Package ‘gep2pep’

Type Package
Title Creation and Analysis of Pathway Expression Profiles (PEPs)
Version 1.3.0
Date 2018-05-15
Author Francesco Napolitano <franapoli@gmail.com>
Maintainer Francesco Napolitano <franapoli@gmail.com>
Description Pathway Expression Profiles (PEPs) are based on the expression of pathways (defined as sets of genes) as opposed to individual genes. This package converts gene expression profiles to PEPs and performs enrichment analysis of both pathways and experimental conditions, such as "drug set enrichment analysis" and "gene2drug" drug discovery analysis respectively.
License GPL-3
Imports repo (>= 2.1.1), foreach, stats, utils, GSEABase, methods, Biobase, XML, rhdf5, digest, iterators
Suggests WriteXLS, testthat, knitr, rmarkdown
RoxygenNote 6.0.1
VignetteBuilder knitr
biocViews GeneExpression, DifferentialExpression, GeneSetEnrichment, DimensionReduction, Pathways
git_url https://git.bioconductor.org/packages/gep2pep
git_branch master
git_last_commit ec3d081
git_last_commit_date 2018-10-30
Date/Publication 2018-11-17

R topics documented:

  gep2pep-package .......................................................... 2
  as.CategorizedCollection ............................................. 3
  buildPEPs ............................................................... 5
  CategorizedCollection ................................................ 7
  CategorizedCollection-class ......................................... 7
  checkRepository ....................................................... 8
Pathway Expression Profiles (PEPs) are based on the expression of pathways (or generic gene sets) belonging to a collection, as opposed to individual genes. gep2pep supports the conversion of gene expression profiles (GEPs) to PEPs and performs enrichment analysis of both pathways and conditions.

gep2pep creates a local repository of gene sets, which can also be imported from the MSigDB [1] database. The local repository is in the repo format. When a GEP, defined as a ranked list of genes, is passed to buildPEPs, the stored database of pathways is used to convert the GEP to a PEP and permanently store the latter.

One type of analysis that can be performed on PEPs and that is directly supported by gep2pep is the Drug-Set Enrichment Analysis (DSEA [2]). It finds pathways that are consistently dysregulated by a set of drugs, as opposed to a background of other drugs. Of course PEPs may refer to non-pharmacological conditions (genetic perturbations, disease states, cell types, etc.) for analogous analyses. See CondSEA function.

A complementary approach is that of finding conditions that consistently dysregulate a set of pathways. This is the pathway-based version of the Gene Set Enrichment Analysis (GSEA). As an application example, this approach can be used to find drugs mimicking the dysregulation of a gene by looking for drugs dysregulating pathways involving the gene (this has been published as the gene2drug tool [3]). See PathSEA.
Both DSEA and gene2drug analyses can be performed using preprocessed data from http://dsea.tigem.it/downloads.php. The data include Connectivity Map [4] GEPs (drug-induced gene expression profiles) converted to PEPs in the form of a gep2pep repository.

Naming conventions:

- pathway: any set of gene identifiers (not necessarily representing a molecular pathway).
- pathway collection: a set of pathways.
- pathway database: a set of pathway collections, like the MSigDB.
- Gene Expression Profile (GEP): a named vector where names are gene identifiers of the same type as those in the pathway database and elements are ranks ranging from 1 to the number of genes.
- Pathway Expression Profile (PEP): a ranked list of pathways, as converted from a GEP according to a pathway collection.
- condition: any transcriptomic-modelled biological state (drug treatment, gene knock-out, disease state, cell type, etc.) characterized by an induced GEP and therefore a PEP.
- gep2pep repository: a pathway database and possibly a related database of PEPs as created by the gep2pep package. It is implemented in repo format.

Author(s)
Francesco Napolitano <franapoli@gmail.com>

References

---

**as.CategorizedCollection**

Converts GeneSetCollection objects to CategorizedCollection objects.

**Description**

Converts GeneSetCollection objects to CategorizedCollection objects.

**Usage**

```r
as.CategorizedCollection(GScollection, category = "uncategorized", subCategory = "uncategorized")
```
### as.CategorizedCollection

**Arguments**

- `GScollection`: An object of class `GeneSetCollection`.
- `category`: The name of the category that all the gene sets will be assigned to (see details).
- `subCategory`: The name of the subcategory that all the gene sets will be assigned to (see details).

**Details**

This function sets the `CollectionType` for each set in the collection to `CategorizedCollection`. If `GScollection` contains `BroadCollection` gene sets, their fields `category` and `subcategory` will be used. Otherwise, the `category` and `subcategory` fields will be used.

**Value**

A `CategorizedCollection` object

**Examples**

```r
## Not run:
## To run this example, first obtain the MSigDB database in XML format (see
## http://software.broadinstitute.org/gsea/downloads.jsp). It is assumed that the database is locally available as the file
## "msigdb_v6.0.xml".

The `\code{importMSigDB.xml} function is just a shortcut to the following:

db <- getBroadSets("msigdb_v6.1.xml")
db <- as.CategorizedCollection(db)

## The database is now in an acceptable format to create a local
## repository using \code{createRepository}

## End(Not run)

## A small sample of the MSigDB as imported by importMSigDB.xml is
## included in gep2pep. The following creates (and deletes) a
## gep2pep repository.

db_sample <- loadSamplePWS()

## The function \code{as.CategorizedCollection} can also be used to
## create arbitrary gene set collections specifying the categories
## and subcategories once for all the sets:

library(GSEABase)
mysets <- as.CategorizedCollection(
  GeneSetCollection(
    list(GeneSet(c("g1", "g2"), setName="set1"),
         GeneSet(c("g3", "g4"), setName="set2"))
  ),
  category="mycategory",
  subCategory="mysubcategory"
)
newCollection <- GeneSetCollection(c(db_sample, mysets))

## The created repository will include both the sample gene sets
## and the two sets just created:

repo_path <- file.path(tempdir(), "gep2pepTemp")
rp <- createRepository(repo_path, newCollection)

## removing temporary repository
unlink(repo_path, TRUE)

---

**buildPEPs**

*Build PEPs from GEPs and stores them in the repository.*

**Description**

Given a matrix of ranked lists of genes (GEPs) and a gep2pep repository, converts GEPs to PEPs and stores the latter in the repository.

**Usage**

`buildPEPs(rp, geps, min_size = 3, max_size = 500, parallel = FALSE, collections = "all", replace_existing = FALSE, progress_bar = TRUE, rawmode_id = NULL, rawmode_outdir = file.path(rp$root(), "raw"))`

**Arguments**

- **rp**: A repository created by `createRepository`.
- **geps**: A matrix of ranks where each row corresponds to a gene and each column to a condition. Each column must include all ranks from 1 to the number of rows. Row and column names must be defined. Row names will be matched against gene identifiers in the pathways collections, and unrecognized gene names will not be used.
- **min_size**: An integer representing the minimum number of genes that must be included in a set before the KS statistic is computed. Smaller gene sets will get ES=NA and p=NA. Default is 3.
- **max_size**: An integer representing the maximum number of genes that must be included in a set before the KS statistic is computed. Larger gene sets will get ES=NA and p=NA. Default is 500.
- **parallel**: If TRUE, gene sets will be processed in parallel. Requires a parallel backend.
- **collections**: A subset of the collection names returned by `getCollections`. If set to "all" (default), all the collections in `rp` will be used.
- **replace_existing**: What to do if PEPs, identified by column names of `geps` are already present in the repository. If set to TRUE, they will be replaced, otherwise they will be skipped and only PEPs of new conditions will be computed and added. Either ways, will throw a warning.
- **progress_bar**: If set to TRUE (default) will show a progress bar updated after conversion of each column of `geps`. 

---


rawmode_id An integer to be appended to files produced in raw mode (see details). If set to NULL (default), raw mode is turned off.

rawmode_outdir A character vector specifying the destination path for files produced in raw mode (by default it is ROOT/raw, where ROOT is the root of the repository). Ignored if rawmode_id is NULL.

Details

By default, output is written to the repository as new items named using the collection name. However, it is possible to avoid the repository and write the output to regular files turning 'raw mode' on through the rawmode_id and rawmode_outdir parameters. This is particularly useful when dealing with very large corpora of GEPs, and conversions are split into independent jobs submitted to a scheduler. At the end, the data will need to be reconstructed and put into the repository using importFromRawMode in order to perform CondSEA or PathSEA analysis.

Value

Nothing. The computed PEPs will be available in the repository.

See Also

buildPEPs

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
## Repo root created.
## Repo created.
## [15:45:06] Storing pathway data for collection: c3_TFT
## [15:45:06] Storing pathway data for collection: c3_MIR
## [15:45:06] Storing pathway data for collection: c4_CGN

rp
## ID Dims Size
## c3_TFT_sets 10 18.16 kB
## c3_MIR_sets 10 17.25 kB
## c4_CGN_sets 10 6.9 kB

## Loading sample gene expression profiles
geps <- loadSampleGEP()

geps[1:3,1:3]
## (+)_chelidonine (+)_isoprenaline (+/_)_catechin
## AKT3 88 117 417
## MED6 357 410 34
## NR2E3 383 121 453

buildPEPs(rp, geps)

rp
## ID Dims Size
## c3_TFT_sets 10 18.16 kB
## c3_MIR_sets 10 17.25 kB
CategorizedCollection

## c4_CGN_sets 10 6.9 kB
## c3_TFT 2 1.07 kB
## c3_MIR 2 1.07 kB
## c4_CGN 2 1.04 kB

unlink(repo_path, TRUE)

CategorizedCollection  Constructor method for objects of class CategorizedCollection.

Description

See CategorizedCollection-class.

Usage

CategorizedCollection(category = "uncategorized",
subCategory = "uncategorized")

Arguments

category A character defining the main category that the gene set belongs to.
subCategory A character defining the secondary category that the gene set belongs to.

Value

An object of class CategorizedCollection.

Examples

library(GSEABase)
gs1 <- GeneSet(setName="set1", setIdentifier="101")
collectionType(gs1) <- CategorizedCollection()

CategorizedCollection-class  A class to contain categorized gene set collection

Description

This class is a simple generalization of the BroadCollection function of GSEABase to store gene sets having assigned categories and subcategories that can be different from those of the MSigDB.

Slots

category A character defining the main category that the gene set belongs to.
subCategory A character defining the secondary category that the gene set belongs to.
checkRepository

Check an existing repository for consistency

Description
Check both repository data consistency (see repo_check from the repo package) and specific gep2pep data consistency.

Usage
checkRepository(rp)

Arguments
rp A repository created by createRepository.

Value
Nothing.

Examples
db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
checkRepository(rp)

unlink(repo_path, TRUE)

clearCache

Clear cached ranked matrices

Description
Clear cached ranked matrices

Usage
clearCache(rp_peps)

Arguments
rp_peps A repository created with createRepository, and containing PEPs created with buildPEPs.

Details
This will clear everything in the repository tagged with "stashed", which by default includes only matrices ranked by some gep2pep functions such as CondSEA.
CondSEA

Value

Nothing, used for side effects

See Also

CondSEA

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

pgset <- c("(+)_chelidonine", "(+/-)_catechin")
psea <- CondSEA(rp, pgset, usecache=TRUE)

## the repository contains cached data
print(rp, all=TRUE)
clearCache(rp)
unlink(repo_path, TRUE)

CondSEA

Performs Condition Set Enrichment Analysis

Description

Condition Set Enrichment Analysis (CondSEA) can be seen as a Gene-SEA performed over rows (as opposed to columns) of a matrix of GEPs. It tells how much a pathway is consistently dysregulated under a set of conditions (such as a set of drug treatments, disease states, cell types, etc.) when compared to a statistical background of other conditions.

Usage

CondSEA(rp_peps, pgset, bgset = "all", collections = "all", details = TRUE, rankingFun = rankPEPsByRows.ES, usecache = FALSE, sortoutput = TRUE)

Arguments

rp_peps A repository created with createRepository, and containing PEPs created with buildPEPs.

pgset A vector of names of conditions. Corresponding PEPs must exist in all the pathway collections currently in rp.

bgset The background against which to compare pgset. If set to all (default), all the remaining PEPs will be used. If provided, the corresponding PEPs must exist in all the pathway collections currently in rp.
collections  A subset of the collection names returned by getCollections. If set to "all" (default), all the collections in rp will be used.
details   If TRUE (default) rank details will be reported for each condition in pgset.
rankingFun  The function used to rank PEPs column-wise. By default rankPEPsByRows.ES is used, which ranks using gene set enrichment scores (see details).
usecache   If set to TRUE, the computed ranked matrix will be stored in the repository (see details). FALSE by default.
sortoutput  If TRUE (default) the output gene sets will be sorted in order of increasing p-value.

Details

For each pathway, all conditions are ranked by how much they dysregulate it (from the most UP-regulating to the most DOWN-regulating). Then, a Kolmogorov-Smirnov (KS) test is performed to compare the ranks assigned to conditions in pgset against the ranks assigned to conditions in bgset. A positive (negative) Enrichment Score (ES) of the KS test indicates whether each pathway is UP- (DOWN-) regulated by pgset as compared to bgset. A p-value is associated to the ES.

When PEPs are obtained from drug-induced gene expression profiles, PathSEA is the Drug-Set Enrichment Analysis [1].

The rankingFun must take in input PEPs like those loaded from the repository and return a matrix of row-wise ranks. Each row must contains ranks from 1 to the number of PEPs minus the number of NAs in the row.

When usecache=TRUE, the ranked matrix is permanently stored in HDF5 format, and subsequent calls to CondSEA will load from the disk the necessary ranks (not the whole matrix). The correct cached data is identified by the alphabetically sorted set union(pgset,bgset), by the collection name, and by the ranking function. Additional calls to CondSEA with variations of these inputs will create additional cache. Cached data is hidden in the repository by default and can be printed with rp_peps$print(all=TRUE), and cleared with clearCache(rp_peps).

Value

A list of 2, by names "CondSEA" and "details". The "CondSEA" entry is a 2-columns matrix including ESs and p-values (see details) for each pathway database and condition. The "details" entry reports the rank of each condition in pgset for each pathway.

References


See Also

getResults, getDetails, clearCache

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)
pgset <- c("(+)_chelidonine", "(+/-)_catechin")
psea <- CondSEA(rp, pgset)
res <- getResults(psea, "c3_TFT")
## getting the names of the top pathways
setId2setName(loadCollection(rp, "c3_TFT"); rownames(res))
unlink(repo_path, TRUE)

createMergedRepository

*Merge multiple PEPs to build a repository of consensus PEPs*

**Description**

Merge multiple PEPs to build a repository of consensus PEPs

**Usage**

```r
createMergedRepository(rpIn_path, rpOut_path, mergestr, progressBar = TRUE, collections = "all")
```

**Arguments**

- `rpIn_path` path to existing gep2pep repository
- `rpOut_path` path where the new merged repository will be created
- `mergestr` a named list of character vectors, each one including a set of PEP names. For each list entry, a consensus PEP will be created and assigned the entry name.
- `progressBar` if TRUE, show a progress bar
- `collections` A subset of the collection names returned by getCollections. If set to "all" (default), all the collections in `rp` will be used.

**Details**

The merging is performed as follows. Given N PEPs, the corresponding consensus PEP will get as enrichment score the average enrichment scores of the N PEPs, and as p-value the composition of the N PEP p-values by Fisher’s method.

**Value**

Nothing, used for side effects.
createRepository

Creates a repository of pathway collections.

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")
rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

mergestr <- list(
    che_iso = c("(+)_chelidonine", "(+)_isoprenaline"),
    cat_mk = c("(+/_)_catechin", "(_)_mk_801")
)
merged_path <- file.path(tempdir(), "gep2pepTempMerged")
createMergedRepository(repo_path, merged_path, mergestr)
unlink(repo_path, TRUE)
unlink(merged_path, TRUE)

createRepository(path, sets, name = NULL, description = NULL)

Arguments

path Path to a non-existing directory where the repository will be created.
sets An object of class CategorizedCollection.
name Name of the repository. Defaults to NULL (a generic name will be given).
description Description of the repository. If NULL (default), a generic description will be given.

Details

sets can be created by importMSigDB.xml or using GSEABase GeneSetCollection class and then converting it to CategorizedCollection. See examples.

Value

An object of class repo that can be passed to gep2pep functions.

See Also

buildPEPs
Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
## Repo root created.
## Repo created.
## [15:45:06] Storing pathway data for collection: c3_TFT
## [15:45:06] Storing pathway data for collection: c3_MIR
## [15:45:06] Storing pathway data for collection: c4_CGN

rp
## ID    Dims  Size
## c3_TFT_sets 10  18.16 kB
## c3_MIR_sets 10  17.25 kB
## c4_CGN_sets 10   6.9 kB

unlink(repo_path, TRUE)

exportSEA

Export CondSEA or PathSEA results to XLS format

Description

The XLS output includes the full CondSEA or PathSEA results, together with additional gene set information for the CondSEA.

Usage

exportSEA(rp, results, outname = NULL)

Arguments

- **rp**: A repository created by `createRepository`.
- **results**: The output of CondSEA or PathSEA.
- **outname**: Name of the XLS file to be created.

Value

Nothing.

See Also

CondSEA, PathSEA
Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

pgset <- c("(+)_chelidonine", "(+/-)_catechin")
psea <- CondSEA(rp, pgset)

## Not run:
exportSEA(rp, psea)
## End(Not run)
unlink(repo_path, TRUE)

gene2pathways Finds pathways including a given gene.

Description

Given a gene, find the set of pathways that involve it in each collection of the repository. This can be used to define a set of pathways for the PathSEA.

Usage

gene2pathways(rp, gene)

Arguments

rp A repository created by createRepository.

gene A gene identifier of the same type as that used to create the repository.

Value

A database of pathways suitable as input to PathSEA.

See Also

createRepository, PathSEA

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)

## Finding all pathways containing "FAM126A":

## Finding all pathways containing "FAM126A":
subpw <- gene2pathways(rp, "FAM126A")

print(names(subpw))

unlink(repo_path, TRUE)

---

**getCollections**

*Returns the names of the pathway collections in a repository.*

**Description**

Given a `gep2pep` repository, returns the names of the stored collections by looking at appropriate repository item names.

**Usage**

```r
getCollections(rp)
```

**Arguments**

- **rp**
  - A repository created by `createRepository`.

**Details**

Each collection in a database has a "category" and a "subcategory" assigned, which are used to build the collection identifier as "category_subcategory". This function obtains the identifiers by looking at data stored in the repository `rp` (entries that are tagged with "sets").

**Value**

Vector of collection names (see details).

**Examples**

```r
db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
## Repo root created.
## Repo created.
## [15:45:06] Storing pathway data for collection: c3_TFT
## [15:45:06] Storing pathway data for collection: c3_MIR
## [15:45:06] Storing pathway data for collection: c4_CGN

getCollections(rp)
## [1] "c3_TFT"  "c3_MIR"  "c4_CGN"

unlink(repo_path, TRUE)
```
getDetails

Extracts the details matrix from CondSEA or PathSEA output

Description

Extracts the details matrix from CondSEA or PathSEA output

Usage

getDetails(analysis, collection)

Arguments

- analysis: The output of either CondSEA or PathSEA.
- collection: One of the names returned by getCollections.

Value

A matrix including the ranks of each pathway (over rows) and each condition (over columns) used as input to CondSEA or PathSEA.

See Also

CondSEA, PathSEA

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

pgset <- c("(+)_chelidonine", "(+/-)_catechin")
psea <- CondSEA(rp, pgset)

getDetails(psea, "c3_TFT")

unlink(repo_path, TRUE)
getResults

Extracts the results matrix from CondSEA or PathSEA output

Description

Extracts the results matrix from CondSEA or PathSEA output

Usage

getResults(analysis, collection)

Arguments

- **analysis**: The output of either CondSEA or PathSEA.
- **collection**: One of the names returned by getCollections.

Value

A 2-columns matrix including ESs and p-values (see details) for each pathway database and condition.

See Also

CondSEA, PathSEA

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, gep)

pgset <- c("(+)_chelidonine", "+/_catechin")
psea <- CondSEA(rp, pgset)

getResults(psea, "c3_TFT")

unlink(repo_path, TRUE)
**importFromRawMode**

Imports PEPs created in raw mode

**Description**

Raw mode is meant to deal with large collections of PEPs (like hundreds of thousands). In this case, problems may arise while trying to convert GEPs by loading all of them in memory at once. Raw mode is meant to be used with HDF5 format, which allows to load subsets of GEPs from the disk. buildPEPs, when used in raw mode, can create the corresponding subsets of PEPs, so that the job can be distributed on a computer cluster. importFromRawMode is meant to join the chunks into HDF5 matrices, which are than stored into the repository. The .loadPEPs function can seamlessly load PEPs stored in normal (RDS) or HDF5 format.

**Usage**

```r
importFromRawMode(rp, path = file.path(rp$root(), "raw"),
                     collections = "all")
```

**Arguments**

- **rp** A repository created by `createRepository`.
- **path** Path were raw PEPs are stored (default is a "raw" directory under the repository root folder).
- **collections** A subset of the collection names returned by `getCollections`. If set to "all" (default), all the collections in rp will be used.

**Details**

PEPs are expect to be found at the specified path and follow the naming convention as generated by `buildPEPs`. According to such convention, each file is named using the format `category_subcategory#chunknumber.RDS`. All non-alphanumeric characters from the original category and subcategory names are replace with an underscore (in rare cases this could create ambiguity that should be manually prevented). All chunks for the same subcategory are joined together following the chunk numbers into a single HDF5 matrix and stored in the repository as an "attachment" (see repo documentation).

Note that raw PEPs (by default everything at repository_root/raw) can be safely removed once they have been imported.

**Value**

Nothing, used for side effects.

**See Also**

`buildPEPs`
Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)

## The following will create PEPs in 2 separate files
geps <- loadSampleGEP()
buildPEPs(rp, geps[,1:2], progress_bar=FALSE,
          rawmode_id=1)
buildPEPs(rp, geps[,3:5], progress_bar=FALSE,
          rawmode_id=2)

## The separate files are then merged into one (possibly big) file
## in HDF5 format
importFromRawMode(rp)

## Now most operations (excluding the addition of new PEPs to
## existing collections) will be available as usual.
unlink(repo_path, TRUE)

importMSigDB.xml

Imports pathways data from an MSigDB XML file.

Description

Creates a GeneSetCollection object using the XML distribution of the MSigDB (see references). The returned object can be passed to createRepository.

Usage

importMSigDB.xml(fname, organism = "Homo Sapiens")

Arguments

fname           Path to an XML file downloaded from MSigDB.
organism        Select only gene sets for a given organism. Default is "Homo Sapiens".

Details

This function now just calls getBroadSets(fname) from the GSEABase package. However, it is left for backward compatibility and as an entry point to package functionalities.

Value

A CategorizedCollection object

References

http://software.broadinstitute.org/gsea/downloads.jsp
Examples

## Not run:

## To run this example, first obtain the MSigDB database in XML
## format (see
## http://software.broadinstitute.org/gsea/downloads.jsp). It is
## assumed that the database is locally available as the file
## "msigdb_v6.0.xml".

db <- importMSigDB.xml("msigdb_v6.0.xml")

## The database is now in an acceptable format to create a local
## repository using createRepository

## End(Not run)

## A small excerpt from the MSigDB is included in gep2pep. The
## following creates (and then deletes) a gep2pep repository.

db_sample <- loadSamplePWS()

repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db_sample)

## removing temporary repository

unlink(repo_path, TRUE)

loadCollection

_loadCollection_ Loads a collection of pathways from the repository

Description

Loads a collection of pathways from the repository

Usage

loadCollection(rp, collection)

Arguments

rp    A repository created by createRepository.
collection One of the names returned by getCollections.

Value

a GeneSetCollection object loaded from the repository rp.

Examples

db <- loadSamplePWS()

repo_path <- file.path(tempdir(), "gep2pepTemp")
loadESmatrix

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()

loadCollection(rp, "c3_TFT")

unlink(repo_path, TRUE)

loadESmatrix

Loads the matrix of Enrichment Scores for a collection

Description

Loads the matrix of Enrichment Scores for a collection

Usage

loadESmatrix(rp, collection)

Arguments

rp A repository created by createRepository.
collection One of the names returned by get Collections.

Value

The matrix of Enrichment Scores (ES) of the Kolmogorov-Smirnov statistic for the pathway collection, if previously computed with buildPEPs. The entry $i,j$ reports the ES for the pathway $i$, condition $j$. If buildPEPs was not run, throws an error.

Examples

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

loadESmatrix(rp, "c3_TFT")[1:5,1:2]

unlink(repo_path, TRUE)
loadPVmatrix loads the matrix of p-values for a collection

**Description**
Loads the matrix of p-values for a collection

**Usage**
loadPVmatrix(rp, collection)

**Arguments**
- **rp**: A repository created by createRepository.
- **collection**: One of the names returned by getCollections.

**Value**
The matrix of p-values (PV) of the Kolmogorov-Smirnov statistic for the pathway collection, if previously computed with buildPEPs. The entry \(i,j\) reports the PV for the pathway \(i\), condition\(j\). If buildPEPs was not run, throws an error.

**Examples**
```r
db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

loadPVmatrix(rp, "c3_TFT")

unlink(repo_path, TRUE)
```

loadSampleGEP loads sample Gene Expression Profiles

**Description**
Loads sample Gene Expression Profiles

**Usage**
loadSampleGEP()

**Value**
Sample gene expression data
loadSamplePWS

Examples

geps <- loadSampleGEP()

Description
Loads sample pathway collections

Usage
loadSamplePWS()

Value
Sample pathway collections in GeneSetCollection format

Examples

geps <- loadSampleGEP()

makeCollectionIDs

Creates a collection label for each pathway.

Description
Given a database, uses "category" and "subcategory" entries to create a vector of collection identifiers. Useful to extract a collection from a database.

Usage
makeCollectionIDs(sets)

Arguments

sets A pathway database in the same format as output by importMSigDB.xml.

Value
A vector of identifiers, one per pathway, with the format: "category_subcategory".

See Also
importMSigDB.xml
openRepository

Opens an existing repository of pathway collections.

Description
The repository must have been created by `createRepository`. Provides an R object to interact with the repository.

Usage
openRepository(path)

Arguments
path Path to a directory where the repository has been created with `createRepository`.

Details
This function only calls the `repo.open` function from the `repo` package on `path`. It is meant to allow users not to explicitly load the `repo` library, unless they want to access advanced features.

Value
An object of class `repo` that can be passed to `gep2pep` functions.

See Also
createRepository

Examples

db <- loadSamplePWS()
ids <- makeCollectionIDs(db)

unique(ids)
## [1] "c3_TFT"  "c3_MIR"  "c4_CGN"

db <- db[ids=="c3_MIR"]

length(db)
## [1] 10

```
# openRepository

db <- loadSamplePWS()
repopath <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repopath, db)
rp2 <- openRepository(repopath)

## rp and rp2 point to the same data:
identical(rp$entries(), rp2$entries())
```
## Description

PathSEA is analogous to the Gene Set Enrichment Analysis (GSEA), but for pathways instead of single genes. It can therefore be used to look for conditions under which a given set of pathways is consistently UP- or DOWN-regulated.

## Usage

PathSEA(rp_peps, pathways, bgsets = "all", collections = "all", subset = "all", details = TRUE, rankingFun = rankPEPsByCols.SPV)

## Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rp_peps</td>
<td>A repository created with createRepository, and containing PEPs created with buildPEPs.</td>
</tr>
<tr>
<td>pathways</td>
<td>A database of pathways in the same format as input to createRepository. PSEA will be performed for each database separately.</td>
</tr>
<tr>
<td>bgsets</td>
<td>Another list like pathways, representing the statistical background for each database. If set to &quot;all&quot; (the default), all pathways that are in the repository and not in pathways will be used.</td>
</tr>
<tr>
<td>collections</td>
<td>A subset of the collection names returned by getCollections. If set to &quot;all&quot; (default), all the collections in rp will be used.</td>
</tr>
<tr>
<td>subset</td>
<td>Character vector including PEP names to be considered (all by default, which may take time).</td>
</tr>
<tr>
<td>details</td>
<td>If TRUE (default) details will be reported for each condition in pgset.</td>
</tr>
<tr>
<td>rankingFun</td>
<td>The function used to rank PEPs column-wise. By default rankPEPsByCols.ES is used, which uses gene set enrichment scores (see details).</td>
</tr>
</tbody>
</table>

## Details

For each condition, all pathways are ranked by how much they are dysregulated by it (from the most UP-regulated to the most DOWN-regulated, according to the corresponding p-values). Then, a Kolmogorov-Smirnov (KS) test is performed to compare the ranks assigned to pathways in pathways against the ranks assigned to pathways in bgsets. A positive (negative) Enrichment Score (ES) of the KS test indicates whether each pathway is UP- (DOWN-) regulated by pgset as compared to bgset. A p-value is associated to the ES.

When PEPs are obtained from drug-induced gene expression profiles, PathSEA can be used together with gene2pathways to perform gene2drug [1] analysis, which predicts which drugs may target a gene of interest (or mimick such effect).

The rankingFun must take in input PEPs like those loaded from the repository and return a matrix of column-wise ranks. Each column must contain ranks from 1 to the number of gene sets minus the number of NAs in the column.
PathSEA

Value

A list of 2, by names "PathSEA" and "details". The "PathSEA" entry is a 2-columns matrix including ESs and p-values for each collection and condition. The "details" entry reports the rank of each pathway in pathways for each condition.

References


See Also

getResults, getDetails

Examples

library(GSEABase)

db <- loadSamplePWS()
repo_path <- file.path(tempdir(), "gep2pepTemp")

rp <- createRepository(repo_path, db)
geps <- loadSampleGEP()
buildPEPs(rp, geps)

pathways <- c("M11607", "M10817", "M16694", ## from c3_TFT
               "M19723", "M5038", "M13419", "M1094") ## from c4_CGN
w <- sapply(db, setIdentifier) %in% pathways

psea <- PathSEA(rp, db[w])
## [15:35:29] Working on collection: c3_TFT
## [15:35:29] Common pathway sets removed from bgset.
## [15:35:29] Column-ranking collection...
## [15:35:29] Computing enrichments...
## [15:35:29] Common pathway sets removed from bgset.
## [15:35:29] Column-ranking collection...
## [15:35:29] Computing enrichments...

getResults(psea, "c3_TFT")
## ES PV
## (_)_mk_801 0.7142857 0.1666667
## (_)_atenolol 0.7142857 0.1666667
## (+)_isoprenaline 0.5714286 0.4000000
## (+/_)_catechin 0.5714286 0.4000000
## (+)_chelidonine 0.3333333 0.9333333

unlink(repo_path, TRUE)
**setId2setName**

Converts gene set IDs to gene set names

**Description**

Converts gene set IDs to gene set names

**Usage**

```r
setId2setName(sets, ids)
```

**Arguments**

- `sets` : An object of class GeneSetCollection
- `ids` : character vector of gene set IDs to be converted to set names

**Value**

A vector of gene set names

**See Also**

CondSEA, PathSEA

**Examples**

```r
collection <- loadSamplePWS()
setId2setName(collection, c("M3128", "M11607"))
```
Index

as.CategorizedCollection, 3
buildPEPs, 2, 5, 8, 9, 25
CategorizedCollection, 7
CategorizedCollection-class, 7
checkRepository, 8
clearCache, 8
CondSEA, 2, 9
createMergedRepository, 11
createRepository, 5, 8, 9, 12, 13–15, 18, 20–22, 24, 25
exportSEA, 13
gene2pathways, 14
gep2pep (gep2pep-package), 2
gep2pep-package, 2
getCollections, 15
getDetails, 16
getResults, 17
importFromRawMode, 18
importMSigDB.xml, 12, 19
loadCollection, 20
loadESmatrix, 21
loadPVmatrix, 22
loadSampleGEP, 22
loadSamplePWS, 23
makeCollectionIDs, 23
openRepository, 24
PathSEA, 2, 14, 25
setId2setName, 27