Package 'bioassayR'

October 8, 2015

Type Package

Title R library for Bioactivity analysis

Version 1.6.1

Date 2015-6-10

Author Tyler Backman, Ronly Schlenk, Thomas Girke

Maintainer Tyler Backman <tyler.backman@ucr.edu>

Depends R (>= 3.1.0), DBI (>= 0.3.1), RSQLite (>= 1.0.0), methods, Matrix, rjson, BiocGenerics (>= 0.13.8)

Imports XML

Suggests BiocStyle, RCurl, ape, ChemmineR, cellHTS2

Description bioassayR provides tools for statistical analysis of small molecule bioactivity data

License Artistic-2.0

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

LazyLoad yes

Collate AllClasses.R AllGenerics.R BioassayDB-accessors.R bioassay-accessors.R loadingData.R queries.R bioassaySet-accessors.R

NeedsCompilation no

R topics documented:

`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 protein targets (target_type for the protein must be 'protein'), 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)
```

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 4 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 represents the structure id (sid) which is often used to identify distinct sources of samples of small molecules carrying a common cid and thought to be structurally identical. The third column is a binary value representing activity (1=active, 0=inactive) 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

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=samp
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

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 binary sparse matrix of assays vs compounds where 0 represents inactive and 1 represents activity

`scores`: Object of class `"dgCMatrix"` numeric activity scores

`targets`: Object of class `"dgCMatrix"` a matrix of the targets for each aid listed in the activity and scores matrix

`sources`: Object of class `"data.frame"` data sources for each assay

`replicates`: Object of class `"factor"` a factor signaling which assays are replicates

`source_id`: Object of class `"integer"` the source_id for each assay as an integer

`assay_type`: Object of class `"character"` the experiment type for each assay

`organism`: Object of class `"character"` scientific name of each target species

`scoring`: Object of class `"character"` scoring method used in the scores matrix

`target_types`: Object of class `"character"` type of target used in assay

Methods

No methods defined with class "bioassaySet" in the signature.

Author(s)

Tyler William H Backman

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 = F)
```

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)
```

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.

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 = T, indexed = F)
```

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)
```

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.

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 binary sparseMatrix object which has an activity value for each distinct target identifier rather than each distinct assay. If any assay for a given target vs compound combination shows active, this combination will be marked active in the resulting object.

Usage

```
perTargetMatrix(assays)
```

Arguments

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

Value

A sparseMatrix which contains a value of 1 for each target vs compound combination which shows activity in at least one parent assay.

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

## 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

`sid` structure identifier

`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="test", targets="116516899", target_types="protein", scores=samp
myassay
```

`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)
```

Arguments

`database`	A `BioassayDB` database to query.
`minTargets`	The minimum number of distinct targets for each returned cid.

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

## 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")
```

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.

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

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 GI 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)
```

Arguments

`database`	A `BioassayDB` database to query.
`target`	A single protein target GI number.
`category`	The specified identifier type to return, such as 'UniProt'.

Value

A `character` vector of the protein target identifiers of the category specified for the target specified.

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
```

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

Index