Package 'customCMPdb'

May 22, 2026

Title Customize and Query Compound Annotation Database

Description This package serves as a query interface for important community collections of small molecules, while also allowing users to include custom compound collections.

Version 1.23.0

Depends R (>= 4.0)

Imports AnnotationHub, RSQLite, XML, utils, ChemmineR, methods, stats, rappdirs, BiocFileCache

Suggests knitr, rmarkdown, testthat, BiocStyle

License Artistic-2.0

biocViews Software, Cheminformatics,AnnotationHubSoftware

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/yduan004/customCMPdb/

BugReports https://github.com/yduan004/customCMPdb/issues

RoxygenNote 7.1.2

NeedsCompilation no

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

git_branch devel

git_last_commit 055aeb1

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-22

Author Yuzhu Duan [aut, cre], Thomas Girke [aut]

Maintainer Yuzhu Duan <yduan004@ucr.edu>

Contents

`customCMPdb-package` *Customize and Query Compound Annotation Database*

Description

This package is served as the query and customization interface for compound annotations from DrugAge, DrugBank, CMAP02 and LINCS databases. It also stores the structure SDF datasets for compounds in the above four databases.

Specifically, the annotation database created by this package is an SQLite database containing 5 tables, including 4 compound annotation tables from DrugAge, DrugBank, CMAP02 and LINCS databases, respectively. The other one is an ID mapping table of ChEMBL IDs to IDs of individual databases. The other 4 datasets stores the structures of compounds in the DrugAge, DrugBank, CMAP02 and LINCS databases in SDF files. For detailed description of the 5 datasets generated by this package, please consult to the vignette of this package by running `browseVignettes("customCMPdb")`. The actual datasets are hosted in `AnnotationHub`.

This package also provides functionalities to customize and query the compound annotation SQLite database. Users could add their customized compound annotation tables to the SQLite database and query both the default (DrugAge, DrugBank, CMAP02, LINCS) and customized annotations by providing ChEMBL ids of the query compounds. The customization and query functions are available at `customAnnot` and `queryAnnotDB`, respectively.

Details

The description of the 5 datasets in this package is as follows.

Annotation SQLite database:

It is a SQLite database storing compound annotation tables for DrugAge, DrugBank, CMAP02 and LINCS, respectively. It also contains an ID mapping table of ChEMBL ID to IDs of individual databases.

DrugAge SDF:

It is an SDF (Structure-Data File) file storing molecular structures of DrugAge compounds. The source DrugAge annotation file was downloaded from here. The extracted csv file only contains drug names, without id mappings to external resources such as PubChem or ChEMBL. The extracted 'drugage.csv' file was further processed by the `processDrugage` function in this package. The result DrugAge annotation table as well as the id-mapping table (DrugAge internal id to ChEMBL ID) were then stored in the SQLite annotation database named as 'compoundCollection'. The drug structures were obtained from PubChem CIDs by `getIds` function from **ChemmineR** package. The `SDFset` object was then written to the `drugage_build2.sdf` file

DrugBank SDF:

This SDF file stores structures of compounds in DrugBank database. The full DrugBank xml file was downloaded from `https://www.drugbank.ca/releases/latest`. The most recent release

version at the time of writing this document is 5.1.5. The extracted xml file was processed by the dbxml2df function in this package. The result DrugBank annotation table was then stored in the compoundCollection SQLite database. The DrugBank to ChEMBL id mappings were obtained from UniChem. The DrugBank SDF file was downloaded from https://www.drugbank.ca/releases/latest#structures. Some validity checks and modifications were made via utilities in the **ChemmineR** package. The results were written to the drugbank_5.1.5.sdf file

CMAP SDF:

The CMAP compound instance table was downloaded from CMAP02 website and processed by the buildCMAPdb function in this package. The result 'cmap.db' contains both compound annotation and structure information. Since the annotation table only contains PubChem CID, the ChEMBL ids were added via PubChem CID to ChEMBL id mappings from UniChem. The CMAP internal IDs were made for ChEMBL id to CMAP id mappings. The structures were written to the cmap02.sdf file

LINCS SDF:

The LINCS compound annotation table was downloaded from GEO. where only compounds type were selected. The LINCS ids were mapped to ChEMBL ids via inchi key. The LINCS compounds structures were obtained from PubChem CIDs via getIds function from **ChemmineR** package. The structures were written to the lincs_pilot1.sdf file

The R script of generating the above 5 datasets is available at the 'inst/scripts/make-data.R' file in this package. The file location can be found by running system.file("scripts/make-data.R",package="customCMP in user's R session or from the GitHub repository of this package.

Author(s)

- Yuzhu Duan (yduan004@ucr.edu)
- Thomas Girke (thomas.girke@ucr.edu)

See Also

customAnnot, queryAnnotDB

Examples

```
library(AnnotationHub)
## Not run:
    ah <- AnnotationHub()

    ## Load compoundCollection annotation SQLite database
    query(ah, c("customCMPdb", "annot_0.1"))
    annot_path <- ah[["AH79563"]]
    library(RSQLite)
    conn <- dbConnect(SQLite(), annot_path)
    dbListTables(conn)
    drugAgeAnnot <- dbReadTable(conn, "drugAgeAnnot")
    head(drugAgeAnnot)
    dbDisconnect(conn)

    ## Load DrugAge SDF file
    query(ah, c("customCMPdb", "drugage_build2"))
    da_path <- ah[["AH79564"]]
    da_sdfset <- ChemmineR::read.SDFset(da_path)

    ## Load DrugBank SDF file
```

```
    query(ah, c("customCMPdb", "drugbank_5.1.5"))
    db_path <- ah[["AH79565"]]
    db_sdfset <- ChemmineR::read.SDFset(db_path)

    ## Load CMAP SDF file
    query(ah, c("customCMPdb", "cmap02"))
    cmap_path <- ah[["AH79566"]]
    cmap_sdfset <- ChemmineR::read.SDFset(cmap_path)

    ## Load LINCS SDF file
    query(ah, c("customCMPdb", "lincs_pilot1"))
    lincs_path <- ah[["AH79567"]]
    lincs_sdfset <- ChemmineR::read.SDFset(lincs_path)

## End(Not run)
```

`addCustomAnnot` *Add/Delete Custom Annotation*

Description

Functions could be used to add/delete user's custom compound annotations from the annotation SQLite database. The added custom compound annotation table should contains a column named as `chembl_id` that represents the ChEMBL ids of the added compounds.

The `listAnnot` function lists the available annotation resources in the SQLite annotation database.

The `defaultAnnot` function sets the annotation SQLite database to the default one by deleting the existing one and re-downloading from AnnotationHub.

Usage

```
addCustomAnnot(annot_tb, id_col = NULL, annot_name, overwrite = FALSE)

deleteAnnot(annot_name)

listAnnot()

defaultAnnot()
```

Arguments

`annot_tb`	data.frame representing the custom annotation table, Note, it should contains a 'chembl_id' column representing the compound ChEMBL ids
`id_col`	column name in `annot_tb` that is used as ID column, this column must contain unique identifiers. If not defined, an automatically generated ID column will be appended.
`annot_name`	character(1), user defined name of the annotation table
`overwrite`	a logical specifying whether to overwrite an existing table or not. Its default is FALSE.

Value

character vector of names of the annotation tables in the SQLite DB

character(1), path to the annotation SQLite database

Examples

```
chembl_id <- c("CHEMBL1000309", "CHEMBL100014", "CHEMBL10",
               "CHEMBL100", "CHEMBL1000", NA)
annot_tb <- data.frame(compound_name=paste0("name", 1:6),
        chembl_id=chembl_id,
        feature1=paste0("f", 1:6),
        feature2=rnorm(6))
addCustomAnnot(annot_tb, annot_name="mycustom3")
deleteAnnot("mycustom3")
annot_names <- listAnnot()
# defaultAnnot()
```

---

buildCMAPdb *Build CMAP Database*

Description

This function builds a SQLite database named as 'cmap.db' that contains id mappings of cmap names to PubChem/DrugBank IDs as well as compound structure information.

Usage

```
buildCMAPdb(dest_dir = ".")
```

Arguments

dest_dir	character(1), destination directory under which the result SQLite database named as 'cmap.db' stored. The default is user's current working directory.

Details

For about 2/3 of the CMAP drugs, one can obtain their PubChem/DrugBank IDs from the DMAP site here: http://bio.informatics.iupui.edu/cmaps. Since this website is no longer supported, the processed CMAP name to PubChem and DrugBank ID mapping table is stored under the "inst/extdata" folder of this package named as "dmap_unique.txt". The SMILES strings for CMAP entries were obtained from ChemBank. Compounds were matched by names using the 'stringdist' library where cmap_name from CMAP were mapped to the closest name in ChemBank.

Value

write "cmap.db" SQLite database to the destination directory defined by user.

Examples

```
library(ChemmineR)
## Query database
# buildCMAPdb(dest_dir="./inst/scripts")
# conn <- initDb("/inst/scripts/cmap.db")
# results <- getAllCompoundIds(conn)
# sdfset <- getCompounds(conn, results, keepOrder=TRUE)
# sdfset
# as.data.frame(datablock2ma(datablock(sdfset)))[1:4,]
# myfeat <- listFeatures(conn)
# feat <- getCompoundFeatures(conn, results, myfeat)
# feat[1:4,]
```

---

buildDrugAgeDB *Build DrugAge Annotation Database*

Description

This function builds the DrugAge annotation SQLite database from the 'drugage_id_mapping' table stored in the 'inst/extdata' directory of this package. The 'drugage_id_mapping.tsv' table contains the DrugAge compounds annotation information (such as species, avg_lifespan_change etc) as well as the compound name to ChEMBL id and PubChem id mappings.

Usage

```
buildDrugAgeDB(
  da_path = system.file("extdata/drugage_id_mapping.tsv", package = "customCMPdb"),
   dest_path
)
```

Arguments

da_path	character(1), file path to the tabular file generated from processDrugage function and manually edited the missing IDs.
dest_path	character(1), destination path of the result DrugAge annotation SQLite database

Details

Part of the id mappings in the 'drugage_id_mapping.tsv' table is generated by the processDrugage function for compound names that have ChEMBL ids from the ChEMBL database (version 24). The missing IDs were added manually. A semi-manual approach was to use this web service: https://cts.fiehnlab.ucdavis.edu/batch. After the semi-manual process, the left ones were manually mapped to ChEMBL, PubChem and DrugBank ids. The mixed items were commented.

Value

DrugAge annotation SQLite database

Examples

```
buildDrugAgeDB(dest_path=tempfile(fileext="_drugage.db"))
```

dbxml2df *Convert drugbank database (xml file) into dataframe.*

Description

Download the original DrugBank database at http://www.drugbank.ca/releases/latest (xml file) into your current working directory and rename as "drugbank.xml" then run: `drugbank_df = dbxml2df(xmlfile="drugbank.xml", version="5.0.10")`.

Usage

```
dbxml2df(xmlfile, version)
```

Arguments

xmlfile	Character(1), file path to the xml file downloaded from the DrugBank website at https://www.drugbank.ca/releases/latest
version	Character(1), DrugBank version of the xml file

Value

Dataframe of drugbank xml database.

Note

This process with take about 20 minutes.

Author(s)

Yuzhu Duan yduan004@ucr.edu

References

http://www.drugbank.ca/releases/latest

See Also

df2SQLite

Examples

```
library(XML)
## Not run:
## download the original drugbank database at
\url{http://www.drugbank.ca/releases/latest} (xml file)
## into your current directory and rename as drugbank.xml

## convert drugbank database (xml file) into dataframe:
drugbank_df <- dbxml2df(xmlfile="drugbank.xml", version="5.0.10")

## End(Not run)
```

df2SQLite *Store drugbank dataframe into an SQLite database*

Description

Store specific version of drugbank dataframe into an SQLite database under user defined directory, the default is user's present working directory of R session

Usage

```
df2SQLite(dbdf, version, dest_dir = ".")
```

Arguments

dbdf	Drugbank dataframe generated by dbxml2df function.
version	Character(1), version of the input drugbank dataframe generated by dbxml2df function
dest_dir	Character(1), destination directory that the result SQLite database stored in. The default is user's current working directory

Value

SQLite database named as "drugbank_<versionNumber>.db" stored under user's present working directory of R session or user's specified directory.

Author(s)

Yuzhu Duan yduan004@ucr.edu

See Also

dbxml2df

Examples

```
library(RSQLite)
## Not run:
# download the original drugbank database (http://www.drugbank.ca/releases/latest) (xml file)
# to your current R working directory, and rename as "drugbank.xml".
# Read in the xml file and convert to a data.frame in R

drugbank_df = dbxml2df(xmlfile="drugbank.xml", version="5.1.5")

# store the converted drugbank dataframe into SQLite database under user's
present R working direcotry, or other directory defined by 'dest_dir'

df2SQLite(dbdf=drugbank_df, version="5.1.5") # set version as version of xml file

## End(Not run)
```

loadAnnot *Load Compound Annotation Database*

Description

The compound annotation tables from different databases/sources are stored in one SQLite database. This function can be used to load the SQLite annotation database

Usage

```
loadAnnot()
```

Examples

```
conn <- loadAnnot()
```

loadSDFwithName *Load Compound Structures from Four Resources*

Description

This function could be used to get SDFset of compounds in CMAP2, LINCS 2017, DrugAge build 2 or DrugBank 5.1.5 databases. The `cid` of the SDFset are compound names instead of their internal IDs.

Usage

```
loadSDFwithName(source = "LINCS")
```

Arguments

`source`	character(1), one of "CMAP2", "LINCS", "DrugBank", "DrugAge"

Value

SDFset object of compounds in the `source` database, the `cid` of the SDFset are compound names.

See Also

SDFset

Examples

```
da_sdf <- loadSDFwithName(source="DrugAge")
```

processDrugage *Process Source DrugAge Dataset*

Description

This function processes the source DrugAge datasets by adding the ChEMBL, PubChem and DrugBank id mapping information to the source DrugAge table which only has compound names without id mapping information. Source file of DrugAge is linked here: http://genomics.senescence.info/drugs/dataset.zip

Usage

```
processDrugage(dest_file = "drugage_id_mapping.tsv", verbose = TRUE)
```

Arguments

dest_file	character(1), file path to the generated DrugAge annotation tabular file with id mappings. The default will write the file named as "drugage_id_mapping.tsv" to user's current working directory.
verbose	logical(1), If descriptive message and list of issues should be included as output

Details

This function only annotates compound names that have ChEMBL ids from the ChEMBL database (version 24). The missing IDs were added manually. A semi-manual approach was to use this web service: https://cts.fiehnlab.ucdavis.edu/batch. After the semi-manual process, the left ones were manually mapped to ChEMBL, PubChem and DrugBank ids. The mixed items were commented.

Value

write the default 'drugage_id_mapping.tsv' file to user's current working directory or the file path defined by users to the dest_dafile argument.

Examples

```
library(ChemmineR)
## Not run:
processDrugage(dest_file="drugage_id_mapping.tsv")
# Now the missing IDs need to be added manually. A semi-manual approach is to
# use this web service: https://cts.fiehnlab.ucdavis.edu/batch

## End(Not run)
```

queryAnnotDB *Annotation Queries with Compound IDs*

Description

This function can be used to query compound annotations from the default resources as well as the custom resources stored in the SQLite annotation database. The default annotation resources are DrugAge, DrugBank, CMAP02 and LINCS. The customized compound annotations could be added/deleted by the `customAnnot` utilities.

Usage

```
queryAnnotDB(
  chembl_id,
  annot = c("drugAgeAnnot", "DrugBankAnnot", "cmapAnnot", "lincsAnnot")
)
```

Arguments

`chembl_id`	character vector of ChEMBL IDs or compound ids from other annotation system..
`annot`	character vector of annotation resources, such as `drugAgeAnnot`, `DrugBankAnnot`, `cmapAnnot`, `lincsAnnot` or names of the annotation tables added by users. The `listAnnot` function lists the available options for the `annot` argument names.

Details

The input of this query function could be a set of ChEMBL IDs, it returns a data.frame storing annotations of the input compounds from selected annotation resources defined by the `annot` argument.

Since in the SQLite annotation database, ID identifiers from different ID systems, such as DrugBank and LINCS, are connected by ChEMBL IDs, it is hard to tell whether two IDs, such as DB00341, BRD-A42571354, refer to the same compound if either of them lack ID mappings to ChEMBL. So for querying compounds that don't have ChEMBL IDs, only one isolated database where the compounds belong to are supported. For example, a compound with LINCS id as "BRD-A00150179" doesn't have the ChEMBL ID mapping, when it is passed to the 'chembl_id' argument, the 'annot' need only to be set as 'lincsAnnot' and the result will be the compound annotation table from the LINCS annotation.

Value

data.frame of annotation result

Examples

```
query_id <- c("CHEMBL1000309", "CHEMBL100014", "CHEMBL100109",
              "CHEMBL100", "CHEMBL1000", "CHEMBL10")
qres <- queryAnnotDB(query_id, annot=c("drugAgeAnnot", "lincsAnnot"))

# Add a custom compound annotation table
chembl_id <- c("CHEMBL1000309", "CHEMBL100014", "CHEMBL10",
```

```
               "CHEMBL100", "CHEMBL1000", NA)
annot_tb <- data.frame(compound_name=paste0("name", 1:6),
        chembl_id=chembl_id,
        feature1=paste0("f", 1:6),
        feature2=rnorm(6))
addCustomAnnot(annot_tb, annot_name="myCustom2")

# query custom annotation
qres2 <- queryAnnotDB(query_id, annot=c("lincsAnnot", "myCustom2"))

# query compounds that don't have ChEMBL IDs
query_id <- c("BRD-A00474148", "BRD-A00150179", "BRD-A00763758", "BRD-A00267231")
qres3 <- queryAnnotDB(chembl_id=query_id, annot=c("lincsAnnot"))
qres3
```

Index