The file extension (".pdf" or ".png") determines the output format.

Value

The function does not return a value. It saves the heatmap to a file.

Examples

```
## Not run:
cyt.heatmap(data = data.df,
scale = "log2",        # Optional scaling
annotation_col_name = "Group",
title = "Heatmap.png")

## End(Not run)
```

`cyt.pca` *Analyze data with Principal Component Analysis (PCA) for cytokines.*

Description

This function performs Principal Component Analysis (PCA) on cytokine data and generates several types of plots, including:

- 2D PCA plots using mixOmics's `plotIndiv` function,
- 3D scatter plots (if `style` is "3d" or "3D" and `comp.num` is 3) via the plot3D package,
- Scree plots showing both individual and cumulative explained variance,
- Loadings plots, and
- Biplots and correlation circle plots.

The function optionally applies a log2 transformation to the numeric data and handles analyses based on either treatment or stimulation groups.

Usage

```
cyt.pca(
  data,
  group.col = NULL,
  trt.col = NULL,
  colors = NULL,
  title,
  ellipse = FALSE,
  comp.num = 2,
  scale = NULL,
  pch.values = NULL,
  style = NULL
)
```

Arguments

`data` A data frame containing cytokine data. It should include at least one column representing grouping information and optionally a second column representing treatment or stimulation.

`group.col` Character. The name of the column containing the grouping information. If not specified and `trt.col` is provided, the treatment column will be used as the grouping variable.

`trt.col` Character. The name of the column containing the treatment (or stimulation) information. If not specified and `group.col` is provided, the grouping column will be used as the treatment variable.

`colors` A vector of colors corresponding to the groups. If set to NULL, a palette is generated using `rainbow()` based on the number of unique groups.

`title` A string specifying the file name of the PDF where the PCA plots will be saved.

`ellipse` Logical. If TRUE, a 95% confidence ellipse is drawn on the PCA individuals plot. Default is FALSE.

`comp.num` Numeric. The number of principal components to compute and display. Default is 2.

`scale` Character. If set to "log2", a log2 transformation is applied to the numeric cytokine measurements (excluding the grouping columns). Default is NULL.

`pch.values` A vector of plotting symbols (pch values) to be used in the PCA plots. Default is NULL.

`style` Character. If set to "3d" or "3D" and `comp.num` equals 3, a 3D scatter plot is generated using the plot3D package. Default is NULL.

Value

A PDF file containing the PCA plots is generated and saved.

Examples

```
## Not run:
data <- cytodata[,-c(1,4, 24)]
data.df <- filter(data, Group != "ND" & Treatment != "Unstimulated")
cyt.pca(data.df, title = "Example PCA Analysis.pdf" ,colors = c("black", "red2"),
scale = "log2", comp.num = 3, pch.values = c(16,4), style = "3D",
group.col = "Group", trt.col = "Treatment")
cyt.pca(data.df, title = "Example PCA Analysis 2.pdf" ,colors = c("black", "red2"),
scale = "log2", comp.num = 2, pch.values = c(16,4), group.col = "Group")

## End(Not run)
```

`cyt.plsda` *Analyze data with Sparse Partial Least Squares Discriminant Analysis (sPLS-DA).*

Description

This function conducts Sparse Partial Least Squares Discriminant Analysis (sPLS-DA) on the provided data. It uses the specified `group.col` (and optionally `trt.col`) to define class labels while assuming the remaining columns contain continuous variables. The function supports a log2 transformation via the `scale` parameter and generates a series of plots, including classification plots, scree plots, loadings plots, and VIP score plots. Optionally, ROC curves are produced when `roc` is `TRUE`. Additionally, cross-validation is supported via LOOCV or Mfold methods. When both `group.col` and `trt.col` are provided and differ, the function analyzes each treatment level separately.

Usage

```
cyt.plsda(
  data,
  group.col = NULL,
  trt.col = NULL,
  colors = NULL,
  title,
  ellipse = FALSE,
  bg = FALSE,
  conf.mat = FALSE,
  var.num,
  cv.opt = NULL,
  fold.num = 5,
  scale = NULL,
  comp.num = 2,
  pch.values,
  style = NULL,
  roc = FALSE
)
```

Arguments

`data` A matrix or data frame containing the variables. Columns not specified by `group.col` or `trt.col` are assumed to be continuous variables for analysis.

`group.col` A string specifying the column name that contains group information. If `trt.col` is not provided, it will be used for both grouping and treatment.

`trt.col` A string specifying the column name for treatments. Default is `NULL`.

`colors` A vector of colors for the groups or treatments. If `NULL`, a random palette (using `rainbow`) is generated based on the number of groups.

`title` A string specifying the file name for saving the PDF output.

`ellipse` Logical. Whether to draw a 95% confidence ellipse on the figures. Default is `FALSE`.

`bg` Logical. Whether to draw the prediction background in the figures. Default is `FALSE`.

`conf.mat` Logical. Whether to print the confusion matrix for the classifications. Default is `FALSE`.

`var.num` Numeric. The number of variables to be used in the PLS-DA model.

`cv.opt` Character. Option for cross-validation method: either "loocv" or "Mfold". Default is `NULL`.

`fold.num` Numeric. The number of folds to use if `cv.opt` is "Mfold". Default is 5.

`scale` Character. Option for data transformation; if set to `"log2"`, a log2 transformation is applied to the continuous variables. Default is `NULL`.

`comp.num` Numeric. The number of components to calculate in the sPLS-DA model. Default is 2.

`pch.values` A vector of integers specifying the plotting characters to be used in the plots.

`style` Character. If set to `"3D"` or `"3d"` and `comp.num` equals 3, a 3D plot is generated using the `plot3D` package. Default is `NULL`.

`roc` Logical. Whether to compute and plot the ROC curve for the model. Default is `FALSE`.

Value

A PDF file containing the classification figures, component figures with Variable of Importance in Projection (VIP) scores, and classifications based on VIP scores greater than 1. ROC curves and confusion matrices are also produced if requested.

Examples

```
## Not run:
# Example using overall analysis (single factor)
cyt.plsda(data = my_data, group.col = "Group", title = "PLSDA_overall.pdf",
          var.num = 25, pch.values = c(16, 4, 3))

# Example using separate group and treatment columns
cyt.plsda(data = my_data, group.col = "Group", trt.col = "Treatment",
title = "PLSDA_byTreatment.pdf",
colors = c("black", "purple", "red2"), ellipse = TRUE, bg = TRUE, conf.mat = TRUE,
var.num = 25, cv.opt = "loocv", comp.num = 2, pch.values = c(16, 4, 3))

# Example with ROC curve and 3D plot (comp.num == 3)
cyt.plsda(data = my_data, group.col = "Group", trt.col = "Treatment",
title = "PLSDA_ROC_3D.pdf", colors = c("black", "purple", "red2"),
 ellipse = TRUE, bg = TRUE, conf.mat = TRUE,
 var.num = 25, cv.opt = "Mfold", fold.num = 5, scale = "log2", comp.num = 3,
 pch.values = c(16, 4, 3), style = "3D", roc = TRUE)

## End(Not run)
```

cyt.rf *Run Random Forest Classification on Cytokine Data*

Description

This function trains and evaluates a Random Forest classification model on cytokine data. It includes feature importance visualization, cross-validation for feature selection, and performance metrics like accuracy, sensitivity, and specificity. Optionally, for binary classification, the function also plots the ROC curve and calculates the AUC.

Usage

```
cyt.rf(
  data,
  group_col,
  ntree = 500,
  mtry = 5,
  train_fraction = 0.7,
  plot_roc = FALSE,
  k_folds = 5,
  step = 0.5,
  run_rfcv = TRUE
)
```

Arguments

`data`	A data frame containing the cytokine data, with one column as the grouping variable and the rest as numerical features.
`group_col`	A string representing the name of the column with the grouping variable (i.e., the target variable for classification).

`ntree` An integer specifying the number of trees to grow in the forest (default is 500).

`mtry` An integer specifying the number of variables randomly selected at each split (default is 5).

`train_fraction` A numeric value between 0 and 1 representing the proportion of data to use for training (default is 0.7).

`plot_roc` A logical value indicating whether to plot the ROC curve and calculate the AUC for binary classification (default is FALSE).

`k_folds` An integer specifying the number of folds for cross-validation (default is 5).

`step` A numeric value specifying the fraction of variables to remove at each step during cross-validation for feature selection (default is 0.5).

`run_rfcv` A logical value indicating whether to run Random Forest cross-validation for feature selection (default is TRUE).

Details

The function fits a Random Forest model to the provided data, splitting it into training and test sets. It calculates key performance metrics such as accuracy, sensitivity, and specificity for both the training and test sets. For binary classification tasks, it can also plot the ROC curve and calculate the AUC. If `run_rfcv` is set to TRUE, Random Forest cross-validation is performed to identify the optimal number of features for classification.

Value

A list containing:

`model` The trained Random Forest model.

`confusion_matrix` The confusion matrix of the test set predictions.

`importance_plot` A ggplot object showing the variable importance plot based on Mean Decrease Gini.

`rfcv_result` Results from Random Forest cross-validation for feature selection (if `run_rfcv` is TRUE).

`importance_data` A data frame containing the variable importance based on the Gini index.

Examples

```
## Not run:
# Example usage:
data.df0 <- cytodata
data.df <- data.frame(data.df0[,1:4], log2(data.df0[,-c(1:4)]))
data.df <- data.df[,-c(1,3,4)]
data.df <- filter(data.df, Group != "ND")

results <- cyt.rf(data = data.df, group_col = 'Group', k_folds = 5,
ntree = 1000, mtry = 4, run_rfcv = TRUE, plot_roc = TRUE)

## End(Not run)
```

cyt.skku *Distribution of the Data Set Shown by Skewness and Kurtosis*

Description

This function computes summary statistics—including sample size, mean, standard error, skewness, and kurtosis—for each numeric measurement column in a data set. If grouping columns are provided via `group.cols`, the function computes the metrics separately for each group defined by the combination of these columns (using the first element as the treatment variable and the second as the grouping variable, or the same column for both if only one is given). When no grouping columns are provided, the entire data set is treated as a single group ("Overall"). A log2 transformation (using a cutoff equal to one-tenth of the smallest positive value in the data) is applied to generate alternative metrics. Histograms showing the distribution of skewness and kurtosis for both raw and log2-transformed data are then generated and saved to a PDF if a file name is provided.

Usage

```
cyt.skku(
  data,
  group.cols = NULL,
  Title = NULL,
  printResRaw = FALSE,
  printResLog = FALSE
)
```

Arguments

`data` A matrix or data frame containing the raw data. If `group.cols` is specified, the columns with names in `group.cols` are treated as grouping variables and all other columns are assumed to be numeric measurement variables.

`group.cols` A character vector specifying the names of the grouping columns. When provided, the first element is treated as the treatment variable and the second as the group variable. If not provided, the entire data set is treated as one group.

`Title` A character string specifying the file name for the PDF file in which the histograms will be saved. If omitted, the plots are generated on the current graphics device.

`printResRaw` Logical. If `TRUE`, the function returns and prints the computed summary statistics for the raw data. Default is `FALSE`.

`printResLog` Logical. If `TRUE`, the function returns and prints the computed summary statistics for the log2-transformed data. Default is `FALSE`.

Details

A cutoff is computed as one-tenth of the minimum positive value among all numeric measurement columns to avoid taking logarithms of zero. When grouping columns are provided, the function loops over unique treatment levels (using the first element of `group.cols` as treatment and the second as group, if available) and computes the metrics for each measurement column within each subgroup. Without grouping columns, the entire data set is analyzed as one overall group.

Value

The function generates histograms of skewness and kurtosis for both raw and log2-transformed data. Additionally, if either `printResRaw` and/or `printResLog` is `TRUE`, the function returns the corresponding summary statistics as a data frame or a list of data frames.

Examples

```
## Not run:
# Example with grouping columns (e.g., "Group" and "Treatment")
data(cytodata)
cyt.skku(cytodata[,-c(1,3,4)], Title = "Skew_and_Kurtosis.pdf", group.cols = c("Group"))

# Example without grouping columns (analyzes the entire data set)
cyt.skku(cytodata[,-c(1,4)], Title = "Skew_and_Kurtosis_Overall.pdf")

## End(Not run)
```

`cyt.ttest` *Two Sample T-test Comparisons*

Description

This function performs pairwise comparisons between two groups for each combination of a categorical predictor (with exactly two levels) and a continuous outcome variable. It first converts any character variables in `data` to factors and applies a log2 transformation to the continuous variables if specified. Depending on the value of `scale`, the function conducts either a two-sample t-test or a Mann-Whitney U test and prints the resulting p-values. An error is thrown if a categorical variable does not have exactly two levels.

Usage

```
cyt.ttest(data, scale = NULL)
```

Arguments

`data` A matrix or data frame containing continuous variables and categorical variables.

`scale` A character value specifying a transformation for continuous variables. Options are `NULL` (default) and `"log2"`. When `scale = "log2"`, a log2 transformation is applied and a two-sample t-test is used; when `scale` is `NULL`, a Mann-Whitney U test is performed.

Value

A list of p-values from the statistical tests for each combination of continuous outcome and categorical predictor is returned.

Examples

```
## Not run:
  data.df <- cytodata[,-c(1,4)]
  data.df <- filter(data.df, Group != "ND", Treatment != "Unstimulated")
  # Two sample T-test
  cyt.ttests(data.df[, c(1,2, 5:6)], scale = "log2")
  # Mann Whitney U Test
  cyt.ttests(data.df[, c(1,2, 5:6)])

## End(Not run)
```

`cyt.volc` *Volcano Plot*

Description

This function subsets the numeric columns from the input data and compares them based on a selected grouping column. It computes the fold changes (as the ratio of means) and associated p-values (using two-sample t-tests) for each numeric variable between two groups. The results are log2-transformed (for fold change) and -log10-transformed (for p-values) to generate a volcano plot.

Usage

```
cyt.volc(
  data,
  group_col,
  cond1 = NULL,
  cond2 = NULL,
  fold_change_thresh = 2,
  p_value_thresh = 0.05,
  top_labels = 10
)
```

Arguments

`data` A matrix or data frame containing the data to be analyzed.

`group_col` A character string specifying the column name used for comparisons (e.g., group, treatment, or stimulation).

`cond1` A character string specifying the name of the first condition for comparison. Default is `NULL`.

`cond2` A character string specifying the name of the second condition for comparison. Default is `NULL`.

`fold_change_thresh` A numeric threshold for the fold change. Default is `2`.

`p_value_thresh` A numeric threshold for the p-value. Default is `0.05`.

`top_labels` An integer specifying the number of top variables to label on the plot. Default is `10`.

Value

A list of volcano plots (as ggplot objects) for each pairwise comparison. Additionally, the function prints the data frame used for plotting (excluding the significance column) from the final comparison.

Note

If cond1 and cond2 are not provided, the function automatically generates all possible pairwise combinations of groups from the specified group_col for comparisons.

Examples

```
## Not run:
  cyt.volc(cytodata, group_col = "Group")
  cyt.volc(cytodata, group_col = "Group", fold_change_thresh = 2, top_labels = 15)

## End(Not run)
```

cyt.xgb *Run XGBoost Classification on Cytokine Data*

Description

This function trains and evaluates an XGBoost classification model on cytokine data. It allows for hyperparameter tuning, cross-validation, and visualizes feature importance.

Usage

```
cyt.xgb(
  data,
  group_col,
  train_fraction = 0.7,
  nrounds = 500,
  max_depth = 6,
  eta = 0.1,
  nfold = 5,
  cv = FALSE,
  objective = "multi:softprob",
  early_stopping_rounds = NULL,
  eval_metric = "mlogloss",
  gamma = 0,
  colsample_bytree = 1,
  subsample = 1,
  min_child_weight = 1,
  top_n_features = 10,
  verbose = 1,
  plot_roc = FALSE
)
```

Arguments

`data` A data frame containing the cytokine data, with one column as the grouping variable and the rest as numerical features.

`group_col` A string representing the name of the column with the grouping variable (i.e., the target variable for classification).

`train_fraction` A numeric value between 0 and 1 representing the proportion of data to use for training (default is 0.7).

`nrounds` An integer specifying the number of boosting rounds (default is 500).

`max_depth` An integer specifying the maximum depth of the trees (default is 6).

`eta` A numeric value representing the learning rate (default is 0.1).

`nfold` An integer specifying the number of folds for cross-validation (default is 5).

`cv` A logical value indicating whether to perform cross-validation (default is FALSE).

`objective` A string specifying the XGBoost objective function (default is "multi:softprob" for multi-class classification).

`early_stopping_rounds`
An integer specifying the number of rounds with no improvement to stop training early (default is NULL).

`eval_metric` A string specifying the evaluation metric (default is "mlogloss").

`gamma` A numeric value for the minimum loss reduction required to make a further partition (default is 0).

`colsample_bytree`
A numeric value specifying the subsample ratio of columns when constructing each tree (default is 1).

`subsample` A numeric value specifying the subsample ratio of the training instances (default is 1).

`min_child_weight`
A numeric value specifying the minimum sum of instance weight needed in a child (default is 1).

`top_n_features` An integer specifying the number of top features to display in the importance plot (default is 10).

`verbose` An integer specifying the verbosity of the training process (default is 1).

`plot_roc` A logical value indicating whether to plot the ROC curve and calculate the AUC for binary classification (default is FALSE).

Details

The function allows for training an XGBoost model on cytokine data, splitting the data into training and test sets. If cross-validation is enabled (`cv = TRUE`), it performs k-fold cross-validation and prints the best iteration based on the evaluation metric. The function also visualizes the top N important features using `xgb.ggplot.importance()`.

Value

A list containing:

`model` The trained XGBoost model.

`confusion_matrix`
The confusion matrix of the test set predictions.

`importance`	The feature importance matrix for the top features.
`class_mapping`	A named vector showing the mapping from class labels to numeric values used for training.
`cv_results`	Cross-validation results, if cross-validation was performed (otherwise NULL).
`plot`	A ggplot object showing the feature importance plot.

Examples

```
## Not run:
# Example usage:
data.df0 <- cytodata
data.df <- data.frame(data.df0[,1:4], log2(data.df0[,-c(1:4)]))
data.df <- data.df[,-c(1,3,4)]
data.df <- filter(data.df, Group != "ND")

cyt.xgb(data = data.df, group_col = 'Group',
nrounds = 500, max_depth = 4, eta = 0.05,
nfold = 5, cv = TRUE, eval_metric = "mlogloss",
early_stopping_rounds = NULL, top_n_features = 10,
verbose = 0, plot_roc = TRUE)

## End(Not run)
```

`cytodata` *Cytokine Profiling Data*

Description

Contains observed values of cytokines and their respective treatment and groups.

Usage

```
cytodata
```

Format

`cytodata`**:**

A data frame with 297 rows and 29 columns:

Group Group assigned to the subjects.

Stimulation Stimulation recieved by subjects.

Treatment Treatment recieved by subjects.

Source

Example data put together for cytokine profiling.

Examples

```
data(cytodata)
```

Index