Package 'bioassayR'

November 11, 2026

Type Package

Title Cross-target analysis of small molecule bioactivity

Version 1.50.0

Date 2024-10-21

Author Tyler Backman, Ronly Schlenk, Thomas Girke

Maintainer Thomas Girke <tgirke@citrus.ucr.edu>

Depends R (>= 3.5.0), DBI (>= 0.3.1), RSQLite (>= 1.0.0), methods, Matrix, rjson, BiocGenerics (>= 0.13.8)

Imports XML, ChemmineR

Suggests BiocStyle, RCurl, biomaRt, knitr, knitcitations, knitrBootstrap, testthat, ggplot2, rmarkdown

Description bioassayR is a computational tool that enables simultaneous analysis of thousands of bioassay experiments performed over a diverse set of compounds and biological targets. Unique features include support for large-scale cross-target analyses of both public and custom bioassays, generation of high throughput screening fingerprints (HTSFPs), and an optional preloaded database that provides access to a substantial portion of publicly available bioactivity data.

URL https://github.com/girke-lab/bioassayR

BugReports https://github.com/girke-lab/bioassayR/issues

License Artistic-2.0

biocViews ImmunoOncology, MicrotitrePlateAssay, CellBasedAssays, Visualization, Infrastructure, DataImport, Bioinformatics, Proteomics, Metabolomics

LazyLoad yes

Collate AllClasses.R AllGenerics.R BioassayDB-accessors.R bioassay-accessors.R loadingData.R queries.R bioassaySet-accessors.R bayesian-cross-reactivity.R similarity.R

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/bioassayR

git_branch RELEASE_3_23

git_last_commit b239b42

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

activeAgainst . . . 3
activeTargets . . . 4
addBioassayIndex . . . 5
addDataSource . . . 6
allCids . . . 7
allTargets . . . 8
assaySetTargets . . . 9
bioactivityFingerprint . . . 10
bioassay-class . . . 11
BioassayDB-class . . . 13
bioassaySet-class . . . 14
connectBioassayDB . . . 15
crossReactivityProbability . . . 16
disconnectBioassayDB . . . 18
dropBioassay . . . 19
dropBioassayIndex . . . 20
getAssay . . . 21
getAssays . . . 22
getBioassaySetByCids . . . 23
inactiveTargets . . . 24
loadBioassay . . . 25
loadIdMapping . . . 26
newBioassayDB . . . 27
parsePubChemBioassay . . . 28
perTargetMatrix . . . 29
queryBioassayDB . . . 31
samplebioassay . . . 32
scaleBioassaySet . . . 33
screenedAtLeast . . . 34
selectiveAgainst . . . 35
targetSelectivity . . . 36
translateTargetId . . . 37
trinarySimilarity . . . 39

Index **41**

activeAgainst *Show compounds active against a specified target*

Description

Returns a data.frame of small molecule cids which show activity against a specified target. Each row name represents a cid which shows activity, and the total screens and the percent active are shown in their respective columns.

Usage

```
activeAgainst(database, target)
```

Arguments

database	A BioassayDB database to query.
target	A string or integer containing a target_id referring to a target of interest.

Value

A data.frame where the row names represent each compound showing activity against the specified target. The second column shows the number of distinct assays in which this cid was screened against the target, and the first column shows the percentage of these which exhibited activity.

Author(s)

Tyler Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## get cids of compounds which show activity against target 116516899
myCids <- row.names(activeAgainst(sampleDB, "166897622"))

## disconnect from database
disconnectBioassayDB(sampleDB)
```

activeTargets *Show targets against which a small molecule is active*

Description

Returns a data.frame of the targets, which a given small molecule (specified by cid) shows activity against. For each target, a single row shows the total number of distinct screens it participated in, and the fraction of those in which it exhibits activity.

Usage

```
activeTargets(database, cid)
```

Arguments

database	A BioassayDB database to query.
cid	A string or integer containing a cid referring to a small molecule.

Value

A data.frame where the row names represent each target the specified compound shows activity against, and the columns show the total screens and the fraction in which the compound was active.

Author(s)

Tyler Backman

See Also

inactiveTargets

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## get targets that compound 2244 shows activity against
myTargets <- row.names(activeTargets(sampleDB, "2244"))

## disconnect from database
disconnectBioassayDB(sampleDB)
```

addBioassayIndex *Index a bioassayR database*

Description

Indexing a bioassayR database before performing queries will drastically improve query performance. However, it will also slow down loading large amounts of additional data. Therefore, we recommend loading the majority of your data, using this function to index, and then performing queries.

Usage

```
addBioassayIndex(database)
```

Arguments

database	A BioassayDB database to be indexed.

Author(s)

Tyler Backman

Examples

```
## create test database
library(bioassayR)
filename <- tempfile()
mydb <- newBioassayDB(filename, indexed=FALSE)

## load any data at this point

## add database index
addBioassayIndex(mydb)

# perform queries here

## close and delete test database
disconnectBioassayDB(mydb)
unlink(filename)
```

addDataSource *Add a new data source to a bioassayR database*

Description

This function adds a new data source (name/description and version) for tracking data within a bioassayR database. This can be used later to identify the source of any specific activity data within the database, or to limit analysis to data from specific source(s).

Usage

```
addDataSource(database, description, version)
```

Arguments

database	A BioassayDB database to add a new data source to.
description	A string containing a name or description of the new data source. This exact value will be used as a key for querying and loading data from this source.
version	A string with the version and/or date of the data source. This can be used to track the date in which a non-version data source was mirrored.

Author(s)

Tyler Backman

Examples

```
## create a test database
library(bioassayR)
filename <- tempfile()
mydb <- newBioassayDB(filename, indexed=FALSE)

## add a new data source
addDataSource(mydb, description="bioassayR_sample", version="1.0")

## list data sources loaded
mydb

## close and delete database
disconnectBioassayDB(mydb)
unlink(filename)
```

allCids *List compound cids in a* BioassayDB*,* bioassay*,* bioassaySet*, or target matrix (*dgCMatrix*) object*

Description

Returns a vector of small molecule cids contained within a BioassayDB, bioassay, bioassaySet, or target matrix (dgCMatrix) object. It can optionally only returned cids labeled as active.

Usage

```
allCids(inputObject, activesOnly = FALSE)
```

Arguments

inputObject	A BioassayDB, bioassay, bioassaySet, or target matrix (dgCMatrix) object to query.
activesOnly	logical. Should only active compounds be returned? Defaults to FALSE.

Value

A vector of distinct small molecule cids. No particular order is guranteed.

Author(s)

Tyler Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## get all compound cids
myCids <- allCids(sampleDB)

## get only active compound cids
activeCids <- allCids(sampleDB, activesOnly = TRUE)

## disconnect from database
disconnectBioassayDB(sampleDB)
```

allTargets *List distinct target(s) in a* BioassayDB*,* bioassay*,* bioassaySet*, or target matrix (*dgCMatrix*) object*

Description

Returns a vector of target ids contained within a BioassayDB, bioassay, bioassaySet, or target matrix (dgCMatrix) object.

Usage

```
allTargets(inputObject)
```

Arguments

inputObject	A BioassayDB, bioassay, bioassaySet, or target matrix (dgCMatrix) object to query.

Value

A vector of distinct target ids. No particular order is guaranteed.

Author(s)

Tyler Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## get all target ids
myTargets <- allTargets(sampleDB)

## disconnect from database
disconnectBioassayDB(sampleDB)
```

assaySetTargets *Return targets of assays in a* bioassaySet *object*

Description

This takes a bioassaySet of multiple assays and returns a vector of the targets of each, with the assay identifiers themselves (aids) as names. If a single assay contains multiple targets, these will all be listed.

Usage

```
assaySetTargets(assays)
```

Arguments

assays	A bioassaySet object with data from multiple assays, some of which may share a common target.

Value

A character vector of the targets of each, with the assay identifiers themselves (aids) as names

Author(s)

Tyler William H Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve three assays
assays <- getAssays(sampleDB, c("673509","103","105"))
assays

## get the targets for these assays
myTargets <- assaySetTargets(assays)
myTargets

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

bioactivityFingerprint

Create an **ChemmineR** FPset *object that contains bioactivity results for a given set of compounds and targets.*

Description

Returns a custom binary descriptor fingerprint for a given set of query cids and target compounds, based on the activity data within a bioassaySet object.

Usage

```
bioactivityFingerprint(bioassaySet, targets = FALSE, summarizeReplicates = "activesFirst")
```

Arguments

bioassaySet	A bioassaySet object to generate fingeprints from. For a given compound set, this can be generated from a database using the getBioassaySetByCids function.
targets	An optional list of target id(s) to consider when creating the binary fingerprint. If a listed target is not in the bioassaySet, or has no active scores it will still be accepted, but create a fingerprint with all zeros for this location. The binary order of this list is preserved, so that direct comparison and combination of resulting FPset objects created with the same target list can be performed. If omitted, the target list for the bioassaySet object will be used, as returned by the allTargets function.
summarizeReplicates	Optionally allows users to choose how replicates (multiple assays sharing common compounds and targets) are resolved if they disagree. If 'activesFirst' any active score will take precedence over an inactive. If 'mode' the resulting score will be computed according to the statistical mode using as.numeric(names(which.max(table(x)))). Users can also optionally pass a function here which (for each cid/target pair) will receive a list of '2' (active) and '1' (inactive) values, and can then return any desired number as a summary to be included in the resulting table. For a large matrix, the default option 'activesFirst' offers the lowest computational overhead.

Value

The returned object is a standard **ChemmineR** FPset object, and can be used as described in the **ChemmineR** documentation. The order and number of binary bits for each compound can be set using the targets option, enabling the combination or comparison of multiple objects created with the same target list. If a single compound has both active and inactive scores for the same target, it will be resolved according to the confictResolver option.

Author(s)

Tyler William H Backman

See Also

Functions: `getBioassaySetByCids`, `getAssays`, `perTargetMatrix`

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve all targets in database
targetList <- allTargets(sampleDB)

## get an activity fingerprint object for selected CIDs
queryCids <- c("2244", "3715", "2662", "3033", "133021",
    "44563999", "44564000", "44564001", "44564002")
myAssaySet <- getBioassaySetByCids(sampleDB, queryCids)
myFp <- bioactivityFingerprint(bioassaySet=myAssaySet)

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

---

bioassay-class *Class* "bioassay"

Description

This class represents the data from a bioassay experiment, where a number of small molecules are screened against a defined target (such as a protein or living organism).

Objects from the Class

Objects can be created by calls of the form `new("bioassay", ...)`.

Slots

`aid`: Object of class `"character"` containing the assay id. For assays sourced from NCBI PubChem, this should be a string containing the PubChem AID (assay identifier).

`source_id`: Object of class `"character"`. This should match the description for a data source loaded via the `addDataSource()` function.

`assay_type`: Object of class `"character"`. A string noting the type of bioactivity experiment, such as "confirmatory" to represent a confirmatory assay.

`organism`: Object of class `"character"`. A string noting the scientific name of the assays target organism.

`scoring`: Object of class `"character"`. A string noting the scoring method used for the bioactivity experiment. For example, IC50 or EC50.

`targets`: Object of class `"character"`. A string or vector of strings containing the target identifier indicating the assay target. In the case of protein targeted assays sourced from NCBI PubChem, this should be a genbank ID.

`target_types`: Object of class `"character"`. A string of text or vector of strings, representing (in the same order) the target types for each target. For example "protein" or "cell."

`scores`: Object of class `"data.frame"` containing the bioactivity data to be loaded. This must be a 3 column data frame, with each row representing the bioactivity results of a single molecule. The first column represents the compound id (cid), which must be a unique value for each structurally distinct molecule. The second column is a binary value representing activity (1=active, 0=inactive, NA=inconclusive or untested) for the given assay. The last column represents a score, scored by the method specified with the `addBioassay()` function. Missing or non-applicable values in any column should be represented by a `NA` value.

Methods

aid `signature(x = "bioassay")`: ...

aid<- `signature(x = "bioassay")`: ...

assay_type `signature(x = "bioassay")`: ...

assay_type<- `signature(x = "bioassay")`: ...

organism `signature(object = "bioassay")`: ...

organism<- `signature(object = "bioassay")`: ...

scores `signature(x = "bioassay")`: ...

scores<- `signature(x = "bioassay")`: ...

scoring `signature(x = "bioassay")`: ...

scoring<- `signature(x = "bioassay")`: ...

show `signature(object = "bioassay")`: ...

source_id `signature(x = "bioassay")`: ...

source_id<- `signature(x = "bioassay")`: ...

target_types `signature(x = "bioassay")`: ...

target_types<- `signature(x = "bioassay")`: ...

targets `signature(x = "bioassay")`: ...

targets<- `signature(x = "bioassay")`: ...

Author(s)

Tyler Backman

See Also

Related classes: bioassaySet, bioAssayDB.

Examples

```
showClass("bioassay")

## create a new bioassay object from sample data
data(samplebioassay)
myassay <- new("bioassay",aid="1000", source_id="test", targets="116516899",
    target_types="protein", scores=samplebioassay)
myassay
```

BioassayDB-class *Class* "BioassayDB"

Description

This class holds a connection to a bioassayR sqlite database.

Objects from the Class

Objects can be created by calls of the form `BioassayDB("datbasePath")`.

Slots

`database`**:** Object of class `"SQLiteConnection"` ~~

Methods

queryBioassayDB `signature(object = "BioassayDB")`: ...

show `signature(object = "BioassayDB")`: ...

Author(s)

Tyler Backman

See Also

Related classes: bioassaySet, bioassay.

Examples

```
showClass("BioassayDB")
```

bioassaySet-class *Class* "bioassaySet"

Description

This class stores a large number of bioactivity scores from multiple assays and experiments as a single sparse matrix.

Objects from the Class

Objects can be created with several functions including `getAssays` and `getBioassaySetByCids`.

Slots

`activity`: Object of class "dgCMatrix" a sparse matrix of assays (rows) vs compounds (columns) where 0 represents untested, NA represents inconclusive, 1 represents inactive, and 2 represents activity

`scores`: Object of class "dgCMatrix" numeric activity scores with the same dimensions as activity

`targets`: Object of class "dgCMatrix" a binary matrix of the targets (columns) for each aid (rows) listed in the activity and scores matrix. A 1 represents a target for the given assay, and a 0 represents that the given target was not used in the assay.

`sources`: Object of class "data.frame" data sources for each assay. There must be three columns titled 'source_id', 'description', and 'version.' Each row represents a data source for these data. The 'source_id' must be a numeric (integer) index that matches to those in the 'source_id' slot.

`source_id`: Object of class "integer" the source_id for each assay as an integer. The length should equal the number of rows in the activity matrix, with element names for each assay id (aid).

`assay_type`: Object of class "character" the experiment type for each assay. The length should equal the number of rows in the activity matrix, with element names for each assay id (aid).

`organism`: Object of class "character" scientific name of each target species. The length should equal the number of rows in the activity matrix, with element names for each assay id (aid).

`scoring`: Object of class "character" scoring method used in the scores matrix. The length should equal the number of rows in the activity matrix, with element names for each assay id (aid).

`target_types`: Object of class "character" type of target for each target id, where the names and order match the columns in the target matrix. The length should equal the number of rows in the activity matrix, with element names for each assay id (aid).

Methods

activity `signature(x = "bioassaySet")`: ...

activity<- `signature(x = "bioassaySet")`: ...

scores `signature(x = "bioassaySet")`: ...

scores<- `signature(x = "bioassaySet")`: ...

targets `signature(object = "bioassaySet")`: ...

targets<- `signature(object = "bioassaySet")`: ...

sources `signature(x = "bioassaySet")`: ...

sources<- `signature(x = "bioassaySet")`: ...

source_id `signature(x = "bioassaySet")`: ...

source_id<- `signature(x = "bioassaySet")`: ...

assay_type `signature(object = "bioassaySet")`: ...

assay_type<- `signature(x = "bioassaySet")`: ...

organism `signature(x = "bioassaySet")`: ...

organism<- `signature(x = "bioassaySet")`: ...

scoring `signature(x = "bioassaySet")`: ...

scoring<- `signature(x = "bioassaySet")`: ...

target_types `signature(x = "bioassaySet")`: ...

target_types<- `signature(x = "bioassaySet")`: ...

Author(s)

Tyler William H Backman

See Also

Related classes: bioassay, bioAssayDB.

Examples

```
showClass("bioassaySet")
```

---

connectBioassayDB *Create a* BioassayDB *object connected to the specified database file*

Description

This function returns a BioassayDB object for working with a pre-existing bioassayR database, already located on the users filesystem. Users can download pre-built databases for use with this feature from http://chemmine.ucr.edu/bioassayr

Usage

```
connectBioassayDB(databasePath, writeable = FALSE)
```

Arguments

databasePath	Full path to the database file to be opened.
writeable	logical. Should the database allow data to be modified and written to?

Value

`BioassayDB` for details see ?"BioassayDB-class"

Author(s)

Tyler Backman

Examples

```
## create a test database
library(bioassayR)
filename <- tempfile()
mydb <- newBioassayDB(filename, indexed=FALSE)
disconnectBioassayDB(mydb)

## connect to test database
mydb <- connectBioassayDB(filename)

## close and delete database
disconnectBioassayDB(mydb)
unlink(filename)
```

---

crossReactivityProbability

Compute the probability that compounds in a compound vs target matrix are promiscuous binders

---

Description

Queries a compound vs target sparse matrix as generated by the `perTargetMatrix` function, and computes the probability $P(theta > threshold)$ for each compound, where theta is the probability that the compound would be active in any given new assay against a novel untested target. This code implements the Bayesian Modeling of Cross-Reactive Compounds method described by Dancik, V. et al. (see references). This method assumes that the number of observed active targets out of total tested targets follows a binomial distribution. A beta conjugate prior distribution is calculated based on the hit ratios (active/total tested) for a reference database.

Usage

```
crossReactivityProbability(inputMatrix,
                           threshold=0.25,
                       prior=list(hit_ratio_mean=0.0126, hit_ratio_sd=0.0375))
crossReactivityPrior(database, minTargets=20, category=FALSE, activesOnly=FALSE)
```

Arguments

inputMatrix	A dgCMatrix sparse matrix as computed by the perTargetMatrix function with the option useNumericScores = FALSE. The cross-reactivity probability will be computed for each compound (column) based on the active and inactive scores present. In most cases, the matrix should be generated with getBioassaySetByCids rather than getAssays, so that it includes all relavent activity data for each compound, rather than a selected set of assays.
threshold	A numeric value between 0 and 1 reflecting the desired hit ratio cutoff for computing the probability a compound is a promiscuous binder. This is the probability $P(theta > threshold)$ if theta is the probability that the compound will be a hit in a new assay. The default of 0.25 was used in Dancik, V. et al. (see references).
prior	A list with elements hit_ratio_mean and hit_ratio_sd representing the mean and standard deviation of hit ratios across a large reference database of highly-screened compounds. This can be generated with crossReactivityPrior and fed to crossReactivityProbability. Computing this for a large database can take a very long time, so defaults are provided based on the April 6th 2016 version of the pre-built protein target only PubChem BioAssay database provided for use with bioassayR. Priors should be recomputed with appropriate reference data if working with a new type of experimental data, i.e. in-vivo rather than in-vitro assays.
database	A BioassayDB database to query, for calculating a prior probability distribution.
minTargets	The minimum number of distinct screened targets for a compound to be included in the prior probability distribution.
category	Include only once in prior hit ratio counts any targets which share a common annotation of this category (as used by the translateTargetId and loadIdMapping functions). For example, with the PubChem BioAssay database one could use "UniProt", "kClust", or "domains" to get selectivity by targets with unique UniProt identifiers, distinct amino acid sequences, or Pfam domains respectively (the latter is also known as domain selectivity).
activesOnly	logical. Should only compounds with at least one active score be used in computing prior? Defaults to FALSE.

Details

This function models the hit-ratio theta (fraction of distinct targets which are active) for a given compound with a standard beta-binomial bayesian model. The observed activity values for a compound tested against N targets with n actives is assumed to follow a binomial distribution:

$$p(n|theta) = \binom{N}{n} theta^n (1 - theta)^{N-n}$$

With a beta conjugate prior distribution where the parameters a and b (alpha and beta) are calculated from the prior mean and standard deviation of hit ratios for a large number of highly screened compounds as follows: $mean = a/(a+b)$ and $sd^2 = ab/((a+b)^2(a+b+1))$. This function then computes and returns the posterior probability $P(theta > threshold)$ using the beta distribution function pbeta.

Value

`crossReactivityProbability` returns an `numeric` vector containing the probability that the hit ratio (active targets / total targets) is greater than value `threshold` for each compound in the `inputMatrix`. `crossReactivityPrior` returns a `list` in the prior format described above.

Author(s)

Tyler Backman

References

Dancik, V. et al. Connecting Small Molecules with Similar Assay Performance Profiles Leads to New Biological Hypotheses. J Biomol Screen 19, 771-781 (2014).

See Also

`pbeta` for the beta distribution function. `perTargetMatrix` `targetSelectivity`

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve activity data for three compounds
assays <- getBioassaySetByCids(sampleDB, c("2244","3715","133021"))

## collapse assays into perTargetMatrix
targetMatrix <- perTargetMatrix(assays)

## compute P(theta > 0.25)
crossReactivityProbability(targetMatrix)

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

`disconnectBioassayDB` *Disconnect the database file from a* `BioassayDB` *object*

Description

This function disconnects the underlying sqlite database from a `BioassayDB` object. This is a critical step for writeable databases, but can be omitted for read only databases.

Usage

```
disconnectBioassayDB(database)
```

Arguments

database	A codeBioassayDB object to be disconnected.

Author(s)

Tyler Backman

Examples

```
## create a test database
library(bioassayR)
filename <- tempfile()
mydb <- newBioassayDB(filename, indexed=FALSE)

## disconnect from database
mydb <- connectBioassayDB(filename)

## delete database file
unlink(filename)
```

---

dropBioassay *Delete an assay from a bioassayR database*

Description

Allows the user to delete all records from the database associated with a given assay identifier.

Usage

```
dropBioassay(database, aid)
```

Arguments

database	A BioassayDB database to remove an assay from.
aid	The assay identifier string (aid), matching an aid for an assay loaded into the database.

Author(s)

Tyler Backman

Examples

```
## create sample database and load with data
myDatabaseFilename <- tempfile()
mydb <- newBioassayDB(myDatabaseFilename, indexed=FALSE)
extdata_dir <- system.file("extdata", package="bioassayR")
assayDescriptionFile <- file.path(extdata_dir, "exampleAssay.xml")
activityScoresFile <- file.path(extdata_dir, "exampleScores.csv")
myAssay <- parsePubChemBioassay("1000", activityScoresFile, assayDescriptionFile)
addDataSource(mydb, description="PubChem BioAssay", version="unknown")
loadBioassay(mydb, myAssay)

## delete the loaded assay
dropBioassay(mydb, "1000")

## disconnect from and delete sample database
disconnectBioassayDB(mydb)
unlink(myDatabaseFilename)
```

`dropBioassayIndex` *Remove index from a bioassayR database*

Description

Indexing a bioassayR database before performing queries will drastically improve query performance. However, it will also slow down loading large amounts of additional data. Therefore, it may be necessary to use this index to remove an index from a database before adding large quantities of data. Afterwards, the index can be re-generated using the `addBioassayIndex` function.

Usage

```
dropBioassayIndex(database)
```

Arguments

`database`	A `BioassayDB` database to have the index removed.

Author(s)

Tyler Backman

Examples

```
## create test database
library(bioassayR)
filename <- tempfile()
mydb <- newBioassayDB(filename, indexed=TRUE)

## remove database index
dropBioassayIndex(mydb)
```

```
## load new data into database here

## reactivate index
addBioassayIndex(mydb)

## close and delete test database
disconnectBioassayDB(mydb)
unlink(filename)
```

---

getAssay *Retrieve a bioassay*

Description

Retrieves a bioassay as a bioassay object from a bioassayR database by identifier.

Usage

```
getAssay(database, aid)
```

Arguments

database	A BioassayDB database to query.
aid	The assay identifier string (aid), matching an aid for an assay loaded into the database.

Value

A bioassay object containing the requested assay.

Author(s)

Tyler Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve an assay
assay <- getAssay(sampleDB, "673509")
assay

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

getAssays *Retrieve multiple bioassays from a database*

Description

Retrieves a list of aids as a single bioassaySet matrix object

Usage

```
getAssays(database, aids)
```

Arguments

database	A BioassayDB database to query.
aids	One or more assay identifier strings (aid), matching aid(s) for assays loaded into the database.

Value

A bioassaySet object containing data from the specified assays.

Author(s)

Tyler William H Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve three assays
assays <- getAssays(sampleDB, c("673509","103","105"))
assays

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

getBioassaySetByCids *Create* bioassaySet *sparse matrix object with activity data only for specified compounds*

Description

Takes a list of compounds, and creates a bioassaySet sparse matrix object with the activity data for these compounds only, not including activity data from other compounds in the same assays.

Usage

```
getBioassaySetByCids(database, cids)
```

Arguments

database	A BioassayDB database to query.
cids	One or more compounds IDs of interest.

Value

A bioassaySet object containing data from the specified cids.

Author(s)

Tyler William H Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve activity data on 3 compounds
activitySet <- getBioassaySetByCids(sampleDB, c("2244","3715","237"))
activitySet

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

`inactiveTargets` *Takes a single cid and returns a table of the proteins it has been found inactive against.*

Description

Returns a `data.frame` of all targets a single cid (compound) has been found inactive against, and the number of times it has been found inactive in distinct assay experiments. If a compound has been found both active and inactive in different assays, it will be listed among these results.

Usage

```
inactiveTargets(database, cid)
```

Arguments

`database`	A `BioassayDB` database to query.
`cid`	A string or integer containing a cid referring to a small molecule.

Value

A `data.frame` where the row names represent each target the specified compound shows inactivity against, and the column shows the number of assays in which it was found to be inactive.

Author(s)

Tyler Backman

See Also

`activeTargets`

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## get targets that compound 2244 shows inactivity against
myCidInactiveTargets <- row.names(inactiveTargets(sampleDB, "2244"))

## disconnect from database
disconnectBioassayDB(sampleDB)
```

loadBioassay *Add an assay to the database*

Description

Loads the results of a bioassay experiment (stored as a bioassay object) into the specified database. The data source specified in the bioassay object be added to the database with addDataSource before loading. If the assay identifier (aid) is already present in the database, an error is returned and no additional data is loaded.

Usage

```
loadBioassay(database, bioassay)
```

Arguments

database	A BioassayDB database to load the data into.
bioassay	A bioassay object containing the data to load.

Author(s)

Tyler Backman

Examples

```
## create sample database
myDatabaseFilename <- tempfile()
mydb <- newBioassayDB(myDatabaseFilename, indexed=FALSE)

## parse example assay data
extdata_dir <- system.file("extdata", package="bioassayR")
assayDescriptionFile <- file.path(extdata_dir, "exampleAssay.xml")
activityScoresFile <- file.path(extdata_dir, "exampleScores.csv")
myAssay <- parsePubChemBioassay("1000", activityScoresFile, assayDescriptionFile)

## load bioassay into database
addDataSource(mydb, description="PubChem BioAssay", version="unknown")
loadBioassay(mydb, myAssay)

## disconnect from and delete sample database
disconnectBioassayDB(mydb)
unlink(myDatabaseFilename)
```

`loadIdMapping` *Load a target identifier mapping into a bioassayR database*

Description

Loads an identifier mapping for a bioassay target (stored in the database as an NCBI GI number) to another protein target naming system. Common uses include UniProt identifiers, similarity clusters, and common names.

Usage

```
loadIdMapping(database, target, category, identifier)
```

Arguments

`database`	A writable `BioassayDB` database to insert data into.
`target`	A single protein target NCBI GI number.
`category`	The specified identifier type of the data being loaded, such as 'UniProt'.
`identifier`	A `character` object with the new identifier. This should be length one, and the function should be re-ran to add multiple identifiers.

Author(s)

Tyler Backman

References

http://www.ncbi.nlm.nih.gov/protein NCBI Protein Database http://www.uniprot.org UniProt Protein Database

See Also

`translateTargetId`

Examples

```
## create sample database
myDatabaseFilename <- tempfile()
mydb <- newBioassayDB(myDatabaseFilename, indexed=FALSE)

## load a sample translation from GI 6686268 to UniProt P11712
loadIdMapping(mydb, "6686268", "UniProt", "P11712")

## get UniProt identifier(s) for GI Number 6686268
UniProtIds <- translateTargetId(mydb, "6686268", "UniProt")
UniProtIds

## disconnect from and delete sample database
```

```
disconnectBioassayDB(mydb)
unlink(myDatabaseFilename)
```

`newBioassayDB` *Create a new bioassayR database*

Description

This function creates a new bioassayR database at the specified filesystem location, and returns a `BioassayDB` object connected to the new database.

Usage

```
newBioassayDB(databasePath, writeable = TRUE, indexed = FALSE)
```

Arguments

`databasePath`	Full path to the database file to be created.
`writeable`	logical. Should the database allow data to be modified and written to?
`indexed`	logical. Should a performance enhancing index be created? The default is false, as typically an index is added only after initial data is loaded. Data loading is much slower into an already indexed database.

Author(s)

Tyler Backman

Examples

```
## get a temporary filename
library(bioassayR)
filename <- tempfile()

## create a new bioassayR database
mydb <- newBioassayDB(filename, indexed=FALSE)

## close and delete database
disconnectBioassayDB(mydb)
unlink(filename)
```

parsePubChemBioassay *Parse PubChem Bioassay Data*

Description

Parses a PubChem Bioassay experimental result from two required files (a csv file and an XML description) into a bioassay object.

Usage

```
parsePubChemBioassay(aid, csvFile, xmlFile, duplicates = "drop",
    missingCid = "drop", scoreRegex = "inhibition|ic50|ki|gi50|ec50|ed50|lc50")
```

Arguments

aid	The assay identifier (aid) for the assay to be parsed.
csvFile	A CSV file for a given assay, as downloaded from PubChem Bioassay.
xmlFile	An XML description file for a given assay, as downloaded from PubChem Bioassay.
duplicates	Specifies how duplicate CIDs in the same assay are treated. If 'drop' is specified, only the first of each duplicated cid is kept and a warning is returned. If 'FALSE' processing will stop with an error if duplicates are present. If 'TRUE' duplicates will be included without warning, which may cause erroneous results with other bioassayR functions that assume a unique cid list for each assay.
missingCid	A value of either 'drop' or a logical value of FALSE. If 'FALSE' processing will stop with an error for any input compounds with an empty cid string. If 'drop' is specified, a warning will be issued and these compounds will be skipped.
scoreRegex	A regular expression (perl compatible, case insensitive) to be matched to the column names in the CSV header, to identify relavent score rows. If any rows match this regex, the first matching row will be used in place of the 'PUBCHEM_ACTIVITY_SCORE' and it's row name will be stored as the assays scoring method. The default will identify most PubChem Bioassays which contain protein target inhibition data. If a matching row contains all empty or non-numeric results, the next matching row is automatically used.

Value

A bioassay object containing the loaded data.

Author(s)

Tyler Backman

References

http://pubchem.ncbi.nlm.nih.gov NCBI PubChem

Examples

```
## get sample data locations
extdata_dir <- system.file("extdata", package="bioassayR")
assayDescriptionFile <- file.path(extdata_dir, "exampleAssay.xml")
activityScoresFile <- file.path(extdata_dir, "exampleScores.csv")

## parse files
myAssay <- parsePubChemBioassay("1000", activityScoresFile, assayDescriptionFile)
myAssay
```

`perTargetMatrix` *Collapse a* `bioassaySet` *object from multiple assays by combining assays with a common target*

Description

Creates a `sparseMatrix` object which has an activity value for each distinct target identifier rather than each distinct assay. Users can optionally choose how replicates are resolved. By default active scores always take preference over inactives: if any assay for a given target vs compound combination shows active, this combination will be marked active in the resulting object. Either binary activity categories or scalar numeric scores can be used. When used with numeric data, this will create a Z-score compound vs. target matrix similar to High Throughput Screening Fingerprints (HTSFPs). This function is not designed for single assays with multiple targets, and if they are present only one of the targets will be considered.

Usage

```
perTargetMatrix(assays, inactives = TRUE, assayTargets = FALSE,
    targetOrder = FALSE, summarizeReplicates = "activesFirst",
    useNumericScores = FALSE)
```

Arguments

`assays`	A `bioassaySet` object with data from multiple assays, some of which may share a common target. If used with `useNumericScores = TRUE` this should be the output of `scaleBioassaySet` if a Z-score matrix is desired.
`inactives`	A logical value. Include both active and inactive scores. If FALSE only active scores are returned. This is only used if `useNumericScores = FALSE`, with numeric scores inactives are always considered.
`assayTargets`	Provide a custom merge table of target identifiers for each assay. For example, if you have clustered the targets of many assays into bins you can here merge by common clusters instead of distinct targets. This must be vector of class `character` with names that correspond to your assay ids (aids) and values that correspond to desired targets or clusteres in the resulting matrix. Names and targets should be represented as a character, even if they are numeric names. Note that if an assay contains multiple targets, only the first is used.

`targetOrder` An optional `character` vector of desired target names in order. This will become the row names in the resulting sparse matrix in exact order, making it possible to coherently bind together sparse matricies of different compounds. If a target is omitted from this list it will be dropped in the result, and if an extra target is included it will show up with all '0' (untested) entries.

`summarizeReplicates`
Optionally allows users to choose how replicates (multiple assays sharing common compounds and targets) are resolved if they disagree. If 'activesFirst' any active score will take precedence over an inactive. If 'mode' the resulting score will be computed according to the statistical mode using `as.numeric(names(which.max(table(x))))`. Users can also optionally pass a function here which (for each cid/target pair) will receive a vector of '2' (active) and '1' (inactive) values (if `useNumericScores = TRUE`), and can then return any desired number as a summary to be included in the resulting table. For a large matrix, the default option 'activesFirst' offers the lowest computational overhead. When used with `useNumericScores = TRUE` the option 'activesFirst' will keep only the replicate with the greatest absolute value. To average across all replicates, one can pass the R function `mean`.

`useNumericScores`
A logical value. Use numeric score rather than binary data to create a scalar compound vs. target matrix. When used with the output of `scaleBioassaySet` this creates a Z-score compound vs. target matrix. `NaN` values are replaced with zeros, as these usually represent assays summarized with `scaleBioassaySet` where all compounds had an identical value. `NA` values are excluded, as these usually represent compounds that have no raw activity score. Warning: Be careful using this feature, as it can average/merge together assays scored by incompatible methods. You should confirm that the assays you are summarizing are scored and summarized in a way that makes sense.

Value

When used with `useNumericScores = FALSE` a `sparseMatrix` which contains a value of 2 for each target vs compound combination which shows activity in at least one parent assay, a value of 1 for inactive combinations, and a value of zero for untested or ambiguous values. Note that this is different from older versions of bioassayR (1.6 and older) which used to return a value of 1 for actives and did not have the option to process inactives. When used with `useNumericScores = TRUE` the raw numeric scores are returned, with replicates summarized as specified with the `summarizeReplicates` option.

Author(s)

Tyler William H Backman

See Also

Functions: `scaleBioassaySet`, `getAssays`, `bioactivityFingerprint`

Examples

```
## connect to a test database
```

```
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## option 1: retrieve all data for three compounds
assays <- getBioassaySetByCids(sampleDB, c("2244","3715","133021"))
assays

## option 2: retrieve all data for three assays
assays <- getAssays(sampleDB, c("673509","103","105"))
assays

## collapse assays into perTargetMatrix
targetMatrix <- perTargetMatrix(assays)
targetMatrix

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

`queryBioassayDB` *Perform a SQL query on a bioassayR database*

Description

Provides extreme query flexibility by allowing the user to perform any SQLite query on a bioassayR database. This allows for analysis beyond that provided by the built in query functions.

Usage

```
queryBioassayDB(object, query)
```

Arguments

`object`	A `BioassayDB` object referring to a bioassayR database.
`query`	A string containing a valid SQLite query (see SQLite documentation for more details).

Value

A `data.frame` containing the results of the specified query.

Author(s)

Tyler Backman

References

http://www.sqlite.org provides a complete reference for SQLite syntax that can be used with this function

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## inspect the structure of the database before forming a query
queryBioassayDB(sampleDB, "SELECT * FROM sqlite_master WHERE type='table'")

## find all activity data for compound cid 2244
queryBioassayDB(sampleDB, "SELECT * FROM activity WHERE cid = '2244'")

## disconnect from database
disconnectBioassayDB(sampleDB)
```

---

samplebioassay *Sample activity data for use with bioassayR*

Description

This is sample bioactivity data, taken from assay identifier (aid) 1000 in the NCBI PubChem Bioassay database. These data are provided for testing the bioassayR library.

Usage

```
data(samplebioassay)
```

Format

A data frame with activity scores for 4 distinct compounds.

`cid` unique compound identifer

`activity` 1=active, 0=inactive, NA=other

`score` activity scores

Source

http://pubchem.ncbi.nlm.nih.gov NCBI PubChem

References

http://pubchem.ncbi.nlm.nih.gov NCBI Pubchem

Examples

```
## create a new bioassay object from these sample data
data(samplebioassay)
myassay <- new("bioassay",aid="1000", source_id="PubChem BioAssay", targets="116516899",
    target_types="protein", scores=samplebioassay)
myassay
```

scaleBioassaySet *Centers and standardizes the numeric activity scores for a* bioassaySet *object (creates Z-scores)*

Description

Converts the numeric activity scores for a bioassaySet object into per-assay Z-scores. Untested '0' values are not considered in computing the value, only actives and inactives. In essence, this is a special version of the R base scale function, which ignores missing entries in a sparse matrix instead of using them as zeros. A primary purpose of this function is to pass scaled results to perTargetMatrix, in order to compute a numeric Z-score compound vs. target matrix.

Usage

```
scaleBioassaySet(bioassaySet, center=TRUE, scale=TRUE)
```

Arguments

bioassaySet	A bioassaySet object with data from multiple assays. This should be created with getAssays rather than getBioassaySetByCids, as the former includes the full assay data whereas the latter omits scores for compounds other than those specified, and therefore will not compute a coherent Z-score.
center	A logical value. If center is TRUE then centering is done by subtracting the assay means (omitting inconclusive NAs) from their corresponding scores, and if center is FALSE, no centering is done.
scale	A logical value. Scaling is done by dividing the (centered) per-assay scores by their standard deviations if center is TRUE, and the root mean square otherwise. If scale is FALSE, no scaling is done.

Value

A bioassaySet object with standardized numeric scores, that can be accessed with scores(bioassaySet).

Author(s)

Tyler William H Backman

See Also

Functions: getAssays, perTargetMatrix

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)
```

```
## retrieve three assays
assays <- getAssays(sampleDB, c("347221","53211","624349"))

## disconnect from sample database
disconnectBioassayDB(sampleDB)

## compute and return standardized scores
scaledAssays <- scaleBioassaySet(assays)

## inspect scaled and unscaled scores
scores(assays)
scores(scaledAssays)

## NOTE: this example only returns non-NA Z-scores if tried with
## real data, not the test database used here
```

`screenedAtLeast` *Return all compounds in the database screened at least 'minTargets' times.*

Description

Returns all compound cids screened against at least 'minTargets' distinct target identifiers. For a very large database (such as PubChem Bioassay) this function may take a long time to run.

Usage

```
screenedAtLeast(database, minTargets, inconclusives=TRUE)
```

Arguments

`database`	A `BioassayDB` database to query.
`minTargets`	The minimum number of distinct targets for each returned cid.
`inconclusives`	Logical. If `TRUE` (the default) screening results with inconclusive (`NA`) are counted towards `minTargets`. If `FALSE` only active and inactive results are counted.

Value

Returns a `character` vector of all CIDs meeting the specified criteria.

Author(s)

Tyler Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## get all CIDS screened against at least 2 distinct targets
highlyScreened <- screenedAtLeast(sampleDB, 2)
highlyScreened

## get all CIDS screened against at least 2 distinct targets with conclusive results
highlyScreened <- screenedAtLeast(sampleDB, 2, inconclusives=FALSE)
highlyScreened

## disconnect from database
disconnectBioassayDB(sampleDB)
```

`selectiveAgainst` *Identify small molecules with selective binding against a target of interest*

Description

Allows the user to find compounds in the database that have been screened against a large number of distinct targets, but show high binding selectivity for a specific target of interest.

Usage

```
selectiveAgainst(database, target, maxCompounds = 10, minimumTargets = 10)
```

Arguments

`database`	A `BioassayDB` database to query.
`target`	A string or integer containing a target_id referring to a target of interest.
`maxCompounds`	An integer representing the number of resulting compounds to return.
`minimumTargets`	An integer representing the minimum number of distinct targets a compound must have been screened against to be included in the results.

Value

A `data.frame` where the row names represent each compound showing binding specificity against the specified target. The first column shows the number of distinct targets each compound shows activity against, and the second column shows the total number of distinct targets it has been screened against.

Author(s)

Tyler Backman

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## find target selective compounds active against a protein of interest
selectiveAgainst(sampleDB, target="166897622", maxCompounds=10,minimumTargets=20)

## disconnect from database
disconnectBioassayDB(sampleDB)
```

---

targetSelectivity *Returns the target selectivity for a specified list of compounds (cids).*

Description

Queries a BioassayDB database and returns the target selectivity of the specified cids.

Usage

```
targetSelectivity(database, cids, scoring = "total",
                  category=FALSE, multiTarget="keepOne")
```

Arguments

database	A BioassayDB database to query.
cids	A string or integer vector containing query cids referring to a small molecules.
scoring	Must be one of two optional scoring methods "total" or "fraction". Fraction returns the target selectivity for each compound as the fraction of screened distinct targets that showed activity in at least one assay. Total returns the total number of active distinct targets for each compound, and does not consider inactive targets in the calculation. If fractional activity is requested, active values take precedence over inactives: if a target is both active and inactive in different assays it will be regarded as active.
category	Include only once in selectivity counts any targets which share a common annotation of this category (as used by the translateTargetId and loadIdMapping functions). For example, with the PubChem BioAssay database one could use "UniProt", "kClust", or "domains" to get selectivity by targets with unique UniProt identifiers, distinct amino acid sequences, or Pfam domains respectively (the latter is also known as domain selectivity).
multiTarget	Decides how selectivity is counted with regard to multi-target assays. If "drop" these assays are excluded entirely. If "keepOne" only the first target in the database is considered. If "all" they are counted separately towards the total.

Value

Returns an numeric vector containing the target selectivity for each query compound. Returned entires are named by their corresponding cid.

Author(s)

Tyler Backman

See Also

translateTargetId loadIdMapping

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## make a vector with compounds of interest
compoundsOfInterest <- c(2244, 2662, 3033)

## get "total" active targets for each compound of interest
targetSelectivity(sampleDB, compoundsOfInterest, scoring="total")

## get fraction of active targets for each compound of interest
targetSelectivity(sampleDB, compoundsOfInterest, scoring="fraction")

## disconnect from database
disconnectBioassayDB(sampleDB)
```

---

translateTargetId *Translate a protein target identifier to another identifier system*

Description

Returns a character vector of the protein target identifiers using the specified category (classification system). This is most often used to translate NCBI Protein GI numbers (as provided with the pre-build PubChem Bioassay database) into UniProt identifiers.

Usage

```
translateTargetId(database, target, category, fromCategory = "GI")
```

Arguments

database	A BioassayDB database to query.
target	A protein target identifier to query (as set by the category option, with a default of "GI").
category	The specified identifier type to return, such as 'UniProt'.
fromCategory	The identifier type of the query (default "GI").

Value

A character vector of the protein target identifiers of the category specified for the target specified. An NA is returned if no matching values exist in the database.

Author(s)

Tyler Backman

References

http://www.ncbi.nlm.nih.gov/protein NCBI Protein Database http://www.uniprot.org UniProt Protein Database

See Also

loadIdMapping

Examples

```
## create sample database
myDatabaseFilename <- tempfile()
mydb <- newBioassayDB(myDatabaseFilename, indexed=FALSE)

## load a sample translation from GI 6686268 to UniProt P11712
loadIdMapping(mydb, "6686268", "UniProt", "P11712")

## get UniProt identifier(s) for GI Number 6686268
UniProtIds <- translateTargetId(mydb, "6686268", "UniProt")
UniProtIds

## get GI identifier(s) for UniProt ID P11712
GIs <- translateTargetId(mydb, "P11712", "GI", "UniProt")
GIs

## disconnect from and delete sample database
disconnectBioassayDB(mydb)
unlink(myDatabaseFilename)
```

`trinarySimilarity` *Computes the tanimoto similarity coefficient between the bioactivity profiles of two compounds, each represented as a column in a compound vs. target sparse matrix*

Description

This computes tanimoto similarity coefficients between bioactivity profiles in a sparse matrix aware way, where only commonly tested targets are considered. The computation is trinary in that each compound is a column in a compound vs target matrix with three possible values (2=active, 1=inactive, 0=untested or inconclusive) as generated by the `perTargetMatrix` function. A comparison will return a value of NA unless one of the two minimum thresholds is satisfied, either a minimum number of shared screened targets, or a minimum number of shared active targets as performed in Dancik, V. et al. (see references).

Usage

```
trinarySimilarity(queryMatrix, targetMatrix,
    minSharedScreenedTargets = 12, minSharedActiveTargets = 3)
```

Arguments

`queryMatrix` This is a compound vs. target sparse matrix representing the bioactivity profiles for one compounds across one or more assays or targets. The format must be a `dgCMatrix` sparse matrix as computed by the `perTargetMatrix` function with the option `useNumericScores = FALSE`. This should be a single column representing the bioactivity profile for a single compound. This can be extracted from a larger compound vs. target sparse matrix with queryMatrix[,colNumber,drop=FALSE] where colNumber is the desired compound column number.

`targetMatrix` This is a compound vs. target sparse matrix representing the bioactivity profiles for one or more compounds across one or more assays or targets. The format must be `dgCMatrix` sparse matrix as computed by the `perTargetMatrix` function with the option `useNumericScores = FALSE`. Similarity will be computed between the query and each column of this matrix individually.

`minSharedScreenedTargets`
A `numeric` value specifying the minimum number of shared screened targets needed for a meaningful similarity computation. If both this threshold and `minSharedActiveTargets` are unsatisfied, the returned result will be `NA` instead of a computed value. The default of 12 was determined taken from Dancik, V. et al. (see references) as experimentally determined to result in meaningful predictions.

`minSharedActiveTargets`
A `numeric` value specifying the minimum number of shared active targets needed for a meaningful similarity computation. If both this threshold and `minSharedScreenedTargets` are unsatisfied, the returned result will be `NA` instead of a computed value. The

default of 3 was determined taken from Dancik, V. et al. (see references) as experimentally determined to result in meaningful predictions.

Value

A `numeric` vector where each element represents the tanimoto similarity between the `queryMatrix` and a given row in the `targetMatrix` where only the shared set of commonly screened targets is considered. If both the `minSharedScreenedTargets` and `minSharedActiveTargets` thresholds are unsatisfied, an `NA` will be returned for the given similarity value. An `NA` will also be returned if the tanimoto coefficient is undefined due to a zero in the denominator, which occurs when neither compound was found active against any of the commonly screened targets.

Author(s)

Tyler Backman

References

Tanimoto similarity coefficient: Tanimoto TT (1957) IBM Internal Report 17th Nov see also Jaccard P (1901) Bulletin del la Societe Vaudoisedes Sciences Naturelles 37, 241-272.

Dancik, V. et al. Connecting Small Molecules with Similar Assay Performance Profiles Leads to New Biological Hypotheses. J Biomol Screen 19, 771-781 (2014).

See Also

`perTargetMatrix` `getBioassaySetByCids` `bioactivityFingerprint`

Examples

```
## connect to a test database
extdata_dir <- system.file("extdata", package="bioassayR")
sampleDatabasePath <- file.path(extdata_dir, "sampleDatabase.sqlite")
sampleDB <- connectBioassayDB(sampleDatabasePath)

## retrieve activity data for three compounds
assays <- getBioassaySetByCids(sampleDB, c("2244","3715","133021"))

## collapse assays into perTargetMatrix
targetMatrix <- perTargetMatrix(assays)

## compute similarity between first column and all columns
queryMatrix <- targetMatrix[,1,drop=FALSE]
trinarySimilarity(queryMatrix, targetMatrix)

## disconnect from sample database
disconnectBioassayDB(sampleDB)
```

Index

∗ **classes**
  bioassay-class, 11
  BioassayDB-class, 13
  bioassaySet-class, 14
∗ **datasets**
  samplebioassay, 32
∗ **utilities**
  activeAgainst, 3
  activeTargets, 4
  addBioassayIndex, 5
  addDataSource, 6
  allCids, 7
  allTargets, 8
  assaySetTargets, 9
  bioactivityFingerprint, 10
  connectBioassayDB, 15
  crossReactivityProbability, 16
  disconnectBioassayDB, 18
  dropBioassay, 19
  dropBioassayIndex, 20
  getAssay, 21
  getAssays, 22
  getBioassaySetByCids, 23
  inactiveTargets, 24
  loadBioassay, 25
  loadIdMapping, 26
  newBioassayDB, 27
  parsePubChemBioassay, 28
  perTargetMatrix, 29
  queryBioassayDB, 31
  scaleBioassaySet, 33
  screenedAtLeast, 34
  selectiveAgainst, 35
  targetSelectivity, 36
  translateTargetId, 37
  trinarySimilarity, 39

activeAgainst, 3
activeTargets, 4, *24*
activity *(*bioassaySet-class*)*, 14
activity,bioassaySet-method *(*bioassaySet-class*)*, 14
activity<- *(*bioassaySet-class*)*, 14
activity<-,bioassaySet-method *(*bioassaySet-class*)*, 14
addBioassayIndex, 5
addDataSource, 6
aid *(*bioassay-class*)*, 11
aid,bioassay-method *(*bioassay-class*)*, 11
aid<- *(*bioassay-class*)*, 11
aid<-,bioassay-method *(*bioassay-class*)*, 11
allCids, 7
allTargets, 8
assay_type *(*bioassay-class*)*, 11
assay_type,bioassay-method *(*bioassay-class*)*, 11
assay_type,bioassaySet-method *(*bioassaySet-class*)*, 14
assay_type<- *(*bioassay-class*)*, 11
assay_type<-,bioassay-method *(*bioassay-class*)*, 11
assay_type<-,bioassaySet-method *(*bioassaySet-class*)*, 14
assaySetTargets, 9

bioactivityFingerprint, 10, *40*
bioassay *(*bioassay-class*)*, 11
bioassay-class, 11
BioassayDB-class, 13
bioassaySet-class, 14

connectBioassayDB, 15
crossReactivityPrior *(*crossReactivityProbability*)*, 16
crossReactivityProbability, 16

disconnectBioassayDB, 18
dropBioassay, 19

dropBioassayIndex, 20

getAssay, 21
getAssays, 22
getBioassaySetByCids, 23, *40*

inactiveTargets, *4*, 24

loadBioassay, 25
loadIdMapping, 26, *37, 38*

newBioassayDB, 27

organism *(*bioassay-class*)*, 11
organism,bioassay-method
*(*bioassay-class*)*, 11
organism,bioassaySet-method
*(*bioassaySet-class*)*, 14
organism<- *(*bioassay-class*)*, 11
organism<-,bioassay-method
*(*bioassay-class*)*, 11
organism<-,bioassaySet-method
*(*bioassaySet-class*)*, 14

parsePubChemBioassay, 28
pbeta, *18*
perTargetMatrix, *18*, 29, *40*

queryBioassayDB, 31
queryBioassayDB,BioassayDB-method
*(*BioassayDB-class*)*, 13

samplebioassay, 32
scaleBioassaySet, 33
scores *(*bioassay-class*)*, 11
scores,bioassay-method
*(*bioassay-class*)*, 11
scores,bioassaySet-method
*(*bioassaySet-class*)*, 14
scores<- *(*bioassay-class*)*, 11
scores<-,bioassay-method
*(*bioassay-class*)*, 11
scores<-,bioassaySet-method
*(*bioassaySet-class*)*, 14
scoring *(*bioassay-class*)*, 11
scoring,bioassay-method
*(*bioassay-class*)*, 11
scoring,bioassaySet-method
*(*bioassaySet-class*)*, 14
scoring<- *(*bioassay-class*)*, 11
scoring<-,bioassay-method
*(*bioassay-class*)*, 11
scoring<-,bioassaySet-method
*(*bioassaySet-class*)*, 14
screenedAtLeast, 34
selectiveAgainst, 35
show *(*bioassay-class*)*, 11
show,bioassay-method *(*bioassay-class*)*,
11
show,BioassayDB-method
*(*BioassayDB-class*)*, 13
show,bioassaySet-method
*(*bioassaySet-class*)*, 14
source_id *(*bioassay-class*)*, 11
source_id,bioassay-method
*(*bioassay-class*)*, 11
source_id,bioassaySet-method
*(*bioassaySet-class*)*, 14
source_id<- *(*bioassay-class*)*, 11
source_id<-,bioassay-method
*(*bioassay-class*)*, 11
source_id<-,bioassaySet-method
*(*bioassaySet-class*)*, 14
sources *(*bioassaySet-class*)*, 14
sources,bioassaySet-method
*(*bioassaySet-class*)*, 14
sources<- *(*bioassaySet-class*)*, 14
sources<-,bioassaySet-method
*(*bioassaySet-class*)*, 14

target_types *(*bioassay-class*)*, 11
target_types,bioassay-method
*(*bioassay-class*)*, 11
target_types,bioassaySet-method
*(*bioassaySet-class*)*, 14
target_types<- *(*bioassay-class*)*, 11
target_types<-,bioassay-method
*(*bioassay-class*)*, 11
target_types<-,bioassaySet-method
*(*bioassaySet-class*)*, 14
targets *(*bioassay-class*)*, 11
targets,bioassay-method
*(*bioassay-class*)*, 11
targets,bioassaySet-method
*(*bioassaySet-class*)*, 14
targets<- *(*bioassay-class*)*, 11
targets<-,bioassay-method
*(*bioassay-class*)*, 11

targets<-,bioassaySet-method
(bioassaySet-class), 14
targetSelectivity, *18*, 36
translateTargetId, *26*, *37*, 37
trinarySimilarity, 39