Package 'KnowSeq'

April 15, 2020

Type Package

Title A R package to extract knowledge by using RNA-seq raw files

Version 1.0.0

Author Daniel Castillo-
Secilla, Juan Manuel Galvez, Francisco Manuel Ortuno, Luis Javier Herrera and Ignacio Rojas.

Maintainer Daniel Castillo Secilla <cased@ugr.es>

Description KnowSeq proposes a whole pipeline that comprises the most relevant steps in the RNA-seq gene expression analysis, with the main goal of extracting biological knowledge from raw data (Differential Expressed Genes, Gene Ontology enrichment, pathway visualization and related diseases). In this sense, KnowSeq allows aligning raw data from the original fastq or sra files, by using the most renowned aligners such as tophat2, hisat2, salmon and kallisto. Nowadays, there is no package that only from the information of the samples to align -included in a text file-, automatically performs the download and alignment of all of the samples. Furthermore, the package includes functions to: calculate the gene expression values; remove batch effect; calculate the Differentially Expressed Genes (DEGs); plot different graphs; and perform the DEGs enrichment with the GO information, pathways visualization and related diseases information retrieval. Moreover, KnowSeq is the only package that allows applying both a machine learning and DEGs enrichment processes just after the DEGs extraction. This idea emerged with the aim of proposing a complete tool to the research community containing all the necessary steps to carry out complete studies in a simple and fast way.

VignetteBuilder knitr

License GPL (>=2)

Depends R (>= 3.6.0), quantreg, mclust, topGO (>= 2.34.0)

Encoding UTF-8

LazyData false

RoxygenNote 6.1.1

biocViews GeneExpression, DifferentialExpression, GeneSetEnrichment, DataImport, Classification, FeatureExtraction, Sequencing, RNASeq, BatchEffect, Normalization, Preprocessing, QualityControl, Genetics, Transcriptomics, Microarray, Metabolomics, Proteomics, Alignment, Pathways, SystemsBiology, MultipleComparison, GO, GraphAndNetwork

Imports stringr, factoextra, kernlab, ggplot2, reshape2, gplots, caret, RCurl, XML, class, praznik, R.utils, e1071, randomForest, httr, jsonlite, sva (>= 3.30.1), cqn (>= 1.28.1),

edgeR (>= 3.24.3), biomaRt (>= 2.38.0), limma (>= 3.38.3), arrayQualityMetrics (>= 3.38.0), tximport (>= 1.10.1), tximportData (>= 1.10.0), rhdf5 (>= 2.26.2), Biobase, multtest, pathview (>= 1.22.3), grDevices, graphics, stats, utils

Suggests knitr

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

git_branch RELEASE_3_10

git_last_commit 9f19e01

git_last_commit_date 2020-02-13

Date/Publication 2020-04-14

R topics documented:

`batchEffectRemoval` *Corrects the batch effect of the data by using the selected method.*

Description

This function corrects the batch effect of the expression matrix indicated by parameter. There are two method to choose such as ComBat or SVA.

Usage

```
batchEffectRemoval(expressionMatrix, labels, method = "combat",
  clusters = 2)
```

Arguments

`expressionMatrix`	The original expression matrix to treat the batch effect.
`labels`	A vector that contains the labels of the samples in expressionMatrix.
`method`	The method that will be used to remove the batch effect. The possibilities are "combat" or "sva". Next release will add RUV.
`clusters`	The number of clusters intrinsic to the expression matrix data which could means different batches. The optimal number of clusters in the expression matrix can be calculated by calling the function `dataPlot`, with the parameter mode equal to "optimalClusters". This parameter is only required when the user selects the combat method.

Value

A matrix with the batch effect corrected for combat or a model for `limmaDEGsExtraction` function in the case of sva.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

expressionMatrixNoBatch <- batchEffectRemoval(expressionMatrix, labels, clusters = 4)
svaMod <- batchEffectRemoval(expressionMatrix, labels, method = "sva")
```

`calculateGeneExpressionValues` *Calculates the gene expression values by using a matrix of counts from RNA-seq.*

Description

Calculates the gene expression values by using a matrix of counts from RNA-seq. Furthermore, the conversion from Ensembl IDs to genes names is performed by default, but can be changed with the parameter genesNames.

Usage

```
calculateGeneExpressionValues(countsMatrix, annotation,
  genesNames = TRUE, notHuman = FALSE, notHumanGeneLengthCSV = "")
```

Arguments

countsMatrix	The original counts matrix returned by countsToMatrix function or a matrix with the gene Ensembl ID in the rows and the samples in the columns that contains the count values.
annotation	A matrix that contains the Ensembl IDs, the gene name and the percentage gene gc content for the genes available in the expression matrix. This annotation could be extracted from the function getAnnotationFromEnsembl.
genesNames	A boolean variable which indicates if the rownames of the expression matrix are the genes Names (Symbols) or the ensembl IDs.
notHuman	A boolean variable which indicates if the gene length file is the default gene length human file or another file indicated by parameter.
notHumanGeneLengthCSV	Path to the CSV file that contains the gene length of the specie to use.

Value

A matrix that contains the gene expression values. The rownames are the genes names or the Ensembl IDs and the colnames are the samples.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

expressionMatrix <- calculateGeneExpressionValues(countsMatrix,myAnnotation, genesNames = TRUE)
```

---

countsToMatrix *countsToMatrix merges in a matrix the information in the count files.*

Description

The function merges in a matrix the information in the count files. It can be used from 1 to N count files. These count files can be created by using the function rawAlignment with the raw files of RNA-seq.

Usage

```
countsToMatrix(csvFile, sep = ",")
```

Arguments

csvFile	The csv that contains the name and the path to each of the count files. The column of the name of the file must be named Run and the column that contains the paths must be named Path. Furthermore, to facilitate the posterior steps, a column named Class that contains the classes for the samples must be required.
sep	The separator character of the csvFile or tsvFile.

Value

A matrix with the ensembl ID in the rows and all the samples of each count files in the columns.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
countsInfo <- read.csv(paste(dir,"/countFiles/mergedCountsInfo.csv",sep = ""))

countsInfo$Path <- paste(dir,"/countFiles/",countsInfo$Run,sep = "")

write.csv(countsInfo, file = "countsInfo.csv")

countsInformation <- countsToMatrix("countsInfo.csv")

countsMatrix <- countsInformation$countsMatrix
labels <- countsInformation$labels

file.remove("countsInfo.csv")
```

---

dataPlot *Plot different graphs depending on the current step of KnowSeq pipeline.*

Description

This function allows to plot different charts only by changing the parameters, for the different KnowSeq pipeline steps. Furthermore, the chosen plot can be saved to PNG and PDF.

Usage

```
dataPlot(
  data,
  labels,
  colours = c("green", "red"),
  main = "",
  ylab = "Expression",
  xlab = "Samples",
  xgrid = FALSE,
  ygrid = FALSE,
  legend = "",
  mode = "boxplot",
  toPNG = FALSE,
  toPDF = FALSE
)
```

Arguments

data	Normally, the data parameter is an expression matrix or data.frame, however for the confusionMatrix plot, the data are a confussion matrix that can be achieved by using the output of any of the machine learning functions of this package.
labels	A vector or factor that contains the labels for each of the samples in the data parameter.

colours	A vector that contains the desired colours to plot the different charts. Example: c("red","green","blue").
main	The title for the plot.
ylab	The description for the y axis.
xlab	The description for the x axis.
xgrid	Shows the x grid into the plot
ygrid	Shows the y grid into the plot
legend	A vector with the elements in the legend of the plot.
mode	The different plots supported by this package. The possibilities are boxplot, orderedBoxplot, genesBoxplot, heatmap, confusionMatrix and classResults.
toPNG	Boolean variable to indicate if a plot would be save to PNG.
toPDF	Boolean variable to indicate if a plot would be save to PDF.

Value

Nothing to return.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

dataPlot(expressionMatrix,labels,mode = "boxplot",toPNG = TRUE,toPDF = TRUE)
dataPlot(DEGsMatrix[1:12,],labels,mode = "orderedBoxplot",toPNG = TRUE,toPDF = TRUE)
dataPlot(DEGsMatrix[1:12,],labels,mode = "genesBoxplot",toPNG = TRUE,toPDF = FALSE)
dataPlot(DEGsMatrix[1:12,],labels,mode = "heatmap",toPNG = TRUE,toPDF = TRUE)
```

DEGsPathwayVisualization

The function uses the DEGs to show graphically the expression of the samples in the pathways in which those genes appear.

Description

The function uses the DEGs to show graphically the expression of the samples in the pathways in which those genes appear. For that, the function makes use of a DEGsMatrix with the expression of the DEGs and the annotation of those DEGs in which appear the pathway or pathways of each DEGs. Internally, the function uses pathview to retrieve and colours the pathways, but a maximum number of 24 samples can be used. Furthermore, the function needs the expression matrix with all the genes in order to use them to colour the rest of the elements in the pathways.

Usage

```
DEGsPathwayVisualization(DEGsMatrix, DEGsAnnotation, expressionMatrix,
  expressionAnnotation, labels)
```

Arguments

DEGsMatrix	A matrix that contains the expression of the DEGs for each samples. This matrix can be achieved by calling the function limmaDEGsExtraction. If the samples are more than 24, only the first 24 will be used to colour the pathways.
DEGsAnnotation	A matrix that contains the gene names and the entrez IDs for the genes available in the DEGs matrix. This annotation can be obtained from the function getAnnotationFromEnsembl.
expressionMatrix	A matrix that contains the expression of the all the genes available for each samples. If the samples are more than 24, only the first 24 will be used to colour the pathways.
expressionAnnotation	A matrix that contains the gene names and the entrez IDs for all the genes available. This annotation can be obtained from the function getAnnotationFromEnsembl.
labels	A vector that contains the labels of the samples for both the DEGsMatrix and the expressionMatrix.

Value

Nothing to return.

Examples

```
## Not run: DEGsPathwayVisualization(DEGsMatrix, myDEGsAnnotation, expressionMatrix, allMyAnnotation, labels)
```

DEGsToDiseases *DEGsToDiseases obtains the information about what diseases are related to the DEGs indicated by parameter.*

Description

The function obtains the information about what diseases are related to the DEGs indicated by parameter. For that, the function makes use of the web platforms gene2Diseases and targetValidation.

Usage

```
DEGsToDiseases(geneList, minCitation = 5, size = 10,
  method = "targetValidation")
```

Arguments

geneList	A list that contains the gene symbols or gene names of the DEGs.
minCitation	Minimum number of citations of each genes in a disease to consider the genes related with the disease.
size	The number of diseases to retrieve from targetValidation
method	The name of the desired web platform to use for the diseases download: genes2Diseases or targetValidation

Value

A list which contains the information about the diseases associated to each genes or to a set of genes.

Examples

```
diseases <- DEGsToDiseases(c("KRT19","BRCA1"))
```

`downloadPublicSeries` *Download automatically samples from NCBI/GEO and ArrayExpress public databases.*

Description

Download automatically samples from series of either microarray and RNA-seq. Furthermore, both NCBI/GEO and ArrayExpress public databases are supported. In the case of Microarray, the raw file are downloaded, if they are available, but for RNA-seq a csv is created with the necessary information to download the samples with the function `rawAlignment`.

Usage

```
downloadPublicSeries(samplesVector)
```

Arguments

`samplesVector`	A vector which contains the different IDs of the wanted series. These IDs are the IDs of the series from NCBI/GEO or ArrayExpress.

Value

Nothing to return.

Examples

```
downloadPublicSeries(c("GSE74251"))
```

`featureSelection` *featureSelection function calculates the optimal order of DEGs to achieve the best result in the posterior machine learning process by using mRMR algorithm or Random Forest.*

Description

featureSelection function calculates the optimal order of DEGs to achieve the best result in the posterior machine learning process by using mRMR algorithm or Random Forest. Furthermore, the ranking is returned and can be used as input of the parameter vars_selected in the machine learning functions.

Usage

```
featureSelection(data, labels, vars_selected, mode = "mrmr")
```

Arguments

data	The data parameter is an expression matrix or data.frame that contains the genes in the columns and the samples in the rows.
labels	A vector or factor that contains the labels for each samples in data parameter.
vars_selected	The genes selected to use in the feature selection process. It can be the final DEGs extracted with the function limmaDEGsExtraction or a custom vector of genes.
mode	The algorithm used to calculate the genes ranking. The possibilities are two: mrmr and rf.

Value

A vector that contains the ranking of genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

featureRanking <- featureSelection(t(DEGsMatrix),labels,rownames(DEGsMatrix))
```

---

fileMove *This function is used to move files to other locations.*

Description

This function is used to move files to other locations.

Usage

```
fileMove(from, to)
```

Arguments

from	The current path to the file.
to	The path to the new location of the file.

Value

nothing to return

Examples

```
## Not run: fileMove("ReferenceFiles/GSE74251.csv","ReferenceFiles/GSE74251Moved.csv")
```

gdcClientDownload *This function downloads a list of controlled files from GDC Portal with the user token and the manifest with the information about the desired controlled files.*

Description

This function downloads a list of controlled files from GDC Portal with the user token and the manifest with the information about the desired controlled files.

Usage

```
gdcClientDownload(tokenPath, manifestPath, data)
```

Arguments

tokenPath	Path to the GDC token
manifestPath	Path to the samples manifest
data	The matrix or data.frame with the information from the Samples Sheet downloaded from GDC Portal.

Value

Nothing to return.

Examples

```
# This function needs the download of the pre-compiled tools supplied by KnowSeq.
## Not run: gdcClientDownload("PathToTheToken", "PathToTheFileWithDownloadInfo", dataMatrix)
```

geneOntologyEnrichment *geneOntologyEnrichment obtains the information about what Gene Ontology terms are related to the DEGs.*

Description

The function obtains the information about GO terms from the three differents ontologies that are related to the DEGs. The function also returns the description about each GO and a list of genes that are inside of each GO.

Usage

```
geneOntologyEnrichment(geneMatrix, labels, identificator = "SYMBOL",
  mapping = "org.Hs.eg.db", nGOs = 10, pvalCutOff = 0.01)
```

Arguments

geneMatrix	A matrix that contains the expression of the DEGs for each samples.
labels	A vector that contains the labels of the samples of the DEGsMatrix.
identificator	The identification methods for the genes. By default the identificator is the gene symbol or gene name.
mapping	The annotation database to map the gene and GOs. By default is prepared for homo sapiens but can be changed for other species.
nGOs	Maximun number of GOs to return of the total amount of top GOs. By default is equal to 10.
pvalCutOff	The maximum p-value to considers that a genes is related with a GO term.

Value

A list that contains a matrix for each of the possible ontologies and a matrix with the GOs for the three ontologies together.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

labelsGo <- gsub("Control",0,labels)
labelsGo <- gsub("Tumor",1,labelsGo)

GOsList <- geneOntologyEnrichment(DEGsMatrix,labelsGo,nGOs = 20,pvalCutOff = 0.001)
```

---

getAnnotationFromEnsembl

getAnnotationFromEnsembl returns the required information about a list of genes from Ensembl biomart.

Description

The function returns the required information about a list of genes from Ensembl biomart. This list of genes can be Ensembl ID, gene names or either of the possible values admited by Ensembl biomart. Furthermore, the reference genome can be chosen depending on the necessity of the user.

Usage

```
getAnnotationFromEnsembl(values, attributes = c("ensembl_gene_id",
  "external_gene_name", "percentage_gene_gc_content", "gene_biotype"),
  filters = "ensembl_gene_id", referenceGenome = 38,
  notHSapiens = FALSE, notHumandataset = "")
```

Arguments

values	A list of genes that contains the names or IDs.
attributes	A vector which contains the different information attributes that the Ensembl biomart admit.
filters	The attributes used as filter to return the rest of the attributes.
referenceGenome	The human reference genome used to return the annotation. The possibilities are two: 37 and 38
notHSapiens	A boolean value that indicates if the user wants the human annotation or another annotation available in BiomaRt. The possible not human dataset can be consulted by calling the following function: biomaRt::listDatasets(useMart("ensembl")).
notHumandataset	A dataset identification from biomaRt::listDatasets(useMart("ensembl")).

Value

A matrix that contains all the information asked to the attributes parameter.

Examples

```
myAnnotation <- getAnnotationFromEnsembl(c("ENSG00000210049","ENSG00000211459","ENSG00000210077"),reference
```

---

hisatAlignment *hisatAlignment allows downloading and processing the fastq samples in a CSV file by using hisat2 aligner.*

Description

This function allows downloading and processing the fastq samples in a CSV file by using hisat2 aligner. This funtion is used internally by `rawAlignment` but it can be used separatelly. Furthermore, the function can downloads the reference files required: FASTA Reference Genome and GTF file.

Usage

```
hisatAlignment(data, downloadRef = FALSE, downloadSamples = FALSE,
  createIndex = TRUE, BAMfiles = TRUE, SAMfiles = TRUE,
  countFiles = TRUE, referenceGenome = 38, customFA = "",
  customGTF = "")
```

Arguments

data	The ID of the variable which contains the samples. Our recommendation is to load this variable from a CSV file.
downloadRef	A logical parameter that represents if the reference files will be downloaded or not.
downloadSamples	A logical parameter that represents if the samples of the CSV file will be downloaded or not.

createIndex	A logical parameter that represents if the index of the aligner would be created or not.
BAMfiles	A logical parameter that represents if the you want the BAM files or not.
SAMfiles	A logical parameter that represents if the you want the SAM files or not.
countFiles	A logical parameter that represents if the you want the Count files or not.
referenceGenome	This parameter allows choosing the reference genome that will be used for the alignment. The options are 37,38 or custom. The two first are human genomes, but with the third option you can choose any genome stored in the computer.
customFA	The path to the custom FASTA file of the reference genome.
customGTF	The path to the custom GTF file.

Value

Nothing to return.

Examples

```
# Due to the high computational cost, we strongly recommend it to see the offical documentation and the complete

dir <- system.file("extdata", package="KnowSeq")

#Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv(paste(dir,"/GSE74251.csv",sep = ""))

## Not run: hisatAlignment(GSE74251csv,downloadRef=FALSE,downloadSamples=FALSE, createIndex = TRUE, BAMfiles
```

---

kallistoAlignment *kallistoAlignment allows downloading and processing the fastq samples in a CSV file by using kallisto aligner.*

Description

This function allows downloading and processing the fastq samples in a CSV file by using kallisto aligner. This funtion is used internally by rawAlignment but it can be used separatelly. Furthermore, the function can download the reference files required: FASTA Reference Genome and GTF file.

Usage

```
kallistoAlignment(data, downloadRef = FALSE, downloadSamples = FALSE,
  createIndex = TRUE, BAMfiles = TRUE, SAMfiles = TRUE,
  countFiles = TRUE, referenceGenome = 38, customFA = "",
  customGTF = "", tx2Counts = tx2Counts)
```

Arguments

data	The ID of the variable which contains the samples. Our recommendation is to load this variable from a CSV file.
downloadRef	A logical parameter that represents if the reference files will be downloaded or not.
downloadSamples	A logical parameter that represents if the samples of the CSV file will be downloaded or not.
createIndex	A logical parameter that represents if the index of the aligner would be created or not.
BAMfiles	A logical parameter that represents if the you want the BAM files or not.
SAMfiles	A logical parameter that represents if the you want the SAM files or not.
countFiles	A logical parameter that represents if the you want the Count files or not.
referenceGenome	This parameter allows choosing the reference genome that will be used for the alignment. The options are 37,38 or custom. The two first are human genomes, but with the third option you can choose any genome stored in the computer.
customFA	The path to the custom FASTA file of the reference genome.
customGTF	The path to the custom GTF file.
tx2Counts	A matrix with two columns that contains the conversion of transcripts IDs to genes IDs. There is more information in the function txiimport.

Value

Nothing to return.

Examples

```
# Due to the high computational cost, we strongly recommend it to see the offical documentation and the complete

dir <- system.file("extdata", package="KnowSeq")

#Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv(paste(dir,"/GSE74251.csv",sep = ""))

## Not run: kallistoAlignment(GSE74251csv,downloadRef=FALSE,downloadSamples=FALSE, createIndex = TRUE, BAMfil
```

---

knn_CV *knn_CV allows assessing the final DEGs through a machine learning step by using k-NN in a cross validation process.*

Description

knn_CV allows assessing the final DEGs through a machine learning step by using k-NN in a cross validation process. This function applies a cross validation of n folds with representation of all classes in each fold. The 80% of the data are used for training and the 20% for test. An optimization of the k neighbours is done at the start of the process.

Usage

```
knn_CV(data, labels, vars_selected, numFold = 10)
```

Arguments

data	The data parameter is an expression matrix or data.frame that contains the genes in the columns and the samples in the rows.
labels	A vector or factor that contains the labels for each of the samples in the data object.
vars_selected	The genes selected to classify by using them. It can be the final DEGs extracted with the function limmaDEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.
numFold	The number of folds to carry out in the cross validation process.

Value

A list that contains four objects. The confusion matrix for each fold, the accuracy, the sensitibity and the specificity for each fold and each genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

knn_CV(t(DEGsMatrix),labels,rownames(DEGsMatrix),3)
```

---

knn_test *knn_test allows assessing the final DEGs through a machine learning step by using k-NN with a test dataset.*

Description

knn_test allows assessing the final DEGs through a machine learning step by using k-NN with a test dataset. An optimization of the k neighbours is done at the start of the process.

Usage

```
knn_test(train, labelsTrain, test, labelsTest, vars_selected)
```

Arguments

train	The train parameter is an expression matrix or data.frame that contains the training dataset with the genes in the columns and the samples in the rows.
labelsTrain	A vector or factor that contains the training labels for each of the samples in the train object.
test	The test parameter is an expression matrix or data.frame that contains the test dataset with the genes in the columns and the samples in the rows.
labelsTest	A vector or factor that contains the test labels for each of the samples in the test object.

vars_selected — The genes selected to classify by using them. It can be the final DEGs extracted with the function limmaDEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.

Value

A list that contains four objects. The confusion matrix, the accuracy, the sensitibity and the specificity for each genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

trainingMatrix <- t(DEGsMatrix)[c(1:4,6:9),]
trainingLabels <- labels[c(1:4,6:9)]
testMatrix <- t(DEGsMatrix)[c(5,10),]
testLabels <- labels[c(5,10)]

results_test_knn <- knn_test(trainingMatrix, trainingLabels, testMatrix, testLabels, rownames(DEGsMatrix)[1:1
```

---

limmaDEGsExtraction — *limmaDEGsExtraction performs the analysis to extract the Differentially Expressed Genes (DEGs) among the classes to compare.*

Description

The function performs the analysis to extract the Differentially Expressed Genes (DEGs) among the classes to compare. The number of final DEGs can change depending on the p-value and the LFC indicated by parameters of the function. Furthermore, the function detects if the number of classes are greater than 2 to perform a multiclass DEGs analysis.

Usage

```
limmaDEGsExtraction(expressionMatrix, labels, pvalue = 0.05, lfc = 1,
  cov = 1, number = Inf, svaCorrection = FALSE, svaMod)
```

Arguments

expressionMatrix — The expressionMatrix parameter is an expression matrix or data.frame that contains the genes in the rows and the samples in the columns.

labels — A vector or factors that contains the labels for each of the samples in the expressionMatrix parameter.

pvalue — The value of the p-value which determines the DEGs. If one or more genes have a p-value lower or equal to the selected p-value, they would be considered as DEGs.

lfc — The value of the LFC which determines the DEGs. If one or more genes have a LFC greater or equal to the selected LFC, they would be considered as DEGs.

cov	This value only works when there are more than two classes in the labels. This parameter stablishs a minimum number of pair of classes combination in which exists differential expression to consider a genes as expressed genes.
number	The maximum number of desired genes as output of limma. As default, the function returns all the extracted DEGs with the selected parameters.
svaCorrection	A logical variable that represents if the model for limma is calculated or indicated by parameter from the output of batchEffectRemoval function by using sva method.
svaMod	The model calculated by batchEffectRemoval function by using sva method.

Value

A list that contains two objects. The table with statistics of the different DEGs and a reduced expression matrix which contains the DEGs and the samples.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

expressionMatrix <- calculateGeneExpressionValues(countsMatrix,myAnnotation, genesNames = TRUE)

DEGsInformation <- limmaDEGsExtraction(expressionMatrix, labels, lfc = 2.0,
pvalue = 0.01, number = Inf)

topTable <- DEGsInformation$Table

DEGsMatrix <- DEGsInformation$DEGsMatrix
```

---

plotConfMatrix *plotConfMatrix plots a confusion matrix with some statistics.*

Description

The function plots a confusion matrix with some statistics. The function is used internally by dataPlot but it can be used separatelly.

Usage

```
plotConfMatrix(data)
```

Arguments

data	A table which contains a confusion matrix.

Value

Nothing to return.

Examples

```
data <- table(as.factor(c(1,2,4,2,4,5)),as.factor(c(1,2,5,4,5,2)))
plotConfMatrix(data)
```

rawAlignment *rawAlignment allows downloading and processing the fastq samples in a CSV file.*

Description

This function allows downloading and processing the fastq samples in a CSV file. Also, different alignment methods can be used such as Tophat2, Salmon, Hisat2 and Kallisto. Finally, the function can downloads the reference files required: FASTA Reference Genome and GTF file.

Usage

```
rawAlignment(data, seq = "tophat2", downloadRef = FALSE,
  downloadSamples = FALSE, createIndex = TRUE, BAMfiles = TRUE,
  SAMfiles = TRUE, countFiles = TRUE, referenceGenome = 38,
  customFA = "", customGTF = "", fromGDC = FALSE, tokenPath = "",
  manifestPath = "", tx2Counts = "")
```

Arguments

data	The ID of the variable which contains the samples. Our recommendation is to load this variable from a CSV file.
seq	This parameter represents the alignment method that will be used in the process. The possibilities are "tophat2" "salmon" "hisat2" and "kallisto".
downloadRef	A logical parameter that represents if the reference files will be downloaded or not.
downloadSamples	A logical parameter that represents if the samples of the CSV file will be downloaded or not.
createIndex	A logical parameter that represents if the index of the aligner would be created or not.
BAMfiles	A logical parameter that represents if the you want the BAM files or not.
SAMfiles	A logical parameter that represents if the you want the SAM files or not.
countFiles	A logical parameter that represents if the you want the Count files or not.
referenceGenome	This parameter allows choosing the reference genome that will be used for the alignment. The options are 37,38 or custom. The two first are human genomes, but with the third option you can choose any genome stored in the computer.
customFA	The path to the custom FASTA file of the reference genome.
customGTF	The path to the custom GTF file.
fromGDC	A logical parameter that allows processing BAM files from GDC portal by using the custom reference genome from GDC.
tokenPath	The path to the GDC portal user token. It is required to downloads the controlled BAM files.
manifestPath	The path to the manifest with the information required to downloads the controlled BAM files selected in GDC Portal.
tx2Counts	A matrix with two columns that contains the conversion of transcripts ID to genes ID. There is more information in the function `tximport`. This parameter is only required with salmon and kallisto.

Value

Nothing to return.

Examples

```
# Due to the high computational cost, we strongly recommend it to see the offical documentation and the complete

dir <- system.file("extdata", package="KnowSeq")

#Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv(paste(dir,"/GSE74251.csv",sep = ""))

## Not run: rawAlignment(GSE74251csv,seq="tophat2",downloadRef=FALSE,downloadSamples=FALSE, createIndex = TRU
```

`rf_CV` *rf_CV allows assessing the final DEGs through a machine learning step by using Random Forest in a cross validation process.*

Description

rf_CV allows assessing the final DEGs through a machine learning step by using Random Forest in a cross validation process. This function applies a cross validation of n folds with representation of all classes in each fold. The 80% of the data are used for training and the 20% for test.

Usage

```
rf_CV(data, labels, vars_selected, numFold = 10)
```

Arguments

`data`	The data parameter is an expression matrix or data.frame that contains the genes in the columns and the samples in the rows.
`labels`	A vector or factor that contains the labels for each of the samples in the data object.
`vars_selected`	The genes selected to classify by using them. It can be the final DEGs extracted with the function `limmaDEGsExtraction` or a custom vector of genes. Furthermore, the ranking achieved by `featureSelection` function can be used as input of this parameter.
`numFold`	The number of folds to carry out in the cross validation process.

Value

A list that contains four objects. The confusion matrix for each fold, the accuracy, the sensitibity and the specificity for each fold and each genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

rf_CV(t(DEGsMatrix),labels,rownames(DEGsMatrix),2)
```

rf_test — *rf_test allows assessing the final DEGs through a machine learning step by using Random Forest with a test dataset.*

Description

rf_test allows assessing the final DEGs through a machine learning step by using Random Forest with a test dataset.

Usage

```
rf_test(train, labelsTrain, test, labelsTest, vars_selected)
```

Arguments

train	The train parameter is an expression matrix or data.frame that contains the training dataset with the genes in the columns and the samples in the rows.
labelsTrain	A vector or factor that contains the training labels for each of the samples in the train object.
test	The test parameter is an expression matrix or data.frame that contains the test dataset with the genes in the columns and the samples in the rows.
labelsTest	A vector or factor that contains the test labels for each of the samples in the test object.
vars_selected	The genes selected to classify by using them. It can be the final DEGs extracted with the function limmaDEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.

Value

A list that contains four objects. The confusion matrix, the accuracy, the sensitibity and the specificity for each genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

trainingMatrix <- t(DEGsMatrix)[c(1:4,6:9),]
trainingLabels <- labels[c(1:4,6:9)]
testMatrix <- t(DEGsMatrix)[c(5,10),]
testLabels <- labels[c(5,10)]

rf_test(trainingMatrix,trainingLabels,testMatrix,testLabels,rownames(DEGsMatrix)[1:10])
```

RNAseqQA *RNAseqQA performs the quality analysis of an expression matrix.*

Description

RNAseqQA performs the quality analysis of an expression matrix. This function adapts the RNA-seq data in order to allows using arrayQualityMetrics expression analysis.

Usage

```
RNAseqQA(expressionMatrix, outdir = "RNAseqQA")
```

Arguments

expressionMatrix	A matrix that contains the gene expression values.
outdir	The output directory to store the report of arrayQualityMetrics

Value

Nothing to return.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

RNAseqQA(expressionMatrix)
```

salmonAlignment *salmonAlignment allows downloading and processing the fastq samples in a CSV file by using salmon aligner.*

Description

This function allows downloading and processing the fastq samples in a CSV file by using salmon aligner. This funtion is used internally by rawAlignment but it can be used separatelly. Furthermore, the function can download the reference files required: FASTA Reference Genome and GTF file.

Usage

```
salmonAlignment(data, downloadRef = FALSE, downloadSamples = FALSE,
  createIndex = TRUE, BAMfiles = TRUE, SAMfiles = TRUE,
  countFiles = TRUE, referenceGenome = 38, customFA = "",
  customGTF = "", tx2Counts = tx2Counts)
```

Arguments

data	The ID of the variable which contains the samples. Our recommendation is to load this variable from a CSV file.
downloadRef	A logical parameter that represents if the reference files will be downloaded or not.
downloadSamples	A logical parameter that represents if the samples of the CSV file will be downloaded or not.
createIndex	A logical parameter that represents if the index of the aligner would be created or not.
BAMfiles	A logical parameter that represents if the you want the BAM files or not.
SAMfiles	A logical parameter that represents if the you want the SAM files or not.
countFiles	A logical parameter that represents if the you want the Count files or not.
referenceGenome	This parameter allows choosing the reference genome that will be used for the alignment. The options are 37,38 or custom. The two first are human genomes, but with the third option you can choose any genome stored in the computer.
customFA	The path to the custom FASTA file of the reference genome.
customGTF	The path to the custom GTF file.
tx2Counts	A matrix with two columns that contains the conversion of transcripts IDs to genes IDs. There is more information in the function txiimport.

Value

Nothing to return.

Examples

```
# Due to the high computational cost, we strongly recommend it to see the offical documentation and the complete

dir <- system.file("extdata", package="KnowSeq")

#Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv(paste(dir,"/GSE74251.csv",sep = ""))

## Not run: salmonAlignment(GSE74251csv,downloadRef=FALSE,downloadSamples=FALSE, createIndex = TRUE, BAMfiles
```

---

sraToFastq *sraToFastq downloads and converts the sra files to fastq files. The function admits both gz and sra formats.*

Description

This function downloads and converts the sra files to fastq files by using the URLs indicated through the urlsVector argument. The function admits both gz and sra formats. This function is used internally by rawAlignment but it can be used separatelly.

Usage

```
sraToFastq(urlsVector)
```

Arguments

urlsVector	A vector that contains a list with the URLs requested.

Value

Nothing.

Examples

```
# This function needs the download of the pre-compiled tools supplied by KnowSeq.

## Not run: sraToFastq(c("http://urlToSRA1", "http://urlToSRA2"))
```

---

svm_CV *svm_CV allows assessing the final DEGs through a machine learning step by using svm in a cross validation process.*

Description

svm_CV allows assessing the final DEGs through a machine learning step by using svm in a cross validation process. This function applies a cross validation of n folds with representation of all classes in each fold. The 80% of the data are used for training and the 20% for test. An optimization of C and G hiperparameters is done at the start of the process.

Usage

```
svm_CV(data, labels, vars_selected, numFold = 10)
```

Arguments

data	The data parameter is an expression matrix or data.frame that contains the genes in the columns and the samples in the rows.
labels	A vector or factor that contains the labels for each of the samples in the data object.
vars_selected	The genes selected to classify by using them. It can be the final DEGs extracted with the function limmaDEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.
numFold	The number of folds to carry out in the cross validation process.

Value

A list that contains four objects. The confusion matrix for each fold, the accuracy, the sensitibity and the specificity for each fold and each genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

svm_CV(t(DEGsMatrix),labels,rownames(DEGsMatrix),2)
```

svm_test *svm_test allows assessing the final DEGs through a machine learning step by using SVM with a test dataset.*

Description

svm_test allows assessing the final DEGs through a machine learning step by using SVM with a test dataset. An optimization of C and G hiperparameters is done at the start of the process.

Usage

```
svm_test(train, labelsTrain, test, labelsTest, vars_selected)
```

Arguments

train	The train parameter is an expression matrix or data.frame that contains the training dataset with the genes in the columns and the samples in the rows.
labelsTrain	A vector or factor that contains the training labels for each of the samples in the train object.
test	The test parameter is an expression matrix or data.frame that contains the test dataset with the genes in the columns and the samples in the rows.
labelsTest	A vector or factor that contains the test labels for each of the samples in the test object.
vars_selected	The genes selected to classify by using them. It can be the final DEGs extracted with the function limmaDEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.

Value

A list that contains four objects. The confusion matrix, the accuracy, the sensitibity and the specificity for each genes.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))

trainingMatrix <- t(DEGsMatrix)[c(1:4,6:9),]
trainingLabels <- labels[c(1:4,6:9)]
testMatrix <- t(DEGsMatrix)[c(5,10),]
testLabels <- labels[c(5,10)]

svm_test(trainingMatrix,trainingLabels,testMatrix,testLabels,rownames(DEGsMatrix)[1:10])
```

tophatAlignment *tophatAlignment allows downloading and processing the fastq samples in a CSV file by using tophat2 aligner.*

Description

This function allows downloading and processing the fastq samples in a CSV file by using tophat2 aligner. This funtion is used internally by `rawAlignment` but it can be used separatelly. Furthermore, the function can download the reference files required: FASTA Reference Genome and GTF file.

Usage

```
tophatAlignment(data, downloadRef = FALSE, downloadSamples = FALSE,
  createIndex = TRUE, BAMfiles = TRUE, SAMfiles = TRUE,
  countFiles = TRUE, referenceGenome = 38, customFA = "",
  customGTF = "")
```

Arguments

`data`	The ID of the variable which contains the samples. Our recommendation is to load this variable from a CSV file.
`downloadRef`	A logical parameter that represents if the reference files will be downloaded or not.
`downloadSamples`	A logical parameter that represents if the samples of the CSV file will be downloaded or not.
`createIndex`	A logical parameter that represents if the index of the aligner would be created or not.
`BAMfiles`	A logical parameter that represents if the you want the BAM files or not.
`SAMfiles`	A logical parameter that represents if the you want the SAM files or not.
`countFiles`	A logical parameter that represents if the you want the Count files or not.
`referenceGenome`	This parameter allows choosing the reference genome that will be used for the alignment. The options are 37,38 or custom. The two first are human genomes, but with the third option you can choose any genome stored in the computer.
`customFA`	The path to the custom FASTA file of the reference genome.
`customGTF`	The path to the custom GTF file.

Value

Nothing to return.

Examples

```
# Due to the high computational cost, we strongly recommend it to see the offical documentation and the complete

dir <- system.file("extdata", package="KnowSeq")
```

```
#Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv(paste(dir,"/GSE74251.csv",sep = ""))

## Not run: tophatAlignment(GSE74251csv,downloadRef=FALSE,downloadSamples=FALSE, createIndex = TRUE, BAMfiles
```

Index