How to Install

Before library installation install required Bioconductor and CRAN packages through this code:

```
bioconductor_packages=c('edgeR','RUVSeq','DESeq2','limma','rhdf5','artMS')

#For R version 3.5> use BiocManager to install required bioconductor packages:
if (length(setdiff(bioconductor_packages, rownames(installed.packages()))) > 0) {
  if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
  BiocManager::install(setdiff(bioconductor_packages, rownames(installed.packages())))
}

#For R version <3.5 use the BiocInstaller to install required bioconductor packages:
source("https://bioconductor.org/biocLite.R")
BiocInstaller::biocLite(bioconductor_packages)

packages=c('magrittr','dplyr','ggplot2','doParallel','foreach','lme4','Rfast','httr')
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}
```

After this you can install package from sourced tar.gz file:

```
install.packages("octad_1.0.tar.gz", repos = NULL, type="source")
```

It takes a few minutes to install the package and verify files. Afterward, the pipeline will be ready to run.

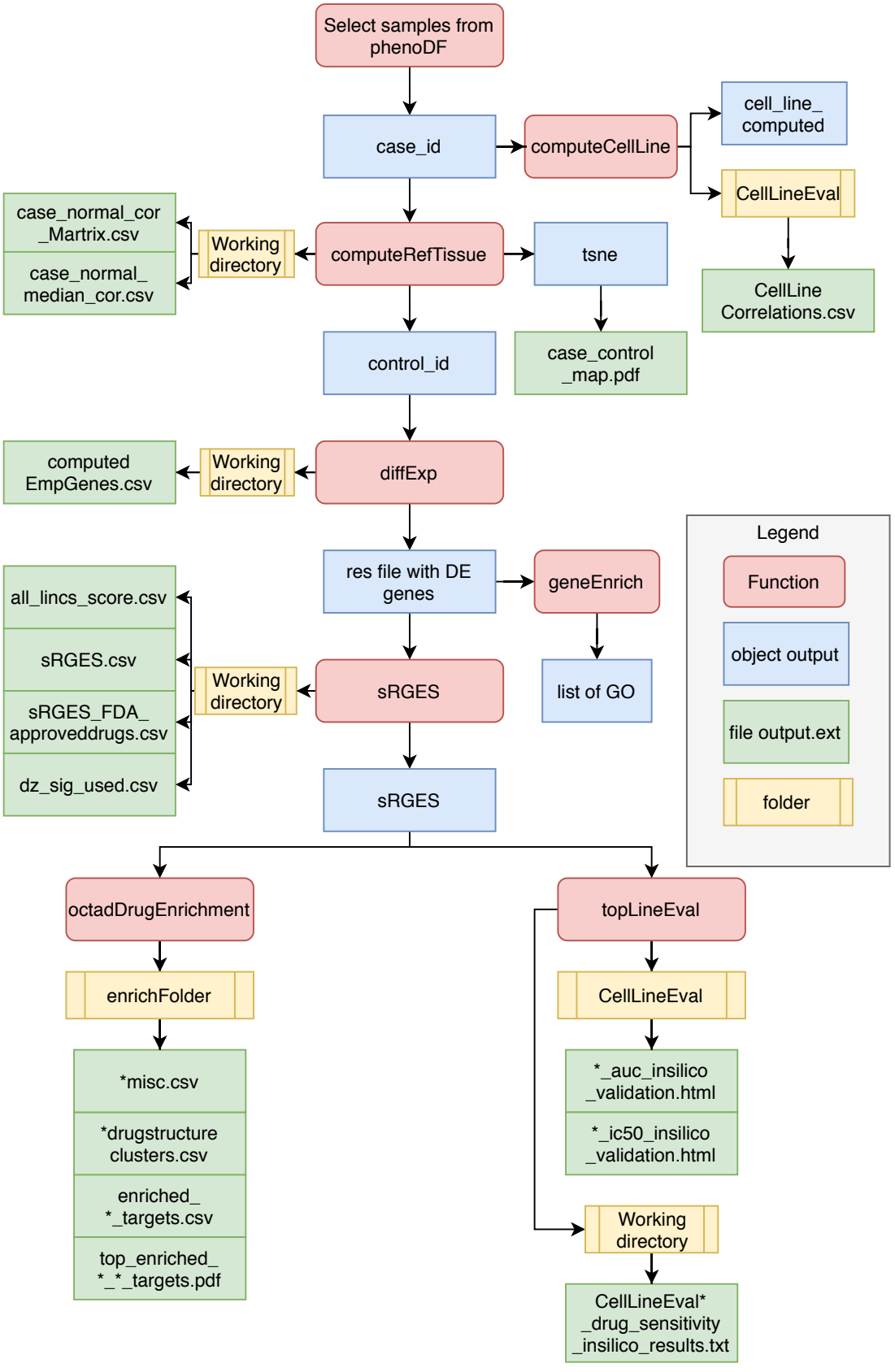
Overview

Several typical usages of the pipeline are sourced in the octad_examples.R, where users can run drug prediction for hepatocellular carcinoma, PI3KCA mutated breast invasive carcinoma, lung adenocarcinoma with MYC amplification and comparison of the metastatic vs primary breast invasive carcinoma.

To obtatin whole results for DE, downloading of the additional OCTAD database octad.counts.and.tpm.h5 from the AWS link is required:

The dataset and code can be also accessed by the link: https://www.synapse.org/#!Synapse:syn22101254/files/ https://chenlab-data-public.s3-us-west-2.amazonaws.com/octad/octad.counts.and.tpm.h5

octad.counts.and.tpm.h5 contains expression of all transcripts, while octad.LINCS.h5, used as a default in this package, only contains expression of 978 genes profiled in the LINCS database.


Select samples from phenoDF
case_id
computeCellLine
cell_line_ computed
CellLineEval
CellLine Correlations.csv
case_normal_cor _Martrix.csv
case_normal_ median_cor.csv
Working directory
computeRefTissue
tsne
case_control _map.pdf
control_id
computed EmpGenes.csv
Working directory
diffExp
res file with DE genes
geneEnrich
list of GO
all_lincs_score.csv
sRGES.csv
sRGES_FDA_ approveddrugs.csv
dz_sig_used.csv
Working directory
sRGES
sRGES
Legend
Function
object output
file output.ext
folder
octadDrugEnrichment
enrichFolder
*misc.csv
*drugstructure clusters.csv
enriched_ *_targets.csv
top_enriched_ *_*_targets.pdf
topLineEval
CellLineEval
*_auc_insilico _validation.html
*_ic50_insilico _validation.html
Working directory
CellLineEval* _drug_sensitivity _insilico_results.txt

Package 'octad'

June 25, 2020

Title Open Cancer TherApeutic Discovery (OCTAD).

Version 1.0.6.25.2020

Description Open Cancer TherApeutic Discovery (OCTAD) package implies sRGES approach for the drug discovery. The essential idea is to identify drugs that reverse the gene expression signature of a disease by tamping down over-expressed genes and stimulating weakly expressed ones.

License `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.0

Depends R (>= 2.10),
magrittr,
dplyr,
ggplot2,
edgeR,
RUVSeq,
DESeq2,
limma,
rhdf5,
doParallel,
foreach,
Rfast

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`computeCellLine` *Compute Correlation between cell lines and vector of case ids.*

Description

Select top CCLE cell lines sharing similar expression profiles with input case samples. Input case sample ids and output correlation scores for every cell line and/or output file. The results could be used for in-silico validation of predictions or used to weight cell lines in RGES computation. `CellLineCorrelations.csv`, correlation between CCLE cell lines and input disease samples.

Usage

```
computeCellLine(case_id,returnDF=T,source='octad.small')
```

Arguments

`case_id`	vector of ids from octad database. Ids can be obtained from `phenoDF`.
`returnDF`	if `TRUE`, file CellLineCorrelations.csv with results are produced in working directory.
`LINCS_overlaps`	vector of cell line ids from octad database. Ids can be obtained from `phenoDF`.
`source`	the file for the octad expression matrix. By default, set to `octad.small` to use only 978 landmark genes profiled in LINCS database. Use `octad.whole` option to compute DE on the whole transcriptome `octad.counts.and.tpm.h5` file. The file should be present in the working directory or the whole path should be included. If `source` is set to `'side'`, the expSet matrix is estimated.
`expSet`	input expression matrix. By default set to `NULL` since the expSet is created based on cases, controls and source file.
`file`	if `expSet='octad.whole'`, source path to `expSet='octad.counts.and.tpm.h5'` file is required if it is not in working directory. By default function seeks for the .h5 file in the working directory.

See Also

`runsRGES`

Examples

```
cell_line_computed=computeCellLine(case_id=case_id,returnDF=T)
computeCellLine(case_id,returnDF=T,source='octad.whole',file='octad.counts.and.tpm.h5')
```

computeRefTissue *Compute correlating reference control samples.*

Description

Compute reference control samples from OCTAD database using precomputed EncoderDF models. Return a vector of an appropriate set of control samples. Besides, if output=TRUE, two files are created in the working directory: case_normal_corMatrix.csv which contains pairwise correlation between case samples vs control samples, and case_normal_median_cor.csv, which contains median correlation values with case samples for returned control samples.

Usage

```
computeRefTissue(case_id)
computeRefTissue(case_id,outputFolder='',output=T,
adjacent=T, expSet = "octad",control_size = 50)
```

Arguments

case_id	vector of cases used to compute references.
source	by default set octad to use autoencoder results for computation. Any other input like 'side' is expSet defined by users.
adjacent	by default set to FALSE. If TRUE, only tissue with sample.type 'adjacent' from phenoDF would be used instead of 'normal'.
expSet	input for expression matrix. By default NULL, since autoencoder results are used.
n_varGenes	number of genes used to select control cases.
method	one of two options is avaliable. random will take a random number of samples from control subset and varGenes (default) will select control samples based on distance between cases and selected samples.
control_size	number of control samples to be selected.
outputFolder	path to output folder. By default, the function produces result files in working directory.
cor_cutoff	cut-off for correlation values, by default cor_cutoff='0'.
output	if TRUE, two output files are produced.

See Also

diffExp.

Examples

```
#select data
HCC_primary=subset(phenoDF,cancer=='Liver Hepatocellular Carcinoma'&
sample.type == 'primary'&data.source == 'TCGA')
#select cases
case_id=HCC_primary$sample.id
#computing reference tissue, by default using small autoEncoder, but can use custom expression set,
#by default output=T and outputFolder option is empty,
```

```
#which creates control corMatrix.csv to working directory
#control_id=computeRefTissue(case_id,outputFolder='',output=T,
expSet = "octad",control_size = 50)
#increase number of samples
control_id=computeRefTissue(case_id, control_size = 100)
#the use adjacent normal tissue samples as control_id allow you to avoid running this function
control_id =subset(phenoDF,cancer=='Liver Hepatocellular Carcinoma'&
          sample.type == 'adjacent' & data.source == 'TCGA')
#by default the autoencoder is used. User can use expression data to compute reference tissue
expression_log2=loadOctadCounts(sample_vector=phenoDF$sample.id,
       type='tpm',file='octad.counts.and.tpm.h5')
control_id=computeRefTissue(case_id, control_size = 50,expSet=expression_log2,source='side')
```

`diffExp` *Compute differential expression*

Description

Compute differential expression for case vs control samples. Will produce the file `computedEmpGenes.csv` listing empiricaly differentially expressed genes used for RNA-Seq normalization.

Usage

```
diffExp(case_id,control_id,source='octad.whole',output=T,normalize_samples=T,
        file='octad.counts.and.tpm.h5')
diffExp(case_id,control_id,source='octad.small',output=T,normalize_samples=T)
```

Arguments

`case_id`	vector of cases used for differential expression.
`control_id`	vector of controls used for differential expression.
`source`	the file for the octad expression matrix. By default, set to `octad.small` to use only 978 landmark genes profiled in LINCS database. Use `octad.whole` option to compute DE on the whole transcriptome `octad.counts.and.tpm.h5` file. The file should be present in the working directory or the whole path should be included. If `source` is set to `'side'`, the expSet matrix is estimated.
`expSet`	input expression matrix. By default set to `NULL` since the expSet is created based on cases, controls and source file.
`file`	if `expSet='octad.whole'`, source path to `expSet='octad.counts.and.tpm.h5'` file is required if it is not in working directory. By default function seeks for the .h5 file in the working directory.
`normalize_samples`	if TRUE, RUVSeq normalization is applied to either EdgeR or DESeq. No normalization needed for limma+voom.
`k`	eiter k=1 (by default), k=2 or k=3, number of factors used in model matrix construction in RUVSeq normalization if `normalize_samples=T`.
`n_topGenes`	number of empiricaly differentially expressed genes estimated for RUVSeq normalization. Default is 5000.
`DE_method`	edgeR, DESeq2 or limma DE analysis.

parallel_cores	number of cores to be used for parallel computing in DESeq2.
output	if TRUE, output files is produced.
outputFolder	path to output folder. By default, the function produces result files in working directory.
annotate	if TRUE, annotation by ENSEMBL gene is performed. If TRUE, make sure row.names of the custom input contain ensembl gene ids.

See Also

computeRefTissue, runsRGES, geneEnrich.

Examples

```
#compute differential expression,
#DE_method='edgeR',
#normalize_samples=T,
#ready source='octad.small' or 'octad.whole' using data for LINCS genes or whole octad dataset,
#IT DOES MATTER!!!, or you can load your own expression data.
#by default output=T and outputFolder is empty sending control corMatrix.csv to working directory.
#and you need to source full path to .h5 file containing count expression data.
#By default files being sourced with package and their names are
#octad.LINCS.h5 for 978 expression signatures and octad.counts.and.tpm.h5
ref_deseq = diffExp(case_id, control_id, DE_method = "edgeR", source='octad.whole',
   file='octad.counts.and.tpm.h5')
#to use the small subset. It is fast but has less coverage of transcriptome and
#lower performance due to normalization and multiple comparison issues.
 res=diffExp(case_id,control_id,expSet='octad.small',output=T)
 ref_edgeR = diffExp(case_id, control_id, DE_method = "edgeR")
 ref_limma = diffExp(case_id, control_id, DE_method = "limma")
 # By default the normalization is performed, but it can be manually turned off:
ref_deseq = diffExp(case_id, control_id, DE_method = "DESeq2", normalize_samples = F, n_topGenes = 500)
 #And filter the result by log2fold-change and adjusted p.value.
 res = ref_deseq[abs(ref_deseq$log2FoldChange) > 2 & ref_deseq$padj < 0.001, ]
 ref_edgeR = diffExp(case_id, control_id, DE_method = "edgeR", normalize_samples = F)
 ref_limma = diffExp(case_id, control_id, DE_method = "limma", normalize_samples = F)
```

---

geneEnrich *Perform functional enrichment on a set of genes.*

Description

Perform functional enrichment analysis for disease signature genes. This function interacts with Enrichr's REST API for enrichment analysis. Databases are specified with underscores for spaces (e.g. "WikiPathways_2016", "KEGG_2019_Human", "GO_Biological_Process_2018"). Other databases are listed on the website (https://amp.pharm.mssm.edu/Enrichr/#stats). A list of data.frame is produced. Every data.frame contain enriched terms for a specific selected database.

Usage

```
GO <- geneEnrich(gene_list,
                     database_list=c( "KEGG_2019_Human",
                     "GO_Biological_Process_2017",
                     "GO_Cellular_Component_2017"),output=F)
```

Arguments

gene_list	a vector of HGNC gene symbols.
database_list	a vector of EnrichR-supported databases.
output	if TRUE dataframe with names of selected databases with GO will be produced.

See Also

diffExp

Examples

```
genes=c('PHF14', 'RBM3', 'MSL1', 'PHF21A', 'ARL10', 'INSR', 'JADE2', 'P2RX7')
db=c('KEGG_2019_Human','KEGG_2015')

enrich=geneEnrich(gene_list=genes,database_list=db)

gene_e = geneEnrich(res$Symbol[res$log2FoldChange > 2],
  database_list = c("KEGG_2019_Human", "GO_Biological_Process_2017"))
dim(gene_e$KEGG_2019_Human)
dim(gene_e$GO_Biological_Process_2017)
```

---

loadOctadCounts *Load octad expression data*

Description

Create TPM or count expression matrix for the selected samples from OCTAD.

Usage

```
loadOctadCounts(c(control_id,case_id),type='tpm',file='octad.counts.and.tpm.h5')
loadOctadCounts(c(control_id,case_id),type='counts',file='octad.counts.and.tpm.h5')
```

Arguments

sample_vector	vector of samples to be selected. Use phenoDF object for sample id selection.
type	either tpm (default) or counts to be returned.
file	full path to octad.counts.and.tpm.h5 file.

See Also

diffExp.

Examples

```
#load expression data for raw counts or tpm values.
expression_tmp=loadOctadCounts(c(control_id,case_id),type='tpm',file='octad.counts.and.tpm.h5')
expression_log2=loadOctadCounts(c(control_id,case_id),type='counts',file='octad.counts.and.tpm.h5')
###
```

octadDrugEnrichment *Compute Drug enrichment*

Description

Perform enrichment analysis of drug hits based on chemical structures, drug-targets, and pharmacological classifications. An enrichment score calculated using ssGSEA and a p-value computed through a permutation test are provided. Following files are created: `enriched_*_targets.csv` and `top_enriched_*_*_targets.pdf`. In the case of chemical structural analysis, additional files are created: `*drugstructureClusters.csv` and `*misc.csv`. The results provide useful information for following candidate selection and experimental design. For example, if two structurally similar drugs are both predicted as top hits, the chance of each drug as a true positive is high.

Usage

```
octadDrugEnrichment(sRGES, target_type = 'chembl_targets')
```

Arguments

`sRGES`	sRGES data frame produced by `runsRGES`.
`target_type`	one or several of `'chembl_targets','mesh','ChemCluster'` databases selected. By deafult only 'chembl_targets' will be used.

See Also

`runsRGES`

Examples

```
octadDrugEnrichment(sRGES = sRGES, target_type = c('chembl_targets','mesh',
                    'ChemCluster'))
```

runsRGES *Compute sRGES*

Description

Compute sRGES, a score indicating the reveral potency of each drug. It first computes RGES (Reverse Gene Expression Score) for individual instances and then summarizes RGES of invididual drugs (one drug may have multiple instances under different treatment conditions). The function returns RGES data.frame containing scores and p.values for every instance. The function also returns `data.frame` with drug id in `pert_iname` collumn, `n` contains the number of instances for this drug, `mean`, `median` and `sd` of `sRGES` RGES sores. Besides, a number of additional files in the sourced directory: `dz_sig_used.csv` contains genes in the disease signature used for computing reverse gene expression scores, `sRGES.csv` contains the same data as returned data.frame and `all__lincs_score.csv` includes information of RGES.

Usage

```
runsRGES(dz_signature,max_gene_size=50,
         permutations=10000)
```

Arguments

dz_signature	disease signature. Make sure input data frame has a gene Symbol column, otherwise an error is produced. It must be an UPPERCASE gene symbol.
choose_fda_drugs	if TRUE, only FDA approved drugs are used.
max_gene_size	maximum number of disease genes used for drug prediction. By default 50 for each side (up/down).
cells	cell ids in lincs_sig_info file used for prediction. By default, all cell lines are used.
outputFolder	folder path to store drug results, by default write results to working directory.
weight_cell_line	by default NULL, if !NULL, an output object from computeCellLine is estimated (see example).
permutations	number of permutations, by default 10000.

See Also

diffExp,octadDrugEnrichment,computeCellLine,topLineEval

Examples

```
sRGES=runsRGES(res,max_gene_size=50,
               permutations=10000)
#with weighted cell lines
lincs_cell_line_weight =computeCellLine(case_id=case_id,returnDF=T, LINCS_overlaps = T)
sRGES=runsRGES(res,max_gene_size=50,
               permutations=100, weight_cell_line = lincs_cell_line_weight)
```

---

topLineEval *Evaluate cell lines*

Description

Evaluate predictions using pharmacogenomics data. Given a cell line, the function computes the correlation between sRGES and drug sensitivity data taken from CTRP. A higher correlation means a better prediction. The cell line could be computed from computeCellLine. It produces CellLineEval*_drug_sensitivity_insilico_results.txt and two .html documents: *_auc_insilico_validation.html (correlation between drug AUC and sRGES in a related cell line; *_ic50_insilico_validation.html (correlation between drug IC50 and sGRES in a related cell line.

Usage

```
topLineEval(topline,mysRGES)
```

Arguments

topline	list of cell lines to be used for prediction.
mysRGES	sRGES data.frame produced by runsRGES.

See Also

runsRGES

Examples

```
topLineEval(topline = c('HEPG2'),mysRGES = sRGES)
```

Index

Function	Message	Possible cause	Solution
computeRefTissue	cannot open file 'as//.../case_normal_corMatrix.csv': No such file or directory	outputFolder	Name of the output folders is incorrect. Replace it with valid name.
	Error in apply(expSet_normal, 1, stats::IQR) : dim(X) must have a positive length	source	Whether source option was modified, no additional matrix was used. Make sure the expSet option is set to custom expression matrix object name
	Error in expSet[, normal_id] : incorrect number of dimensions	expSet	Custom matrix input has not enough samples for computation or missing case samples in expression matrix. Make sure after filter out case samples there are still samples.
diffExp	Expression data not sourced, please, modify expSet option	expSet	User decided to use custom file input but not sourced an object. Make sure an expSet option is defined with object name containing expression data
	Please, source case ids and control ids vector	case_id or control_id	Either case or source vector is not sourced
	empty output	annotate	If TRUE, annotation by ENSEMBL gene would be performed. If TRUE, make sure row.names of the custom input contain ensembl gene ids.
	Error in h5checktypeOrOpenLoc(file, readonly = TRUE, native = native) : Error in h5checktypeOrOpenLoc(). Cannot open file. File 'octad.counts.and.tpm.h5' does not exist.	source	if source='octad.whole', function estimates octad.counts.and.tpm.h5 to be stored in working directory or full path sourced through file option

Function	Message	Possible cause	Solution
runsRGES	Disease signature input not found	dz_signature	Source disease signature with columns Symbol and log2FoldChange
	Either Symbol or log2FoldChange collumn in Disease signature is missing	dz_signature	Source disease signature with columns Symbol and log2FoldChange
	Warning message: In dir.create(outputFolder) : cannot create dir '', reason 'Invalid argument'	OutputFolder	Wrong name of output folder
computeCellLine	Case ids vector input not found	case_id	Case vector is not sourced
octadDrugEnrichment	Error in file(file, ifelse(append, "a", "w")) : cannot open the connection	enrichFolder	Wrong name of output folder
	sRGES input not found	sRGES	sRGES is not sourced
	Error in .local(expr, gset.idx.list, ...) : No identifiers in the gene sets could be matched to the identifiers in the expression data.	sRGES	Make sure column pert_iname column not empty
	Either sRGES or pert_iname collumn in Disease signature is missing	sRGES	Make sure sRGES tibble input contains both pert_iname and sRGES columns
topLineEval	Error in `[.data.frame`(x, r, vars, drop = drop) : undefined columns selected	topline	Make sure your cell lines vector is valid. You can compare output with computeCellLine output