Package ‘cTRAP’

May 3, 2024

Title Identification of candidate causal perturbations from differential gene expression data

Version 1.22.0

Description Compare differential gene expression results with those from known cellular perturbations (such as gene knock-down, overexpression or small molecules) derived from the Connectivity Map. Such analyses allow not only to infer the molecular causes of the observed difference in gene expression but also to identify small molecules that could drive or revert specific transcriptomic alterations.

Depends R (>= 4.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

biocViews DifferentialExpression, GeneExpression, RNASeq, Transcriptomics, Pathways, ImmunoOncology, GeneSetEnrichment


BugReports https://github.com/nuno-agostinho/cTRAP/issues

Suggests testthat, knitr, covr, rmarkdown, spelling, biomaRt, remotes

RoxygenNote 7.3.1

Imports AnnotationDbi, AnnotationHub, binr, cowplot, data.table, dplyr, DT, fastmatch, fgsea, ggplot2, ggrepel, graphics, highcharter, htmltools, htt, limma, methods, parallel, pbapply, purrr, qs, R.utils, readxl, reshape2, rhtml, rlang, scales, shiny (>= 1.7.0), shinyjson, stats, tibble, tools, utils

VignetteBuilder knitr

Language en-GB

Collate 'utils.R' 'CMap.R' 'ENCORE.R' 'cTRAP-package.r'
  'cmapR_subset.R' 'compare.R' 'drugSensitivity.R'
  'drugSetEnrichment.R' 'floweRy.R' 'plots.R' 'shinyInterface.R'
  'shinyInterface_session.R'
Contents

.git_url https://git.bioconductor.org/packages/cTRAP
.git_branch RELEASE_3_19
.git_last_commit 3cc210c
.git_last_commit_date 2024-04-30
.Repository Bioconductor 3.19
.Date/Publication 2024-05-03
.Author Bernardo P. de Almeida [aut],
       Nuno Saraiva-Agostinho [aut, cre],
       Nuno L. Barbosa-Morais [aut, led]
.Maintainer Nuno Saraiva-Agostinho <nunodanielagostinho@gmail.com>

Contents

.plotBubbles .............................................. 4
.prepareNavPage ......................................... 4
.traceInList ............................................. 4
.analyseDrugSetEnrichment ................................ 5
.as.table.referenceComparison ............................ 6
.calculateCellLineMean ..................................... 7
.calculateEvenlyDistributedBins ......................... 8
.checkColnames ............................................ 9
.chunkColumns ............................................. 9
.closeOpenHandles ........................................ 10
.cmapMetadata ............................................. 10
.cmapPerturbationsCompounds ............................. 10
.cmapPerturbationsKD ...................................... 11
.compareQuantile .......................................... 12
.compareWithAllMethods .................................... 12
.convertENSEMBLtoGeneSymbols ............................ 14
.convertGeneIdentifiers ................................... 14
.counts ..................................................... 15
.cTRAP ..................................................... 16
.diffExprStat ............................................. 18
dimnames.expressionDrugSensitivityAssociation ............ 18
.downloadENCODEknockdownMetadata ......................... 19
.downloadIfNotFound ...................................... 19
.ENCODEmetadata .......................................... 20
.filterCMapMetadata ....................................... 20
.findIntersectingCompounds ................................ 21
.fix.datatypes ........................................... 22
.GCT-class ................................................. 22
.getCMapConditions ....................................... 23
.getCMapPerturbationTypes ................................ 24
.getENCODEcontrols ....................................... 25
.HTMLfast .................................................. 25
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>launchCMapDataLoader</td>
<td>26</td>
</tr>
<tr>
<td>launchDiffExprLoader</td>
<td>27</td>
</tr>
<tr>
<td>launchDrugSetEnrichmentAnalyser</td>
<td>27</td>
</tr>
<tr>
<td>launchMetadataViewer</td>
<td>28</td>
</tr>
<tr>
<td>launchResultPlotter</td>
<td>29</td>
</tr>
<tr>
<td>listExpressionDrugSensitivityAssociation</td>
<td>29</td>
</tr>
<tr>
<td>loadCMapData</td>
<td>30</td>
</tr>
<tr>
<td>loadCMapZscores</td>
<td>31</td>
</tr>
<tr>
<td>loadCTRPgeneExpression</td>
<td>32</td>
</tr>
<tr>
<td>loadDrugDescriptors</td>
<td>33</td>
</tr>
<tr>
<td>loadENCODESample</td>
<td>34</td>
</tr>
<tr>
<td>loadENCODESamples</td>
<td>34</td>
</tr>
<tr>
<td>loadExpressionDrugSensitivityAssociation</td>
<td>35</td>
</tr>
<tr>
<td>loadNCI60drugSensitivity</td>
<td>36</td>
</tr>
<tr>
<td>matchStatsWithDrugSetsID</td>
<td>37</td>
</tr>
<tr>
<td>parseCMapID</td>
<td>38</td>
</tr>
<tr>
<td>performDifferentialExpression</td>
<td>38</td>
</tr>
<tr>
<td>performGSEA</td>
<td>39</td>
</tr>
<tr>
<td>plot.perturbationChanges</td>
<td>40</td>
</tr>
<tr>
<td>plot.referenceComparison</td>
<td>42</td>
</tr>
<tr>
<td>plotDrugSetEnrichment</td>
<td>44</td>
</tr>
<tr>
<td>plotESplot</td>
<td>45</td>
</tr>
<tr>
<td>plotGSEA</td>
<td>45</td>
</tr>
<tr>
<td>plotMetricDistribution</td>
<td>46</td>
</tr>
<tr>
<td>plotSingleCor</td>
<td>47</td>
</tr>
<tr>
<td>plotTargetingDrugsVSsimilarPerturbations</td>
<td>47</td>
</tr>
<tr>
<td>predictTargetingDrugs</td>
<td>49</td>
</tr>
<tr>
<td>prepareCMapPerturbations</td>
<td>50</td>
</tr>
<tr>
<td>prepareDrugSets</td>
<td>52</td>
</tr>
<tr>
<td>prepareENCODEgeneExpression</td>
<td>53</td>
</tr>
<tr>
<td>prepareExpressionDrugSensitivityAssociation</td>
<td>54</td>
</tr>
<tr>
<td>prepareGSEAgenesets</td>
<td>54</td>
</tr>
<tr>
<td>prepareSetsCompoundInfo</td>
<td>55</td>
</tr>
<tr>
<td>prepareStatsCompoundInfo</td>
<td>56</td>
</tr>
<tr>
<td>prepareWordBreak</td>
<td>56</td>
</tr>
<tr>
<td>print.similarPerturbations</td>
<td>57</td>
</tr>
<tr>
<td>processByChunks</td>
<td>57</td>
</tr>
<tr>
<td>processIds</td>
<td>58</td>
</tr>
<tr>
<td>rankAgainstReference</td>
<td>59</td>
</tr>
<tr>
<td>rankColumns</td>
<td>60</td>
</tr>
<tr>
<td>rankSimilarPerturbations</td>
<td>61</td>
</tr>
<tr>
<td>readGctxIds</td>
<td>63</td>
</tr>
<tr>
<td>readGctxxMeta</td>
<td>64</td>
</tr>
<tr>
<td>stripStr</td>
<td>65</td>
</tr>
<tr>
<td>subsetData</td>
<td>65</td>
</tr>
<tr>
<td>subsetDim</td>
<td>65</td>
</tr>
<tr>
<td>subsetToIds</td>
<td>66</td>
</tr>
</tbody>
</table>

**Index** 67
.plotBubbles  

*Plot packed bubbles*

**Description**

Plot packed bubbles

**Usage**

```
.plotBubbles(data, title, colour = "orange")
```

**Arguments**

- `data`: Data to plot
- `title`: Character: plot title
- `colour`: Character: bubble colour

**Value**

highchart object

---

.prepareNavPage  

*Prepare Shiny page template*

**Description**

Prepare Shiny page template

**Usage**

```
.prepareNavPage(...)  
```

**Value**

HTML elements

---

.traceInList  

*Find an item in list of lists and return its coordinates*

**Description**

Find an item in list of lists and return its coordinates

**Usage**

```
.traceInList(ll, item)
```
**analyseDrugSetEnrichment**

*Analyse drug set enrichment*

**Description**

Analyse drug set enrichment

**Usage**

```r
analyseDrugSetEnrichment(
    sets,
    stats,
    col = NULL,
    nperm = 10000,
    maxSize = 500,
    ...,
    keyColSets = NULL,
    keyColStats = NULL
)
```

**Arguments**

- `sets` Named list of characters: named sets containing compound identifiers (obtain drug sets by running `prepareDrugSets()`)
- `stats` Named numeric vector or either a `similarPerturbations` or a `targetingDrugs` object (obtained after running `rankSimilarPerturbations` or `predictTargetingDrugs`, respectively)
- `col` Character: name of the column to use for statistics (only required if class of `stats` is either `similarPerturbations` or `targetingDrugs`)
- `nperm` Number of permutations to do. Minimal possible nominal p-value is about $1/nperm$
- `maxSize` Maximal size of a gene set to test. All pathways above the threshold are excluded.
- `...` Arguments passed on to `fgsea::fgseaSimple`
- `minSize` Minimal size of a gene set to test. All pathways below the threshold are excluded.
- `scoreType` This parameter defines the GSEA score type. Possible options are ("std", "pos", "neg"). By default ("std") the enrichment score is computed as in the original GSEA. The "pos" and "neg" score types are intended to be used for one-tailed tests (i.e. when one is interested only in positive ("pos") or negative ("neg") enrichment).
- `nproc` If not equal to zero sets `BPPARAM` to use `nproc` workers (default = 0).
- `gseaParam` GSEA parameter value, all gene-level statis are raised to the power of `gseaParam` before calculation of GSEA enrichment scores.
BPPARAM Parallelization parameter used in bplapply. Can be used to specify cluster to run. If not initialized explicitly or by setting ‘nproc’ default value ‘bpparam()’ is used.

keyColSets Character: column from sets to compare with column keyColStats from stats; automatically selected if NULL

keyColStats Character: column from stats to compare with column keyColSets from sets; automatically selected if NULL

Value

Enrichment analysis based on GSEA

See Also

Other functions for drug set enrichment analysis: loadDrugDescriptors(), plotDrugSetEnrichment(), prepareDrugSets()

Examples

descriptors <- loadDrugDescriptors()
drugSets <- prepareDrugSets(descriptors)

# Analyse drug set enrichment in ranked targeting drugs for a differential expression profile
data("diffExprStat")
gdsc <- loadExpressionDrugSensitivityAssociation("GDSC")
predicted <- predictTargetingDrugs(diffExprStat, gdsc)
analyseDrugSetEnrichment(drugSets, predicted)

as.table.referenceComparison

Cross Tabulation and Table Creation

Description

Cross Tabulation and Table Creation

Usage

## S3 method for class 'referenceComparison'
as.table(x, ..., clean = TRUE)

Arguments

x referenceComparison object
... Extra parameters not currently used
clean Boolean: only show certain columns (to avoid redundancy)?
**calculateCellLineMean**

*Calculate cell line mean*

**Description**

Calculate cell line mean

**Usage**

```r
calculateCellLineMean(data, cellLine, metadata, rankPerCellLine)
```

**Arguments**

- `data` Data table: comparison against CMap data
- `cellLine` Character: perturbation identifiers as names and respective cell lines as values
- `metadata` Data table: data metadata
- `rankPerCellLine` Boolean: rank results based on both individual cell lines and mean scores across cell lines (TRUE) or based on mean scores alone (FALSE)? If `cellLineMean = FALSE`, individual cell line conditions are always ranked.

**Value**

A list with two items:

- `data` input data with extra rows containing cell line average scores (if calculated)
- `rankingInfo` data table with ranking information
- `metadata` metadata associated with output data, including for identifiers regarding mean cell line scores

**See Also**

Other functions related with the ranking of CMap perturbations: `filterCMapMetadata()`, `getCMapConditions()`, `getCMapPerturbationTypes()`, `loadCMapData()`, `loadCMapZscores()`, `parseCMapID()`, `plot.perturbationChanges()`, `plot.referenceComparison()`, `plotTargetingDrugsVSSimilarPerturbations()`, `prepareCMapPerturbations()`, `print.similarPerturbations()`

Other functions related with the prediction of targeting drugs: `listExpressionDrugSensitivityAssociation()`, `loadExpressionDrugSensitivityAssociation()`, `plot.referenceComparison()`, `plotTargetingDrugsVSSimilarPerturbations()`, `predictTargetingDrugs()`
calculateEvenlyDistributedBins

Calculate evenly-distributed bins

Description

Calculate evenly-distributed bins

Usage

```r
calculateEvenlyDistributedBins(
  numbers,
  maxBins = 15,
  k = 5,
  minPoints = NULL,
  ...,
  ids = NULL
)
```

Arguments

- **numbers**: Numeric
- **maxBins**: Numeric: maximum number of bins for numeric columns
- **k**: Numeric: constant; the higher the constant, the smaller the bin size (check `minpts`)
- **minPoints**: Numeric: minimum number of points in a bin (if NULL, the minimum number of points is the number of non-missing values divided by `maxBins` divided by `k`)
- **...**: Arguments passed on to `binr::bins`
  - `max.breaks` Used for initial cut. If `exact.groups` is `FALSE`, bins are merged until there's no bins with fewer than `length(x) / max.breaks` points. In `bins`, one of `max.breaks` and `minpts` must be supplied.
  - `exact.groups` if `TRUE`, the result will have exactly the number of `target.bins`; if `FALSE`, the result may contain fewer than `target.bins` bins
  - `verbose` Indicates verbose output.
  - `errthresh` If the error is below the provided value, stops after the first rough estimate of the bins.

Value

Factor containing the respective group of each element in `numbers`
checkColnames

Check whether test_names are columns in the data.frame

Description
Check whether test_names are columns in the data.frame

Usage
checkColnames(test_names, df, throw_error = TRUE)

Arguments
- test_names: a vector of column names to test
- df: the data.frame to test against
- throw_error: boolean indicating whether to throw an error if any test_names are not found in df

Value
boolean indicating whether or not all test_names are columns of df

Source
https://github.com/cmap/cmapR

chunkColumns
Assign columns into chunks

Description
Assign columns into chunks

Usage
chunkColumns(x, nrows, chunkGiB)

Arguments
- x: Vector of elements
- nrows: Numeric: number of rows
- chunkGiB: Numeric: size (in gibibytes) of chunks to load reference file; only if argument reference is a file path

Value
List of chunks with equally distributed columns
closeOpenHandles  

**Description**

Close open handles

**Usage**

closeOpenHandles()

**Value**

Closes all open identifiers

---

cmapMetadata  

**Description**

CMap metadata obtained by running the following code:

```r
cmapMetadata <- filterCMapMetadata("cmapMetadata.txt", cellLine = "HEPG2", timepoint = "2 h")
```

---

cmapPerturbationsCompounds  

**Description**

CMap perturbations sample for small molecules obtained by running the following code:

```r
cellLine <- c("HepG2", "HUH7")
cmapMetadataCompounds <- filterCMapMetadata("cmapMetadata.txt", cellLine=cellLine, timepoint="24 h", dosage="5 \u00B5M", perturbationType="Compound")
cmapPerturbationsCompounds <- prepareCMapPerturbations( cmapMetadataCompounds, "cmapZscores.gctx", "cmapGeneInfo.txt", "cmapCompoundInfo_drugs.txt", loadZscores=TRUE)
```

# Remove non-ASCII characters for portability reasons
CMap perturbations sample for knockdown experiments obtained by running the following code:

```r
# Code for loading CMap gene KD HepG2 data
cellLine <- "HepG2"
cmapMetadataKD <- filterCMapMetadata("cmapMetadata.txt", cellLine=cellLine,
      perturbationType="Consensus signature from shRNAs targeting the same gene")
cmapPerturbationsKD <- prepareCMapPerturbations(cmapMetadataKD, "cmapZscores.gctx", "cmapGeneInfo.txt",
      loadZscores=TRUE)
data("diffExprStat")
compareKD <- rankSimilarPerturbations(diffExprStat, cmapPerturbationsKD)

# Select only some perturbations (to reduce file size)
filter <- c(head(order(compareKD$spearman_rank)),
      tail(order(compareKD$spearman_rank)),
      head(order(compareKD$pearson_rank)),
      tail(order(compareKD$pearson_rank)),
      head(order(compareKD$gsea_rank)),
      tail(order(compareKD$gsea_rank)))
filter <- unique(compareKD[[1]][filter])
cmapPerturbationsKD <- cmapPerturbationsKD[, filter]

# Remove non-ASCII characters for portability reasons
metadata <- attr(cmapPerturbationsCompounds, "metadata")
metadata$pert_idose <- gsub("\\u00B5", "micro", metadata$pert_idose)
metadata$pert_dose_unit <- gsub("\\u00B5", "micro", metadata$pert_dose_unit)
attr(cmapPerturbationsCompounds, "metadata") <- metadata
```
compareQuantile  

*Compare vector against its quantile*

**Description**

Check which elements of the vector are lower/greater than or equal to the quantile of a given vector.

**Usage**

```r
compareQuantile(vec, prob, lte = FALSE)
```

**Arguments**

- **vec**: Numeric vector
- **prob**: Numeric: probability value between [0,1] to produce sample quantiles
- **lte**: Boolean: check if values are <= quantile? If FALSE, checks if values are >= quantile

**Value**

Boolean vector regarding compared elements

---

compareWithAllMethods  

*Compare reference using all methods*

**Description**

Compare reference using all methods

**Usage**

```r
compareWithAllMethods(
    method,          
    input,           
    reference,       
    geneSize = 150, 
    cellLines = NULL, 
    cellLineMean = "auto", 
    rankPerCellLine = FALSE, 
    threads = 1,     
    chunkGiB = 1,    
    verbose = FALSE  
)
```
compareWithAllMethods

Arguments

method Character: comparison method (spearman, pearson or gsea; multiple methods may be selected at once)

input Named numeric vector of differentially expressed genes whose names are gene identifiers and respective values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics); or character of gene symbols composing a gene set that is tested for enrichment in reference data (only used if method includes gsea)

reference Data matrix or character object with file path to CMap perturbations (see prepareCMapPerturbations()) or gene expression and drug sensitivity association (see loadExpressionDrugSensitivityAssociation())

geneSize Numeric: number of top up-/down-regulated genes to use as gene sets to test for enrichment in reference data; if a 2-length numeric vector, the first index is the number of top up-regulated genes and the second index is the number of down-regulated genes used to create gene sets; only used if method includes gsea and if input is not a gene set

cellLines Integer: number of unique cell lines

cellLineMean Boolean: add rows with the mean of method across cell lines? If cellLineMean = "auto" (default), rows will be added when data for more than one cell line is available.

rankPerCellLine Boolean: rank results based on both individual cell lines and mean scores across cell lines (TRUE) or based on mean scores alone (FALSE)? If cellLineMean = FALSE, individual cell line conditions are always ranked.

threads Integer: number of parallel threads

chunkGiB Numeric: size (in gibibytes) of chunks to load reference file; only if argument reference is a file path

verbose Boolean: print additional details?

rankByAscending Boolean: rank values based on their ascending (TRUE) or descending (FALSE) order?

Value

List of data tables with correlation and/or GSEA score results

GSEA score

When method = "gsea", weighted connectivity scores (WTCS) are calculated (https://clue.io/connectopedia/cmap_algorithms).
# convertENSMBLtoGeneSymbols

*Convert ENSEMBL gene identifiers to gene symbols*

## Description
Convert ENSEMBL gene identifiers to gene symbols

## Usage

```r
convertENSMBLtoGeneSymbols(
  genes,
  dataset = "hsapiens_gene_ensembl",
  mart = "ensembl"
)
```

## Arguments

- **genes**: Character: ENSEMBL gene identifiers
- **dataset**: Character: biomaRt dataset name
- **mart**: Character: biomaRt database name

## Value
Named character vector where names are the input ENSEMBL gene identifiers and the values are the matching gene symbols

# convertGeneIdentifiers

*Convert gene identifiers*

## Description
Convert gene identifiers

## Usage

```r
convertGeneIdentifiers(
  genes,
  annotation = "Homo sapiens",
  key = "ENSEMBL",
  target = "SYMBOL",
  ignoreDuplicatedTargets = TRUE
)
```
**Arguments**

genes | Character: genes to be converted
annotation | OrgDb with genome wide annotation for an organism or character with species name to query OrgDb, e.g. "Homo sapiens"
key | Character: type of identifier used, e.g. ENSEMBL; read ?AnnotationDbi::columns
target | Character: type of identifier to convert to; read ?AnnotationDbi::columns
ignoreDuplicatedTargets | Boolean: if TRUE, identifiers that share targets with other identifiers will not be converted

**Value**

Character vector of the respective targets of gene identifiers. The previous identifiers remain other identifiers have the same target (in case ignoreDuplicatedTargets = TRUE) or if no target was found.

**Examples**

genes <- c("ENSG00000012048", "ENSG00000083093", "ENSG00000141510", "ENSG00000051180")
convertGeneIdentifiers(genes)
convertGeneIdentifiers(genes, key="ENSEMBL", target="UNIPROT")

# Explicit species name to automatically look for its OrgDb database
sp <- "Homo sapiens"
genes <- c("ENSG00000012048", "ENSG00000083093", "ENSG00000141510", "ENSG00000051180")
convertGeneIdentifiers(genes, sp)

# Alternatively, set the annotation database directly
ah <- AnnotationHub::AnnotationHub()
sp <- AnnotationHub::query(ah, c("OrgDb", "Homo sapiens"))[1]
columns(sp) # these attributes can be used to change the attributes
convertGeneIdentifiers(genes, sp)

**counts** | Gene expression data sample

**Description**

Gene expression data sample obtained by running the following code:

data("ENCODEmetadata")
ENCODEsamples <- loadENCODEsamples(ENCODEmetadata)[[1]]
counts <- prepareENCODEgeneExpression(ENCODEsamples)
# Remove low coverage (at least 10 counts shared across two samples)
minReads <- 10
minSamples <- 2
filter <- rowSums(counts[, -c(1, 2)] >= minReads) >= minSamples
counts <- counts[filter, ]

# Convert ENSEMBL identifier to gene symbol
counts$gene_id <- convertGeneIdentifiers(counts$gene_id)

cTRAP package

Description

Compare differential gene expression results with those from big datasets (e.g. CMap), allowing to infer which types of perturbations may explain the observed difference in gene expression.

Optimised to run in ShinyProxy with Celery/Flower backend with argument shinyproxy = TRUE.

Usage

cTRAP(
  ..., 
  commonPath = "data", 
  expire = 14, 
  fileSizeLimitMiB = 50, 
  flowerURL = NULL, 
  port =getOption("shiny.port"), 
  host =getOption("shiny.host", "127.0.0.1")
)

Arguments

... Objects
commonPath Character: path where to store data common to all sessions
expire Character: days until a session expires (message purposes only)
fileSizeLimitMiB Numeric: file size limit in MiB
flowerURL Character: Flower REST API’s URL (NULL to avoid using Celery/Flower backend)
port The TCP port that the application should listen on. If the port is not specified, and the shiny.port option is set (with options(shiny.port = XX)), then that port will be used. Otherwise, use a random port between 3000:8000, excluding ports that are blocked by Google Chrome for being considered unsafe: 3659, 4045, 5060, 5061, 6000, 6566, 6665:6669 and 6697. Up to twenty random ports will be tried.
host The IPv4 address that the application should listen on. Defaults to the shiny.host option, if set, or "127.0.0.1" if not. See Details.
Details

**Input**: To use this package, a named vector of differentially expressed gene metric is needed, where its values represent the significance and magnitude of the differentially expressed genes (e.g. t-statistic) and its names are gene symbols.

**Workflow**: The differentially expressed genes will be compared against selected perturbation conditions by:

- Spearman or Pearson correlation with z-scores of differentially expressed genes after perturbations from CMap. Use function `rankSimilarPerturbations` with method = "spearman" or method = "pearson"
- Gene set enrichment analysis (GSEA) using the (around) 12000 genes from CMap. Use function `rankSimilarPerturbations` with method = gsea.

Available perturbation conditions for CMap include:

- Cell line(s).
- Perturbation type (gene knockdown, gene upregulation or drug intake).
- Drug concentration.
- Time points.

Values for each perturbation type can be listed with `getCMapPerturbationTypes()`

**Output**: The output includes a data frame of ranked perturbations based on the associated statistical values and respective p-values.

Value

Launches result viewer and plotter (returns NULL)

Author(s)

**Maintainer**: Nuno Saraiva-Agostinho <nunodanielagostinho@gmail.com>

Authors:

- Bernardo P. de Almeida
- Nuno L. Barbosa-Morais [lead]

See Also

Useful links:

- [https://nuno-agostinho.github.io/cTRAP](https://nuno-agostinho.github.io/cTRAP)
- [https://github.com/nuno-agostinho/cTRAP](https://github.com/nuno-agostinho/cTRAP)
- Report bugs at [https://github.com/nuno-agostinho/cTRAP/issues](https://github.com/nuno-agostinho/cTRAP/issues)

Other visual interface functions: `launchCMapDataLoader()`, `launchDiffExprLoader()`, `launchDrugSetEnrichmentAnalyser()`, `launch_MetadataViewer()`, `launchResultPlotter()`
**diffExprStat**  
*Differential expression’s t-statistics sample*

**Description**

Differential expression’s t-statistics sample obtained by running the following code:

```r
data("counts")

# Perform differential gene expression analysis
diffExpr <- performDifferentialExpression(counts)

# Get t-statistics of differential expression with respective gene names
diffExprStat <- diffExpr$t
names(diffExprStat) <- diffExpr$Gene_symbol
```

**dimnames.expressionDrugSensitivityAssociation**

*Operations on expressionDrugSensitivityAssociation objects*

**Description**

Operations on expressionDrugSensitivityAssociation objects

**Usage**

```r
## S3 method for class 'expressionDrugSensitivityAssociation'
dimnames(x)

## S3 method for class 'expressionDrugSensitivityAssociation'
dim(x)

## S3 method for class 'expressionDrugSensitivityAssociation'
x[i, j, drop = FALSE, ...]
```

**Arguments**

- `x`  
  An expressionDrugSensitivityAssociation object
- `i, j`  
  Character or numeric indexes specifying elements to extract
- `drop`  
  Boolean: coerce result to the lowest possible dimension?
- `...`  
  Extra arguments given to other methods

**Value**

Subset, dimension or dimension names
downloadENCODENoiseDownMetadata

*Description*

Download metadata for ENCODE knockdown experiments

*Usage*

```r
downloadENCODENoiseDownMetadata(
  cellLine = NULL,
  gene = NULL,
  file = "ENCODEmetadata.rds"
)
```

*Arguments*

- `cellLine` Character: cell line
- `gene` Character: target gene
- `file` Character: RDS filepath with metadata (if file doesn’t exist, it will be created)

*Value*

Data frame containing ENCODE knockdown experiment metadata

*See Also*

Other functions related with using ENCODE expression data: `loadENCODEsamples()`, `performDifferentialExpression()`, `prepareENCODEgeneExpression()`

*Examples*

```r
downloadENCODENoiseDownMetadata("HepG2", "EIF4G1")
```

downloadIfNotFound

*Description*

Download data if given file is not found

*Usage*

```r
downloadIfNotFound(link, file, ask = FALSE, toExtract = NULL)
```
filterCMapMetadata

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link</td>
<td>Character</td>
<td>link to download file</td>
</tr>
<tr>
<td>file</td>
<td>Character</td>
<td>filepath</td>
</tr>
<tr>
<td>ask</td>
<td>Boolean</td>
<td>ask to download file?</td>
</tr>
<tr>
<td>toExtract</td>
<td>Character</td>
<td>files to extract (if NULL, extract all)</td>
</tr>
</tbody>
</table>

Value

Download file if file is not found

Description

ENCODE metadata sample obtained by running the following code:

```r
gene <- "EIF4G1"
cellLine <- "HepG2"
ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

table(ENCODEmetadata$'Experiment target')
length(unique(ENCODEmetadata$'Experiment target'))
```

Description

Filter CMap metadata

Usage

```r
filterCMapMetadata(
    metadata,    # Link to download file
    cellLine = NULL,    # Cell line
    timepoint = NULL,    # Time point
    dosage = NULL,    # Dosage
    perturbationType = NULL
)
```
findIntersectingCompounds

Arguments

metadata  Data frame (CMap metadata) or character (respective filepath)
cellLine  Character: cell line (if NULL, all values are loaded)
timepoint  Character: timepoint (if NULL, all values are loaded)
dosage  Character: dosage (if NULL, all values are loaded)
perturbationType  Character: type of perturbation (if NULL, all perturbation types are loaded)

Value

Filtered CMap metadata

See Also

Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), getCMapConditions(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations(), prepareCMapPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Examples

cmapMetadata <- loadCMapData("cmapMetadata.txt", "metadata")
filterCMapMetadata(cmapMetadata, cellLine="HEPG2", timepoint="2 h", dosage="25 ng/mL")

findIntersectingCompounds  
\textit{Check for intersecting compounds across specific columns on both datasets}

Description

Check for intersecting compounds across specific columns on both datasets

Usage

findIntersectingCompounds(data1, data2, keys1 = NULL, keys2 = NULL)

Value

List containing three elements: matching compounds commonCompounds between column key 1 and key 2 from the first and second datasets, respectively
A function to adjust the data types for columns of a metadata frame. GCT(X) parsing initially returns data frames of row and column descriptors where all columns are of type character. This is inconvenient for analysis, so the goal of this function is to try and guess the appropriate data type for each column.

```
fix.datatypes(meta)
```

Arguments:

- `meta`: a data.frame

Details:

This is a low-level helper function which most users will not need to access directly.

Value:

- `meta`: the same data frame with (potentially) adjusted column types

See Also:

Other GCTX parsing functions: `processIds()`, `readGctxIds()`, `readGctxMeta()`
getCMapConditions

Slots

mat  a numeric matrix
rid  a character vector of row ids
cid  a character vector of column ids
rdesc a data.frame of row descriptors
rdesc a data.frame of column descriptors
src  a character indicating the source (usually file path) of the data

Source

https://github.com/cmap/cmapR

See Also

http://clue.io/help for more information on the GCT format

Description

Downloads metadata if not available

Usage

getcMapConditions(
  metadata,
  cellLine = NULL,
  timepoint = NULL,
  dosage = NULL,
  perturbationType = NULL,
  control = FALSE
)

Arguments

metadata  Data frame (CMap metadata) or character (respective filepath)
cellLine  Character: cell line (if NULL, all values are loaded)
timepoint Character: timepoint (if NULL, all values are loaded)
dosage  Character: dosage (if NULL, all values are loaded)
perturbationType  Character: type of perturbation (if NULL, all perturbation types are loaded)
control  Boolean: show controls for perturbation types?
getCMapPerturbationTypes

Value
List of conditions in CMap datasets

See Also
Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations(), prepareCMapPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Examples

## Not run:
cmapMetadata <- loadCMapData("cmapMetadata.txt", "metadata")

## End(Not run)
getCMapConditions(cmapMetadata)

getcMapPerturbationTypes

Get CMap perturbation types

Description
Get CMap perturbation types

Usage
getcMapPerturbationTypes(control = FALSE)

Arguments
control Boolean: return perturbation types used as control?

Value
Perturbation types and respective codes as used by CMap datasets

See Also
Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations(), prepareCMapPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Examples
getcMapPerturbationTypes()
getENCODEcontrols

Get experiments files for a given control

Description

Get experiments files for a given control

Usage

getENCODEcontrols(control, table)

Arguments

control Character: control identifier
table Data frame

Value

Character vector with respective experiment identifiers

HTMLfast

Faster version of shiny::HTML

Description

Faster version of shiny::HTML

Usage

HTMLfast(text)

Arguments

text Character: text

Value

HTML element
launchCMapDataLoader  Load CMap data via a visual interface

Description
Load CMap data via a visual interface

Usage
launchCMapDataLoader(
    metadata = "cmapMetadata.txt",
    zscores = "cmapZscores.gctx",
    geneInfo = "cmapGeneInfo.txt",
    compoundInfo = "cmapCompoundInfo.txt",
    cellLine = NULL,
    timepoint = NULL,
    dosage = NULL,
    perturbationType = NULL
)

Arguments
metadata    Data frame (CMap metadata) or character (respective filepath)
zscores     Data frame (GCTX z-scores) or character (respective filepath to load data from file)
geneInfo    Data frame (CMap gene info) or character (respective filepath to load data from file)
compoundInfo Data frame (CMap compound info) or character (respective filepath to load data from file)
cellLine    Character: cell line (if NULL, all values are loaded)
timepoint   Character: timepoint (if NULL, all values are loaded)
dosage      Character: dosage (if NULL, all values are loaded)
perturbationType Character: type of perturbation (if NULL, all perturbation types are loaded)

Value
CMap data

See Also
Other visual interface functions: cTRAP(), launchDiffExprLoader(), launchDrugSetEnrichmentAnalyser(), launchMetadataViewer(), launchResultPlotter()
launchDiffExprLoader

Load differential expression data via a visual interface

Description
Currently only supports loading data from ENCODE knockdown experiments

Usage
launchDiffExprLoader(
  cellLine = NULL,
  gene = NULL,
  file = "ENCODEmetadata.rds",
  path = "."
)

Arguments

  cellLine  Character: cell line
  gene     Character: target gene
  file     Character: RDS filepath with metadata (if file doesn’t exist, it will be created)
  path     Character: path where to download files

Value
Differential expression data

See Also
Other visual interface functions: cTRAP(), launchCMapDataLoader(), launchDrugSetEnrichmentAnalyser(), launchMetadataViewer(), launchResultPlotter()

launchDrugSetEnrichmentAnalyser

View and plot results via a visual interface

Description
View and plot results via a visual interface

Usage
launchDrugSetEnrichmentAnalyser(sets, ...)

Arguments

sets    Named list of characters: named sets containing compound identifiers (obtain drug sets by running `prepareDrugSets()`)

...    Objects

Value

Launches result viewer and plotter (returns NULL)

See Also

Other visual interface functions: `cTRAP()`, `launchCMapDataLoader()`, `launchDiffExprLoader()`, `launchMetadataViewer()`, `launchResultPlotter()`
launchResultPlotter  View and plot results via a visual interface

Description
View and plot results via a visual interface

Usage
launchResultPlotter(...)

Arguments
...

Objects

Value
Launches result viewer and plotter (returns NULL)

See Also
Other visual interface functions: cTRAP(), launchCMapDataLoader(), launchDiffExprLoader(), launchDrugSetEnrichmentAnalyser(), launchMetadataViewer()

listExpressionDrugSensitivityAssociation
List available gene expression and drug sensitivity correlation matrices

Description
List available gene expression and drug sensitivity correlation matrices

Usage
listExpressionDrugSensitivityAssociation(url = FALSE)

Arguments
url

Boolean: return download link?

Value
Character vector of available gene expression and drug sensitivity correlation matrices
loadCMapData

Description
Load CMap data (if not found, file will be automatically downloaded)

Usage
loadCMapData(
  file,
  type = c("metadata", "geneInfo", "zscores", "compoundInfo"),
  zscoresID = NULL
)

Arguments
file Character: path to file
type Character: type of data to load (metadata, geneInfo, zscores or compoundInfo)
zscoresID Character: identifiers to partially load z-scores file (for performance reasons; if NULL, all identifiers will be loaded)

Value
Metadata as a data table

Note
If type = "compoundInfo", two files from The Drug Repurposing Hub will be downloaded containing information about drugs and perturbations. The files will be named file with _drugs and _samples before their extension, respectively.

See Also
Other functions related with the prediction of targeting drugs: as.table.referenceComparison(), loadExpressionDrugSensitivityAssociation(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations(), predictTargetingDrugs()
loadCMapZscores

Examples

# Load CMap metadata (data is automatically downloaded if not available)
cmapMetadata <- loadCMapData("cmapMetadata.txt", "metadata")

# Load CMap gene info
loadCMapData("cmapGeneInfo.txt", "geneInfo")
## Not run:
# Load CMap zscores based on filtered metadata
cmapMetadataKnockdown <- filterCMapMetadata(
  cmapMetadata, cellLine="HepG2",
  perturbationType="Consensus signature from shRNAs targeting the same gene")
loadCMapData("cmapZscores.gctx.gz", "zscores", cmapMetadataKnockdown$sig_id)

## End(Not run)

loadCMapZscores

Load matrix of CMap perturbation’s differential expression z-scores (optional)

Description

Load matrix of CMap perturbation’s differential expression z-scores (optional)

Usage

loadCMapZscores(data, inheritAttrs = FALSE, verbose = TRUE)

Arguments

data perturbationChanges object

inheritAttrs Boolean: convert to perturbationChanges object and inherit attributes from data?

verbose Boolean: print additional details?

Value

Matrix containing CMap perturbation z-scores (genes as rows, perturbations as columns)

See Also

Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(),getCMapConditions(),getCMapPerturbationTypes(),loadCMapData(), parseCMapID(),plot.perturbationChanges(),plot.referenceComparison(),plotTargetingDrugsVSsimilarPerturbations(),prepareCMapPerturbations(),print.similarPerturbations(),rankSimilarPerturbations()
loadCTRPgeneExpression

Load CTRP data

Description

If given paths direct to non-existing files, those files will be downloaded

Usage

loadCTRPgeneExpression(
  geneExpressionFile = "CTRP 2.1/geneExpr.txt",
  geneInfoFile = "CTRP 2.1/geneInfo.txt",
  cellLineInfoFile = "CTRP 2.1/cellLineInfo.txt"
)

loadCTRPdrugSensitivity(
  drugSensitivityFile = "CTRP 2.1/drugSensitivity.txt",
  experimentFile = "CTRP 2.1/experimentInfo.txt",
  compoundFile = "CTRP 2.1/compoundInfo.txt"
)

loadCTRPcompoundInfo(compoundFile = "CTRP 2.1/compoundInfo.txt")

loadNCI60geneExpression(
  file = "NCI60/geneExpr.xls",
  cellLineInfoFile = "cellLineInfo.xls"
)

loadGDSC7file(file, filename, type, ...)

loadGDSC7cellLineInfo(file = "GDSC_7/cellLineInfo.xlsx")

loadGDSC7compoundInfo(file = "GDSC_7/compoundInfo.xlsx")

loadGDSC7geneExpression(file = "GDSC_7/geneExpr.txt")

loadGDSC7drugSensitivity(file = "GDSC_7/drugs.xlsx")

Examples

metadata <- loadCMapData("cmapMetadata.txt", "metadata")
metadata <- filterCMapMetadata(metadata, cellLine="HepG2")
## Not run:
perts <- prepareCMapPerturbations(metadata, "cmapZscores.gctx",
  "cmapGeneInfo.txt")
zscores <- loadCMapZscores(perts[, 1:10])
## End(Not run)
loadDrugDescriptors

Arguments

geneExpressionFile
  Character: path to file with gene expression
geneInfoFile
  Character: path to file with gene information
cellLineInfoFile
  Character: path to file with cell line information
drugSensitivityFile
  Character: path to file with drug sensitivity
experimentFile
  Character: path to file with experiment information
compoundFile
  Character: path to file with compound information
file
  Character: file path

Value

  Data frame

Description

  Load table with drug descriptors

Usage

loadDrugDescriptors(
  source = c("NCI60", "CMap"),
  type = c("2D", "3D"),
  file = NULL,
  path = NULL
)

Arguments

  source
    Character: source of compounds used to calculate molecular descriptors (NCI60 or CMap)
  type
    Character: load 2D or 3D molecular descriptors
  file
    Character: filepath to drug descriptors (automatically downloaded if file does not exist)
  path
    Character: folder where to find files (optional; file may contain the full filepath if preferred)

Value

  Data table with drug descriptors
See Also

Other functions for drug set enrichment analysis: `analyseDrugSetEnrichment()`, `plotDrugSetEnrichment()`, `prepareDrugSets()`

Examples

```r
loadDrugDescriptors()
```

---

### loadENCODEsample

Load ENCODE sample

#### Usage

```r
loadENCODEsample(metadata, replicate, control = FALSE, path = ".")
```

#### Arguments

- `metadata`: Data frame: ENCODE metadata
- `replicate`: Number: replicate
- `control`: Boolean: load control experiment?
- `path`: Character: path where to download files

#### Value

Data table with ENCODE sample data

---

### loadENCODESamples

Load ENCODE samples

#### Usage

```r
loadENCODESamples(metadata, path = ".")
```

#### Arguments

- `metadata`: Character: ENCODE metadata
- `path`: Character: path where to download files

#### Description

Samples are automatically downloaded if they are not found in the current working directory.
loadExpressionDrugSensitivityAssociation

Value
List of loaded ENCODE samples

See Also
Other functions related with using ENCODE expression data: downloadENCODEknockdownMetadata(), performDifferentialExpression(), prepareENCODEgeneExpression()

Examples
if (interactive()) {
  # Load ENCODE metadata for a specific cell line and gene
  cellLine <- "HepG2"
  gene <- c("EIF4G1", "U2AF2")
  ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

  # Load samples based on filtered ENCODE metadata
  loadENCODEsamples(ENCODEmetadata)
}

loadExpressionDrugSensitivityAssociation

Load gene expression and drug sensitivity correlation matrix

Description
Load gene expression and drug sensitivity correlation matrix

Usage
loadExpressionDrugSensitivityAssociation(
  source,
  file = NULL,
  path = NULL,
  rows = NULL,
  cols = NULL,
  loadValues = FALSE
)

Arguments

source Character: source of matrix to load; see listExpressionDrugSensitivityAssociation

file Character: filepath to gene expression and drug sensitivity association dataset (automatically downloaded if file does not exist)

path Character: folder where to find files (optional; file may contain the full filepath if preferred)

rows Character or integer: rows
loadNCI60drugSensitivity

cols  Character or integer: columns
loadValues  Boolean: load data values (if available)? If FALSE, downstream functions will load and process directly from the file chunk by chunk, resulting in a lower memory footprint

Value

Correlation matrix between gene expression (rows) and drug sensitivity (columns)

See Also

Other functions related with the prediction of targeting drugs: as.table.referenceComparison(), listExpressionDrugSensitivityAssociation(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations(), predictTargetingDrugs()

Examples

gdsc <- listExpressionDrugSensitivityAssociation()[[1]]
loadExpressionDrugSensitivityAssociation(gdsc)

loadNCI60drugSensitivity

Load CTRP data

Description

If given paths direct to non-existing files, those files will be downloaded

Usage

loadNCI60drugSensitivity(file = "NCI60/drugSensitivity.xls")

Arguments

file  Character: file path

Value

Data frame
matchStatsWithDrugSetsID

Match identifiers between data and drug sets

Description

Match identifiers between data and drug sets

Usage

matchStatsWithDrugSetsID(
  sets,
  stats,
  col = "values",
  keyColSets = NULL,
  keyColStats = NULL
)

Arguments

sets: Named list of characters: named sets containing compound identifiers (obtain drug sets by running prepareDrugSets())
stats: Named numeric vector or either a similarPerturbations or a targetingDrugs object (obtained after running rankSimilarPerturbations or predictTargetingDrugs, respectively)
col: Character: name of the column to use for statistics (only required if class of stats is either similarPerturbations or targetingDrugs)
keyColSets: Character: column from sets to compare with column keyColStats from stats; automatically selected if NULL
keyColStats: Character: column from stats to compare with column keyColSets from sets; automatically selected if NULL

Value

Statistic values from input data and corresponding identifiers as names (if no match is found, the original identifier from argument stats is used)
### parseCMapID

**Parse CMap identifier**

**Description**

Parse CMap identifier

**Usage**

```r
parseCMapID(id, cellLine = FALSE)
```

**Arguments**

- `id`: Character: CMap identifier
- `cellLine`: Boolean: if TRUE, return cell line information from CMap identifier; else, return the CMap identifier without the cell line

**Value**

Character vector with information from CMap identifiers

**See Also**

Other functions related with the ranking of CMap perturbations: `as.table.referenceComparison()`, `filterCMapMetadata()`, `getCMapConditions()`, `getCMapPerturbationTypes()`, `loadCMapData()`, `loadCMapZscores()`, `plot.perturbationChanges()`, `plot.referenceComparison()`, `plotTargetingDrugsVSsimilarPerturbations()`, `prepareCMapPerturbations()`, `print.similarPerturbations()`, `rankSimilarPerturbations()`

**Examples**

```r
parseCMapID(id, cellLine=TRUE)
parseCMapID(id, cellLine=FALSE)
```

---

### performDifferentialExpression

*Perform differential gene expression based on ENCODE data*

**Description**

Perform differential gene expression based on ENCODE data

**Usage**

```r
performDifferentialExpression(counts)
```
performGSEA

Arguments

counts Data frame: gene expression

Value

Data frame with differential gene expression results between knockdown and control

See Also

Other functions related with using ENCODE expression data: downloadENCODEknockdownMetadata(), loadENCODEsamples(), prepareENCODEgeneExpression()

Examples

if (interactive()) {
  cellLine <- "HepG2"
gene <- "EIF4G1"
ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

# Download samples based on filtered ENCODE metadata
ENCODEsamples <- loadENCODEsamples(ENCODEmetadata)[[1]]

counts <- prepareENCODEgeneExpression(ENCODEsamples)

# Remove low coverage (at least 10 counts shared across two samples)
minReads <- 10
minSamples <- 2
filter <- rowSums(counts[, -c(1, 2)] >= minReads) >= minSamples
counts <- counts[filter, ]

# Convert ENSEMBL identifier to gene symbol
counts$gene_id <- convertGeneIdentifiers(counts$gene_id)

diffExpr <- performDifferentialExpression(counts)
}

performGSEA Perform GSEA

Description

Perform GSEA

Usage

performGSEA(pathways, stats)
plot.perturbationChanges

Operations on a perturbationChanges object

Description

Operations on a perturbationChanges object

Usage

## S3 method for class 'perturbationChanges'
plot(
  x,
  perturbation,
  input,
  method = c("spearman", "pearson", "gsea"),
  geneSize = 150,
  genes = c("both", "top", "bottom"),
  ...,
  title = NULL
)

## S3 method for class 'perturbationChanges'
 x[i, j, drop = FALSE, ...]

## S3 method for class 'perturbationChanges'
dim(x)

## S3 method for class 'perturbationChanges'
dimnames(x)

Arguments

x perturbationChanges object

perturbation Character (perturbation identifier) or a similarPerturbations table (from which the respective perturbation identifiers are retrieved)

input Named numeric vector of differentially expressed genes whose names are gene identifiers and respective values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics); or character of gene symbols composing a gene set that is tested for enrichment in reference data (only used if method includes gsea)

method Character: comparison method (spearman, pearson or gsea; multiple methods may be selected at once)
geneSize Numeric: number of top up-/down-regulated genes to use as gene sets to test for enrichment in reference data; if a 2-length numeric vector, the first index is the number of top up-regulated genes and the second index is the number of down-regulated genes used to create gene sets; only used if method includes gsea and if input is not a gene set

genes Character: when plotting gene set enrichment analysis (GSEA), plot most up-regulated genes (genes = "top"), most down-regulated genes (genes = "bottom") or both (genes = "both"); only used if method = "gsea" and geneset = NULL

title Character: plot title (if NULL, the default title depends on the context; ignored when plotting multiple perturbations)
i, j Character or numeric indexes specifying elements to extract

drop Boolean: coerce result to the lowest possible dimension?

Value Subset, plot or return dimensions or names of a perturbationChanges object

See Also Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getGeneSetList(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plotTargetingDrugsVSSimilarPerturbations(), prepareCMapPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Examples

data("diffExprStat")
data("cmapPerturbationsKD")

compK <- rankSimilarPerturbations(diffExprStat, cmapPerturbationsKD)
EIF4GIknockdown <- grep("EIF4GI", compK[[1]], value=TRUE)
plot(cmapPerturbationsKD, EIF4GIknockdown, diffExprStat, method="spearman")
plot(cmapPerturbationsKD, EIF4GIknockdown, diffExprStat, method="pearson")
plot(cmapPerturbationsKD, EIF4GIknockdown, diffExprStat, method="gsea")

data("cmapPerturbationsCompounds")
pert <- "CVD001_HEPG2_24H:BRD-A14014306-001-01-1:4.1"
plot(cmapPerturbationsCompounds, pert, diffExprStat, method="spearman")
plot(cmapPerturbationsCompounds, pert, diffExprStat, method="pearson")
plot(cmapPerturbationsCompounds, pert, diffExprStat, method="gsea")

# Multiple cell line perturbations
pert <- "CVD001_HEPG2_24H:BRD-A14014306-001-01-1:4.1"
plot(cmapPerturbationsCompounds, pert, diffExprStat, method="spearman")
plot(cmapPerturbationsCompounds, pert, diffExprStat, method="pearson")
plot(cmapPerturbationsCompounds, pert, diffExprStat, method="gsea")
Description

If `element = NULL`, comparison is plotted based on all elements. Otherwise, show scatter or GSEA plots for a single element compared with previously given differential expression results.

Usage

```r
## S3 method for class 'referenceComparison'
plot(
  x,
  element = NULL,
  method = c("spearman", "pearson", "gsea", "rankProduct"),
  n = c(3, 3),
  showMetadata = TRUE,
  plotNonRankedPerturbations = FALSE,
  alpha = 0.3,
  genes = c("both", "top", "bottom"),
  ...
)
```

Arguments

- `x` referenceComparison object: obtained after running `rankSimilarPerturbations()` or `predictTargetingDrugs()`
- `element` Character: identifier in the first column of `x`
- `method` Character: method to plot results; choose between spearman, pearson, gsea or rankProduct (the last one is only available if `element = NULL`)
- `n` Numeric: number of top and bottom genes to label (if a vector of two numbers is given, the first and second numbers will be used as the number of top and bottom genes to label, respectively); only used if `element = NULL`
- `showMetadata` Boolean: show available metadata information instead of identifiers (if available)? Only used if `element = NULL`
- `plotNonRankedPerturbations` Boolean: plot non-ranked data in grey? Only used if `element = NULL`
- `alpha` Numeric: transparency; only used if `element = NULL`
- `genes` Character: when plotting gene set enrichment analysis (GSEA), plot most up-regulated genes (genes = "top"), most down-regulated genes (genes = "bottom") or both (genes = "both"); only used if `method = "gsea"` and `geneset = NULL`
- `...` Extra arguments currently not used
plot.referenceComparison

zscores  Data frame (GCTX z-scores) or character (respective filepath to load data from file)

title  Character: plot title (if NULL, the default title depends on the context; ignored when plotting multiple perturbations)

Value

Plot illustrating the reference comparison

See Also

Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getCMAPConditions(), getCMAPPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plotTargetingDrugsVSsimilarPerturbations(), prepareCMapPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Other functions related with the prediction of targeting drugs: as.table.referenceComparison(), listExpressionDrugSensitivityAssociation(), loadExpressionDrugSensitivityAssociation(), plotTargetingDrugsVSsimilarPerturbations(), predictTargetingDrugs()

Examples

# Example of a differential expression profile
data("diffExprStat")

## Not run:
# Download and load CMap perturbations to compare with
cellLine <- "HepG2"
cmapMetadataKD <- filterCMapMetadata(
  "cmapMetadata.txt", cellLine=cellLine,
  perturbationType="Consensus signature from shRNAs targeting the same gene")
cmapPerturbationsKD <- prepareCMapPerturbations(
  cmapMetadataKD, "cmapZscores.gctx", "cmapGeneInfo.txt", loadZscores=TRUE)

## End(Not run)

# Rank similar CMap perturbations
compareKD <- rankSimilarPerturbations(diffExprStat, cmapPerturbationsKD)

# Plot ranked list of CMap perturbations
plot(compareKD, method="spearman")
plot(compareKD, method="spearman", n=c(7, 3))
plot(compareKD, method="pearson")
plot(compareKD, method="gsea")

# Plot results for a single perturbation
pert <- compareKD[[1, 1]]
plot(compareKD, pert, method="spearman", zscores=cmapPerturbationsKD)
plot(compareKD, pert, method="pearson", zscores=cmapPerturbationsKD)
plot(compareKD, pert, method="gsea", zscores=cmapPerturbationsKD)
# Predict targeting drugs based on a given differential expression profile

gdsc <- loadExpressionDrugSensitivityAssociation("GDSC 7")
predicted <- predictTargetingDrugs(diffExprStat, gdsc)

# Plot ranked list of targeting drugs
plot(predicted, method="spearman")
plot(predicted, method="spearman", n=c(7, 3))
plot(predicted, method="pearson")
plot(predicted, method="gsea")

# Plot results for a single targeting drug

drug <- predicted$compound[[4]]
plot(predicted, drug, method="spearman")
plot(predicted, drug, method="pearson")
plot(predicted, drug, method="gsea")

plotDrugSetEnrichment  *Plot drug set enrichment*

**Description**

Plot drug set enrichment

**Usage**

```
plotDrugSetEnrichment(
  sets,  
  stats,  
  col = "rankProduct_rank",  
  selectedSets = NULL,  
  keyColSets = NULL,  
  keyColStats = NULL
)
```

**Arguments**

- **sets**: Named list of characters: named sets containing compound identifiers (obtain drug sets by running `prepareDrugSets()`)
- **stats**: Named numeric vector or either a `similarPerturbations` or a `targetingDrugs` object (obtained after running `rankSimilarPerturbations` or `predictTargetingDrugs`, respectively)
- **col**: Character: name of the column to use for statistics (only required if class of `stats` is either `similarPerturbations` or `targetingDrugs`)
- **selectedSets**: Character: drug sets to plot (if `NULL`, plot all)
- **keyColSets**: Character: column from `sets` to compare with column `keyColStats` from `stats`; automatically selected if `NULL`
- **keyColStats**: Character: column from `stats` to compare with column `keyColSets` from `sets`; automatically selected if `NULL"
**plotESplot**

**Value**

List of GSEA plots per drug set

**See Also**

Other functions for drug set enrichment analysis: `analyseDrugSetEnrichment()`, `loadDrugDescriptors()`, `prepareDrugSets()`

**Examples**

```r
descriptors <- loadDrugDescriptors()
drugSets <- prepareDrugSets(descriptors)

# Analyse drug set enrichment in ranked targeting drugs for a differential
# expression profile
data("diffExprStat")
gdsc <- loadExpressionDrugSensitivityAssociation("GDSC")
predicted <- predictTargetingDrugs(diffExprStat, gdsc)
plotDrugSetEnrichment(drugSets, predicted)
```

---

**plotESplot**

*Render GSEA enrichment plot*

**Description**

Render GSEA enrichment plot

**Usage**

```r
plotESplot(enrichmentScore, gseaStat, compact = FALSE)
```

**Value**

GSEA enrichment plot

---

**plotGSEA**

*Plot gene set enrichment analysis (GSEA)*

**Description**

Plot gene set enrichment analysis (GSEA)
plotGSEA

Usage

plotGSEA(
  stats,
  geneset,
  genes = c("both", "top", "bottom"),
  title = "GSEA plot",
  gseaParam = 1,
  compact = FALSE
)

Arguments

stats Named numeric vector: statistics
genes Character: when plotting gene set enrichment analysis (GSEA), plot most up-regulated genes (genes = "top"), most down-regulated genes (genes = "bottom") or both (genes = "both"); only used if method = "gsea" and geneset = NULL

title Character: plot title (if NULL, the default title depends on the context; ignored when plotting multiple perturbations)
gseaParam Numeric: GSEA-like parameter
compact Boolean: render a compact version of the GSEA plot?

Value

Grid of plots illustrating a GSEA plot

plotMetricDistribution

Plot metric distribution

Description

Plot metric distribution

Usage

plotMetricDistribution(stat, compact = FALSE)

Value

Metric distribution plot
**plotSingleCorr**

Render scatter plot to show a single relationship

### Description
Render scatter plot to show a single relationship

### Usage
plotSingleCorr(perturbation, ylabel, diffExprGenes, title = NULL)

### Arguments
- **perturbation**: List of named numeric vectors containing the differential expression profile score per gene for a perturbation; each perturbation of the list will be rendered with a different colour
- **ylabel**: Character: Y axis label
- **diffExprGenes**: Named numeric vector
- **title**: Character: plot title (if NULL, the default title depends on the context; ignored when plotting multiple perturbations)

### Value
Scatter plot

**plotTargetingDrugsVSsimilarPerturbations**

Plot similar perturbations against predicted targeting drugs

### Description
Plot similar perturbations against predicted targeting drugs

### Usage
plotTargetingDrugsVSsimilarPerturbations(targetingDrugs, similarPerturbations, column, labelBy = "pert_iname", quantileThreshold = 0.25, showAllScores = FALSE, keyColTargetingDrugs = NULL, keyColSimilarPerturbations = NULL)
plotTargetingDrugsVSsimilarPerturbations

Arguments

targetingDrugs   targetingDrugs object
similarPerturbations   similarPerturbations object
column   Character: column to plot (must be available in both databases)
labelBy   Character: column in as.table(similarPerturbations) or as.table(targetingDrugs) to be used for labelling
quantileThreshold   Numeric: quantile (between 0 and 1) to highlight values of interest
showAllScores   Boolean: show all scores? If FALSE, only the best score per compound will be plotted
keyColTargetingDrugs   Character: column from targetingDrugs to compare with column keyColSimilarPerturbations from similarPerturbations; automatically selected if NULL
keyColSimilarPerturbations   Character: column from similarPerturbations to compare with column keyColTargetingDrugs from targetingDrugs; automatically selected if NULL

Value

ggplot2 plot

See Also

Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getCMapConditions(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plot.referenceComparison(), prepareCMapPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Other functions related with the prediction of targeting drugs: as.table.referenceComparison(), listExpressionDrugSensitivityAssociation(), loadExpressionDrugSensitivityAssociation(), plot.referenceComparison(), predictTargetingDrugs()

Examples

# Rank similarity against CMap compound perturbations
similarPerts <- rankSimilarPerturbations(diffExprStat, cmapPerturbationsCompounds)

# Predict targeting drugs
gdsc <- loadExpressionDrugSensitivityAssociation("GDSC 7")
predicted <- predictTargetingDrugs(diffExprStat, gdsc)
plotTargetingDrugsVSsimilarPerturbations(predicted, similarPerts, "spearman_rank")
**predictTargetingDrugs**  
*Predict targeting drugs*

**Description**

Identify compounds that may target the phenotype associated with a user-provided differential expression profile by comparing such against a correlation matrix of gene expression and drug sensitivity.

**Usage**

```r
predictTargetingDrugs(
  input,
  expressionDrugSensitivityCor,
  method = c("spearman", "pearson", "gsea"),
  geneSize = 150,
  isDrugActivityDirectlyProportionalToSensitivity = NULL,
  threads = 1,
  chunkGiB = 1,
  verbose = FALSE
)
```

**Arguments**

- **input**
  Named numeric vector of differentially expressed genes whose names are gene identifiers and respective values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics); or character of gene symbols composing a gene set that is tested for enrichment in reference data (only used if method includes gsea)

- **expressionDrugSensitivityCor**
  Matrix or character: correlation matrix of gene expression (rows) and drug sensitivity (columns) across cell lines or path to file containing such data; see `loadExpressionDrugSensitivityAssociation()`.

- **method**
  Character: comparison method (spearman, pearson or gsea; multiple methods may be selected at once)

- **geneSize**
  Numeric: number of top up-/down-regulated genes to use as gene sets to test for enrichment in reference data; if a 2-length numeric vector, the first index is the number of top up-regulated genes and the second index is the number of down-regulated genes used to create gene sets; only used if method includes gsea and if input is not a gene set

- **isDrugActivityDirectlyProportionalToSensitivity**
  Boolean: are the values used for drug activity directly proportional to drug sensitivity? If NULL, the argument expressionDrugSensitivityCor must have a non-NULL value for attribute isDrugActivityDirectlyProportionalToSensitivity.

- **threads**
  Integer: number of parallel threads
prepareCMapPerturbations

chunkGiB

Numeric: if second argument is a path to an HDF5 file (.h5 extension), that file is loaded and processed in chunks of a given size in gibibytes (GiB); lower values decrease peak RAM usage (see details below)

verbose

Boolean: print additional details?

Value

Data table with correlation and/or GSEA score results

Process data by chunks

If a file path to a valid HDF5 (.h5) file is provided instead of a data matrix, that file can be loaded and processed in chunks of size chunkGiB, resulting in decreased peak memory usage.

The default value of 1 GiB (1 GiB = 1024^3 bytes) allows loading chunks of ~10000 columns and 14000 rows (10000 * 14000 * 8 bytes / 1024^3 = 1.04 GiB).

GSEA score

When method = "gsea", weighted connectivity scores (WTCS) are calculated (https://clue.io/connectopedia/cmap_algorithms).

See Also

Other functions related with the prediction of targeting drugs: as.table.referenceComparison(), listExpressionDrugSensitivityAssociation(), loadExpressionDrugSensitivityAssociation(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations()

Examples

# Example of a differential expression profile
data("diffExprStat")

# Load expression and drug sensitivity association derived from GDSC data
gdsc <- loadExpressionDrugSensitivityAssociation("GDSC 7")

# Predict targeting drugs on a differential expression profile
predictTargetingDrugs(diffExprStat, gdsc)

prepareCMapPerturbations

Prepare CMap perturbation data

Description

Prepare CMap perturbation data
prepareCMapPerturbations

Usage

prepareCMapPerturbations(
    metadata,
    zscores,
    geneInfo,
    compoundInfo = NULL,
    ...,
    loadZscores = FALSE
)

Arguments

metadata Data frame (CMap metadata) or character (respective filepath to load data from file)
zscores Data frame (GCTX z-scores) or character (respective filepath to load data from file)
geneInfo Data frame (CMap gene info) or character (respective filepath to load data from file)
compoundInfo Data frame (CMap compound info) or character (respective filepath to load data from file)
... Arguments passed on to filterCMapMetadata
cellLine Character: cell line (if NULL, all values are loaded)
timepoint Character: timepoint (if NULL, all values are loaded)
dosage Character: dosage (if NULL, all values are loaded)
perturbationType Character: type of perturbation (if NULL, all perturbation types are loaded)
loadZscores Boolean: load matrix of perturbation z-scores? Not recommended in systems with less than 30GB of RAM; if FALSE, downstream functions will load and process the file directly chunk by chunk, resulting in a lower memory footprint

Value

CMap perturbation data attributes and filename

See Also

Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getCMapConditions(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plot.referenceComparison(), plotTargetingDrugsVSSimilarPerturbations(), print.similarPerturbations(), rankSimilarPerturbations()

Examples

metadata <- loadCMapData("cmapMetadata.txt", "metadata")
metadata <- filterCMapMetadata(metadata, cellLine="HepG2")
## Not run:
prepareCMapPerturbations(metadata, "cmapZscores.gctx", "cmapGeneInfo.txt")
prepareDrugSets

Prepares drug sets from a table with compound descriptors

Description

Create a list of drug sets for each character and numeric column. For each character column, drugs are split across that column’s unique values (see argument maxUniqueElems). For each numeric column, drugs are split across evenly-distributed bins.

Usage

```r
prepareDrugSets(
  table,
  id = 1,
  maxUniqueElems = 15,
  maxBins = 15,
  k = 5,
  minPoints = NULL
)
```

Arguments

- `table` Data frame: drug descriptors
- `id` Integer or character: index or name of the identifier column
- `maxUniqueElems` Numeric: ignore character columns with more unique elements than maxUniqueElems
- `maxBins` Numeric: maximum number of bins for numeric columns
- `k` Numeric: constant; the higher the constant, the smaller the bin size (check minpts)
- `minPoints` Numeric: minimum number of points in a bin (if NULL, the minimum number of points is the number of non-missing values divided by maxBins divided by k)

Value

Named list of characters: named drug sets with respective compound identifiers as list elements

See Also

Other functions for drug set enrichment analysis: `analyseDrugSetEnrichment()`, `loadDrugDescriptors()`, `plotDrugSetEnrichment()`

Examples

```r
descriptors <- loadDrugDescriptors("NCI60")
prepareDrugSets(descriptors)
```
prepareENCODEgeneExpression

Load ENCODE gene expression data

Description

Load ENCODE gene expression data

Usage

prepareENCODEgeneExpression(samples)

Arguments

samples List of loaded ENCODE samples

Value

Data frame containing gene read counts

See Also

convertGeneIdentifiers()

Other functions related with using ENCODE expression data: downloadENCODEknockdownMetadata(), loadENCODEsamples(), performDifferentialExpression()

Examples

if (interactive()) {
  # Load ENCODE metadata for a specific cell line and gene
  cellLine <- "HepG2"
  gene <- "EIF4G1"
  ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

  # Load samples based on filtered ENCODE metadata
  ENCODEsamples <- loadENCODEsamples(ENCODEmetadata)[[1]]

  prepareENCODEgeneExpression(ENCODEsamples)
}
prepareExpressionDrugSensitivityAssociation

Prepare gene expression and drug sensitivity correlation matrix

Description
Prepare gene expression and drug sensitivity correlation matrix

Usage
prepareExpressionDrugSensitivityAssociation(
  dataset = c("GDSC 7", "CTRP 2.1", "NCI60"),
  method = "spearman"
)

Arguments

  dataset  Character: dataset to use (CTRP, GDSC or NCI60)
  method   Character: correlation method to use between gene expression and drug sensitivity

Details
If path directs to non-existing files, data will be downloaded.

Value
Correlation matrix between gene expression and drug sensitivity

prepareGSEAgenesets

Prepare GSEA gene sets

Description
Prepare GSEA gene sets

Usage
prepareGSEAgenesets(input, geneSize)
prepareSetsCompoundInfo

Arguments

- **input**
  Named numeric vector of differentially expressed genes whose names are gene identifiers and respective values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics); or character of gene symbols composing a gene set that is tested for enrichment in reference data (only used if method includes gsea)

- **geneSize**
  Numeric: number of top up-/down-regulated genes to use as gene sets to test for enrichment in reference data; if a 2-length numeric vector, the first index is the number of top up-regulated genes and the second index is the number of down-regulated genes used to create gene sets; only used if method includes gsea and if input is not a gene set

Value

List of gene sets

prepareSetsCompoundInfo

*Get drug sets’ compound info*

Description

Get drug sets’ compound info

Usage

prepareSetsCompoundInfo(sets)

Arguments

- **sets**
  Named list of characters: named sets containing compound identifiers (obtain drug sets by running prepareDrugSets())

Value

List containing drug sets’ compound info
prepareStatsCompoundInfo

*Prepare stats’ compound information*

**Description**

Prepare stats’ compound information

**Usage**

```
prepareStatsCompoundInfo(stats)
```

**Arguments**

- **stats**
  Named numeric vector or either a `similarPerturbations` or a `targetingDrugs` object (obtained after running `rankSimilarPerturbations` or `predictTargetingDrugs`, respectively)

**Value**

List containing stats’ compound info

prepareWordBreak

Create word break opportunities (for HTML) using given characters

**Description**

Create word break opportunities (for HTML) using given characters

**Usage**

```
prepareWordBreak(
  str,
  pattern = c(".", "-", "\", "/", ",", ",", ",", "+", "+"),
  html = TRUE
)
```

**Arguments**

- **str**
  Character: text

- **pattern**
  Character: pattern(s) of interest to be used as word break opportunities

- **html**
  Boolean: convert to HTML?

**Value**

String containing HTML elements
print.similarPerturbations

Print a similarPerturbations object

Description

Print a similarPerturbations object

Usage

## S3 method for class 'similarPerturbations'
print(x, perturbation = NULL, ...)

Arguments

x similarPerturbations object
perturbation Character (perturbation identifier) or numeric (perturbation index)
... Extra parameters passed to print

Value

Information on perturbationChanges object or on specific perturbations

See Also

Other functions related with the ranking of CMap perturbations: as.table.referenceComparison(), filterCMapMetadata(), getCMapConditions(), getCMapPerturbationTypes(), loadCMapData(), loadCMapZscores(), parseCMapID(), plot.perturbationChanges(), plot.referenceComparison(), plotTargetingDrugsVSsimilarPerturbations(), prepareCMapPerturbations(), rankSimilarPerturbations()

processByChunks

Process data by chunks

Description

Process data by chunks

Usage

processByChunks(
  data,
  FUN,
  num,
  ...,
  threads = 1,
  chunkGiB = 1,
  verbose = FALSE
)
Arguments

- **data**: Character containing a HDF5 file path (allowing partial loading) or data matrix (processed as single chunk if data matrix)
- **FUN**: Function: function to run for each chunk
- **num**: Numeric: numbers of methods to run per chunk
- **...**: Arguments passed to `FUN`
- **threads**: Integer: number of parallel threads
- **chunkGiB**: Numeric: size (in gibibytes) of chunks to load reference file; only if argument `reference` is a file path
- **verbose**: Boolean: print additional details?

Value

Results of running `FUN`

Note

All rows from file are currently loaded when processing chunks.

---

**processIds**

Return a subset of requested GCTX row/column ids out of the universe of all ids

Description

Return a subset of requested GCTX row/column ids out of the universe of all ids

Usage

```
processIds(ids, all_ids, type = "rid")
```

Arguments

- **ids**: vector of requested ids. If NULL, no subsetting is performed
- **all_ids**: vector of universe of ids
- **type**: flag indicating the type of ids being processed

Details

This is a low-level helper function which most users will not need to access directly

Value

A list with the following elements:
- **ids**: a character vector of the processed ids
- **idx**: an integer list of their corresponding indices in `all_ids`
rankAgainstReference

Source

https://github.com/cmap/cmapR

See Also

Other GCTX parsing functions: fix.datatypes(), readGctxIds(), readGctxMeta()

Description

Compare multiple methods and rank against reference accordingly

Usage

rankAgainstReference(
  input, reference,
  method = c("spearman", "pearson", "gsea"),
  geneSize = 150,
  cellLines = NULL,
  cellLineMean = "auto",
  rankByAscending = TRUE,
  rankPerCellLine = FALSE,
  threads = 1,
  chunkGiB = 1,
  verbose = FALSE
)

Arguments

input Named numeric vector of differentially expressed genes whose names are gene identifiers and respective values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics); or character of gene symbols composing a gene set that is tested for enrichment in reference data (only used if method includes gsea)

reference Data matrix or character object with file path to CMap perturbations (see prepareCMapPerturbations()) or gene expression and drug sensitivity association (see loadExpressionDrugSensitivityAssociation())

method Character: comparison method (spearman, pearson or gsea; multiple methods may be selected at once)

geneSize Numeric: number of top up-/down-regulated genes to use as gene sets to test for enrichment in reference data; if a 2-length numeric vector, the first index is the number of top up-regulated genes and the second index is the number of down-regulated genes used to create gene sets; only used if method includes gsea and if input is not a gene set
cellLines  Integer: number of unique cell lines

cellLineMean  Boolean: add rows with the mean of method across cell lines? If cellLineMean = "auto" (default), rows will be added when data for more than one cell line is available.

rankByAscending  Boolean: rank values based on their ascending (TRUE) or descending (FALSE) order?

rankPerCellLine  Boolean: rank results based on both individual cell lines and mean scores across cell lines (TRUE) or based on mean scores alone (FALSE)? If cellLineMean = FALSE, individual cell line conditions are always ranked.

threads  Integer: number of parallel threads

chunkGiB  Numeric: if second argument is a path to an HDF5 file (.h5 extension), that file is loaded and processed in chunks of a given size in gibibytes (GiB); lower values decrease peak RAM usage (see details below)

verbose  Boolean: print additional details?

**Value**

Data table with correlation and/or GSEA score results

**Process data by chunks**

If a file path to a valid HDF5 (.h5) file is provided instead of a data matrix, that file can be loaded and processed in chunks of size chunkGiB, resulting in decreased peak memory usage.

The default value of 1 GiB (1 GiB = 1024^3 bytes) allows loading chunks of ~10000 columns and 14000 rows (10000 * 14000 * 8 bytes / 1024^3 = 1.04 GiB).

**GSEA score**

When method = "gsea", weighted connectivity scores (WTCS) are calculated (https://clue.io/connectopedia/cmap_algorithms).

---

**rankColumns**  Rank columns in a dataset

**Description**

Rank columns in a dataset

**Usage**

rankColumns(table, rankingInfo, rankByAscending = TRUE, sort = FALSE)
Arguments

- **table**: Data table: data; first column must be identifiers
- **rankingInfo**: Data table: boolean values of which rows to rank based on columns (column names to be ranked must exactly match those available in argument `table`); first column must be identifiers
- **rankByAscending**: Boolean: rank values based on their ascending (TRUE) or descending (FALSE) order?
- **sort**: Boolean: sort data based on rank product’s rank (if multiple methods are available) or by available ranks

Details

The rank product’s rank is calculated if more than one method is ranked.

Value

Data table with the contents of `table` and extra columns with respective rankings

Note

The first column of `data` and `rankingInfo` must contain common identifiers.

---

`rankSimilarPerturbations`

*Rank differential expression profile against CMap perturbations by similarity*

Description

Compare differential expression results against CMap perturbations.

Usage

```r
rankSimilarPerturbations(
  input,
  perturbations,
  method = c("spearman", "pearson", "gsea"),
  geneSize = 150,
  cellLineMean = "auto",
  rankPerCellLine = FALSE,
  threads = 1,
  chunkGiB = 1,
  verbose = FALSE
)
```
rankSimilarPerturbations

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>Named numeric vector of differentially expressed genes whose names are gene identifiers and respective values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics); or character of gene symbols composing a gene set that is tested for enrichment in reference data (only used if method includes gsea)</td>
</tr>
<tr>
<td>perturbations</td>
<td>perturbationChanges object: CMap perturbations (check prepareCMapPerturbations())</td>
</tr>
<tr>
<td>method</td>
<td>Character: comparison method (spearman, pearson or gsea; multiple methods may be selected at once)</td>
</tr>
<tr>
<td>geneSize</td>
<td>Numeric: number of top up-/down-regulated genes to use as gene sets to test for enrichment in reference data; if a 2-length numeric vector, the first index is the number of top up-regulated genes and the second index is the number of down-regulated genes used to create gene sets; only used if method includes gsea and if input is not a gene set</td>
</tr>
<tr>
<td>cellLineMean</td>
<td>Boolean: add rows with the mean of method across cell lines? If cellLineMean = &quot;auto&quot; (default), rows will be added when data for more than one cell line is available.</td>
</tr>
<tr>
<td>rankPerCellLine</td>
<td>Boolean: rank results based on both individual cell lines and mean scores across cell lines (TRUE) or based on mean scores alone (FALSE)? If cellLineMean = FALSE, individual cell line conditions are always ranked.</td>
</tr>
<tr>
<td>threads</td>
<td>Integer: number of parallel threads</td>
</tr>
<tr>
<td>chunkGiB</td>
<td>Numeric: if second argument is a path to an HDF5 file (.h5 extension), that file is loaded and processed in chunks of a given size in gibibytes (GiB); lower values decrease peak RAM usage (see details below)</td>
</tr>
<tr>
<td>verbose</td>
<td>Boolean: print additional details?</td>
</tr>
</tbody>
</table>

Value

Data table with correlation and/or GSEA score results

Process data by chunks

If a file path to a valid HDF5 (.h5) file is provided instead of a data matrix, that file can be loaded and processed in chunks of size chunkGiB, resulting in decreased peak memory usage.

The default value of 1 GiB (1 GiB = 1024^3 bytes) allows loading chunks of ~10000 columns and 14000 rows (10000 * 14000 * 8 bytes / 1024^3 = 1.04 GiB).

GSEA score

When method = "gsea", weighted connectivity scores (WTCS) are calculated (https://clue.io/connectopedia/cmap_algorithms).
See Also
Other functions related with the ranking of CMap perturbations: \texttt{as.table.referenceComparison()}, \texttt{filterCMapMetadata()}, \texttt{getCMapConditions()}, \texttt{getCMapPerturbationTypes()}, \texttt{loadCMapData()}, \texttt{loadCMapZscores()}, \texttt{parseCMapID()}, \texttt{plot.perturbationChanges()}, \texttt{plot.referenceComparison()}, \texttt{plotTargetingDrugsVSsimilarPerturbations()}, \texttt{prepareCMapPerturbations()}, \texttt{print.similarPerturbations()}

Examples

# Example of a differential expression profile
data("diffExprStat")

## Not run:
# Download and load CMap perturbations to compare with
celline <- c("HepG2", "HUH7")
cmapMetadataCompounds <- filterCMapMetadata(
  "cmapMetadata.txt", cellline=celline, timepoint="24 h",
  dosage="5 \mu M", perturbationType="Compound")
cmapPerturbationsCompounds <- prepareCMapPerturbations(
  cmapMetadataCompounds, "cmapZscores.gctx", "cmapGeneInfo.txt",
  "cmapCompoundInfo_drugs.txt", loadZscores=TRUE)

## End(Not run)
perturbations <- cmapPerturbationsCompounds

# Rank similar CMap perturbations (by default, Spearman's and Pearson's
# correlation are used, as well as GSEA with the top and bottom 150 genes of
# the differential expression profile used as reference)
rankSimilarPerturbations(diffExprStat, perturbations)

# Rank similar CMap perturbations using only Spearman's correlation
rankSimilarPerturbations(diffExprStat, perturbations, method="spearman")

---

\texttt{readGctxIds} \hspace{1cm} \textit{Read GCTX row or column ids}

\textbf{Description}

Read GCTX row or column ids

\textbf{Usage}

\texttt{readGctxIds(gctx\_path, dimension = "row")}

\textbf{Arguments}

- \texttt{gctx\_path} \hspace{1cm} path to the GCTX file
- \texttt{dimension} \hspace{1cm} which ids to read (row or column)
readGctxMeta

Value

a character vector of row or column ids from the provided file

Source

https://github.com/cmap/cmapR

See Also

Other GCTX parsing functions: fix.datatypes(), processIds(), readGctxMeta()

---

readGctxMeta  
*Parse row or column metadata from GCTX files*

Description

Parse row or column metadata from GCTX files

Usage

```r
readGctxMeta(
  gctx_path,
  dimension = "row",
  ids = NULL,
  set_annot_rownames = TRUE
)
```

Arguments

- `gctx_path`: the path to the GCTX file
- `dimension`: which metadata to read (row or column)
- `ids`: a character vector of a subset of row/column ids for which to read the metadata
- `set_annot_rownames`: a boolean indicating whether to set the rownames attribute of the returned data.frame to the corresponding row/column ids.

Value

a data.frame of metadata

Source

https://github.com/cmap/cmapR

See Also

Other GCTX parsing functions: fix.datatypes(), processIds(), readGctxIds()
**stripStr**

*Strip non-alpha-numeric characters from a string*

**Description**
Strip non-alpha-numeric characters from a string

**Usage**

```r
stripStr(str)
```

**Arguments**

- `str` : Character

**Value**
Character without non-alphanumeric values

---

**subsetData**

*Subset data by rows and/or columns*

**Description**
Subset data by rows and/or columns

**Usage**

```r
subsetData(x, i, j, rowAttr, colAttr, nargs, ...)
```

**Value**
Subset data

---

**subsetDim**

*Subset rows or columns based on a given index*

**Description**
Subset rows or columns based on a given index

**Usage**

```r
subsetDim(k, dims, nargs, areCols = TRUE)
```

**Value**
Subset rows/columns
Description

Do a robust data.frame subset to a set of ids

Usage

subsetToIds(df, ids)

Arguments

df   data.frame to subset
ids  the ids to subset to

Value

a subset version of df

Source

https://github.com/cmap/cmapR
Index

* **GCTX parsing functions**
  fix.datatypes, 22
  processIds, 58
  readGctxIds, 63
  readGctxMeta, 64
* **functions for drug set enrichment analysis**
  analyseDrugSetEnrichment, 5
  loadDrugDescriptors, 33
  plotDrugSetEnrichment, 44
  prepareDrugSets, 52
* **functions for gene expression pre-processing**
  convertGeneIdentifiers, 14
* **functions related with the prediction of targeting drugs**
  as.table.referenceComparison, 6
  listExpressionDrugSensitivityAssociation, 29
  loadExpressionDrugSensitivityAssociation, 35
  plot.referenceComparison, 42
  plotTargetingDrugsVSsimilarPerturbations, 47
  predictTargetingDrugs, 49
* **functions related with the ranking of CMap perturbations**
  as.table.referenceComparison, 6
  filterCMapMetadata, 20
  getCMapConditions, 23
  getCMAPerturbationTypes, 24
  loadCMapData, 30
  loadCMapZscores, 31
  parseCMapID, 38
  plot.perturbationChanges, 40
  plot.referenceComparison, 42
  plotTargetingDrugsVSsimilarPerturbations, 47
  prepareCMapPerturbations, 50
  print.similarPerturbations, 57
  rankSimilarPerturbations, 61
* **functions related with using ENCODE expression data**
  downloadENCODEknockdownMetadata, 19
  loadENCODExamples, 34
  performDifferentialExpression, 38
  prepareENCODExgeneExpression, 53
* **internal**
  .plotBubbles, 4
  .prepareNavPage, 4
  .traceInList, 4
  calculateCellLineMean, 7
  calculateEvenlyDistributedBins, 8
  checkColnames, 9
  chunkColumns, 9
  closeOpenHandles, 10
  cmapMetadata, 10
  cmapPerturbationsCompounds, 10
  cmapPerturbationsKD, 11
  compareQuantile, 12
  compareWithAllMethods, 12
  counts, 15
  diffExprStat, 18
  downloadIfNotFound, 19
  ENCODEmetadata, 20
  findIntersectingCompounds, 21
  fix.datatypes, 22
  GCT-class, 22
  getENCODExcontrols, 25
  HTMLfast, 25
  loadCTRPgeneExpression, 32
  loadENCODExsample, 34
  loadNCI60drugSensitivity, 36
  matchStatsWithDrugSetsID, 37
  performGSEA, 39
  plotGSEplot, 45
  plotGSEA, 45
  plotMetricDistribution, 46
plotSingleCorr, 47
prepareExpressionDrugSensitivityAssociation, 54
prepareGSEAgenesets, 54
prepareSetsCompoundInfo, 55
prepareStatsCompoundInfo, 56
prepareWordBreak, 57
processByChunks, 57
processIds, 58
rankAgainstReference, 59
rankColumns, 60
readGctxIds, 63
readGctxMeta, 64
stripStr, 65
subsetData, 65
subsetDim, 65
subsetToIds, 66
visual interface functions
cTRAP, 16
launchCMapDataLoader, 26
launchDiffExprLoader, 27
launchDrugSetEnrichmentAnalyser, 27
launchMetadataViewer, 28
launchResultPlotter, 29
data.frame, 9, 66
diffExprStat, 18
dim.expressionDrugSensitivityAssociation
(dimnames.expressionDrugSensitivityAssociation), 18
dim.perturbationChanges
(plot.perturbationChanges), 40
dimnames.expressionDrugSensitivityAssociation, 18
dimnames.perturbationChanges
(plot.perturbationChanges), 40
downloadENCODEknockdownMetadata, 19, 35, 39, 53
downloadIfNotFound, 19
ENCODEmetadata, 20
fgsea::fgseaSimple, 5
filterCMapMetadata, 7, 21, 24, 30, 31, 38, 41, 43, 48, 51, 57, 63
findIntersectingCompounds, 21
fix.datatypes, 22, 59, 64
GCT-class, 22
getCMapConditions, 7, 21, 23, 24, 30, 31, 38, 41, 43, 48, 51, 57, 63
getENCODEcontrols, 25
HTMLfast, 25
listExpressionDrugSensitivityAssociation, 7, 29, 35, 36, 43, 48, 50
launchCMapDataLoader, 17, 26, 27–29
launchDiffExprLoader, 17, 26, 27, 28, 29
launchDrugSetEnrichmentAnalyser, 17, 26, 27, 28, 29
launchMetadataViewer, 17, 26–28, 29
launchResultPlotter, 17, 26–28, 29
listExpressionDrugSensitivityAssociation, 7, 29, 35, 36, 43, 48, 50
INDEX

loadCMapData, 7, 21, 24, 30, 31, 38, 41, 43, 48, 51, 57, 63
loadCMapZscores, 7, 21, 24, 30, 31, 38, 41, 43, 48, 51, 57, 63
loadCTRPcompoundInfo, 32
loadCTRPdrugSensitivity, 32
loadCTRPgeneExpression, 32
loadDrugDescriptors, 6, 33, 45, 52
loadENCODEsample, 34
loadENCODEsamples, 19, 34, 39, 53
loadExpressionDrugSensitivityAssociation, 7, 13, 30, 35, 43, 48–50, 59
loadGDSC7cellLineInfo, 32
loadGDSC7compoundInfo, 32
loadGDSC7drugSensitivity, 32
loadGDSC7file, 32
loadGDSC7geneExpression, 32
loadNCI60drugSensitivity, 36
loadNCI60geneExpression, 32
matchStatsWithDrugSetsID, 37
parseCMapID, 7, 21, 24, 30, 31, 38, 41, 43, 48, 51, 57, 63
performDifferentialExpression, 19, 35, 38, 53
performGSEA, 39
plot.perturbationChanges, 7, 21, 24, 30, 31, 38, 40, 43, 48, 51, 57, 63
plot.referenceComparison, 7, 21, 24, 30, 31, 36, 38, 41, 42, 48, 50, 51, 57, 63
plotDrugSetEnrichment, 6, 34, 44, 52
plotESplot, 45
plotGSEA, 45
plotMetricDistribution, 46
plotSingleCorr, 47
plotTargetingDrugsVSSimilarPerturbations, 7, 21, 24, 30, 31, 36, 38, 41, 43, 47, 50, 51, 57, 63
predictTargetingDrugs, 5, 7, 30, 36, 37, 42–44, 48, 49, 56
prepareCMapPerturbations, 7, 13, 21, 24, 30, 31, 38, 41, 43, 48, 50, 57, 59, 62, 63
prepareDrugSets, 6, 34, 45, 52
prepareENCODEgeneExpression, 19, 35, 39, 53
prepareExpressionDrugSensitivityAssociation, 54
prepareGSEAgeneSets, 54
prepareSetsCompoundInfo, 55
prepareStatsCompoundInfo, 56
prepareWordBreak, 56
print.similarPerturbations, 7, 21, 24, 30, 31, 38, 41, 43, 48, 51, 57, 63
processByChunks, 57
processIds, 22, 58, 64
rankAgainstReference, 59
rankColumns, 60
rankSimilarPerturbations, 5, 7, 17, 21, 24, 30, 31, 37, 38, 41–44, 48, 51, 56, 57, 61
readGctxIds, 22, 59, 63, 64
readGctxMeta, 22, 59, 64, 64
stripStr, 65
subsetData, 65
subsetDim, 65
subsetToIds, 66