Package 'DaMiRseq'

April 3, 2025

Type Package

Date 2021-11-20

Title Data Mining for RNA-seq data: normalization, feature selection and classification

Version 2.19.0

Author Mattia Chiesa <mattia.chiesa@cardiologicomonzino.it>, Luca Piacentini <luca.piacentini@cardiologicomonzino.it>

Maintainer Mattia Chiesa <mattia.chiesa@cardiologicomonzino.it>

Description The DaMiRseq package offers a tidy pipeline of data mining procedures to identify transcriptional biomarkers and exploit them for both binary and multi-class classification purposes. The package accepts any kind of data presented as a table of raw counts and allows including both continous and factorial variables that occur with the experimental setting. A series of functions enable the user to clean up the data by filtering genomic features and samples, to adjust data by identifying and removing the unwanted source of variation (i.e. batches and confounding factors) and to select the best predictors for modeling. Finally, a ``stacking" ensemble learning technique is applied to build a robust classification model. Every step includes a checkpoint that the user may exploit to assess the effects of data management by looking at diagnostic plots, such as clustering and heatmaps, RLE boxplots, MDS or correlation plot.

License GPL (>= 2)

Encoding UTF-8

LazyData true

biocViews Sequencing, RNASeq, Classification, ImmunoOncology

VignetteBuilder knitr

Imports DESeq2, limma, EDASeq, RColorBrewer, sva, Hmisc, pheatmap, FactoMineR, corrplot, randomForest, e1071, caret, MASS, lubridate, plsVarSel, kknn, FSelector, methods, stats, utils,

graphics, grDevices, reshape2, ineq, arm, pls, RSNNS, edgeR, plyr

Suggests BiocStyle, knitr, testthat

Depends R (>= 3.4), SummarizedExperiment, ggplot2

RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/DaMiRseq

git_branch devel

git_last_commit 83ca733

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-04-03

Contents

`DaMiR.Allplot` *Quality assessment and visualization of expression data*

Description

This is a helper function to easily draw (1) clustering dendrogram and heatmap of a sample-per-sample correlation matrix, (2) multidimensional scaling plots (MDS), (3) relative log expression (RLE) boxplots of expression data, (4) a sample-by-sample expression value distribution, and (5) a class average expression value distribution

Usage

```
DaMiR.Allplot(
  data,
  df,
  type = c("spearman", "pearson"),
 what = c("all", "all_w_PCA", "MDS", "PCA", "heatmap", "RLEbox", "distr", "avg_distr")
)
```

Arguments

`data`	A SummarizedExperiment object or a matrix or a data.frame where rows and cols should be, respectively, observations and features
`df`	A data frame with class and known variables (or a subset of them); at least one column with 'class' label must be included
`type`	A character string specifing the metric to be applied to correlation analysis. Either "spearman" or "pearson" is allowed; default is "spearman"
`what`	A character string specifing the plots to be shown 'all', 'all_w_PCA', 'MDS','PCA','heatmap','RLEbox', 'distr', 'avg_distr' are allowed; default is "all"

Details

Please be sure that NAs are not present in `df`'s columns. Plots will not be drawn in the presence of NAs.

Value

A dendrogram and heatmap, MDS plot(s), a RLE boxplot, a sample-by-sample expression value distribution, and a class average expression value distribution

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(data_norm)
data(df)
# Draw clustering dendrogram and heatmap, MDS, RLE boxplot:
DaMiR.Allplot(data=data_norm, df=df[,5,drop=FALSE])
```

`DaMiR.Clustplot` *Expression data clustering and heatmap*

Description

The function helps to draw a clustering dendrogram and a heatmap of expression data.

Usage

```
DaMiR.Clustplot(
  data,
  df,
  type_row = c("euclidean", "correlation"),
  type_col = c("euclidean", "correlation")
)
```

Arguments

`data`	A SummarizedExperiment object or a matrix or a data.frame where rows and cols should be, respectively, observations and features
`df`	A data frame with class and (optionally) known variables; at least one column with 'class' label must be included
`type_row`	The metric to be used to cluster rows. Either "euclidean" or "correlation" is allowed; default is "euclidean"
`type_col`	The metric to be used to cluster cols. Either "euclidean" or "correlation" is allowed; default is "euclidean"

Value

A clustering dendrogram and heatmap.

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(data_norm)
data(df)
# use the first 100 genes:
data_norm_red<-data_norm[1:100,]
# Draw heatmap: samples (cols) per genes (rows)
# and use variable annotation:
DaMiR.Clustplot(data=data_norm_red,
df=df, type_row="correlation", type_col="correlation")
```

`DaMiR.corrplot` *Correlation Plot*

Description

This function easily draws the correlation plot of surrogate variables (sv) and variables.

Usage

```
DaMiR.corrplot(sv, df, type = c("pearson", "spearman"), sig.level = 0.01)
```

Arguments

`sv`	The matrix of sv identified by `DaMiR.SV` function
`df`	A data frame with class and known variables; at least one column with 'class' label must be included
`type`	Type of correlation metric to be applied; default is "pearson"
`sig.level`	The significance level of the correlation; default is 0.0001

Details

Factorial variables are allowed. They will be tranformed as numeric before applying the `rcorr` function of `Hmisc`.The `corrplot` function, which draws the plot, marks with a cross all the correlations that do not reach the significance threshold defined in the `sig.level` argument.This plot allows the user to identify those sv that present significant correlations with either technical and biological known variables. Notably, none of the sv should present signifcant correlation with "class" variable.

Value

A correlation plot between sv and known variables.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

DaMiR.SV

Examples

```
# use example data:
data(df)
data(sv)
# Draw correlation plot:
#DaMiR.corrplot(sv=sv, df=df, type = "pearson", sig.level=0.01)
```

DaMiR.EnsembleLearning

Build Classifier using 'Staking' Ensemble Learning strategy.

Description

This function implements a 'Stacking' ensemble learning strategy. Users can provide heterogeneous features (other than genomic features) which will be taken into account during classification model building.

Usage

```
DaMiR.EnsembleLearning(
  data,
  classes,
  variables,
  fSample.tr = 0.7,
  fSample.tr.w = 0.7,
  iter = 100,
  cl_type = c("RF", "kNN", "SVM", "LDA", "LR", "NB", "NN", "PLS")
)
```

Arguments

data	A transposed data frame of normalized expression data. Rows and Cols should be, respectively, observations and features
classes	A class vector with nrow(data) elements. Each element represents the class label for each observation. More than two different class labels are handled.
variables	An optional data frame containing other variables (but without 'class' column). Each column represents a different covariate to be considered in the model
fSample.tr	Fraction of samples to be used as training set; default is 0.7
fSample.tr.w	Fraction of samples of training set to be used during weight estimation; default is 0.7
iter	Number of iterations to assess classification accuracy; default is 100

`cl_type` List of weak classifiers that will compose the meta-learners. Only "RF", "kNN", "SVM", "LDA", "LR", "NB", "NN", "PLS" are allowed. Default is c("RF", "LR", "kNN", "LDA", "NB", "SVM")

Details

To assess the robustness of a set of predictors, a specific 'Stacking' strategy has been implemented. First, a training set (TR1) and a test set (TS1) are generated by 'bootstrap' sampling. Then, sampling again from TR1 subset, another pair of training (TR2) and test set (TS2) are obtained. TR2 is used to train Random Forest (RF), Naive Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbour (kNN), Linear Discriminant Analysis (LDA) and Logistic Regression (LR) classifiers, whereas TS2 is used to test their accuracy and to calculate weights. The decision rule of 'Stacking' classifier is made by a linear combination of the product between weigths (w) and predictions (Pr) of each classifier; for each sample k, the prediction is computed by:

$$Pr_{k,Ensemble} = w_{RF} * Pr_{k,RF} + w_{NB} * Pr_{k,NB} + w_{SVM} * Pr_{k,SVM} + w_{k,kNN} * Pr_{k,kNN} + w_{k,LDA} * Pr_{k,LDA} + w_{k,LR} * Pr_k$$

$$Pr_{k,Ensemble} = sum(w[RF] * Pr[k,i]), i = 1, N$$

Performance of 'Stacking' classifier is evaluated by using TS1. This process is repeated several times (default 100 times).

Value

A list containing:

- A matrix of accuracies of each classifier in each iteration.
- A matrix of weights used for each classifier in each iteration.
- A list of all models generated in each iteration.
- A violin plot of model accuracy obtained for each iteration.

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(selected_features)
data(df)
set.seed(1)
# only for the example:
# speed up the process setting a low 'iter' argument value;
# for real data set use default 'iter' value (i.e. 100) or higher:
# Classification_res <- DaMiR.EnsembleLearning(selected_features,
# classes=df$class, fSample.tr=0.6, fSample.tr.w=0.6, iter=3,
#  cl_type=c("RF","kNN"))
```

`DaMiR.EnsembleLearning2cl` *Build a Binary Classifier using 'Staking' Learning strategy.*

Description

This function implements a 'Stacking' ensemble learning strategy. Users can provide heterogeneous features (other than genomic features) which will be taken into account during classification model building. A 'two-classes' classification task is addressed.

Usage

```
DaMiR.EnsembleLearning2cl(
  data,
  classes,
  variables,
  fSample.tr = 0.7,
  fSample.tr.w = 0.7,
  iter = 100,
  cl_type = c("RF", "kNN", "SVM", "LDA", "LR", "NB", "NN", "PLS")
)
```

Arguments

`data`	A transposed data frame of normalized expression data. Rows and Cols should be, respectively, observations and features
`classes`	A class vector with `nrow(data)` elements. Each element represents the class label for each observation. Two different class labels are allowed
`variables`	An optional data frame containing other variables (but without 'class' column). Each column represents a different covariate to be considered in the model
`fSample.tr`	Fraction of samples to be used as training set; default is 0.7
`fSample.tr.w`	Fraction of samples of training set to be used during weight estimation; default is 0.7
`iter`	Number of iterations to assess classification accuracy; default is 100
`cl_type`	List of weak classifiers that will compose the meta-learners. "RF", "kNN", "SVM", "LDA", "LR", "NB", "NN", "PLS" are allowed. Default is c("RF", "LR", "kNN", "LDA", "NB", "SVM")

Details

To assess the robustness of a set of predictors, a specific 'Stacking' strategy has been implemented. First, a training set (TR1) and a test set (TS1) are generated by 'bootstrap' sampling. Then, sampling again from TR1 subset, another pair of training (TR2) and test set (TS2) are obtained. TR2 is used to train Random Forest (RF), Naive Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbour (kNN), Linear Discriminant Analysis (LDA) and Logistic Regression (LR) classifiers,

whereas TS2 is used to test their accuracy and to calculate weights. The decision rule of 'Stacking' classifier is made by a linear combination of the product between weigths (w) and predictions (Pr) of each classifier; for each sample k, the prediction is computed by:

$$Pr_{k,Ensemble} = w_{RF} * Pr_{k,RF} + w_{NB} * Pr_{k,NB} + w_{SVM} * Pr_{k,SVM} + w_{k,kNN} * Pr_{k,kNN} + w_{k,LDA} * Pr_{k,LDA} + w_{k,LR} * Pr_k$$

Performance of 'Stacking' classifier is evaluated by using TS1. This process is repeated several times (default 100 times).

Value

A list containing:

- A matrix of accuracies of each classifier in each iteration.
- A matrix of weights used for each classifier in each iteration.
- A list of all models generated in each iteration.
- A violin plot of model accuracy obtained for each iteration.

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(selected_features)
data(df)
set.seed(1)
# only for the example:
# speed up the process setting a low 'iter' argument value;
# for real data set use default 'iter' value (i.e. 100) or higher:
#  Classification_res <- DaMiR.EnsembleLearning(selected_features,
# classes=df$class, fSample.tr=0.6, fSample.tr.w=0.6, iter=3,
# cl_type=c("RF","kNN"))
```

DaMiR.EnsembleLearningNcl

Build a Multi-Class Classifier using 'Staking' Learning strategy.

Description

This function implements a 'Stacking' ensemble learning strategy. Users can provide heterogeneous features (other than genomic features) which will be taken into account during classification model building. A 'multi-classes' classification task is addressed.

Usage

```
DaMiR.EnsembleLearningNcl(
  data,
  classes,
  variables,
  fSample.tr = 0.7,
  fSample.tr.w = 0.7,
  iter = 100,
  cl_type = c("RF", "kNN", "SVM", "LDA", "LR", "NB", "NN", "PLS")
)
```

Arguments

data	A transposed data frame of normalized expression data. Rows and Cols should be, respectively, observations and features
classes	A class vector with nrow(data) elements. Each element represents the class label for each observation. More than two different class labels are allowed
variables	An optional data frame containing other variables (but without 'class' column). Each column represents a different covariate to be considered in the model
fSample.tr	Fraction of samples to be used as training set; default is 0.7
fSample.tr.w	Fraction of samples of training set to be used during weight estimation; default is 0.7
iter	Number of iterations to assess classification accuracy; default is 100
cl_type	List of weak classifiers that will compose the meta-learners. "RF", "kNN", "SVM", "LDA", "LR", "NB", "NN", "PLS" are allowed. Default is c("RF", "LR", "kNN", "LDA", "NB", "SVM")

Details

To assess the robustness of a set of predictors, a specific 'Stacking' strategy has been implemented. First, a training set (TR1) and a test set (TS1) are generated by 'bootstrap' sampling. Then, sampling again from TR1 subset, another pair of training (TR2) and test set (TS2) are obtained. TR2 is used to train Random Forest (RF), Naive Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbour (kNN), Linear Discriminant Analysis (LDA) and Logistic Regression (LR) classifiers, whereas TS2 is used to test their accuracy and to calculate weights. The decision rule of 'Stacking' classifier is made by a linear combination of the product between weigths (w) and predictions (Pr) of each classifier; for each sample k, the prediction is computed by:

$$Pr_{k,Ensemble} = w_{RF} * Pr_{k,RF} + w_{NB} * Pr_{k,NB} + w_{SVM} * Pr_{k,SVM} + w_{k,kNN} * Pr_{k,kNN} + w_{k,LDA} * Pr_{k,LDA} + w_{k,LR} * Pr_k$$

Performance of 'Stacking' classifier is evaluated by using TS1. This process is repeated several times (default 100 times).

Value

A matrix of accuracies of each classifier in each iteration.

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(selected_features)
data(df)
set.seed(1)
# only for the example:
# speed up the process setting a low 'iter' argument value;
# for real data set use default 'iter' value (i.e. 100) or higher:
# Classification_res <- DaMiR.EnsembleLearning(selected_features,
# classes=df$class, fSample.tr=0.6, fSample.tr.w=0.6, iter=3,
# cl_type=c("RF","kNN"))
```

DaMiR.EnsL_Predict *Predict new samples class*

Description

The best model learned by the DaMiR.EnsL_Train functionn is tested on a new dataset, in order to predict the samples class

Usage

```
DaMiR.EnsL_Predict(data, bestModel)
```

Arguments

data	A SummarizedExperiment object or a data frame/matrix of normalized expression data. Rows and Cols should be observations and features, respectively.
bestModel	The best model, selected between those trained by the DaMiR.EnsL_Train function.

Details

This function implements the prediction step on new data, given a model learned by DaMiR.EnsL_Train

Value

A matrix containing the predictions

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(selected_features)
data(df)
```

`DaMiR.EnsL_Test` *Test Binary Classifiers*

Description

This function tests the models learned by the DaMiR.EnsL_Train function, on a test set

Usage

```
DaMiR.EnsL_Test(data, classes, EnsL_model)
```

Arguments

`data`	A SummarizedExperiment object or a data frame/matrix of normalized expression data. Rows and Cols should be observations and features, respectively.
`classes`	A class vector with `nrow(data)` elements. Each element represents the class label for each observation. Two different class labels are allowed. Note. this argument should not be set when 'data' is a SummarizedExperiment object
`EnsL_model`	A list with the models trained by DaMiR.EnsL_Train function.

Details

This function implements the test step of DaMiR.EnsembleLearning2cl function

Value

A dataframe containing the predictions on the testset

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(selected_features)
data(df)
set.seed(1)
# only for the example:
# speed up the process setting a low 'iter' argument value;
# for real data set use default 'iter' value (i.e. 100) or higher:
#  Tr_res <- DaMiR.EnsL_Train(
```

```
#  selected_features,classes=df$class, fSample.tr.w=0.6, iter=3,
# cl_type=c("RF","LR"))
# DaMiR.EnsembleLearning2cl_Test(selected_features,
#classes=df$class,Tr_res)
```

`DaMiR.EnsL_Train` *Train a Binary Classifier using 'Staking' Learning strategy.*

Description

This function learn a meta learner by a 'Stacking' strategy. Users can provide heterogeneous features (other than genomic features) which will be taken into account during classification model building. A 'two-classes' classification task isaddressed.

Usage

```
DaMiR.EnsL_Train(
  data,
  classes,
  variables,
  fSample.tr.w = 0.7,
  cl_type = c("RF", "SVM", "LDA", "LR", "NB", "NN", "PLS")
)
```

Arguments

`data`	A SummarizedExperiment object or a data frame/matrix of normalized expression data. Rows and Cols should be observations and features, respectively.
`classes`	A class vector with `nrow(data)` elements. Each element represents the class label for each observation. Two different class labels are allowed. Note. this argument should not be set when 'data' is a SummarizedExperiment object
`variables`	An optional data frame containing other variables (but without 'class' column). Each column represents a different covariate to be considered in the model
`fSample.tr.w`	Fraction of samples of training set to be used during weight estimation; default is 0.7
`cl_type`	List of weak classifiers that will compose the meta-learners. "RF", "SVM", "LDA", "LR", "NB", "NN", "PLS" are allowed. Default is c("RF", "LR", "LDA", "NB", "SVM")

Details

This function implements the training step of DaMiR.EnsembleLearning2cl function

Value

A list containing:

- The models of each classifier used to build the Ensemble meta-learner with the median or the best accuracy (over the iteration) for the Ensemble classifier;
- the weights associated to each weak classifier;

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(selected_features)
data(df)
set.seed(1)
# For the example:
# speed up the process setting a low 'iter' argument value;
# for real data set use default 'iter' value (i.e. 100) or higher:
#  Classification_res <- DaMiR.EnsL_Train(
#  selected_features,classes=df$class, fSample.tr.w=0.6, iter=3,
# cl_type=c("RF","LR"))
```

DaMiR.FBest *Select best predictors to build Classification Model*

Description

This function allows the user to select a subset of predictors; the number of predictors can be defined by user or selected automatically.

Usage

```
DaMiR.FBest(
  data,
  ranking,
  autoselect = c("no", "yes"),
  n.pred = 10,
  th.zscore = 2
)
```

Arguments

data	A transposed data frame of expression data. Rows and Cols should be, respectively, observations and features
ranking	A data frame with importance score for each feature, generated by DaMiR.FSort
autoselect	A flag to specify how to select predictors: • "no" (default) - Manually: users can specify the number of best predictors, setting n.pred argument • "yes" - Automatically: users have to specify the importance threshold defined by the th.zscore argument; features will be accordingly selected
n.pred	If autoselect="no" then the user have to specify the number of predictors; default is 10
th.zscore	Threshold of scaled importance score (Z-score); default value is 2

Value

A list containing:

- A data frame of normalized expression data of the most important selected predictors.
- A vector with predictors name.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

DaMiR.FSort

Examples

```
# use example data:
data(data_reduced)
data(data_relief)
# select the first 8 predictors rankad by imporatance:
selected_features <- DaMiR.FBest(data_reduced, data_relief, n.pred = 8)
# select predictors by importance but automatically:
selected_features <- DaMiR.FBest(data_reduced, data_relief,
autoselect = "yes", th.zscore = 1.5)
```

DaMiR.FReduct *Remove highly correlated features, based on feature-per-feature correlation.*

Description

This function allows the user to remove highly correlated features.

Usage

```
DaMiR.FReduct(data, th.corr = 0.85, type = c("spearman", "pearson"))
```

Arguments

data	A transposed data frame or matrix of normalized expression data. Rows and Cols should be, respectively, observations and features
th.corr	Feature-per-feature correlation threshold; default is 0.85
type	Type of correlation metric to be applied; default is "spearman"

Details

This function produces an absolute correlation matrix that it is then used to reduce pair-wise correlations. When two features present a correlation higher than that defined by the user in th.corr argument, the function, first, calculates the mean absolute correlation of each feature and, then, removes the feature with the largest mean absolute correlation.

Value

An expression matrix without highly correlated features.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

rcorr, findCorrelation

Examples

```
# use example data:
data(data_reduced)
# reduce the number of features:
data_Reduced <- DaMiR.FReduct(data_reduced,
th.corr = 0.75, type = "pearson")
```

DaMiR.FSelect *Feature selection for classification*

Description

This function identifies the class-correlated principal components (PCs) which are then used to implement a backward variable elimination procedure for the removal of non informative features.

Usage

```
DaMiR.FSelect(
  data,
  df,
  th.corr = 0.6,
  type = c("spearman", "pearson"),
  th.VIP = 3,
  nPlsIter = 1
)
```

Arguments

data	A transposed data frame or a matrix of normalized expression data. Rows and Cols should be, respectively, observations and features
df	A data frame with known variables; at least one column with 'class' label must be included
th.corr	Minimum threshold of correlation between class and PCs; default is 0.6. Note. If df$class has more than two levels, this option is disable and the number of PCs is set to 3.
type	Type of correlation metric; default is "spearman"
th.VIP	Threshold for bve_pls function, to remove non-important variables; default is 3
nPlsIter	Number of times that bve_pls has to run. Each iteration produces a set of selected features, usually similar to each other but not exacly the same! When nPlsIter is > 1, the intersection between each set of selected features is performed; so that, only the most robust features are selected. Default is 1

Details

The function aims to reduce the number of features to obtain the most informative variables for classification purpose. First, PCs obtained by principal component analysis (PCA) are correlated with "class". The correlation threshold is defined by the user in th.corr argument. The higher is the correlation, the lower is the number of PCs returned. Importantly, if df$class has more than two levels, the number of PCs is automatically set to 3. In a binary experimental setting, users should pay attention to appropriately set the th.corr argument because it will also affect the total number of selected features that ultimately depend on the number of PCs. The bve_pls function of plsVarSel package is, then, applied. This function exploits a backward variable elimination

procedure coupled to a partial least squares approach to remove those variable which are less informative with respect to class. The returned vector of variables is further reduced by the following DaMiR.FReduct function in order to obtain a subset of non correlated putative predictors.

Value

A list containing:

- An expression matrix with only informative features.
- A data frame with class and optional variables information.

Author(s)

Mattia Chiesa, Luca Piacentini

References

Tahir Mehmood, Kristian Hovde Liland, Lars Snipen and Solve Saebo (2011). A review of variable selection methods in Partial Least Squares Regression. Chemometrics and Intelligent Laboratory Systems 118, pp. 62-69.

See Also

- bve_pls
- DaMiR.FReduct

Examples

```
# use example data:
data(data_norm)
data(df)
# extract expression data from SummarizedExperiment object
# and transpose the matrix:
t_data<-t(assay(data_norm))
t_data <- t_data[,seq_len(100)]
# select class-related features
data_reduced <- DaMiR.FSelect(t_data, df,
th.corr = 0.7, type = "spearman", th.VIP = 1)
```

DaMiR.FSort *Order features by importance, using RReliefF filter*

Description

This function implements a procedure in order to rank features by their importance evaluated by RReliefF score.

Usage

```
DaMiR.FSort(data, df, fSample = 1)
```

Arguments

data	A transposed data frame of expression data, i.e. transformed counts by vst or rlog. A log2 transformed expression matrix is also accepted. Rows and Cols should be, respectively, observations and features
df	A data frame with class and known variables; at least one column with 'class' label must be included
fSample	Fraction of sample to be used for the implementation of RReliefF algorithm; default is 1

Details

This function is very time-consuming when the number of features is high. We observed there is a quadratic relationship between execution time and the number of features. Thus, we have also implemented a formula which allows the users to estimate the time to perform this step, given the number of features. The formula is:

$$T = 0.0011 * N^2 - 0.1822 * N + 27.092$$

where T = Time and N = Number of genes. We strongly suggest to filter out non informative features before performing this step.

Value

A data frame with two culmuns, where features are sorted by importance scores:

- RReliefF score - Calculated by `relief` function, implemented in `FSelector` package;
- scaled.RReliefF score - Z-score value, computed for each RReliefF score.

A plot with the first 50 features ordered by their importance.

Author(s)

Mattia Chiesa, Luca Piacentini

References

Marko Robnik-Sikonja, Igor Kononenko: An adaptation of Relief for attribute estimation in regression. In: Fourteenth International Conference on Machine Learning, 296-304, 1997

See Also

`relief`, `DaMiR.FSelect`, `DaMiR.FReduct`

Examples

```
# use example data:
data(data_reduced)
data(df)
# rank features by importance:
df.importance <- DaMiR.FSort(data_reduced[,1:10],
 df, fSample = 0.75)
```

DaMiR.goldenDice *Generate a Number to Set Seed*

Description

This function implements a formula based on current date and time.

Usage

```
DaMiR.goldenDice()
```

Details

The number is generated by combining current seconds (S), minutes (Mi), hours (H), days (D), months (Mo), years (Y) and golden ratio (ϕ), in the form:

$$Num = (S * Mi + H * D * Mo/D)^{\phi}$$

Value

An integer number.

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
gen_numb <- DaMiR.goldenDice()
set.seed(gen_numb)
```

DaMiR.iTSadjust *Batch correction of normalized Independent Test Set*

Description

This function aims to perform a batch correction on a normalized independent test set, exploiting the ComBat function of the sva package.

Usage

```
DaMiR.iTSadjust(adj_Learning_set, norm_Ind_Test_set, iTS_batch)
```

Arguments

adj_Learning_set	A SummarizedExperiment object or a data frame/matrix of adjusted and normalized data, obtained by the DaMiR.SVadjust function.
norm_Ind_Test_set	A data frame or a matrix of normalized data. The independent test set is supposed to be already normlaized by the DaMiR.iTSnorm function
iTS_batch	(Optional). A factor or a data.frame, containing information regarding experimental batches of the independent test set. Users can ignore this argument, if the independent test set is deemed a single experimental batch.

Details

The function applied a batch correction procedure to the independent test set, normalized by DaMiR.iTSnorm.

Value

A matrix containing a normalized and adjusted expression matrix (log2 scale).

Author(s)

Mattia Chiesa, Luca Piacentini

References

Jeffrey T. Leek, W. Evan Johnson, Hilary S. Parker, Elana J. Fertig, Andrew E. Jaffe and John D. Storey (2016). sva: Surrogate Variable Analysis. R package version 3.22.0.

See Also

ComBat

Examples

```
# use example data:
data(SE)
```

`DaMiR.iTSnorm` *Normalization of Independent Test Set*

Description

This function aims to normalize properly an actual independent test set by taking information from the Learning set that will be used to transform the new sample(s).

Usage

```
DaMiR.iTSnorm(
  Learning_set,
  Ind_Test_set,
  normtype = c("vst", "rlog", "logcpm"),
  method = c("precise", "quick")
)
```

Arguments

`Learning_set`	A SummarizedExperiment object or a data frame/matrix of raw count data. The learning set is supposed to be a raw counts dataset of the expressed features (not all features). Rows and Cols should be features and samples, respectively.
`Ind_Test_set`	A SummarizedExperiment object or a data frame/matrix of raw count data. The independent test set is supposed to be a raw counts dataset with the same features of 'Learning_set'. Rows and Cols should be features and samples, respectively.
`normtype`	Type of normalization to be applied: `varianceStabilizingTransformation` (`vst`), `rlog` or `logcpm` are allowed; default is "`vst`".
`method`	Type of method to estimate the dispersion, applied to the independent test set to normalize data. Only 'precise' and 'quick' are allowed. In the first case, the dispersion is estimated by the Learning set and applied to the independent test set. In the second case, is estimated from the independent test set. Default is "precise". See details in dispersionFunction

Details

The Learning_set is supposed to be a raw counts dataset of the expressed features. Moreover, the independent test set is supposed to be a raw counts dataset with the same features of 'Learning_set'. The independent test set is normalized, taking into account the dispersion parameter, estimated by the Learning set ('precise' method) or by the independent test set itself ('quick' method).

Value

A matrix containing a normalized expression matrix (log2 scale)

Author(s)

Mattia Chiesa, Luca Piacentini

References

Michael I Love, Wolfgang Huber and Simon Anders (2014): Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2. Genome Biology

See Also

`varianceStabilizingTransformation`, `rlog` `cpm`

Examples

```
# use example data:
data(SE)
```

`DaMiR.makeSE` *Import RNA-Seq count data and variables*

Description

This is an helper function that allows the user to simultaneously import counts, class (mandatory) and variables (optional) data, and creates a `SummarizedExperiment` object.

Usage

```
DaMiR.makeSE(x, y)
```

Arguments

`x`	A tab-delimited file which contains RNA-Seq count data. Each row is a feature (i.e. gene, transcript, exon etc.) and each column is a sample
`y`	A tab-delimited file which contains experiment information. Each row is a sample and each column is a variable. This file must contain at least one column which represent 'class' information for data adjustment and classification; the class column must be labeled as 'class'

Details

Before creating a `SummarizedExperiment` object, the function performs some checks on input data to ensure that only a matrix of raw counts is accordingly loaded. Other checks allows the identification of missing data (NA) in the data frame of the variables of interest.

Value

A SummarizedExperiment object containing raw counts, class and (optionally) variables of interest.

Author(s)

Mattia Chiesa, Luca Piacentini

References

Morgan M, Obenchain V, Hester J and Pag\'es H (2016). SummarizedExperiment: SummarizedExperiment container. R package version 1.4.0.

See Also

SummarizedExperiment

Examples

```
rawdata.path <- system.file(package = "DaMiRseq","extdata")
# import tab-delimited files:
# sample data are a small subset of Genotype-Tissue Expression (GTEx)
# RNA-Seq database (dbGap Study Accession: phs000424.v6.p1):
count_data <- read.delim(file.path(rawdata.path, "counts_import.txt"))
variables_data <- read.delim(file.path(rawdata.path, "annotation_import.txt"))
# create a SummarizedExperiment object:
SE <- DaMiR.makeSE(count_data, variables_data)
print(SE)
```

DaMiR.MDSplot *Plot multidimentional scaling (MDS)*

Description

A MDS plot is drawn in order to visualize class clustering.

Usage

```
DaMiR.MDSplot(data, df, type = c("spearman", "pearson"))
```

Arguments

data	A SummarizedExperiment object or a matrix or a data.frame where rows and cols should be, respectively, observations and features
df	A data frame with class; it can be directly subset from data
type	A character string specifing the metric to be applied to correlation analysis. Either "spearman" or "pearson" is allowed; default is "spearman"

Details

The MDS plot is drawn taking as input a dissimilarity matrix produced by either a sample-per-sample Pearson's or Spearman's correlation of normalized expression data.

Value

A MDS plot, using only 'class' information

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(data_reduced)
data(df)
# Draw MDS:
DaMiR.MDSplot(data=data_reduced, df=df, type="pearson")
```

DaMiR.ModelSelect *Select the best classification model*

Description

This function selects the bestmodels learned by the DaMiR.EnsL_Train and a tested by DaMiR.EnsL_Test.

Usage

```
DaMiR.ModelSelect(
  df,
  type.sel = c("mode", "median", "greater"),
  th.sel = 0.85,
  npred.sel = c("min", "rnd"),
  metric.idx = 1,
  npred.idx = 2
)
```

Arguments

df	A data frame of performance metrics. At least two columns representing a specific classification metrics (e.g., Accuracy) and the number of predictors must be provided. Additionally, other classification metrics (e.g., MCC, Sensitivity, Specificity,PPV, NPV, AUC, ...) can be appended (from the third column onwards) and used for the evaluation, by correctly setting the 'metric.idx' parameter.

type.sel	The method to select the best models. Only "mode","median" and "greater" values are allowed. For a specific classification metrics, "mode" selects all models whose score is the mode of all scores; "median" selects all models whose score is the median of all scores; and, "greater" selects all models whose score is greater than the value specified in "th.sel". Default: "mode".
th.sel	Threshold for the evaluation of the performance when "type.sel" is equal to "greater". Default: 0.85
npred.sel	The method to select the best model. Only "min" and "rnd" values are allowed. Taking into account the subset of models found by 'type.sel', this parameter selects one single model with the minimum number of predictors ("min") or randomly ("rnd"). Default: "min".
metric.idx	The index of the 'df' column (i.e., classification metrics) to be considered for the models evaluation. Default: 1.
npred.idx	The index of the 'df' column representing the number of predictors. Default: 2.

Details

This function finds the best model, taking into account specific classification metrics.

Value

The index of df (row), representing the model selected and a bubble chart

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
set.seed(1)
```

DaMiR.normalization *Filter non Expressed and 'Hypervariant' features and Data Normalization*

Description

Features will be firstly filtered based on their expression value and/or by their variability across samples; features will be then normalized.

Usage

```
DaMiR.normalization(
  data,
  minCounts = 10,
  fSample = 0.5,
  hyper = c("yes", "no"),
  th.cv = 3,
  type = c("vst", "rlog", "logcpm"),
  nFitType = c("parametric", "local", "mean")
)
```

Arguments

data	A SummarizedExperiment object
minCounts	Minimum reads counts; default is 10
fSample	Fraction of samples with minCounts counts; default is 0.5
hyper	Flag to enable gene filtering by Coefficient of Variation (CV); default is "yes"
th.cv	Threshold of minimum CV to consider a feature 'Hypervariant' accross samples; default is 3
type	Type of normalization to be applied: varianceStabilizingTransformation (vst), rlog or logcpm are allowed; default is "vst"
nFitType	Type of method to estimate the dispersion by vst or rlog. Default is "parametric".

Details

Before normalization step, this function allows the user to filter features by:

- Expression - Features will be filtered out whether their reads count do not reach a minCounts in at least fSample of samples;
- CV - The CV of each feature is individually calculated for each sample class. Featurers with both class CV greater than th.cv will be discarded. Computing a class restricted CV may prevent the removal of hypervariant features that may be specifically associated with a certain class. This could be important, for example, for immune genes whose expression under definite conditions may unveil peculiar class-gene association.

Finally, expressed features will be normalized by varianceStabilizingTransformation (default) or rlog, both implemented in DESeq2 package. We suggest to use varianceStabilizingTransformation to speed up the normalization process because rlog is very time-consuming despite the two methods produce quite similar results.

Value

A SummarizedExperiment object which contains a normalized expression matrix (log2 scale) and the data frame with 'class' and (optionally) variables.

Author(s)

Mattia Chiesa, Luca Piacentini

References

Michael I Love, Wolfgang Huber and Simon Anders (2014): Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2. Genome Biology

See Also

varianceStabilizingTransformation, rlog

Examples

```
# use example data:
data(SE)
# perform normalization on a subset of data:
SE_sub<-SE[1:1000, c(1:3, 21:23)]
data_norm <- DaMiR.normalization(SE_sub, minCounts=10, fSample=0.8,
hyper="yes", th.cv = 2.5)
```

DaMiR.sampleFilt *Filter Samples by Mean Correlation Distance Metric*

Description

This function implements a sample-per-sample correlation. Samples with a mean correlation lower than a user's defined threshold will be filtered out.

Usage

```
DaMiR.sampleFilt(data, th.corr = 0.9, type = c("spearman", "pearson"))
```

Arguments

data	A SummarizedExpression object
th.corr	Threshold of mean correlation; default is 0.9
type	Type of correlation metric; default is "spearman"

Details

This step introduces a sample quality checkpoint. Global gene expression should, in fact, exhibit a high correlation among biological replicates; conversely, low correlated samples may be suspected to bear some technical artifact (e.g. poor RNA or library preparation quality), despite they may have passed sequencing quality checks. If not assessed, these samples may, thus, negatively affect all the downstream analysis. This function looks at the mean absolute correlation of each sample and removes those samples with a mean correlation lower than the value set in th.corr argument. This threshold may be specific for different experimental setting but should be as high as possible. For sequencing data we suggest to set th.corr greater than 0.85.

Value

A SummarizedExperiment object which contains a normalized and filtered expression matrix (log2 scale) and a filtered data frame with 'class' and (optionally) variables.

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(data_norm)
# filter out samples with Pearson's correlation <0.92:
data_filt<- DaMiR.sampleFilt(data_norm, th.corr=0.92, type ="pearson")
```

DaMiR.SV *Identification of Surrogate Variables*

Description

This function returns a matrix of surrogate variables (sv) using the implementation by Chiesa-Piacentini or the sva method by Leek et al.

Usage

```
DaMiR.SV(
  data,
  method = c("fve", "leek", "be"),
  th.fve = 0.95,
  second.var = NULL
)
```

Arguments

data	A SummarizedExpression object
method	The method used to identify sv. If missing, the "fve" method will be selected. Otherwise the method "leek" or "be" should be choosen
th.fve	This argument sets the threshold of maximum fraction of variance explained (fve) to be used in conjunction with "fve" method; default is 0.95
second.var	A factor or a numeric vector corresponding to an additional variable to take into account during the sv identification. This variable together with 'class' in the data object will be used to design the model matrix (~ class + second.var)

Details

This function helps the user to identify the appropriate number of sv: it is possible to select a different strategy to be used by changing the option in `method` argument. Three methods are available:

- "be" - this option uses the `num.sv` function of `sva` package with default parameters;
- "leek" - The same of before but with asymptotic approach proposed by Leek;
- "fve" - This method is introduced in `DaMiRseq` package, and integrates part of `sva` function with custom code. Briefly, we computed eigenvalues of `data` using code already implemented in `sva` function and then, we calculated the squared of each eigenvalues. Thus, the ratio between each "squared eigenvalue" and the sum of them were calculated. These values represent a surrogate measure of the "Percentage of Explained Variance" (pve) obtained by principal component analysis (PCA), and their cumulative sum can be used to select sv.

Value

A matrix of sv. A plot with the sv identified by "fve" method is also returned. A red dot shows the maximum number of variables to be included for a specific "fve".

Author(s)

Mattia Chiesa, Luca Piacentini

References

Jeffrey T. Leek, W. Evan Johnson, Hilary S. Parker, Elana J. Fertig, Andrew E. Jaffe and John D. Storey (2016). sva: Surrogate Variable Analysis. R package version 3.22.0.

See Also

`sva`

Examples

```
# use example data:
data(data_norm)
sv <- DaMiR.SV(data_norm, method = "fve", th.fve=0.95)
```

DaMiR.SVadjust *Remove variable effects from expression data*

Description

This function removes surrogate or other confounding variable effects from normalized expression data by the usage of `removeBatchEffect` function of `limma` package.

Usage

```
DaMiR.SVadjust(data, sv, n.sv)
```

Arguments

data	A SummarizedExpression object
sv	The matrix of surrogate variables identified by DaMiR.SV function
n.sv	The number of surrogate variables to be used to adjust the data

Value

A SummarizedExpression object containing a matrix of log-expression values with sv effects removed and the data frame of the variables.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

removeBatchEffect, DaMiR.SV

Examples

```
# use example data:
data(data_norm)
data(sv)
data_adjust <- DaMiR.SVadjust(data_norm, sv = sv, n.sv = 3)
```

DaMiR.transpose *Matrix transposition and replacement of '.' and '-' special characters*

Description

This function transposes matrix and replaces '.' and '-' special characters.

Usage

```
DaMiR.transpose(data)
```

Arguments

data	Matrix of normalized expression data, i.e. transformed counts by vst or rlog. A log2 transformed expression matrix is also accepted

Value

Normalized matrix in which each row is a sample and each column is a feature

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
data(data_norm)
data.transposed <- DaMiR.transpose(assay(data_norm))
```

---

data_min *Example gene-expression dataset for DaMiRseq package*

Description

A dataset with a small dimension of normalized expression data in DaMiRseq package

Usage

```
data_min
```

Format

A data frame with 40 samples (rows) and 87 genes (columns)

Value

An example dataset for `DaMiRseq` package

---

data_norm *A dataset with a normalized matrix to test several DaMiRseq functions: sample data are a subset of Genotype-Tissue Expression (GTEx) RNA-Seq database (dbGap Study Accession: phs000424.v6.p1)*

Description

A dataset with a normalized matrix to test several DaMiRseq functions: sample data are a subset of Genotype-Tissue Expression (GTEx) RNA-Seq database (dbGap Study Accession: phs000424.v6.p1)

Usage

```
data_norm
```

Format

A SummarizedExperiment object containing an assay of 4897 genes (rows) and 40 samples (columns) and a colData with 5 variables

Value

An example dataset for DaMiRseq package

---

data_reduced *Example gene-expression dataset for DaMiRseq package*

Description

A dataset with a small dimension of normalized expression data in DaMiRseq package

Usage

```
data_reduced
```

Format

A list with:

data reduced expression matrix

variables a data frame with variables

Value

An example dataset for DaMiRseq package

---

data_relief *Example ranking dataset for DaMiRseq package*

Description

A data frame with relieF and scaled reliefF scores for each gene

Usage

```
data_relief
```

Format

A dataframe with 87 genes (rows) and 2 variables (columns):

reliefF Score reliefF score for each gene

scaled reliefF Score scaled reliefF score for each gene, by z-score

Value

An example dataset for DaMiRseq package

---

df *Example gene-expression dataset for DaMiRseq package*

Description

A data frame with class and covariates information

Usage

```
df
```

Format

A dataframe with 40 samples (rows) and 5 variables (columns):

center center where sample has been collected

sex sample's gender

age sample's age

death kind of sample's death, based on Hardy scale

class sample's class

Value

An example dataset for DaMiRseq package

SE *Example gene-expression dataset for DaMiRseq package*

Description

A dataset with count matrix to test several DaMiRseq functions. To show package functionality in a reasonably execution time, sample data are a subset of Genotype-Tissue Expression (GTEx) RNA-Seq database (dbGap Study Accession: phs000424.v6.p1). Samples incude 20 Anterior Cingulate Cortex (ACC) tissues and 20 Frontal Cortex (FC) tissues. 21363 genes have been preaviously selected to have 5 read counts in at least 60

Usage

```
SE
```

Format

A SummarizedExperiment object containing an assay of 21363 randomly selected genes (rows) and 40 samples (columns) and a colData with 5 variables

Value

An example dataset for DaMiRseq package

---

selected_features *Example gene-expression dataset for DaMiRseq package*

Description

A dataset with normalized expression data to build classification models in DaMiRseq package

Usage

```
selected_features
```

Format

A dataframe with 40 samples (rows) and 7 variables (genes):

Value

An example dataset for DaMiRseq package

SEtest_norm *A sample dataset with a normalized count matrix for "testthat" functions.*

Description

A sample dataset with a normalized count matrix for "testthat" functions.

Usage

```
SEtest_norm
```

Format

A SummarizedExperiment object containing an assay of 100 genes (rows) and 11 samples (columns) and a colData with 5 variables

Value

An example dataset for DaMiRseq package

sv *Example Surrogate Variables dataset for DaMiRseq package*

Description

A dataset with surrogate variables to test DaMiRseq functions

Usage

```
sv
```

Format

A matrix with 40 samples (rows) and 4 surrogate variables (columns):

Value

An example dataset for `DaMiRseq` package

Index