

Package ‘cTRAP’

May 15, 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

URL <https://nuno-agostinho.github.io/cTRAP>,
<https://github.com/nuno-agostinho/cTRAP>

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, htr, limma, methods, parallel, pbapply, purrr, qs, R.utils, readxl, reshape2, rhdf5, rlang, scales, shiny (>= 1.7.0), shinycssloaders, stats, tibble, tools, utils

VignetteBuilder knitr

Language en-GB

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

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 2024-05-15

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 |

| | |
|---|----|
| launchCMapDataLoader | 26 |
| launchDiffExprLoader | 27 |
| launchDrugSetEnrichmentAnalyser | 27 |
| launchMetadataViewer | 28 |
| launchResultPlotter | 29 |
| listExpressionDrugSensitivityAssociation | 29 |
| loadCMapData | 30 |
| loadCMapZscores | 31 |
| loadCTRPgeneExpression | 32 |
| loadDrugDescriptors | 33 |
| loadENCODEsample | 34 |
| loadENCODEsamples | 34 |
| loadExpressionDrugSensitivityAssociation | 35 |
| loadNCI60drugSensitivity | 36 |
| matchStatsWithDrugSetsID | 37 |
| parseCMapID | 38 |
| performDifferentialExpression | 38 |
| performGSEA | 39 |
| plot.perturbationChanges | 40 |
| plot.referenceComparison | 42 |
| plotDrugSetEnrichment | 44 |
| plotESplot | 45 |
| plotGSEA | 45 |
| plotMetricDistribution | 46 |
| plotSingleCorr | 47 |
| plotTargetingDrugsVSSimilarPerturbations | 47 |
| predictTargetingDrugs | 49 |
| prepareCMapPerturbations | 50 |
| prepareDrugSets | 52 |
| prepareENCODegeneExpression | 53 |
| prepareExpressionDrugSensitivityAssociation | 54 |
| prepareGSEAgeneSets | 54 |
| prepareSetsCompoundInfo | 55 |
| prepareStatsCompoundInfo | 56 |
| prepareWordBreak | 56 |
| print.similarPerturbations | 57 |
| processByChunks | 57 |
| processIds | 58 |
| rankAgainstReference | 59 |
| rankColumns | 60 |
| rankSimilarPerturbations | 61 |
| readGctxIds | 63 |
| readGctxMeta | 64 |
| stripStr | 65 |
| subsetData | 65 |
| subsetDim | 65 |
| subsetToIds | 66 |

`.plotBubbles` *Plot packed bubbles*

Description

Plot packed bubbles

Usage

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

Arguments

| | |
|---------------------|--------------------------|
| <code>data</code> | Data to plot |
| <code>title</code> | Character: plot title |
| <code>colour</code> | 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

```
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 <code>prepareDrugSets()</code>) |
| stats | Named numeric vector or either a <code>similarPerturbations</code> or a <code>targetingDrugs</code> 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 <code>similarPerturbations</code> or <code>targetingDrugs</code>) |
| 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)? |

Value

Complete table with metadata based on a targetingDrugs object

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()`, `rankSimilarPerturbations()`

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

calculateCellLineMean *Calculate cell line mean*

Description

Calculate cell line mean

Usage

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

Arguments

| | |
|------------------------------|---|
| <code>data</code> | Data table: comparison against CMap data |
| <code>cellLine</code> | Character: perturbation identifiers as names and respective cell lines as values |
| <code>metadata</code> | Data table: data metadata |
| <code>rankPerCellLine</code> | Boolean: rank results based on both individual cell lines and mean scores across cell lines (TRUE) or based on mean scores alone (FALSE)? If <code>cellLineMean = FALSE</code> , 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

```
calculateEvenlyDistributedBins
```

Calculate evenly-distributed bins

Description

Calculate evenly-distributed bins

Usage

```
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 <code>length(x) / max.breaks</code> 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 | <i>Check whether test_names are columns in the data.frame</i> |
|---------------|---|

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 | <i>Assign columns into chunks</i> |
|--------------|-----------------------------------|

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 *Close open handles*

Description

Close open handles

Usage

```
closeOpenHandles()
```

Value

Closes all open identifiers

cmapMetadata *CMap metadata*

Description

CMap metadata obtained by running the following code:

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

cmapPerturbationsCompounds
CMap perturbations sample for small molecules

Description

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

```
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
```

```
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
```

cmapPerturbationsKD *CMap perturbations sample for knockdown experiments*

Description

CMap perturbations sample for knockdown experiments obtained by running the following code:

```
# 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(cmapPerturbationsKD, "metadata")
metadata$pert_idose <- gsub("\u00B5", "micro", metadata$pert_idose)
metadata$pert_dose_unit <- gsub("\u00B5", "micro", metadata$pert_dose_unit)
attr(cmapPerturbationsKD, "metadata") <- metadata
```

| | |
|-----------------|--|
| compareQuantile | <i>Compare vector against its quantile</i> |
|-----------------|--|

Description

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

Usage

```
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 | <i>Compare reference using all methods</i> |
|-----------------------|--|

Description

Compare reference using all methods

Usage

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

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).

`convertENSEMBLtoGeneSymbols`*Convert ENSEMBL gene identifiers to gene symbols*

Description

Convert ENSEMBL gene identifiers to gene symbols

Usage

```
convertENSEMBLtoGeneSymbols(  
  genes,  
  dataset = "hsapiens_gene_ensembl",  
  mart = "ensembl"  
)
```

Arguments

| | |
|----------------------|-------------------------------------|
| <code>genes</code> | Character: ENSEMBL gene identifiers |
| <code>dataset</code> | Character: biomaRt dataset name |
| <code>mart</code> | 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

```
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("ENSG0000012048", "ENSG0000083093", "ENSG0000141510",
           "ENSG0000051180")
convertGeneIdentifiers(genes)
convertGeneIdentifiers(genes, key="ENSEMBL", target="UNIPROT")

# Explicit species name to automatically look for its OrgDb database
sp <- "Homo sapiens"
genes <- c("ENSG0000012048", "ENSG0000083093", "ENSG0000141510",
           "ENSG0000051180")
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

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

| | |
|-------------------------------|--|
| <code>...</code> | Objects |
| <code>commonPath</code> | Character: path where to store data common to all sessions |
| <code>expire</code> | Character: days until a session expires (message purposes only) |
| <code>fileSizeLimitMiB</code> | Numeric: file size limit in MiB |
| <code>flowerURL</code> | Character: Flower REST API's URL (NULL to avoid using Celery/Flower backend) |
| <code>port</code> | The TCP port that the application should listen on. If the <code>port</code> is not specified, and the <code>shiny.port</code> option is set (with <code>options(shiny.port = XX)</code>), 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. |
| <code>host</code> | The IPv4 address that the application should listen on. Defaults to the <code>shiny.host</code> option, if set, or <code>"127.0.0.1"</code> 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) 12 000 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://github.com/nuno-agostinho/cTRAP>
- Report bugs at <https://github.com/nuno-agostinho/cTRAP/issues>

Other visual interface functions: `launchCMapDataLoader()`, `launchDiffExprLoader()`, `launchDrugSetEnrichmentAnal`, `launchMetadataViewer()`, `launchResultPlotter()`

| | |
|--------------|--|
| diffExprStat | <i>Differential expression's t-statistics sample</i> |
|--------------|--|

Description

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

```
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 | <i>Operations on expressionDrugSensitivityAssociation objects</i> |
|---|---|

Description

Operations on expressionDrugSensitivityAssociation objects

Usage

```
## 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

`downloadENCODEknockdownMetadata`*Download metadata for ENCODE knockdown experiments*

Description

Download metadata for ENCODE knockdown experiments

Usage

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

Arguments

| | |
|-----------------------|---|
| <code>cellLine</code> | Character: cell line |
| <code>gene</code> | Character: target gene |
| <code>file</code> | 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

```
downloadENCODEknockdownMetadata("HepG2", "EIF4G1")
```

`downloadIfNotFound`*Download data if given file is not found*

Description

Download data if given file is not found

Usage

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

Arguments

| | |
|-----------|--|
| link | Character: link to download file |
| file | Character: filepath |
| ask | Boolean: ask to download file? |
| toExtract | Character: files to extract (if NULL, extract all) |

Value

Download file if file is not found

| | |
|----------------|-------------------------------|
| ENCODEmetadata | <i>ENCODE metadata sample</i> |
|----------------|-------------------------------|

Description

ENCODE metadata sample obtained by running the following code:

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

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

| | |
|--------------------|-----------------------------|
| filterCMapMetadata | <i>Filter CMap metadata</i> |
|--------------------|-----------------------------|

Description

Filter CMap metadata

Usage

```
filterCMapMetadata(
  metadata,
  cellLine = NULL,
  timepoint = NULL,
  dosage = NULL,
  perturbationType = NULL
)
```

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\(\)](#), [plot.TargetingDrugsVSSimilarPerturbations\(\)](#), [prepareCMapPerturbations\(\)](#), [print.similarPerturbations\(\)](#), [rankSimilarPerturbations\(\)](#)

Examples

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

findIntersectingCompounds

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

`fix.datatypes`*Adjust the data types for columns of a meta data frame*

Description

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.

Usage

```
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

Source

<https://github.com/cmap/cmapR>

See Also

Other GCTX parsing functions: [processIds\(\)](#), [readGctxIds\(\)](#), [readGctxMeta\(\)](#)

`GCT-class`*An S4 class to represent a GCT object*

Description

The GCT class serves to represent annotated matrices. The `mat` slot contains said data and the `rdesc` and `cdesc` slots contain data frames with annotations about the rows and columns, respectively

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

getCMapConditions *List available conditions in CMap datasets*

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? |

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\(\)](#), [plot.TargetingDrugsVSsimilarPerturbations\(\)](#), [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\(\)](#), [getCMapConditions\(\)](#), [loadCMapData\(\)](#), [loadCMapZscores\(\)](#), [parseCMapID\(\)](#), [plot.perturbationChanges\(\)](#), [plot.referenceComparison\(\)](#), [plot.TargetingDrugsVSsimilarPerturbations\(\)](#), [prepareCMapPerturbations\(\)](#), [print.similarPerturbations\(\)](#), [rankSimilarPerturbations\(\)](#)

Examples

```
getCMapPerturbationTypes()
```

| | |
|-------------------|--|
| getENCODEcontrols | <i>Get experiments files for a given control</i> |
|-------------------|--|

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 | <i>Faster version of shiny::HTML</i> |
|----------|--------------------------------------|

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\(\)](#)

launchMetadataViewer *View metadata via a visual interface*

Description

View metadata via a visual interface

Usage

```
launchMetadataViewer(...)
```

Arguments

| | |
|-----|---------|
| ... | Objects |
|-----|---------|

Value

Metadata viewer (returns NULL)

See Also

Other visual interface functions: [cTRAP\(\)](#), [launchCMapDataLoader\(\)](#), [launchDiffExprLoader\(\)](#), [launchDrugSetEnrichmentAnalyser\(\)](#), [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

See Also

Other functions related with the prediction of targeting drugs: [as.table.referenceComparison\(\)](#), [loadExpressionDrugSensitivityAssociation\(\)](#), [plot.referenceComparison\(\)](#), [plotTargetingDrugsVSSimilarPerturbations\(\)](#), [predictTargetingDrugs\(\)](#)

Examples

```
listExpressionDrugSensitivityAssociation()
```

loadCMapData

Load CMap data

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 ranking of CMap perturbations: [as.table.referenceComparison\(\)](#), [filterCMapMetadata\(\)](#), [getCMapConditions\(\)](#), [getCMapPerturbationTypes\(\)](#), [loadCMapZScores\(\)](#), [parseCMapID\(\)](#), [plot.perturbationChanges\(\)](#), [plot.referenceComparison\(\)](#), [plotTargetingDrugsVSSimilarPerturbations\(\)](#), [prepareCMapPerturbations\(\)](#), [print.similarPerturbations\(\)](#), [rankSimilarPerturbations\(\)](#)

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 | <i>Load matrix of CMap perturbation's differential expression z-scores (optional)</i> |
|-----------------|---|

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\(\)](#)

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)

```

```
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")

```


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

loadDrugDescriptors *Load table with drug descriptors*

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

```
loadDrugDescriptors()
```

| | |
|------------------|---------------------------|
| loadENCODEsample | <i>Load ENCODE sample</i> |
|------------------|---------------------------|

Description

Load ENCODE sample

Usage

```
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 | <i>Load ENCODE samples</i> |
|-------------------|----------------------------|

Description

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

Usage

```
loadENCODEsamples(metadata, path = ".")
```

Arguments

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

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 |

| | |
|------------|---|
| 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\(\)](#), [plotTargetingDrugsVsimilarPer](#), [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

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

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

```
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\(\)](#), [plotTargetingDrugsVSsimilarP](#), [prepareCMapPerturbations\(\)](#), [print.similarPerturbations\(\)](#), [rankSimilarPerturbations\(\)](#)

Examples

```
id <- c("CVD001_HEPG2_24H:BRD-K94818765-001-01-0:4.8",
        "CVD001_HEPG2_24H:BRD-K96188950-001-04-5:4.3967",
        "CVD001_HUH7_24H:BRD-A14014306-001-01-1:4.1")
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

```
performDifferentialExpression(counts)
```

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()) {
  # Download ENCODE metadata for a specific cell line and gene
  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)

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

performGSEA

Perform GSEA

Description

Perform GSEA

Usage

```
performGSEA(pathways, stats)
```

Value

List with results of running GSEA

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 |
| ... | Extra arguments |
| 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\(\)](#), [getCMapConditions\(\)](#), [getCMapPerturbationTypes\(\)](#), [loadCMapData\(\)](#), [loadCMapZscores\(\)](#), [parseCMapID\(\)](#), [plot.referenceComparison\(\)](#), [plotTargetingDrugsVSSimilarPerturbations\(\)](#), [prepareCMapPerturbations\(\)](#), [print.similarPerturbations\(\)](#), [rankSimilarPerturbations\(\)](#)

Examples

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

compareKD <- rankSimilarPerturbations(diffExprStat, cmapPerturbationsKD)
EIF4G1knockdown <- grep("EIF4G1", compareKD[[1]], value=TRUE)
plot(cmapPerturbationsKD, EIF4G1knockdown, diffExprStat, method="spearman")
plot(cmapPerturbationsKD, EIF4G1knockdown, diffExprStat, method="pearson")
plot(cmapPerturbationsKD, EIF4G1knockdown, 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_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")
```

```
plot.referenceComparison
```

Plot data comparison

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

```
## 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"),
  ...,
  zscores = NULL,
  title = NULL
)
```

Arguments

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

| | |
|---------|---|
| 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 <code>prepareDrugSets()</code>) |
| stats | Named numeric vector or either a <code>similarPerturbations</code> or a <code>targetingDrugs</code> 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 <code>similarPerturbations</code> or <code>targetingDrugs</code>) |
| selectedSets | Character: drug sets to plot (if NULL, plot all) |
| keyColSets | Character: column from sets to compare with column <code>keyColStats</code> from stats; automatically selected if NULL |
| keyColStats | Character: column from stats to compare with column <code>keyColSets</code> from sets; automatically selected if NULL |

Value

List of GSEA plots per drug set

See Also

Other functions for drug set enrichment analysis: [analyseDrugSetEnrichment\(\)](#), [loadDrugDescriptors\(\)](#), [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)

plotDrugSetEnrichment(drugSets, predicted)
```

| | |
|------------|------------------------------------|
| plotESplot | <i>Render GSEA enrichment plot</i> |
|------------|------------------------------------|

Description

Render GSEA enrichment plot

Usage

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

Value

GSEA enrichment plot

| | |
|----------|---|
| plotGSEA | <i>Plot gene set enrichment analysis (GSEA)</i> |
|----------|---|

Description

Plot gene set enrichment analysis (GSEA)

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
)
```

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

```
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 |

| | |
|----------|---|
| 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 ($10000 * 14000 * 8 \text{ bytes} / 1024^3 = 1.04 \text{ GiB}$).

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\(\)](#), [plotTargetingDrugsVSimilarPerturbations\(\)](#)

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

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")
```

```
## End(Not run)
```

prepareDrugSets *Prepare 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

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

Arguments

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

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

```
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)
```

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 <code>similarPerturbations</code> or a <code>targetingDrugs</code> object (obtained after running <code>rankSimilarPerturbations</code> or <code>predictTargetingDrugs</code> , 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 | <i>Return a subset of requested GCTX row/column ids out of the universe of all ids</i> |
|------------|--|

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`

Source

<https://github.com/cmap/cmapR>

See Also

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

rankAgainstReference *Compare multiple methods and rank against reference accordingly*

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 <code>prepareCMapPerturbations()</code>) or gene expression and drug sensitivity association (see <code>loadExpressionDrugSensitivityAssociation()</code>) |
| 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 ($10000 * 14000 * 8 \text{ bytes} / 1024^3 = 1.04 \text{ GiB}$).

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

```
rankSimilarPerturbations(  
  input,  
  perturbations,  
  method = c("spearman", "pearson", "gsea"),  
  geneSize = 150,  
  cellLineMean = "auto",  
  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) |
| perturbations | perturbationChanges object: CMap perturbations (check prepareCMapPerturbations()) |
| 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 |
| 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: 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 ($10000 * 14000 * 8 \text{ bytes} / 1024^3 = 1.04 \text{ GiB}$).

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: [as.table.referenceComparison\(\)](#), [filterCMapMetadata\(\)](#), [getCMapConditions\(\)](#), [getCMapPerturbationTypes\(\)](#), [loadCMapData\(\)](#), [loadCMapZscores\(\)](#), [parseCMapID\(\)](#), [plot.perturbationChanges\(\)](#), [plot.referenceComparison\(\)](#), [plotTargetingDrugsVSsimilarPerturbations\(\)](#), [prepareCMapPerturbations\(\)](#), [print.similarPerturbations\(\)](#)

Examples

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

## Not run:
# Download and load CMap perturbations to compare with
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)

## 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")
```

readGctxIds

Read GCTX row or column ids

Description

Read GCTX row or column ids

Usage

```
readGctxIds(gctx_path, dimension = "row")
```

Arguments

| | |
|-----------|-----------------------------------|
| gctx_path | path to the GCTX file |
| dimension | which ids to read (row or column) |

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

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

Arguments

| | |
|---------------------------------|--|
| <code>gctx_path</code> | the path to the GCTX file |
| <code>dimension</code> | which metadata to read (row or column) |
| <code>ids</code> | a character vector of a subset of row/column ids for which to read the metadata |
| <code>set_annot_rownames</code> | a boolean indicating whether to set the <code>rownames</code> attribute of the returned <code>data.frame</code> 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 | <i>Strip non-alpha-numeric characters from a string</i> |
|----------|---|

Description

Strip non-alpha-numeric characters from a string

Usage

```
stripStr(str)
```

Arguments

| | |
|-----|-----------|
| str | Character |
|-----|-----------|

Value

Character without non-alphanumeric values

| | |
|------------|---|
| subsetData | <i>Subset data by rows and/or columns</i> |
|------------|---|

Description

Subset data by rows and/or columns

Usage

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

Value

Subset data

| | |
|-----------|--|
| subsetDim | <i>Subset rows or columns based on a given index</i> |
|-----------|--|

Description

Subset rows or columns based on a given index

Usage

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

Value

Subset rows/columns

| | |
|-------------|---|
| subsetToIds | <i>Do a robust <code>data.frame</code> subset to a set of ids</i> |
|-------------|---|

Description

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

Usage

```
subsetToIds(df, ids)
```

Arguments

| | |
|-----|-----------------------------------|
| df | <code>data.frame</code> 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
 - getCMapPerturbationTypes, 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
 - loadENCODEsamples, 34
 - performDifferentialExpression, 38
 - prepareENCODEgeneExpression, 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
 - getENCODEcontrols, 25
 - HTMLfast, 25
 - loadCTRPgeneExpression, 32
 - loadENCODEsample, 34
 - loadNCI60drugSensitivity, 36
 - matchStatsWithDrugSetsID, 37
 - performGSEA, 39
 - plotESplot, 45
 - plotGSEA, 45
 - plotMetricDistribution, 46

- plotSingleCorr, [47](#)
- prepareExpressionDrugSensitivityAssociation, [54](#)
- prepareGSEAgeneSets, [54](#)
- prepareSetsCompoundInfo, [55](#)
- prepareStatsCompoundInfo, [56](#)
- prepareWordBreak, [56](#)
- 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](#)
- .plotBubbles, [4](#)
- .prepareNavPage, [4](#)
- .traceInList, [4](#)
- [.expressionDrugSensitivityAssociation
 - (dimnames.expressionDrugSensitivityAssociation), [18](#)
- [.perturbationChanges
 - (plot.perturbationChanges), [40](#)
- analyseDrugSetEnrichment, [5](#), [34](#), [45](#), [52](#)
- as.table.referenceComparison, [6](#), [21](#), [24](#), [30](#), [31](#), [36](#), [38](#), [41](#), [43](#), [48](#), [50](#), [51](#), [57](#), [63](#)
- binr::bins, [8](#)
- calculateCellLineMean, [7](#)
- calculateEvenlyDistributedBins, [8](#)
- checkColnames, [9](#)
- chunkColumns, [9](#)
- closeOpenHandles, [10](#)
- cmapMetadata, [10](#)
- cmapPerturbationsCompounds, [10](#)
- cmapPerturbationsKD, [11](#)
- compareAgainstCMap
 - (rankSimilarPerturbations), [61](#)
- compareQuantile, [12](#)
- compareWithAllMethods, [12](#)
- convertENSEMBLtoGeneSymbols, [14](#)
- convertGeneIdentifiers, [14](#), [53](#)
- counts, [15](#)
- cTRAP, [16](#), [26–29](#)
- cTRAP-package (cTRAP), [16](#)
- 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](#), [20](#), [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](#)
- getCMapPerturbationTypes, [7](#), [21](#), [24](#), [24](#), [30](#), [31](#), [38](#), [41](#), [43](#), [48](#), [51](#), [57](#), [63](#)
- getENCODEcontrols, [25](#)
- HTMLfast, [25](#)
- launchCMapDataLoader, [17](#), [26](#), [27–29](#)
- launchDiffExprLoader, [17](#), [26](#), [27](#), [28](#), [29](#)
- launchDrugSetEnrichmentAnalyser, [17](#), [26](#), [27](#), [27](#), [28](#), [29](#)
- launchMetadataViewer, [17](#), [26–28](#), [28](#), [29](#)
- launchResultPlotter, [17](#), [26–28](#), [29](#)
- listExpressionDrugSensitivityAssociation, [7](#), [29](#), [35](#), [36](#), [43](#), [48](#), [50](#)

- 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
(loadCTRPgeneExpression), [32](#)
- loadCTRPdrugSensitivity
(loadCTRPgeneExpression), [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
(loadCTRPgeneExpression), [32](#)
- loadGDSC7compoundInfo
(loadCTRPgeneExpression), [32](#)
- loadGDSC7drugSensitivity
(loadCTRPgeneExpression), [32](#)
- loadGDSC7file (loadCTRPgeneExpression),
[32](#)
- loadGDSC7geneExpression
(loadCTRPgeneExpression), [32](#)
- loadNCI60drugSensitivity, [36](#)
- loadNCI60geneExpression
(loadCTRPgeneExpression), [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](#)
- plotTargetingDrugsVsimilarPerturbations,
[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](#)