Package ‘gep2pep’

October 16, 2018

Type Package

Title Creation and Analysis of Pathway Expression Profiles (PEPs)

Version 1.0.0

Date 2017-09-11

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

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 RELEASE_3_7

git_last_commit bc54ceb

git_last_commit_date 2018-04-30

Date/Publication 2018-10-15

R topics documented:

`gep2pep-package` *gep2pep: creation and analysis of Pathway Expression Profiles*

Description

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.

Details

`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

[1] Subramanian A. et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. PNAS 102, 15545-15550 (2005). [2] Napolitano F. et al, Drug-set enrichment analysis: a novel tool to investigate drug mode of action. Bioinformatics 32, 235-241 (2016). [3] Napolitano F. et al, gene2drug: a Computational Tool for Pathway-based Rational Drug Repositioning, bioRxiv (2017) 192005; doi: https://doi.org/10.1101/192005 [4] Lamb, J. et al. The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease. Science 313, 1929-1935 (2006).

`as.CategorizedCollection` *Converts GeneSetCollection objects to CategorizedCollection objects.*

Description

Converts GeneSetCollection objects to CategorizedCollection objects.

Usage

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

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

```
## 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 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()
db_sample <- as.CategorizedCollection(db_sample)

## The function 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, parallel = FALSE, collections = "all",
  replace_existing = FALSE, progress_bar = TRUE)
```

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.
`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 coversion of each column of `geps`.

Value

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

See Also

buildPEPs

Examples

```
db <- loadSamplePWS()
db <- as.CategorizedCollection(db)
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
##   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 existyng 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()
db <- as.CategorizedCollection(db)
repo_path <- file.path(tempdir(), "gep2pepTemp")

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

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`.

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` can be used to perform Drug-Set Enrichment Analysis [1].

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

[1] Napolitano F. et al, Drug-set enrichment analysis: a novel tool to investigate drug mode of action. Bioinformatics 32, 235-241 (2016).

See Also

getResults, getDetails

Examples

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

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

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

getResults(psea, "c3_TFT")

unlink(repo_path, TRUE)
```

`createRepository` *Creates a repository of pathway collections.*

Description

Given a database of collections, stores them in a local repository to be used by gep2pep functions.

Usage

```
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()
db <- as.CategorizedCollection(db)
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()
db <- as.CategorizedCollection(db)
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()
db <- as.CategorizedCollection(db)
repo_path <- file.path(tempdir(), "gep2pepTemp")

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

```
## 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

```
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

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

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

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

getResults(psea, "c3_TFT")

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)
```

Arguments

`fname`	Path to an XML file downloaded from MSigDB.

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()
db_sample <- as.CategorizedCollection(db_sample)

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

```
## removing temporary repository
unlink(repo_path, TRUE)
```

`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()
db <- as.CategorizedCollection(db)
repo_path <- file.path(tempdir(), "gep2pepTemp")

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 `getCollections`.

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`, conditionj. If `buildPEPs` was not run, throws an error.

Examples

```
db <- loadSamplePWS()
db <- as.CategorizedCollection(db)
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`, conditionj. If `buildPEPs` was not run, throws an error.

Examples

```
db <- loadSamplePWS()
db <- as.CategorizedCollection(db)
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

Examples

```
geps <- loadSampleGEP()
dim(geps)
## [1] 485   5
```

`loadSamplePWS` *Loads sample pathway collections*

Description

Loads sample pathway collections

Usage

```
loadSamplePWS()
```

Value

Sample pathway collections in `GeneSetCollection` format

Examples

```
geps <- loadSampleGEP()
geps
## GeneSetCollection
##   names: AAANWWTGC_UNKNOWN, AAAYRNCTG_UNKNOWN, ..., MORF_BUB3 (30 total)
##   unique identifiers: MEF2C, ATP1B1, ..., SLBP (5778 total)
##   types in collection:
##     geneIdType: SymbolIdentifier (1 total)
##     collectionType: BroadCollection (1 total)
```

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

Examples

```
db <- loadSamplePWS()
db <- as.CategorizedCollection(db)
ids <- makeCollectionIDs(db)

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

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

length(db)
## [1] 10
```

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

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

## rp and rp2 point to the same data:
identical(rp$entries(), rp2$entries())
## > [1] TRUE

unlink(repo_path, TRUE)
```

PathSEA	*Performs Pathway Set Enrichment Analysis (PSEA)*

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",
  details = TRUE)
```

Arguments

`rp_peps`	A repository created with `createRepository`, and containing PEPs created with `buildPEPs`.
`pathways`	A database of pathways in the same format as input to `createRepository`. PSEA will be performed for each database separately.
`bgsets`	Another list like `pathways`, representing the statistical background for each database. If set to "all" (the default), all pathways that are in the repository and not in `pathways` will be used.
`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) details will be reported for each condition in `pgset`.

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-regulatied, 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).

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

[1] Napolitano F. et al, gene2drug: a Computational Tool for Pathway-based Rational Drug Repositioning, bioRxiv (2017) 192005; doi: https://doi.org/10.1101/192005

See Also

getResults, getDetails

Examples

```
library(GSEABase)

db <- loadSamplePWS()
db <- as.CategorizedCollection(db)
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] done.
## [15:35:29] Working on collection: C4_CGN
## [15:35:29] Common pathway sets removed from bgset.
## [15:35:29] Column-ranking collection...
## [15:35:29] Computing enrichments...
## [15:35:29] done.

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)
```

Index