Package 'octad'

March 17, 2022

Title Open Cancer TherApeutic Discovery (OCTAD)

Version 0.99.41

Description OCTAD provides a platform for virtually screening compounds targeting precise cancer patient groups. The essential idea is to identify drugs that reverse the gene expression signature of disease by tamping down over-expressed genes and stimulating weakly expressed ones. The package offers deep-learning based reference tissue selection, disease gene expression signature creation, pathway enrichment analysis, drug reversal potency scoring, cancer cell line selection, drug enrichment analysis and in silico hit validation. It currently covers ~20,000 patient tissue samples covering 50 cancer types, and expression profiles for ~12,000 distinct compounds.

License Artistic-2.0

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Depends R (>= 4.1.0),
magrittr,
dplyr,
ggplot2,
edgeR,
RUVSeq,
DESeq2,
limma,
rhdf5,
doParallel,
foreach,
Rfast,
octad.db,
stats,
httr,
ExperimentHub

Imports EDASeq,
GSVA,
data.table,
htmlwidgets,
plotly,
reshape2

Suggests knitr,
rmarkdown

VignetteBuilder knitr

biocViews Classification, GeneExpression, Pharmacogenetics, Pharmacogenomics, Software, GeneSetEnrichment

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 =case_id,expSet=NULL,LINCS_overlaps = TRUE,
source='octad.small',file=NULL,returnDF = FALSE)
```

Arguments

`case_id`	vector of ids from octad database. Ids can be obtained from `phenoDF`.
`returnDF`	by default `FALSE`, if `TRUE`, file CellLineCorrelations.csv with results are produced in working directory.
`LINCS_overlaps`	vector of cell line ids from octad database. If `TRUE`, overlap with LINCS cells database wll be performed
`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.

Value

topline	data.frame with row.names as cell line names and column medcor containing values for correlation between set of samples from case_id and cell lines.

See Also

runsRGES

Examples

```
phenoDF=(ExperimentHub()[["EH7274"]])
HCC_primary=subset(phenoDF,cancer=='liver hepatocellular carcinoma'&
sample.type == 'primary') #select data
case_id=HCC_primary$sample.id #select cases
cell_line_computed=computeCellLine(case_id=case_id,returnDF=FALSE,source='octad.small')
```

---

computeRefTissue *Compute correlating reference control samples.*

Description

Compute reference control samples from OCTAD database using precomputed EncoderDF models.

Usage

```
computeRefTissue(case_id=NULL,adjacent=FALSE,source='octad',n_varGenes = 500,
method='varGenes',expSet=NULL,control_size = length(case_id),
outputFolder='',cor_cutoff='0',output=TRUE)
```

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.

Value

Return

control_id	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
: contains pairwise correlation between case samples vs control samples.

case_normal_median_cor.csv
: contains median correlation values with case samples for returned control samples.

See Also

diffExp.

Examples

```
#select data
phenoDF=(ExperimentHub()[["EH7274"]])
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=TRUE and outputFolder option is empty,
#which creates control corMatrix.csv to working directory
control_id=computeRefTissue(case_id,outputFolder='',output=TRUE,
expSet = "octad",control_size = 50)
```

---

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.small',
file='octad.counts.and.tpm.h5',normalize_samples=TRUE,k=1,
expSet=NULL,n_topGenes=500,DE_method='edgeR',
parallel_cores = 2,output=TRUE,outputFolder='', annotate=TRUE)
```

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=TRUE.
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.

Value

res	data.frame with list of differentially expressed genes.
computedEmpGenes.csv	data.frame listing empiricaly differentially expressed genes used for RNA-Seq normalization.

See Also

computeRefTissue, runsRGES, geneEnrich.

Examples

```
phenoDF=(ExperimentHub()[["EH7274"]])
HCC_primary=subset(phenoDF,cancer=='liver hepatocellular carcinoma'&
sample.type == 'primary') #select data
case_id=HCC_primary$sample.id #select cases
HCC_adjacent=subset(phenoDF,cancer=='liver hepatocellular carcinoma'&
sample.type == 'adjacent'&data.source == 'TCGA') #select data
control_id=HCC_adjacent$sample.id #select cases
res=diffExp(case_id,control_id,source='octad.small',output=FALSE)
```

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

```
geneEnrich(gene_list=NULL,
                       database_list=NULL,output=FALSE)
```

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.

Value

GO_enriched	list with result of GO analysis. Every leaf is a data.frame containing GO enrichment for specific database.

See Also

diffExp

Examples

```
#gene list
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)
#output
gene_e = geneEnrich(genes,  database_list = db)
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(sample_vector='',type='tpm',file='')
```

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.

Value

exprData	matrix with either log2 corrected counts or tmp matrix for selected samples.

See Also

diffExp.

Examples

```
phenoDF=(ExperimentHub()[["EH7274"]])
#load expression data for raw counts or tpm values.
HCC_primary=subset(phenoDF,cancer=='liver hepatocellular carcinoma'&
sample.type == 'primary') #select data
#case_id=HCC_primary$sample.id #select cases
#expression_tmp=loadOctadCounts(case_id,type='tpm',file='octad.counts.and.tpm.h5')
#expression_log2=loadOctadCounts(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.

Usage

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

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.
enrichFolder	folder to store output.

Value

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.

exprData	matrix with either log2 corrected counts or tmp matrix for selected samples.

See Also

runsRGES

Examples

```
#load example sRGES computed by runsRGES() function for HCC vs liver adjacent tissues on octad.small dataset
sRGES=ExperimentHub()[["EH7279"]]
#run drug enrichment
octadDrugEnrichment(sRGES = sRGES, target_type = c('chembl_targets'))
```

---

res_example — *Differential expression example for HCC vs adjacent liver tissue computed in diffExp() function*

Description

Differential expression example for HCC vs adjacent liver tissue computed in diffExp() function

Usage

```
res_example
```

Format

A data.frame with 963 rows and 18 variables:

identifier Ensg ID

log2FoldChange Log2 fold-change

logCPM log CPM value

LR LR value

pvalue p.value

padj FDR

tax_id taxon id

GeneID Gene id

LocusTag Locus tag

chromosome Chromosome

map_location Chromosome location

description Full gene name

type type of gene

Symbol_autho HGNC symbol

other Gene function

`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).

Usage

```
runsRGES(dz_signature=NULL,choose_fda_drugs = FALSE,max_gene_size=500,
cells=NULL,outputFolder=NULL,weight_cell_line=NULL,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.

Value

The function returns RGES data.frame
containing scores and p.values for every instance. data.frame contains 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.

all__lincs_score.csv
includes information of RGES.

See Also

diffExp,octadDrugEnrichment,computeCellLine,topLineEval

Examples

```
#load differential expression example for HCC
#vs adjacent liver tissue computed in diffExp() function
res=ExperimentHub()[["EH7278"]]
res=subset(res,abs(log2FoldChange)>1&padj<0.001)
#run sRGES computation
runsRGES(dz_signature=res,choose_fda_drugs = FALSE,max_gene_size=100,permutations=1000)
```

---

sRGES_example *Data of computed example sRGEs for HCC vs liver adjacent tissues on octad.small dataset*

Description

Data of computed example sRGEs for HCC vs liver adjacent tissues on octad.small dataset

Usage

```
sRGES_example
```

Format

A tibble with 12,442 rows and 6 variables:

pert_iname dbl Year price was recorded

mean mean sRGES for obtained drug if n>1

n times this drug was obtained

median median sRGES for drug if n>1

sd standart deviation for obtained drug if n>1

sRGES sRGES score of the drug

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.

Usage

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

Arguments

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

Value

The function produces 3 feils in the output directory:

`CellLineEval*_drug_sensitivity_insilico_results.txt`
with drug sensitivity information.

`*_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.

See Also

runsRGES

Examples

```
#load example sRGES computed by runsRGES() function for HCC
#vs liver adjacent tissues on octad.small dataset
sRGES=ExperimentHub()[["EH7279"]]
#Pick up cell lines
topLineEval(topline = c('HEPG2'),mysRGES = sRGES)
```

Index