Package 'multiSight'

October 21, 2021

Title Multi-omics Classification, Functional Enrichment and Network Inference analysis

Version 1.1.0

Description multiSight is an R package providing an user-friendly graphical interface to analyze your omic datasets in a multi-omics manner based on Stouffer's p-value pooling and multi-block statistical methods.
For each omic dataset you furnish, multiSight provides classification models with feature selection you can use as biosignature:
(i) To forecast phenotypes (e.g. to diagnostic tasks, histological subtyping),
(ii) To design Pathways and gene ontology enrichments (Over Representation Analysis),
(iii) To build Network inference linked to PubMed querying to make assumptions easier and data-driven.

biocViews Software, RNASeq, miRNA, Network, NetworkInference, DifferentialExpression, Classification, Pathways, GeneSetEnrichment

License CeCILL + file LICENSE

Imports golem, config, R6, shiny, shinydashboard, DT, dplyr, stringr, anyLib, caret, biosigner, mixOmics, stats, DESeq2, clusterProfiler, rWikiPathways, ReactomePA, enrichplot, ppcor, metap, infotheo, igraph, networkD3, easyPubMed, utils, htmltools, rmarkdown

Encoding UTF-8

LazyData false

BugReports https://github.com/Fjeanneret/multiSight/issues

RoxygenNote 7.1.1

Suggests org.Mm.eg.db, rlang, markdown, attempt, processx, testthat, knitr, BiocStyle

VignetteBuilder knitr

Depends R (>= 4.1)

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

git_branch master

git_last_commit 00961ec

git_last_commit_date 2021-05-19

Date/Publication 2021-10-21

Author Florian Jeanneret [cre, aut] (<https://orcid.org/0000-0002-9301-4019>), Stephane Gazut [aut]

Maintainer Florian Jeanneret <florian.jeanneret@cea.fr>

R topics documented:

assessPerformance_Biosigner
MLmodels biosigner function

Description

A biosigner models assessing function. to display by confusion matrices for SVM and RF models.

Usage

```
assessPerformance_Biosigner(modelList, dataTest)
```

Arguments

modelList	Models list computed and returned inside runSVMRFmodels_Biosigner() results.
dataTest	List of new omic datasets and samples for same omics that training to test model performances. Returned by splitDatatoTrainTest().

Value

List of performances for svm and rf models (confusion matrices).

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#biosignerRes <- runSVMRFmodels_Biosigner(data.train)
data("biosignerRes", package = "multiSight")
biosignerModels <- biosignerRes$model #list of SVM/RF models for each omic.
biosignerFeats <- biosignerRes$biosignature #selected features for each omic.
perfBiosigner <- assessPerformance_Biosigner(biosignerModels, data.test)
perfBiosigner$svm$rnaRead # perf for SVM for rnaRead data block.
perfBiosigner$rf$rnaRead # perf for RF for rnaRead data block.
```

assessPerformance_Diablo *MLmodels diablo function*

Description

A diablo models assessing function.

Usage

```
assessPerformance_Diablo(splsdaModel, dataTest)
```

Arguments

splsdaModel	sPLS-DA model computed and returned inside runSPLSDAmodels_Diablo() results.
dataTest	List of new omic data sets and samples for same omics that training to test model performances. Returned by splitDatatoTrainTest().

Value

Confusion matrix for sPLS-DA model

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
perfDiablo <- assessPerformance_Diablo(diabloModels, data.test)
perfDiablo$Omic1 # sPLS-DA's perf for omic1 data block.
```

biosignerRes *Biosigner results*

Description

Biosigner results list:

model SVM and RF models

biosignature Selected features of omic data sets

Usage

```
biosignerRes
```

Format

An object of class `list` of length 2.

Source

Returned by runSVMRFmodels_Biosigner() function.

`buildFeatTable` *Models util function*

Description

To build detailed feature tables outputs for selected ones by machine learning models. Relative mean values by label class then log2FoldChange and p.adj values if DESeq2 have been computed are indicated.

Usage

```
buildFeatTable(featVec, omicBlock, Y, deTable = NULL)
```

Arguments

`featVec`	Selected features vector of one omic dataset.
`omicBlock`	Omic dataset of selected features.
`Y`	Omic sample classes.
`deTable`	(optional) Deseq2 results table for an omic dataset.

Value

Returns table of features selected by classification model and relative values.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
diabloFeatTable <- buildFeatTable(diabloFeats[[1]],
```

```
  omic2[[1]],
  omic2$Y)
diabloFeatTable
```

---

convertToEntrezid *Understand utils function*

Description

To convert features to entrezid to enrich.

Usage

```
convertToEntrezid(featList, fromDbList, organismDb)
```

Arguments

featList	Feature lists from each omic data block.
fromDbList	Database names vector. One by omic block (e.g. for 2 omics list(omic1 = "SYMBOL", omic2 = "ENSEMBL")). Returned by getDbFromInput() for app.
organismDb	Organism database to convert features.

Value

featConverted

Examples

```
if (requireNamespace("org.Mm.eg.db", quietly = TRUE))
{
    library(org.Mm.eg.db, warn.conflicts = FALSE)
    data("omic2", package = "multiSight")
    splitData <- splitDatatoTrainTest(omic2, 0.8)
    data.train <- splitData$data.train
    data.test <- splitData$data.test

    diabloRes <- runSPLSDA(data.train)
    diabloModels <- diabloRes$model #sPLS-DA model using all omics.
    diabloFeats <- diabloRes$biosignature #selected features for each omic.
    id_db <- list(omic1 = "ENSEMBL", omic2 = "ENSEMBL")
    convFeat <- convertToEntrezid(diabloFeats, id_db, "org.Mm.eg.db")


    featList <- list(Omic1 = c("ENSMUSG00000039621",
                           "ENSMUSG00000038733",
                           "ENSMUSG00000062031"),
                     Omic2 = c("ENSMUSG00000031170",
```

```
                            "ENSMUSG00000077495",
                            "ENSMUSG00000042992"))
    dbList <- list(Omic1 = "ENSEMBL",
                   Omic2 = "ENSEMBL")

    convFeat <- convertToEntrezid(featList, dbList, "org.Mm.eg.db")
}
```

`correlationNetworkInference` *Correlation network inference*

Description

Correlation network inference

Usage

```
correlationNetworkInference(concatenatedMatrix, valueThreshold)
```

Arguments

`concatenatedMatrix`	A concatenated matrix of all omic selected features. Returned by getDataSelectedFeatures().
`valueThreshold`	Correlation absolute value threshold to select relevant values.

Value

Each result for a correlation network of selected features according to threshold values.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
omicMatrices <- getDataSelectedFeatures(omic2, diabloFeats)
correlationNetworkInference(omicMatrices, 0.8)
```

deseqRes *multiSight results*

Description

DESeq2 results object obtained with several omic data sets.

Usage

```
deseqRes
```

Format

An object of class `list` of length 2.

Source

Returned by multiSight runMultiDeseqAnalysis() function

diabloRes *Diablo results*

Description

Diablo results list:

design Covariance matrix design to maximize

model sPLS-DA model

biosignature Selected features of omic data sets

Usage

```
diabloRes
```

Format

An object of class `list` of length 3.

Source

Returned by runSPLSDA() function.

enrichResList *multiSight results*

Description

enrichRes object obtained with several omic data sets.

Usage

```
enrichResList
```

Format

An object of class `list` of length 2.

Source

Returned by multiSight runMultiEnrichment() function in enrichTables$pathways$reactome$enrichObj slot

enrichWP *Util function From clusterProfiler*

Description

Downloads wikiPathways data and computes ORA analysis with provided features.

Usage

```
enrichWP(gene, organism, pAdjustMethod, minGSSize, maxGSSize)
```

Arguments

`gene`	Features to enrich.
`organism`	Value chosen by user in Home tab..
`pAdjustMethod, minGSSize, maxGSSize`	Numeric values chosen by user in Biological Insights tab.

Value

enrichResult object with wikiPAthways database used.

getDataSelectedFeatures

Select only data values from diablo features selected.

Description

Select only data values from diablo features selected.

Usage

```
getDataSelectedFeatures(matrixDataList, featureList = NULL)
```

Arguments

matrixDataList	A matrix list according to omic type with sample ID as columns and features as rows.
featureList	A features list for each omic type.

Value

A concatenated matrix of all omic selected features.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
omicMatrices <- getDataSelectedFeatures(omic2, diabloFeats)
```

mod_hypothesisGenerator_server

hypothesisGenerator Server Function

Description

hypothesisGenerator Server Function

Usage

```
mod_hypothesisGenerator_server(
  input,
  output,
  session,
  obj,
  biosignature,
  featMethod
)
```

Arguments

input, output, session	Internal parameters for shiny.
obj	R6 object to wrap all data from different analysis.
biosignature	Discriminant biological features selected.
featMethod	character string to specify feature selection method (e.g. "diablo");

Value

Network inferences and PubMed queries results.

---

mod_understand_server *understand Server Function*

Description

understand Server Function

Usage

```
mod_understand_server(input, output, session, startSignal)
```

Arguments

input, output, session	Internal parameters for shiny.
startSignal	input$start from Start button in Home tab.

Value

Biological insights from databases chosen by user by hypergeomtric tests (ORA) on DESeq2 or diablo (sPLS-DA) selected features.

mod_understand_ui *understand UI Function*

Description

A shiny Module.

Usage

```
mod_understand_ui(id)
```

Arguments

id	Internal parameter for shiny.

Value

Displays UI output for Biological Insigths tab.

mod_user_input_server *user_input Server Function*

Description

user_input Server Function

Usage

```
mod_user_input_server(input, output, session)
```

Arguments

input, output, session	Internal parameters for shiny

Value

Launches multi-omic data analysis and saves results.

mutualInformationNI *Mutual Information network inference*

Description

Mutual Information network inference

Usage

```
mutualInformationNI(concatenatedMatrix, valueThreshold)
```

Arguments

concatenatedMatrix	A concatenated matrix of all omic selected features. Returned by getDataSelectedFeatures().
valueThreshold	Mutual Information Value threshold to select relevant values

Value

Each result for a mutual information network of selected features according to threshold values.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
omicMatrices <- getDataSelectedFeatures(omic2, diabloFeats)
mutualInformationNI(omicMatrices, 0.8)
```

omic2 *Multi-omic data with 2 omics.*

Description

A dataset containing simulated multi-omic data with 30 samples.

Usage

```
omic2
```

Format

A list of 2 dataframes and 1 factor vector 30 rows:

rnaRead transcriptomic simulated data

dnaRead genomic simulated data

Y 30 samples' classes

Source

MOSim package used to simulate omic data.

`partialCorrelationNI` *Partial Correlation network inference*

Description

Partial Correlation network inference

Usage

```
partialCorrelationNI(concatenatedMatrix, valueThreshold)
```

Arguments

`concatenatedMatrix`	A concatenated matrix of all omic selected features. Returned by getDataSelectedFeatures().
`valueThreshold`	Partial Correlation Value threshold to select relevant values

Value

Each result for a partial correlation network of selected features according to threshold values.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
omicMatrices <- getDataSelectedFeatures(omic2, diabloFeats)
partialCorrelationNI(omicMatrices, 0.8)
```

pubmedInsight *Select only data values from diablo features selected.*

Description

Select only data values from diablo features selected.

Usage

```
pubmedInsight(session, input, output, query, featMethod)
```

Arguments

input, output, session	Internal parameters for shiny
query	character vector to query pubmed database
featMethod	character string to specify feature selection method (e.g. "diablo")

Value

uiOutput, a html content to embed in client side.

---

runFeatureNumberTuning *diablo util function*

Description

A feature number tuning function of splsda model for each component of each data block.

Usage

```
runFeatureNumberTuning(dataTrainList, YClassVector, ncomp, design)
```

Arguments

dataTrainList	List of data block training part.
YClassVector	List of your sample classes vector provided
ncomp	Component number in splsda model obtained from runComponentNumberTest function launched by runSPLSDAmodels_Diablo function.
design	Covariance matrix design obtained from

Value

Grid of number of features to select by sPLS-DA model for each component and omic.

runMultiDeseqAnalysis *Enrichment function*

Description

Runs DESEQ2 analysis on all omic data blocks.

Usage

```
runMultiDeseqAnalysis(omicDataList, padjUser)
```

Arguments

omicDataList	List of omic data blocks with Y class vector.
padjUser	Threshold for p-value adjusted to select features according to padj values in DESeq2 table.

Value

List of DESeq2's Differential Expression tables for all omic datasets (e.g. BaseMean, Log2FoldChange, padj columns).

Examples

```
data("omic2", package = "multiSight")
#deseqRes <- runMultiDeseqAnalysis(omic2, 0.05)
data("deseqRes", package = "multiSight")
print(deseqRes$DEtable$rnaRead)
```

runMultiEnrichment *Main understand module function*

Description

Launches functional enrichment of features provided for every databases furnished. Run by utils function runMultiOmicEnrichment

Usage

```
runMultiEnrichment(
  omicSignature,
  databasesChosen,
  organismDb,
  pvAdjust = "BH",
  minGSSize = 5,
  maxGSSize = 800,
  pvStouffer = 0.1
)
```

Arguments

omicSignature	Feature lists from each omic data block.
databasesChosen	Which biological database c(reactome, kegg, wikiPathways, MF, CC, BP)
organismDb	Organism provided by user in Home tab.
pvAdjust	pv adjust method (e.g "BH" for Benjamini-Hochberg)
minGSSize, maxGSSize, pvStouffer	Numeric values chosen by user in ui.

Value

Wraps in obj enrichment results for all databases and all omics.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

diabloRes <- runSPLSDA(data.train)
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
id_db <- list(omic1 = "ENSEMBL", omic2 = "ENSEMBL")

if (requireNamespace("org.Mm.eg.db", quietly = TRUE))
{
    library(org.Mm.eg.db, warn.conflicts = FALSE) #Organism's database
    featList <- list(Omic1 = c("ENSMUSG00000039621",
                           "ENSMUSG00000038733",
                           "ENSMUSG00000062031"),
                     Omic2 = c("ENSMUSG00000031170",
                           "ENSMUSG00000077495",
                           "ENSMUSG00000042992"))
    dbList <- list(Omic1 = "ENSEMBL",
                Omic2 = "ENSEMBL")
    convFeat <- convertToEntrezid(featList, dbList, "org.Mm.eg.db")
```

```
    ## To enrich features
    database <- c("reactome", "MF")
    #runMultiEnrichment_result <- runMultiEnrichment(databasesChosen = database,
    #                                omicSignature = convFeat,
    #                                organismDb = "org.Mm.eg.db")
}
```

`runMultiEnrichment_result` *multiSight results*

Description

multiOmicEnrichment results object obtained with several omic data sets.

Usage

```
runMultiEnrichment_result
```

Format

An object of class `list` of length 2.

Source

Returned by multiSight runMultiEnrichment() function

`runSPLSDA` *MLmodels diablo function*

Description

A model and selection features function from mixOmics.

Usage

```
runSPLSDA(dataTrain)
```

Arguments

`dataTrain`	Data train set to build classification models. Returned by splitDatatoTrainTest().

Value

Returns sPLS-DA model, performances, features selected.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#diabloRes <- runSPLSDA(data.train)
data("diabloRes", package = "multiSight")
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
```

---

runSVMRFmodels_Biosigner

MLmodels biosigner function

Description

A model and selection features function from biosigner.

Usage

```
runSVMRFmodels_Biosigner(dataTrain)
```

Arguments

dataTrain	Data train set to build classification models. Returned by splitDatatoTrainTest(). List of Omic blocks and Y class vector.

Value

Models and features selected for each omic block in dataTrain.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

#biosignerRes <- runSVMRFmodels_Biosigner(data.train)
data("biosignerRes", package = "multiSight")
biosignerModels <- biosignerRes$model #list of SVM/RF models for each omic.
biosignerFeats <- biosignerRes$biosignature #selected features for each omic.
```

run_app *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(...)
```

Arguments

...	A series of options to be used inside the app.

Value

Launches RShiny app

splitDatatoTrainTest *MLmodels util function*

Description

To split list of omic data in data train and data test subsets.

Usage

```
splitDatatoTrainTest(MultiOmicData, freq = 0.8)
```

Arguments

MultiOmicData	list of your data blocks
freq	Split proportion of train samples

Value

Return two data sets: first to train model and second to assess it.

Examples

```
data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test
```

stoufferTable *Stouffer's p-value computing with all pvalues from all enrichment tables provided and all gene sets to build enrichment maps.*

Description

Stouffer's p-value computing with all pvalues from all enrichment tables provided and all gene sets to build enrichment maps.

Usage

```
stoufferTable(enrichmentResult)
```

Arguments

enrichmentResult
: enrichResults@result list.

Value

enrichment table results merged with Stouffer's p-value non-weighted and weighted.

Examples

```
data(enrichResList, package = "multiSight")
enrichResList # list of enrichRes objects (e.g. enrichKEGG() results)
multiOmicRes <- stoufferTable(enrichResList)
multiOmicRes$table # table with stouffer's values
multiOmicRes$moEnrichRes # enrichRes object for clusterProfiler plots


data("omic2", package = "multiSight")
splitData <- splitDatatoTrainTest(omic2, 0.8)
data.train <- splitData$data.train
data.test <- splitData$data.test

diabloRes <- runSPLSDA(data.train)
diabloModels <- diabloRes$model #sPLS-DA model using all omics.
diabloFeats <- diabloRes$biosignature #selected features for each omic.
id_db <- list(omic1 = "ENSEMBL", omic2 = "ENSEMBL")
if (requireNamespace("org.Mm.eg.db", quietly = TRUE))
{
    library(org.Mm.eg.db, warn.conflicts = FALSE)#'
    convFeat <- convertToEntrezid(diabloFeats, id_db, "org.Mm.eg.db")
    database <- c("reactome", "MF")
    #enrichTables <- runMultiEnrichment(databasesChosen = database,
    #  omicSignature = convFeat,
    #  organismDb = "org.Mm.eg.db")
    # enrichmentTables <- enrichTables$pathways$reactome$enrichObj
    #enrichResList # list of enrichRes objects (e.g. enrichKEGG() results)
```

```
    data(enrichResList, package = "multiSight")
    multiOmicTable <- stoufferTable(enrichResList)
}
```

Index