Package 'MSstatsSampleSize'

October 14, 2021

Type Package

Title Simulation tool for optimal design of high-dimensional MS-based proteomics experiment

Version 1.6.0

Date 2020-03-03

Description The packages estimates the variance in the input protein abundance data and simulates data with predefined number of biological replicates based on the variance estimation.
It reports the mean predictive accuracy of the classifier and mean protein importance over multiple iterations of the simulation.

Imports ggplot2, BiocParallel, caret, gridExtra, reshape2, stats, utils, grDevices, graphics, MSstats

Suggests BiocStyle, knitr, rmarkdown, testthat

VignetteBuilder knitr

biocViews MassSpectrometry, Proteomics, Software, DifferentialExpression, Classification, PrincipalComponent, ExperimentalDesign, Visualization

License Artistic-2.0

Depends R (>= 3.6)

Encoding UTF-8

LazyData true

URL `http://msstats.org`

BugReports `https://groups.google.com/forum/#!forum/msstats`

RoxygenNote 7.0.2

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

git_branch RELEASE_3_13

git_last_commit 0e335a0

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

Author Ting Huang [aut, cre],
Meena Choi [aut],
Olga Vitek [aut]

Maintainer Ting Huang <thuang0703@gmail.com>

R topics documented:

classification_results 2
designSampleSizeClassification 3
designSampleSizeClassificationPlots 5
designSampleSizeHypothesisTestingPlot 8
designSampleSizePCAplot 10
estimateVar 12
meanSDplot 13
MSstatsSampleSize 15
OV_SRM_train 15
OV_SRM_train_annotation 16
simulateDataset 17
simulated_datasets 19
variance_estimation 20

Index **22**

`classification_results` *Example of output from designSampleSizeClassification function*

Description

It is the output of `designSampleSizeClassification` function with a list of `simulated_datasets` generated under same protein number and sample size. The list should include the required elements as below.

Usage

```
classification_results
```

Format

A list with five elements

Details

- num_proteins : the number of simulated proteins
- num_samples : a vector with the number of simulated samples in each condition
- results : a list with 'num_proteins' elements. Each element has (1) classification models trained on each simulated dataset; (2) the predictive accuracy on the validation set predicted by the corresponding classification model.
- mean_predictive_accuracy : the mean predictive accuracy over all the simulated datasets.
- mean_feature_importance : the mean protein importance vector over all the simulated datasets, the length of which is 'num_proteins'.
- predictive_accuracy : a vector of predictive accuracy on each simulated dataset.
- feature_importance : a matrix of feature importance, where rows are proteins and columns are simulated datasets. the length of which is 'num_proteins'.

Examples

```
classification_results$num_proteins
classification_results$num_samples
classification_results$mean_predictive_accuracy
head(classification_results$mean_feature_importance)
```

designSampleSizeClassification

Estimate the mean predictive accuracy and mean protein importance over all the simulated datasets

Description

Estimate the mean predictive accuracy and mean protein importance over all the simulated datasets

Usage

```
designSampleSizeClassification(
  simulations,
  classifier = "rf",
  top_K = 10,
  parallel = FALSE
)
```

Arguments

`simulations`	A list of simulated datasets It should be the name of the output of `simulateDataset` function.

`classifier` A string specifying which classfier to use. This function uses function 'train' from package caret. The options are 1) rf (random forest calssifier, default option). 2) nnet (neural network), 3) svmLinear (support vector machines with linear kernel), 4) logreg (logistic regression), and 5) naive_bayes (naive_bayes).

`top_K` the number of proteins selected as important features (biomarker candidates). All the proteins are ranked in descending order based on its importance to separate different groups and the 'top_K' proteins are selected as important features.

`parallel` Default is FALSE. If TRUE, parallel computation is performed.

Details

This function fits the classification model, in order to classify the subjects in each simulated training dataset (the output of `simulateDataset`). Then the fitted model is validated on the (simulated) validation set (the output of `simulateDataset`). Two performance are reported :

(1) the mean predictive accuracy : The function trains classifier on each simulated training dataset and reports the predictive accuracy of the trained classifier on the validation data (output of `simulateDataset` function). Then these predictive accuracies are averaged over all the simulation.

(2) the mean protein importance : It represents the importance of a protein in separating different groups. It is estimated on each simulated training dataset using function 'varImp' from package caret. Please refer to the help file of 'varImp' about how each classifier calculates the protein importance. Then these importance values for each protein are averaged over all the simulation.

The list of classification models trained on each simulated dataset, the predictive accuracy on the validation set predicted by the corresponding classification model and the importance value for all the proteins estimated by the corresponding classification model are also reported.

Value

num_proteins is the number of simulated proteins. It should be the same as one of the output from *simulateDataset*, called *num_proteins*

num_samples is a vector with the number of simulated samples in each condition. It should be the same as one of the output from *simulateDataset*, called *num_samples*

mean_predictive_accuracy is the mean predictive accuracy over all the simulated datasets, which have same 'num_proteins' and 'num_samples'.

mean_feature_importance is the mean protein importance vector over all the simulated datasets, the length of which is 'num_proteins'.

predictive_accuracy is a vector of predictive accuracy on each simulated dataset.

feature_importance is a matrix of feature importance, where rows are proteins and columns are simulated datasets.

results is the list of classification models trained on each simulated dataset and the predictive accuracy on the validation set predicted by the corresponding classification model.

Author(s)

Ting Huang, Meena Choi, Olga Vitek

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

# num_simulations = 10: simulate 10 times
# expected_FC = "data": fold change estimated from OV_SRM_train
# select_simulated_proteins = "proportion":
# select the simulated proteins based on the proportion of total proteins
# simulate_valid = FALSE: use input OV_SRM_train as validation set
# valid_samples_per_group = 50: 50 samples per condition
simulated_datasets <- simulateDataset(data = OV_SRM_train,
                                      annotation = OV_SRM_train_annotation,
                                      num_simulations = 10,
                                      expected_FC = "data",
                                      list_diff_proteins =  NULL,
                                      select_simulated_proteins = "proportion",
                                      protein_proportion = 1.0,
                                      protein_number = 1000,
                                      samples_per_group = 50,
                                      simulate_valid = FALSE,
                                      valid_samples_per_group = 50)

# run classification on simulated datasets without parallel computation
classification_results <- designSampleSizeClassification(simulations = simulated_datasets,
                                                         parallel = FALSE)

classification_results$num_proteins

# a vector with the number of simulated samples in each condition
classification_results$num_samples

# the mean predictive accuracy over all the simulated datasets,
# which have same 'num_proteins' and 'num_samples'
classification_results$mean_predictive_accuracy

# the mean protein importance vector over all the simulated datasets,
# the length of which is 'num_proteins'.
head(classification_results$mean_feature_importance)
```

---

designSampleSizeClassificationPlots
Visualization for sample size calculation in classification

Description

To illustrate the mean classification accuracy and protein importance under different sample sizes through predictive accuracy plot and protein importance plot.

Usage

```
designSampleSizeClassificationPlots(
  data,
  list_samples_per_group,
  num_important_proteins_show = 10,
  protein_importance_plot = TRUE,
  predictive_accuracy_plot = TRUE,
  x.axis.size = 10,
  y.axis.size = 10,
  protein_importance_plot_width = 3,
  protein_importance_plot_height = 3,
  predictive_accuracy_plot_width = 4,
  predictive_accuracy_plot_height = 4,
  ylimUp_predictive_accuracy = 1,
  ylimDown_predictive_accuracy = 0,
  address = ""
)
```

Arguments

`data` A list of outputs from function `designSampleSizeClassification`. Each element represents the results under a specific sample size. The input should include at least two simulation results with different sample sizes.

`list_samples_per_group` A vector includes the different sample sizes simulated. This is required. The number of simulated sample sizes in the input 'data' should be equal to the length of list_samples_per_group

`num_important_proteins_show` The number of proteins to show in protein importance plot.

`protein_importance_plot` TRUE(default) draws protein importance plot.

`predictive_accuracy_plot` TRUE(default) draws predictive accuracy plot.

`x.axis.size` Size of x-axis labeling in predictive accuracy plot and protein importance plot. Default is 10.

`y.axis.size` Size of y-axis labels in predictive accuracy plot and protein importance plot. Default is 10.

`protein_importance_plot_width` Width of the saved pdf file for protein importance plot. Default is 3.

`protein_importance_plot_height` Height of the saved pdf file for protein importance plot. Default is 3.

`predictive_accuracy_plot_width` Width of the saved pdf file for predictive accuracy plot. Default is 4.

`predictive_accuracy_plot_height` Height of the saved pdf file for predictive accuracy plot. Default is 4.

`ylimUp_predictive_accuracy`
: The upper limit of y-axis for predictive accuracy plot. Default is 1. The range should be 0 to 1.

`ylimDown_predictive_accuracy`
: The lower limit of y-axis for predictive accuracy plot. Default is 0.0. The range should be 0 to 1.

`address`
: the name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of 'PredictiveAccuracyPlot.pdf' and 'ProteinImportancePlot.pdf'. The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

Details

This function visualizes for sample size calculation in classification. Mean predictive accuracy and mean protein importance under each sample size is from the input 'data', which is the output from function `designSampleSizeClassification`.

To illustrate the mean predictive accuracy and protein importance under different sample sizes, it generates two types of plots in pdf files as output: (1) The predictive accuracy plot, The X-axis represents different sample sizes and y-axis represents the mean predictive accuracy. The reported sample size per condition can be used to design future experiment

(2) The protein importance plot includes multiple subplots. The number of subplots is equal to 'list_samples_per_group'. Each subplot shows the top 'num_important_proteins_show' most important proteins under each sample size. The Y-axis of each subplot is the protein name and X-axis is the mean protein importance under the sample size.

Value

predictive accuracy plot is the mean predictive accuracy under different sample sizes. The X-axis represents different sample sizes and y-axis represents the mean predictive accuracy.

protein importance plot includes multiple subplots. The number of subplots is equal to 'list_samples_per_group'. Each subplot shows the top 'num_important_proteins_show' most important proteins under each sample size. The Y-axis of each subplot is the protein name and X-axis is the mean protein importance under the sample size.

Author(s)

Ting Huang, Meena Choi, Olga Vitek.

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

# simulate different sample sizes
# 1) 10 biological replicats per group
# 2) 25 biological replicats per group
# 3) 50 biological replicats per group
# 4) 100 biological replicats per group
list_samples_per_group <- c(10, 25, 50, 100)

# save the simulation results under each sample size
multiple_sample_sizes <- list()
for(i in seq_along(list_samples_per_group)){
    # run simulation for each sample size
    simulated_datasets <- simulateDataset(data = OV_SRM_train,
                                          annotation = OV_SRM_train_annotation,
                                          num_simulations = 10, # simulate 10 times
                                          expected_FC = "data",
                                          list_diff_proteins =  NULL,
                                          select_simulated_proteins = "proportion",
                                          protein_proportion = 1.0,
                                          protein_number = 1000,
                                          samples_per_group = list_samples_per_group[i],
                                          simulate_valid = FALSE,
                                          valid_samples_per_group = 50)

    # run classification performance estimation for each sample size
    res <- designSampleSizeClassification(simulations = simulated_datasets,
                                          parallel = TRUE)

    # save results
    multiple_sample_sizes[[i]] <- res
}

## make the plots
designSampleSizeClassificationPlots(data = multiple_sample_sizes,
                                    list_samples_per_group = list_samples_per_group)
```

designSampleSizeHypothesisTestingPlot

Sample size calculation plot for hypothesis testing

Description

Calculate sample size for future experiments based on intensity-based linear model.

Usage

```
designSampleSizeHypothesisTestingPlot(
  data,
  annotation,
  desired_FC = "data",
  select_testing_proteins = "proportion",
  protein_proportion = 1,
  protein_number = 1000,
  FDR = 0.05,
  power = 0.9,
  height = 5,
  width = 5,
  address = ""
)
```

Arguments

`data` Protein abundance data matrix. Rows are proteins and columns are biological replicates (samples).

`annotation` Group information for samples in data. ‘BioReplicate’ for sample ID and ‘Condition’ for group information are required. ‘BioReplicate’ information should match with column names of ‘data’.

`desired_FC` the range of a desired fold change. The first option (Default) is "data", indicating the range of the desired fold change is directly estimated from the input ‘data’, which are the minimal fold change and the maximal fold change in the input ‘data’. The second option is a vector which includes the lower and upper values of the desired fold change (For example, c(1.25,1.75)).

`select_testing_proteins`
the standard to select the proteins for hypothesis testing and sample size calculation. The variance (and the range of desired fold change if desiredFC = "data") for sample size calculation will be estimated from the selected proteins. It can be 1) "proportion" of total number of proteins in the input data or 2) "number" to specify the number of proteins. "proportion" indicates that user should provide the value for ‘protein_proportion’ option. "number" indicates that user should provide the value for ‘protein_number’ option.

`protein_proportion`
Proportion of total number of proteins in the input data to test. For example, input data has 1,000 proteins and user selects ‘protein_proportion=0.1’. Proteins are ranked in decreasing order based on their mean abundance across all the samples. Then, 1,000 * 0.1 = 100 proteins will be selected from the top list to test. Default is 1.0, which meaans that all the proteins will be used.

`protein_number` Number of proteins to test. For example, ‘protein_number=1000’. Proteins are ranked in decreasing order based on their mean abundance across all the samples and top ‘protein_number’ proteins will be selected to test. Default is 1000.

`FDR` a pre-specified false discovery ratio (FDR) to control the overall false positive. Default is 0.05

`power` a pre-specified statistical power which defined as the probability of detecting a true fold change. You should input the average of power you expect. Default is 0.9

`height` Height of the saved pdf file. Default is 5.

`width` Width of the saved pdf file. Default is 5.

`address` The name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of ‘HypothesisTestingSampleSizePlot.pdf’. The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

Details

The function fits intensity-based linear model on the input ‘data’. Then it uses the fitted models and the fold changes estimated from the models to calculate sample size for hypothesis testing through ‘designSampleSize’ function from MSstats package. It outputs the minimal number of biological replciates per condition to acquire the expected FDR and power under different fold changes.

Value

sample size plot for hypothesis testing : the plot for the minimal number of biological replciates per condition to acquire the expected FDR and power under different fold changes.

data frame with columns desiredFC, numSample, FDR, power and CV

Author(s)

Ting Huang, Meena Choi, Olga Vitek

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

# sample size plot for hypothesis testing
HT_res <- designSampleSizeHypothesisTestingPlot(data = OV_SRM_train,
                                                annotation= OV_SRM_train_annotation,
                                                desired_FC = "data",
                                                select_testing_proteins = "proportion",
                                                protein_proportion = 1.0,
                                                protein_number = 1000,
                                                FDR=0.05,
                                                power=0.9)

# data frame with columns desiredFC, numSample, FDR, power and CV
head(HT_res)
```

---

designSampleSizePCAplot

PCA plot for each simulation

Description

PCA plot for each simulation

Usage

```
designSampleSizePCAplot(
  simulations,
  which.PCA = "all",
  x.axis.size = 10,
  y.axis.size = 10,
  dot.size = 3,
  legend.size = 7,
  width = 6,
  height = 5,
  address = ""
)
```

Arguments

`simulations` A list of simulated datasets. It should be the output of `simulateDataset` function.

`which.PCA` Select one PCA plot to show. It can be "all", "allonly", or "simulationX". X should be index of simulation, such as "simulation1" or "simulation5". Default is "all", which generates all the plots. "allonly" generates the PCA plot for the whole input dataset. "simulationX" generates the PCA plot for a specific simulated dataset (given by index).

`x.axis.size` size of x-axis labeling in PCA Plot. Default is 10.

`y.axis.size` size of y-axis labels. Default is 10.

`dot.size` size of dots in PCA plot. Default is 3.

`legend.size` size of legend above Profile plot. Default is 7.

`width` width of the saved pdf file. Default is 6.

`height` height of the saved pdf file. Default is 5.

`address` the name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of ‘PCAPlot.pdf’. The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

Details

This function draws PCA plot for the whole input dataset and each simulated dataset in ‘simulations’ (input for this function). It outputs the number of simulations plus 1 of PCA plots. The first page shows a PCA plot for the input preliminary dataset. Each of the following pages shows a PCA plot under one simulation. x-axis of PCA plot is the first component and y-axis is the second component. This function can be used to validate whether the simulated dataset looks consistent with the input dataset.

Value

PCA plot : x-axis of PCA plot is the first component and y-axis is the second component.

Author(s)

Ting Huang, Meena Choi, Olga Vitek

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

# num_simulations = 10: simulate 10 times
# expected_FC = "data": fold change estimated from OV_SRM_train
# select_simulated_proteins = "proportion":
# select the simulated proteins based on the proportion of total proteins
# simulate_valid = FALSE: use input OV_SRM_train as validation set
# valid_samples_per_group = 50: 50 samples per condition
simulated_datasets <- simulateDataset(data = OV_SRM_train,
                                      annotation = OV_SRM_train_annotation,
                                      num_simulations = 10,
                                      expected_FC = "data",
                                      list_diff_proteins =  NULL,
                                      select_simulated_proteins = "proportion",
                                      protein_proportion = 1.0,
                                      protein_number = 1000,
                                      samples_per_group = 50,
                                      simulate_valid = FALSE,
                                      valid_samples_per_group = 50)

# output a PDF file with multiple PCA plots
designSampleSizePCAplot(simulated_datasets)
```

estimateVar

Estimate the mean abundance and variance of each protein in each condition.

Description

Estimate the mean abundance and variance of each protein in each condition.

Usage

```
estimateVar(data, annotation)
```

Arguments

`data` Data matrix with protein abundance. Rows are proteins and columns are Biological replicates or samples.

`annotation` Group information for samples in data. ‘Run’ for MS run, ‘BioReplicate’ for biological subject ID and ‘Condition’ for group information are required. ‘Run’ information should be the same with the column of ‘data’. Multiple ‘Run’ may come from same ‘BioReplicate’.

Details

The function fits intensity-based linear model on the input data 'data'. This function outputs variance components and mean abundance for each protein.

Value

model is the list of linear models trained for each protein.

mu is the mean abundance matrix of each protein in each phenotype group.

sigma is the sd matrix of each protein in each phenotype group.

promean is the mean abundance vector of each protein across all the samples.

protein is proteins, correpsonding to the rows in *mu* and *sigma* or the element of *promean*.

Author(s)

Ting Huang, Meena Choi, Olga Vitek

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

# estimate the mean protein abunadnce and variance in each condition
variance_estimation <- estimateVar(data = OV_SRM_train,
                                   annotation = OV_SRM_train_annotation)

# the mean protein abundance in each condition
head(variance_estimation$mu)

# the standard deviation in each condition
head(variance_estimation$sigma)

# the mean protein abundance across all the conditions
head(variance_estimation$promean)
```

meanSDplot *Mean-SD plot*

Description

Draw the plot for the mean protein abundance vs standard deviation in each condition. The 'lowess' function is used to fit the LOWESS smoother between mean protein abundance and standard deviation.

Usage

```
meanSDplot(
  data,
  x.axis.size = 10,
  y.axis.size = 10,
  smoother_size = 1,
  xlimUp = 30,
  ylimUp = 3,
  height = 4,
  width = 4,
  address = ""
)
```

Arguments

data	A list with mean protein abundance matrix and standard deviation matrix. It should be the output of estimateVar function.
x.axis.size	Size of x-axis labeling in Mean-SD Plot. Default is 10.
y.axis.size	Size of y-axis labels. Default is 10.
smoother_size	Size of lowess smoother. Default is 1.
xlimUp	The upper limit of x-axis for mean-SD plot. Default is 30.
ylimUp	The upper limit of y-axis for mean-SD plot. Default is 3.
height	Height of the saved pdf file. Default is 4.
width	Width of the saved pdf file. Default is 4.
address	The name of folder that will store the results. Default folder is the current working directory. The other assigned folder has to be existed under the current working directory. An output pdf file is automatically created with the default name of ‘MeanSDPlot.pdf’. The command address can help to specify where to store the file as well as how to modify the beginning of the file name. If address=FALSE, plot will be not saved as pdf file but showed in window.

Value

meanSDplot is the plot for the mean protein abundance (X-axis) vs standard deviation (Y-axis) in each condition.

Author(s)

Ting Huang, Meena Choi, Olga Vitek

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

variance_estimation <- estimateVar(data = OV_SRM_train,
                                   annotation = OV_SRM_train_annotation)
```

```
meanSDplot(variance_estimation)
```

MSstatsSampleSize *MSstatsSampleSize: A package for optimal design of high-dimensional MS-based proteomics experiment*

Description

A set of functions for sample size calculation. The packages estimates the variance in the input protein abundance data and simulates data with pre-defined number of biological replicates based on the variance estimation. It reports the mean predictive accuracy of the classifier and mean protein importance over multiple iterations of the simulation.

functions

- `estimateVar` : estimate the mean abundance and variance of each protein in each condition.
- `meanSDplot` : draw the plot for the mean protein abundance vs standard deviation in each condition.
- `simulateDataset` : simulate datasets with the pre-defined size based on the preliminary data.
- `designSampleSizeClassification` : estimate the mean predictive accuracy and protein importance over all the simulated datasets.
- `designSampleSizePCAplot` : make PCA plots with the first two components for each simulated dataset.
- `designSampleSizeClassificationPlots` : visualization for sample size calculation in classification.
- `designSampleSizeHypothesisTestingPlot` : Sample size calculation plot for hypothesis testing.

OV_SRM_train *The training set from a study for subjects with ovarian cancer*

Description

It is a protein abundance data matrix, where rows are proteins and columns are samples. It includes log2 protein intensities for 67 proteins among 173 biological subjects from control and cancer groups. It is the input for estimateVar and simulateDataset function, with annotation file. It should be prepared by users.

Usage

```
OV_SRM_train
```

Format

A numeric matrix with 67 rows and 173 columns.

References

Huttenhain R and Choi M et al. (2019). A targeted mass spectrometry strategy for developing proteomic biomarkers: a case study of epithelial ovarian cancer. Mol Cell Proteomics 18(9):1836-1850. doi:10.1074/mcp.RA118.001221.

Examples

```
head(OV_SRM_train)
```

`OV_SRM_train_annotation`

Annotation file for `OV_SRM_train`*,*

Description

Annotation of example data, `OV_SRM_train`, in this package. It should be prepared by users. The variables are as follows:

Usage

```
OV_SRM_train_annotation
```

Format

A data frame with 173 rows and 2 variables.

Details

- BioReplicate : Unique ID for biological subject. It should be the same as the column names of `OV_SRM_train`
- Condition : Condition for BioReplicate (ex. Healthy, Cancer, Time0)

References

Huttenhain R and Choi M et al. (2019). A targeted mass spectrometry strategy for developing proteomic biomarkers: a case study of epithelial ovarian cancer. Mol Cell Proteomics 18(9):1836-1850. doi:10.1074/mcp.RA118.001221.

Examples

```
head(OV_SRM_train_annotation)
```

simulateDataset *Simulate datasets with the given number of biological replicates and proteins based on the input* data

Description

Simulate datasets with the given number of biological replicates and proteins based on the input *data*

Usage

```
simulateDataset(
  data,
  annotation,
  num_simulations = 10,
  expected_FC = "data",
  list_diff_proteins = NULL,
  select_simulated_proteins = "proportion",
  protein_proportion = 1,
  protein_number = 1000,
  samples_per_group = 50,
  simulate_validation = FALSE,
  valid_samples_per_group = 50
)
```

Arguments

`data` Protein abundance data matrix. Rows are proteins and columns are biological replicates (samples).

`annotation` Group information for samples in data. 'BioReplicate' for sample ID and 'Condition' for group information are required. 'BioReplicate' information should match with column names of 'data'.

`num_simulations`
Number of times to repeat simulation experiments (Number of simulated datasets). Default is 10.

`expected_FC` Expected fold change of proteins. The first option (Default) is "data", indicating the fold changes are directly estimated from the input 'data'. The second option is a vector with predefined fold changes of listed proteins. The vector names must match with the unique information of Condition in 'annotation'. One group must be selected as a baseline and has fold change 1 in the vector. The user should provide list_diff_proteins, which users expect to have the fold changes greater than 1. Other proteins that are not available in 'list_diff_proteins' will be expected to have fold change = 1

`list_diff_proteins`
Vector of proteins names which are set to have fold changes greater than 1 between conditions. If user selected 'expected_FC= "data" ', this should be NULL.

`select_simulated_proteins`
: The standard to select the simulated proteins among data. It can be 1) "proportion" of total number of proteins in the input data or 2) "number" to specify the number of proteins. "proportion" indicates that user should provide the value for ‘protein_proportion’ option. "number" indicates that user should provide the value for ‘protein_number’ option.

`protein_proportion`
: Proportion of total number of proteins in the input data to simulate. For example, input data has 1,000 proteins and user selects ‘protein_proportion=0.1’. Proteins are ranked in decreasing order based on their mean abundance across all the samples. Then, 1,000 * 0.1 = 100 proteins will be selected from the top list to simulate. Default is 1.0, which meaans that all the proteins will be used.

`protein_number`
: Number of proteins to simulate. For example, ‘protein_number=1000’. Proteins are ranked in decreasing order based on their mean abundance across all the samples and top ‘protein_number’ proteins will be selected to simulate. Default is 1000.

`samples_per_group`
: Number of samples per group to simulate. Default is 50.

`simulate_validation`
: Default is FALSE. If TRUE, simulate the validation set; otherwise, the input ‘data’ will be used as the validation set.

`valid_samples_per_group`
: Number of validation samples per group to simulate. This option works only when user selects ‘simulate_validation=TRUE’. Default is 50.

Details

This function simulate datasets with the given numbers of biological replicates and proteins based on the input dataset (input for this function). The function fits intensity-based linear model on the input *data* in order to get variance and mean abundance, using `estimateVar` function. Then it uses variance components and mean abundance to simulate new training data with the given sample size and protein number. It outputs the number of simulated proteins, a vector with the number of simulated samples in a condition, the list of simulated training datasets, the input preliminary dataset and the (simulated) validation dataset.

Value

num_proteins is the number of simulated proteins. It should be set up by parameters, named *protein_proportion* or *protein_number*

num_samples is a vector with the number of simulated samples in each condition. It should be same as the parameter, *samples_per_group*

input_X is the input protein abundance matrix ‘data’.

input_Y is the condition vector for the input ‘data.

simulation_train_Xs is the list of simulated protein abundance matrices. Each element of the list represents one simulation.

simulation_train_Ys is the list of simulated condition vectors. Each element of the list represents one simulation.

valid_X is the validation protein abundance matrix, which is used for classification.

valid_Y is the condition vector of validation samples.

Author(s)

Ting Huang, Meena Choi, Olga Vitek.

Examples

```
data(OV_SRM_train)
data(OV_SRM_train_annotation)

# num_simulations = 10: simulate 10 times
# expected_FC = "data": fold change estimated from OV_SRM_train
# select_simulated_proteins = "proportion":
# select the simulated proteins based on the proportion of total proteins
# simulate_validation = FALSE: use input OV_SRM_train as validation set
# valid_samples_per_group = 50: 50 samples per condition
simulated_datasets <- simulateDataset(data = OV_SRM_train,
                                      annotation = OV_SRM_train_annotation,
                                      num_simulations = 10,
                                      expected_FC = "data",
                                      list_diff_proteins =  NULL,
                                      select_simulated_proteins = "proportion",
                                      protein_proportion = 1.0,
                                      protein_number = 1000,
                                      samples_per_group = 50,
                                      simulate_validation = FALSE,
                                      valid_samples_per_group = 50)

# the number of simulated proteins
simulated_datasets$num_proteins

# a vector with the number of simulated samples in each condition
simulated_datasets$num_samples

# the list of simulated protein abundance matrices
# Each element of the list represents one simulation
head(simulated_datasets$simulation_train_Xs[[1]]) # first simulation

# the list of simulated condition vectors
# Each element of the list represents one simulation
head(simulated_datasets$simulation_train_Ys[[1]]) # first simulation
```

`simulated_datasets` *Example of output from simulateDataset function*

Description

It is the output of simulateDataset function with two inputs: OV_SRM_train and OV_SRM_train_annotation. The list should include the required elements as below.

Usage

```
simulated_datasets
```

Format

A list with eight elements

Details

- num_proteins : the number of simulated proteins
- num_samples : a vector with the number of simulated samples in each condition
- simulation_train_Xs : the list of simulated protein abundance matrices. Each element of the list represents one simulation
- simulation_train_Ys : the list of simulated condition vectors(simulation_train_Xs). Each element of the list represents one simulation
- input_X : the input protein abundance matrix 'OV_SRM_train'.
- input_Y : is the condition vector for the input 'OV_SRM_train'.
- valid_X: the validation protein abundance matrix, which is used for classification
- valid_Y : the condition vector of validation samples (valid_X)

Examples

```
simulated_datasets$num_proteins
simulated_datasets$num_samples
head(simulated_datasets$simulation_train_Xs[[1]])
head(simulated_datasets$simulation_train_Ys[[1]])
```

variance_estimation *Example of output from estimateVar function*

Description

It is the output of estimateVar function with two inputs: OV_SRM_train and OV_SRM_train_annotation. The list should include the required elements as below.

Usage

```
variance_estimation
```

Format

A list with five elements

Details

- model : the list of linear models trained for each protein.
- mu : the mean abundance matrix of each protein in each condition
- sigma : the standard deviation matrix of each protein in each condition
- promean: the mean abundance vector of each protein across all the samples.
- protein : proteins, correpsonding to the rows in *mu* and *sigma* or the element of *promean*

Examples

```
head(variance_estimation$mu)
head(variance_estimation$sigma)
head(variance_estimation$promean)
```

Index

∗ **datasets**
- classification_results, 2
- OV_SRM_train, 15
- OV_SRM_train_annotation, 16
- simulated_datasets, 19
- variance_estimation, 20

classification_results, 2

designSampleSizeClassification, *2*, 3, *6*, *7*, *15*
designSampleSizeClassificationPlots, 5, *15*
designSampleSizeHypothesisTestingPlot, 8, *15*
designSampleSizePCAplot, 10, *15*

estimateVar, 12, *14*, *15*, *18*, *20*

meanSDplot, 13, *15*
MSstatsSampleSize, 15

OV_SRM_train, 15, *16*, *20*
OV_SRM_train_annotation, 16, *20*

simulated_datasets, *2*, 19
simulateDataset, *3*, *4*, *11*, *15*, 17, *20*

variance_estimation, 20