Package 'KnowSeq'

October 24, 2025

Type Package

Title KnowSeq R/Bioc package: The Smart Transcriptomic Pipeline

Version 1.23.0

Description

KnowSeq proposes a novel methodology that comprises the most relevant steps in the Transcriptomic gene expression analysis. KnowSeq expects to serve as an integrative tool that allows to process and extract relevant biomarkers, as well as to assess them through a Machine Learning approaches. Finally, the last objective of KnowSeq is the biological knowledge extraction from the biomarkers (Gene Ontology enrichment, Pathway listing and Visualization and Evidences related to the addressed disease). Although the package allows analyzing all the data manually, the main strenght of KnowSeq is the possibilty of carrying out an automatic and intelligent HTML report that collect all the involved steps in one document. It is important to highligh that the pipeline is totally modular and flexible, hence it can be started from whichever of the different steps. KnowSeq expects to serve as a novel tool to help to the experts in the field to acquire robust knowledge and conclusions for the data and diseases to study.

License GPL (>=2)

Depends R (>= 4.0), cqn (>= 1.28.1)

Encoding UTF-8

LazyData false

RoxygenNote 7.1.1

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

Imports stringr, methods, ggplot2 (>= 3.3.0), jsonlite, kernlab, rlist, rmarkdown, reshape2, e1071, randomForest, caret, XML, praznik, R.utils, httr, sva (>= 3.30.1), edgeR (>= 3.24.3), limma (>= 3.38.3), grDevices, graphics, stats, utils, Hmisc (>= 4.4.0), gridExtra

VignetteBuilder knitr

Suggests knitr

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

git_branch devel

git_last_commit 23de926

git_last_commit_date 2025-04-15

Repository Bioconductor 3.23

Date/Publication 2025-10-24

Author Daniel Castillo-Secilla [aut, cre],
Juan Manuel Galvez [ctb],
Francisco Carrillo-Perez [ctb],
Marta Verona-Almeida [ctb],
Daniel Redondo-Sanchez [ctb],
Francisco Manuel Ortuno [ctb],
Luis Javier Herrera [ctb],
Ignacio Rojas [ctb]

Maintainer Daniel Castillo-Secilla <cased@ugr.es>

Contents

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",
  batchGroups = c()
)
```

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.
batchGroups	A numeric vector with the different known batch groups for the samples.

Value

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

Examples

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

batchGroups <- c(1,1,1,1,2,2,1,2,1,2)

expressionMatrixNoBatch <- batchEffectRemoval(expressionMatrix, labels, batchGroups = batchGroups)
expressionMatrixNoBatch <- 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 = "",
  Ensembl_ID = TRUE
)
```

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 `getGenesAnnotation`.
`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.
`Ensembl_ID`	A boolean variable which indicate if the counts matrix contains Ensembl_ID(TRUE) or genes names(FALSE).

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 = ",", extension = "")
```

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.
extension	The extension of the count file. Set to count by default.

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", extension = 'count')

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("red", "green"),
  main = "",
  ylab = "Expression",
  xlab = "Samples",
  xgrid = FALSE,
  ygrid = FALSE,
  legend = "",
  mode = "boxplot",
  heatmapResultsN = 0,
  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, classResults and heatmapResults.

heatmapResultsN	Number of genes to show when mode is equal to heatmapResults.
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)

results <- knn_trn(t(DEGsMatrix), labels, rownames(DEGsMatrix), 3)
dataPlot(results, labels = "", mode = "heatmapResults", main = "Plot to show indicators of trained model")
```

DEGsEvidences *DEGsEvidences function returns for each DEG a list of evidences that correlate it with the studied disease.*

Description

DEGsEvidences function returns for each DEG a list of evidences that correlate it with the studied disease.

Usage

```
DEGsEvidences(geneList, disease, size = 10, verbose = TRUE)
```

Arguments

geneList	A list that contains the gene symbols or gene names of the DEGs.
disease	The name of a disease in order to obtain related evidences from target validation by using the DEGs indicated in the geneList parameter.
size	The number of diseases to retrieve from targetValidation
verbose	Boolean that indicates if progress messages are printed to stdout

Value

A list which names are genes from geneList and which contains related evidences for each gene in geneList and indicated disease.

Examples

```
evidences <- DEGsEvidences(c("KRT19","BRCA1","TYMP"),'cancer')
```

`DEGsExtraction` *DEGsExtraction 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

```
DEGsExtraction(
  expressionMatrix,
  labels,
  pvalue = 0.05,
  lfc = 1,
  cov = 1,
  nmax = 1,
  multiDegsMethod = "cov",
  number = Inf,
  CV = FALSE,
  numFolds = 5
)
```

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 establishes a minimum number of pair of classes combination in which exists differential expression to consider a genes as expressed genes.

nmax	This value only works when there are more than two classes in the labels. NMAX indicates the maximum number of DEGs selected for each class pair comparison.
multiDegsMethod	Select the multiclass extraction method for the process: cov or nmax
number	The maximum number of desired genes as output of limma. As default, the function returns all the extracted DEGs with the selected parameters.
CV	A boolean value that has to be setted to TRUE if the user would to run a Cross-Validation DEGs extraction process.
numFolds	This parameter indicates the number of folds for the Cross-Validation process.

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

topTable <- DEGsInformation$Table

DEGsMatrix <- DEGsInformation$DEGsMatrix
```

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, size = 10, disease = "", getEvidences = FALSE)
```

Arguments

geneList	A list that contains the gene symbols or gene names of the DEGs.
size	The number of diseases to retrieve from targetValidation
disease	Query a specific disease instead of retrieving the whole list of related diseases.
getEvidences	Boolean. If true, for each gene, a list of found evidences for each disease will be returned.

Value

A list which contains the information about the diseases associated to each genes or to a set of genes. If getEvidences is TRUE, found evidences for each case will be returned too.

Examples

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

DEGsToPathways *The function uses the DEGs to retrieves the different pathways in which those DEGs involve any interaction.*

Description

The function uses the DEGs to retrieves the different pathways in which those DEGs involve any interaction.

Usage

```
DEGsToPathways(geneList)
```

Arguments

geneList	A list which contains the DEGs that will be used to retrieve the related pathways to them.

Value

A list with the pathways that contain relation to the DEGs within the geneList parameter.

Examples

```
DEGsToPathways(c("BRCA1","MLANA"))
```

`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. Furthermore, the ranking is returned and can be used as input of the parameter vars_selected in the machine learning functions.*

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",
  disease = "",
  maxGenes = ncol(data)
)
```

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 DEGsExtraction or a custom vector of genes.
mode	The algorithm used to calculate the genes ranking. The possibilities are three: mrmr, rf and da.
disease	The name of a disease in order to calculate the Disease Association ranking by using the DEGs indicated in the vars_selected parameter.
maxGenes	Integer that indicated the maximum number of genes to be returned.

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),mode='mrmr')
```

---

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(manifestPath, controlled = FALSE, tokenPath = "")
```

Arguments

manifestPath	Path to the samples manifest
controlled	Parameter that indicates if data to download are controlled or not
tokenPath	Path to the GDC token if data are controlled

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(
  geneList,
  geneType = "ENTREZ_GENE_ID",
  ontologies = c("BP", "CC", "MF"),
  pvalCutOff = 1
)
```

Arguments

geneList	A list that contains entrez gene id of the DEGs. Entrez gene id can be obtained using getAnnotationFromEnsembl function.
geneType	A string indicating the type of genes in geneList, it must be one of indicated in DAVIDs API documentation.
ontologies	A list that contains ontologies to be searchs. Values must be contained in the following three: BP, CC, MF.
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

```
## Not run: GOsList <- geneOntologyEnrichment(data$entrezgene_id,geneType='ENTREZ_GENE_ID',pvalCutOff=0.1)
```

getGenesAnnotation *getGenesAnnotation 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

```
getGenesAnnotation(
  values,
 attributes = c("ensembl_gene_id", "external_gene_name", "percentage_gene_gc_content",
    "entrezgene_id"),
  filter = "ensembl_gene_id",
  notHSapiens = FALSE,
  notHumandataset = "",
  referenceGenome = 38
)
```

Arguments

values	A list of genes that contains the names or IDs or "allGenome" string, which indicates that all genome will be returned.
attributes	A vector which contains the different information attributes that the Ensembl biomart admit.
filter	The attribute used as filter to return the rest of the attributes.
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")).
referenceGenome	Integer that indicates used reference genome. It must be 37 or 38.

Value

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

Examples

```
myAnnotation <- getGenesAnnotation(c("KRT19","BRCA1"), filter="external_gene_name",notHSapiens=FALSE)
```

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 = "",
  hisatParameters = "-p 8 --dta-cufflinks"
)
```

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.

hisatParameters
Parameter that allow to modify the default configuration for the Hisat2 aligner.

Value

Nothing to return.

Examples

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

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 = TR
```

---

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, bestK)
```

Arguments

train The train parameter is an expression matrix or data.frame that contains the train dataset with the genes in the columns and the samples in the rows.

labelsTrain A vector or factor that contains the train 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 DEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.

bestK Best K selected during the training phase.

Value

A list that contains six objects. The confusion matrix for each fold, the accuracy, the sensitivity, the specificity and the F1-Scores for each gene, and the predictions made.

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)]
bestK <- 3 # the one that has been selected
results_test_knn <- knn_test(trainingMatrix, trainingLabels, testMatrix, testLabels, rownames(DEGsMatrix)[1:10],
```

---

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

Description

knn_trn 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_trn(data, labels, vars_selected, numFold = 10, LOOCV = FALSE)
```

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 `DEGsExtraction` 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.
`LOOCV`	Logical parameter to choose between Loo-CV and KFold-CV.

Value

A list that contains seven objects. The confusion matrix for each fold, the accuracy, the sensitivity, the specificity and the F1-Scores for each fold and each genes, the best k found for the knn algorithm after tuning, and the predictions made.

Examples

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

knn_trn(t(DEGsMatrix)[,1:10],labels,rownames(DEGsMatrix)[1:10],3)
```

---

knowseqReport *knowseqReport creates a report for a given set of genes which their label.*

Description

knowseqReport creates a report for a given set of genes which their label. This provide an html file with all the information that can be obtained for a certain set of genes (as GO, pathway visualization, associated diseases) and their labels (machine learning process).

Usage

```
knowseqReport(
  data,
  labels,
  MLTest = FALSE,
  testData = "",
  testLabels = "",
  outdir = "knowSeq-report",
  qualityAnalysis = TRUE,
  batchEffectTreatment = TRUE,
  geneOntology = TRUE,
  getPathways = TRUE,
  getDiseases = TRUE,
  lfc = 2,
  pvalue = 0.01,
  cov = 2,
  featureSelectionMode = "nofs",
  disease = "",
  subdiseases = c(""),
  maxGenes = 150,
  clasifAlgs = c("knn", "rf", "svm"),
  metrics = c("accuracy", "specificity", "sensitivity")
)
```

Arguments

data	A matrix that contains the gene expression.
labels	A vector or factor that contains the labels for each of the samples in the data object.
MLTest	This parameter enables the classification process for a test dataset.
testData	A matrix that contains the unseen samples for the test process.
testLabels	A vector or factor that contains the labels for the unseen samples for the test process.
outdir	The output directory to store the report.
qualityAnalysis	A logical parameter that indicates if the user wants to perform the quality anaylisis or not.
batchEffectTreatment	A logical parameter that indicates if the user wants to perform the batch effect treatment or not.
geneOntology	A logical parameter that indicates if the user wants to show genes ontologies or not.
getPathways	A logical parameter that indicates if the user wants to show genes pathways or not.
getDiseases	A logical parameter that indicates if the user wants to show genes related diseases or not.
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.
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.
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.
featureSelectionMode	String that indicates which feature selection algorithm is going to be used. Possible values are: mrmr, rf or da. By default, no feature selection algorithm will be applied.
disease	String that indicates from which disease wants the user wants to know if selected genes are related to. Found evidences will be shown for each subdiseases. Default empty, this means that all related diseases, and found evidences, will be shown.
subdiseases	String that indicates the name of a particular subtype from disease, which the user to know if selected genes are related to. Found evidences will be shown. Default empty, this means that there are not subtypes of disease to look for, all found evidences for disease will be shown.
maxGenes	Integer that indicates the maximun number of genes which information will be shown and that will be used to train models.

clasifAlgs	A vector with including algorithms names that will be used in training cv.
metrics	A list with metrics that the user wants to be shown in machine learning process. Metrics could be accuracy, specificity and/or sensitivity.

Value

Nothing to return.

Examples

```
dir <- system.file("extdata", package="KnowSeq")
load(paste(dir,"/expressionExample.RData",sep = ""))
## Not run: knowseqReport(expressionMatrix,labels,'knowSeq-report',clasifAlgs=c('rf'),disease='lung-cancer',max
## Not run: knowseqReport(expressionMatrix,labels,'knowSeq-report',clasifAlgs=c('rf'),disease='lung-cancer',sub
```

---

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, samples can be aligned by using hisat2. Finally, the function can downloads the reference files required: FASTA Reference Genome and GTF file.

Usage

```
rawAlignment(
  data,
  downloadRef = FALSE,
  downloadSamples = FALSE,
  createIndex = TRUE,
  BAMfiles = TRUE,
  SAMfiles = TRUE,
  countFiles = TRUE,
  referenceGenome = 38,
  customFA = "",
  customGTF = "",
  fromGDC = FALSE,
  tokenPath = "",
  manifestPath = "",
  hisatParameters = "-p 8 --dta-cufflinks"
)
```

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.

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.

hisatParameters
: Parameter that allow to modify the default configuration for the Hisat2 aligner.

Value

Nothing to return.

Examples

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

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,downloadRef=FALSE,downloadSamples=FALSE, createIndex = TRUE, BAMfiles = TRUE
```

---

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, bestParameters)
```

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 DEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.
bestParameters	Best values for ntree and mtry parameters selected during the training phase.

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)]
bestParameters <- 30
rf_test(trainingMatrix, trainingLabels, testMatrix, testLabels,rownames(DEGsMatrix)[1:10], bestParameters = best
```

---

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

Description

rf_trn 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_trn(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 DEGsExtraction 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_trn(t(DEGsMatrix)[,1:10],labels,rownames(DEGsMatrix)[1:10],2)
```

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

Description

RNAseqQA performs the quality analysis of an expression matrix. This function generates different plots over expression data in order to detect possible outliers.

Usage

```
RNAseqQA(
  expressionMatrix,
  outdir = "SamplesQualityAnalysis",
  toPNG = TRUE,
  toPDF = TRUE,
  toRemoval = FALSE
)
```

Arguments

expressionMatrix	A matrix that contains the gene expression values.
outdir	The output directory to store the report of arrayQualityMetrics
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.
toRemoval	Boolean variable to indicate if detected outliers will be removed from original data

Value

A list containing found outliers for each realized test or corrected data if toRemoval is TRUE.

Examples

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

---

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 identifier argument. The function admits both gz and sra formats. This function is used internally by rawAlignment but it can be used separatelly.

Usage

```
sraToFastq(identifier)
```

Arguments

identifier	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("SRA1")
```

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, bestParameters)
```

Arguments

train	The train parameter is an expression matrix or data.frame that contains the train dataset with the genes in the columns and the samples in the rows.
labelsTrain	A vector or factor that contains the train 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 DEGsExtraction or a custom vector of genes. Furthermore, the ranking achieved by featureSelection function can be used as input of this parameter.
bestParameters	Best values for C and gamma parameters selected during the training phase.

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_svm_cv <- svm_trn(trainingMatrix, trainingLabels, rownames(DEGsMatrix)[1:10], 2)
bestParameters <- results_svm_cv$bestParameters
svm_test(trainingMatrix, trainingLabels, testMatrix, testLabels,rownames(DEGsMatrix)[1:10], bestParameters)
```

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

Description

svm_trn 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_trn(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 DEGsExtraction 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 five objects. The confusion matrix for each fold, the accuracy, the sensitibity and the specificity for each fold and each genes, and a vector with the best parameters found for the SVM algorithm after tuning.

Examples

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

svm_trn(t(DEGsMatrix)[,1:10], labels, rownames(DEGsMatrix)[1:10], 2)
```

Index