Package ‘MetaDBparse’

May 3, 2021

Type Package
Title Annotate Mass over Charge Values with Databases and Formula Prediction
Version 2.0.0
Author Joanna Wolthuis
Maintainer Joanna Wolthuis <jcwolthuis93@gmail.com>
Description Provides parsing functionality for over 30 metabolomics databases, with most available without having to create an account on given websites. Once parsed, calculates given adducts and isotope patterns and inserts into one big database which can be used to annotate unknown m/z values. Furthermore, formulas can be predicted for a given m/z, and these can be matched to ChemSpider, PubChem, SUPERNATURAL II, KNApSAcK and ChemIDplus for further annotation. Current databases available: HMDB, ChEBI, LMDB, BMDB, MCDB, ECMDB, Wiki-data, mVOC, VMH, T3DB, Exposome Explorer, FooDB, MetaCyc (requires account), DrugBank (requires account), ReSPECT, MaConDa, Blood Exposome DB, KEGG, SM-PDB, LIPIID MAPS, MetaboLights, DimeDB, Phenol Explorer, MassBank, YMDB, PAMDB, ANPDB, Metabolomics Workbench, PharmGKB, Reactome, mVOC and STOFF. Featured in the 'MetaboShiny' package (Wolthuis, J. (2019) <doi:10.1007/s11306-020-01717-8>).
License Apache License 2.0
Encoding UTF-8
LazyData true
Depends R (>= 4.0.0), rvest, RCurl, XML, SPARQL
Imports methods, pacman, rdk, rJava, parallel, phapply, enviPat, data.table, RSQLite, DBI, gsubfn, utils, base, stringr, WikiDataQueryServiceR, webchem, jsonlite, R.utils, KEGGREST, zip, ChemmineR, xml2, stringi, reshape2, Hmisc, httr, RJSONIO, readxl, cmmr, progress, Rdisop, rlist
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2021-05-03 14:00:11 UTC
### R topics documented:

- adducts ........................................ 3
- adduct_rules .................................. 4
- build.ANPDB .................................. 4
- build.BLOODEXPOSOME ...................... 5
- build.BMDB .................................. 5
- build.CHEBI .................................. 6
- build.DIMEDB .................................. 7
- build.DRUGBANK ................................ 7
- build.ECMDB ................................... 8
- build.EXPOSOMEEXPLORER .................... 9
- build.FOODB ................................... 9
- build.HMDB ................................... 10
- build.KEGG ................................... 11
- build.LIPIDMAPS ................................ 11
- build.LMDB ................................... 12
- build.MACONDA ................................ 13
- build.MASSBANK ................................ 13
- build.MCDB ................................... 14
- build.METABOLIGHTS ................. 15
- build.METACYC ................................ 15
- build.mVOC ................................... 16
- build.PAMDB .................................. 17
- build.PHARMGKB ................................ 17
- build.PHENOLEXPLORER ..................... 18
- build.REACTOME ................................ 19
- build.RESPECT .................................. 19
- build.RMDB ................................... 20
- build.SMPDB .................................. 21
- build.STOFF ................................... 21
- build.T3DB ................................... 22
- build.VMH ................................... 23
- build.WIKIDATA .............................. 23
- build.WIKIPATHWAYS ....................... 24
- build.YMDB ................................... 25
- buildBaseDB ................................... 25
- buildExtDB ................................... 27
- checkAdductRule ......................... 28
- chemspiderInfo .............................. 29
- cleanDB ................................... 29
- countAdductRuleMatches ................... 30
- doAdduct ................................... 31
- doBT ................................... 31
- doIsotopes ................................... 32
- downloadBT ................................... 33
- filterFormula .................................. 34
- getFormula ................................... 34
adducts

getPredicted .................................................. 35
iatom.to.charge .................................................. 36
iatom.to.formula ............................................... 37
iatom.to.smiles .................................................. 38
is.empty ......................................................... 38
isotopes .......................................................... 39
lmdb ............................................................... 39
openBaseDB ....................................................... 40
pubChemInfo ....................................................... 40
removeDB ......................................................... 41
removeFailedStructures ......................................... 41
revertAdduct ..................................................... 42
sdfStream.joanna ............................................... 42
searchCMMR ....................................................... 43
searchFormula ................................................... 44
searchFormulaWeb ................................................ 45
searchMZ .......................................................... 46
searchMZonline ................................................... 47
searchRev ......................................................... 48
showAllBase ...................................................... 49
smiles.to.iatom .................................................. 49
writeDB .......................................................... 50

Index 51

adducts  Adduct table

Description
Table with all adducts included by default in MetaDBparse.

Usage
adducts

Format
An object of class data.table (inherits from data.frame) with 69 rows and 11 columns.
adduct_rules

*Adduct rule table*

**Description**
Table with all adduct rules included by default in MetaDBparse.

**Usage**
adduct_rules

**Format**
An object of class `data.table` (inherits from `data.frame`) with 10 rows and 3 columns.

---

build.ANPDB

*Build ANPDB*

**Description**
Parses the ANPDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**
`build.ANPDB(outfolder, testMode = FALSE)`

**Arguments**
- `outfolder` Which folder to save temp files to?
- `testMode` run in test mode? Only parses first ten compounds

**Value**
data table with parsed database

**See Also**
download.file fread

**Examples**
```r
## Not run: build.ANPDB(outfolder=tempdir(), testMode=TRUE)
```
build.BLOODEXPOSOME Build BLOOD EXPOSOME DB

Description

Parses the BLOOD EXPOSOME DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.BLOODEXPOSOME(outfolder, testMode = FALSE)

Arguments

outfolder Which folder to save temp files to?
testMode run in test mode? Only parses first ten compounds

Value
data table with parsed database

See Also
download.file read.xlsx pbapply read_json

Examples

## Not run: build.BLOODEXPOSOME(outfolder=tempdir(), testMode=TRUE)

build.BMDB Build BMDB

Description

Parses the BMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.BMDB(outfolder, testMode = FALSE)

Arguments

outfolder Which folder to save temp files to?
testMode run in test mode? Only parses first ten compounds
Value

data table with parsed database

See Also

getURL str_match download.file unzip pboptions connections xmlValue, xmlEventParse

Examples

## Not run: build.BMDB(outfolder=tempdir(), testMode=TRUE)

---

### Build CHEBI

Parses CHEBI, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

#### Usage

```r
build.CHEBI(outfolder)
```

#### Arguments

- **outfolder** Which folder to save temp files to?

#### Value

data table with parsed database

#### See Also

getURL download.file as.data.table datablock2ma, datablock

#### Examples

```r
## Not run: build.CHEBI(outfolder=tempdir())
```
build.DIMEDB

**Build DIMEDB**

**Description**

Parses the DIMEDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```
build.DIMEDB(outfolder, testMode = FALSE)
```

**Arguments**

- `outfolder` Which folder to save temp files to?
- `testMode` run in test mode? Only parses first ten compounds

**Value**

data table with parsed database

**See Also**

`pbapply download.file, unzip fread cast capitalize`

**Examples**

```
## Not run: build.DIMEDB(outfolder=tempdir(), testMode=TRUE)
```

---

build.DRUGBANK

**Build DRUGBANK**

**Description**

Parses the DRUGBANK DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```
build.DRUGBANK(outfolder)
```

**Arguments**

- `outfolder` Which folder to save temp files to?
Details

Requires account creation! Then please download the full XML database from the website and place in databases/drugbank_source folder. Create it if it doesn’t exist yet please!

Value

data table with parsed database

See Also

unzip str_match getURL readHTMLTable,xmlToList,xmlValue,xmlEventParse as.data.table pboptions

Description

Parses the ECMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.ECMDB(outfolder, testMode = FALSE)

Arguments

outfolder Which folder to save temp files to?
testMode run in test mode? Only parses first ten compounds

Value

data table with parsed database

See Also

getCode str_match download.file,unzip fromJSON rbindlist

Examples

## Not run: build.ECMDB(outfolder=tempdir(), testMode=TRUE)
Build EXPOSOME EXPLORER

**Description**

Parses the EXPOSOME EXPLORER DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.EXPOSOMEEXPLORER(outfolder, testMode = FALSE)
```

**Arguments**

- `outfolder`: Which folder to save temp files to?
- `testMode`: run in test mode? Only parses first ten compounds

**Value**

data table with parsed database

**See Also**

download.file, unzip, str_match, getURL, fread, pbapply, capitalize

**Examples**

```r
## Not run: build.EXPOSOMEEXPLORER(outfolder=tempdir(), testMode=TRUE)
```

Build FOODB

**Description**

Parses the FOODB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.FOODB(outfolder)
```

**Arguments**

- `outfolder`: Which folder to save temp files to?
Value
data table with parsed database

See Also
download.file, untar, getURL, str_match, fread

Examples
## Not run: build.FOODB(outfolder=tempdir())

---

**build.HMDB**

Build HMDB

Description
Parses the HMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage
build.HMDB(outfolder, testMode = FALSE)

Arguments
- **outfolder**: Which folder to save temp files to?
- **testMode**: run in test mode? Only parses first ten compounds

Value
data table with parsed database

See Also
download.file, unzip, getURL, readHTMLTable, xmlValue, xmlEventParse, bindlist, pboptions, connections, str_match

Examples
## Not run: build.HMDB(outfolder=tempdir(), testMode=TRUE)
**build.KEGG**

---

**Build KEGG**

**Description**

Parses the KEGG DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

build.KEGG(outfolder, testMode = FALSE)

**Arguments**

- `outfolder`: Which folder to save temp files to?
- `testMode`: run in test mode? Only parses first ten compounds

**Value**

data table with parsed database

**See Also**

pbapply, keggFind, keggGet, getURL, str_match, rbindlist, load.molecules, get.smiles

**Examples**

```r
## Not run: build.KEGG(outfolder=tempdir(), testMode=TRUE)
```

---

**build.LIPIDMAPS**

---

**Build LIPID MAPS**

**Description**

Parses the LIPID MAPS DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

build.LIPIDMAPS(outfolder, testMode = FALSE, apikey)

**Arguments**

- `outfolder`: Which folder to save temp files to?
- `testMode`: run in test mode? Only parses first ten compounds
- `apikey`: ChemSpider API key
build.LMDB

Build LMDB

Description

Parses the LMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.LMDB(outfolder)

Arguments

outfolder Which folder to save temp files to?

Value

data table with parsed database

See Also

getURL str_match

Examples

## Not run: build.LIPIDMAPS(outfolder=tempdir(), testMode=TRUE)
**build.MACONDA**  

**Build MACONDA**

**Description**

Parses MACONDA, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.MACONDA(outfolder, testMode = FALSE, conn, apikey)
```

**Arguments**

- `outfolder`: Which folder to save temp files to?
- `testMode`: run in test mode? Only parses first ten compounds
- `conn`: Connection to extended database (MaConDa writes directly to there due to anomalous adducts)
- `apikey`: ChemSpider API key

**Value**

data table with parsed database

**See Also**

str_match, download.file, unzip, fread, pbapply, SQLite, dbDisconnect, fn

**Examples**

```r
## Not run: build.MACONDA(outfolder=tempdir(), testMode=TRUE)
```

---

**build.MASSBANK**  

**Build MASSBANK DB**

**Description**

Parses MASSBANK, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.MASSBANK(outfolder, testMode = FALSE)
```
Arguments

  outfolder   Which folder to save temp files to?
  testMode   run in test mode? Only parses first ten compounds

Value

data table with parsed database

See Also

  str_match download.file, unzip, pbapply, rbindlist

Examples

## Not run: build.MASSBANK(outfolder=tempdir(), testMode=TRUE)

## Not run: build.MCDB(outfolder=tempdir(), testMode=TRUE)
build.METABOLIGHTS

Build METABOLIGHTS DB

Description

Parses the METABOLIGHTS DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.METABOLIGHTS(outfolder, testMode = FALSE)

Arguments

outfolder Which folder to save temp files to?
testMode run in test mode? Only parses first ten compounds

Value
data table with parsed database

See Also
download.file xmlToList pbapply rbindlist read_json

Examples

## Not run: build.METABOLIGHTS(outfolder=tempdir(), testMode=TRUE)

build.METACYC

Build METACYC

Description

Parses METACYC, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.METACYC(outfolder)

Arguments

outfolder Which folder to save temp files to?
Details

Requires account creation! Then download SmartTable from 'https://metacyc.org/group?id=biocyc17-31223-3787684059#' as 'All_compounds_of_MetaCyc.txt' and save in the databases/metacyc_source folder.

Value

data table with parsed database

See Also

getURL, str_match, fread, pbapply

Description

Parses the mVOC db, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.mVOC(outfolder, testMode = FALSE)

Arguments

outfolder Which folder to save temp files to?
testMode run in test mode? Only parses first ten compounds

Value

data table with parsed database

See Also

getNodeSet, xmlAttrs, readHTMLTable, pbapply, rbindlist, getURL, str_match

Examples

## Not run: build.mVOC(outfolder=tempdir(), testMode=TRUE)
**build.PAMDB**

**Build PAMDB**

**Description**

Parses the PAMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.PAMDB(outfolder, testMode = FALSE)
```

**Arguments**

- `outfolder`: Which folder to save temp files to?
- `testMode`: run in test mode? Only parses first ten compounds

**Value**

data table with parsed database

**See Also**

`download.file` `as.data.table` `read_xlsx` `getURL` `str_match`

**Examples**

```r
## Not run: build.PAMDB(outfolder=tempdir(), testMode=TRUE)
```

---

**build.PHARMGKB**

**Build PharmGKB**

**Description**

Parses PharmGKB drugs and chemicals (drug metabolites, etc.), returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.PHARMGKB(outfolder)
```

**Arguments**

- `outfolder`: Which folder to save temp files to?
Value

data table with parsed database

See Also

download.file, unzip, getURL, str_match, fread

Examples

## Not run: build.PHARMGKB(outfolder=tempdir())

---

**build.PHENOLEXPLORER**  
Build PHENOL EXPLORER DB

Description

Parses the PHENOL EXPLORER DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.PHENOLEXPLORER(outfolder, testMode = FALSE)

Arguments

outfolder  
Which folder to save temp files to?

testMode  
run in test mode? Only parses first ten compounds

Value

data table with parsed database

See Also

getURL, str_match, download.file, unzip, read_xlsx, fread

Examples

## Not run: build.PHENOLEXPLORER(outfolder=tempdir(), testMode=TRUE)
**build.REACTOME**

**Build REACTOME db**

**Description**

Parses the REACTOME db, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Parses the METABOLOMICSWORKBENCH db, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.REACTOME(outfolder)
```

```r
build.METABOLOMICSWORKBENCH(outfolder)
```

**Arguments**

- **outfolder** Which folder to save temp files to?

**Value**

- data table with parsed database
- data table with parsed database

**See Also**

- `as.data.table`
- `html_nodes`
- `read_html`
build.RMDB

Build RMDB (deprecated)

Description

Parses the RMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.RMDB(outfolder, testMode = FALSE)

Arguments

outfolder Which folder to save temp files to?
testMode run in test mode? Only parses first ten compounds

Value

Message that RMDB is deprecated

Examples

## Not run: build.RMDB(outfolder=tempdir(), testMode=TRUE)
**build.SMPDB**

**Build SMPDB**

**Description**

Parses the SMPDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.SMPDB(outfolder, testMode = FALSE)
```

**Arguments**

- `outfolder` Which folder to save temp files to?
- `testMode` run in test mode? Only parses first ten compounds

**Value**

data table with parsed database

**See Also**

download.file, unzip, getURL, str_match, pbapply, fread, rbindlist

**Examples**

```r
## Not run: build.SMPDB(outfolder=tempdir(), testMode=TRUE)
```

---

**build.STOFF**

**Build STOFF db**

**Description**

Parses the STOFF db, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.STOFF(outfolder)
```

**Arguments**

- `outfolder` Which folder to save temp files to?
Value

data table with parsed database

See Also

download.file.unzip as.data.table read_excel

Examples

## Not run: build.STOFF(outfolder=tempdir())

---

build.T3DB  

Build T3DB

Description

Parses the T3DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

Usage

build.T3DB(outfolder, testMode = FALSE)

Arguments

  outfolder  Which folder to save temp files to?
  testMode   run in test mode? Only parses first ten compounds

Value

data table with parsed database

See Also

download.file.unzip getURL str_match fread

Examples

## Not run: build.T3DB(outfolder=tempdir(), testMode=TRUE)
build.VMH  

**Build VMH**

**Description**

Parses the VMH DB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.VMH(outfolder, testMode = FALSE)
```

**Arguments**

- `outfolder` Which folder to save temp files to?
- `testMode` run in test mode? Only parses first ten compounds

**Value**

data table with parsed database

**See Also**

`getURL str_match pbapply GET.content_type.content rbindlist`

**Examples**

```r
# Not run: build.VMH(outfolder=tempdir(), testMode=TRUE)
```

build.WIKIDATA  

**Build Wikidata**

**Description**

Parses wikidata chemical compound database, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.WIKIDATA(outfolder)
```

**Arguments**

- `outfolder` Which folder to save temp files to?
Value
data table with parsed database

See Also
query_wikidata as.data.table

Examples
## Not run: build.WIKIDATA(outfolder=tempdir())
**Description**

Parses the YMDB, returns data table with columns compoundname, description, charge, formula and structure (in SMILES)

**Usage**

```r
build.YMDB(outfolder)
```

**Arguments**

- `outfolder`: Which folder to save temp files to?

**Value**

data table with parsed database

**See Also**

`download.file`, `unzip`, `as.data.table`, `fread`, `rbindlist`, `datablock2ma`, `datablock pbapply getURL`, `str_match`

**Examples**

```r
## Not run: build.YMDB(outfolder=tempdir())
```

---

**buildBaseDB**

**Build the base database**

**Description**

This is a large wrapper function that calls upon all individual database parsers, cleans the resulting database and saves it in a SQLite database.

**Usage**

```r
buildBaseDB(
  outfolder, 
  dbname, 
  custom_csv_path = NULL, 
  smitype = "Canonical", 
  silent = TRUE, 
  cl = 0, 
)```

```r
test = FALSE,
doBT = FALSE,
btOpts = "phaseII:1",
bLoc,
skipClean = F
)
```

### Arguments

- **outfolder**: In which folder are you building your databases? Temp folders etc. will be put here.
- **dbname**: Which database do you want to build? Options: chebi,maconda,kegg,bloodexposome,dimedb,expoexplorer,foodb,drugbank,lipidmaps,massbank,metabolights,metacyc,phenolexplorer,respect,wikidata,wikipathways,t3db,vmh,hmdb,smpdb,lmdb,ymdb,ecmdb,bmdb,rmdb,stoff,anpdb,mcdb,mvoc,pamdb
- **custom_csv_path**: `PARAM_DESCRIPTION`, Default: NULL
- **smitype**: Which SMILES format do you want?, Default: 'Canonical'
- **silent**: Suppress warnings?, Default: TRUE
- **cl**: parallel::makeCluster object for multithreading, Default: 0
- **test**: Run in test mode? Makes an incomplete ver of db, but is faster.
- **doBT**: Do a biotransformation step using Biotransformer?
- **btOpts**: Biotransfomer -q options. Defaults to phase II transformation only.
- **btLoc**: Location of Biotransformer JAR file. Needs to be executable!
- **skipClean**: Skip cleaning step? Cleaning step uses SMILES to acquire formula, charge, and transforms SMILES into 'smitype' dialect.

### Value

Nothing, writes SQLite database to 'outfolder'.

### See Also

`fread`, `as.data.table`, `dbDisconnect`

### Examples

```r
## Not run: buildBaseDB(outfolder = tempdir(), "lmdb", test=TRUE)
```
buildExtDB

Build external database using a given base database

Description

Wrapper function that takes a base database, an existing (or not) external database, and fills the extended database with adduct and isotope variants of the compounds in the base database.

Usage

```
buildExtDB(
  outfolder, 
  ext.dbname = "extended", 
  base.dbname, 
  cl = 0, 
  blocksize = 600, 
  mzrange = c(60, 600), 
  adduct_table = adducts, 
  adduct_rules = adduct_rules, 
  silent = silent, 
  use.rules = TRUE, 
  count.isos = F, 
  all.isos = T
)
```

Arguments

- `outfolder`: Which folder are your databases in?
- `ext.dbname`: Extended database name (without .db suffix), Default: 'extended'
- `base.dbname`: Base database name (without .db suffix)
- `cl`: parallel::makeCluster object for multithreading, Default: 0
- `blocksize`: How many compounds to process simultaneously? Higher means more memory spikes but faster building, Default: 600
- `mzrange`: Range of m/zs to include in database, Default: c(60, 600)
- `adduct_table`: Adduct table, Default: adducts
- `adduct_rules`: Adduct rule table, Default: adduct_rules
- `silent`: Silence warnings?, Default: silent
- `use.rules`: Use adduct rules?, Default: TRUE
- `count.isos`: Add columns for amounts of 2H, 13C, 15N atoms? Useful for heavy isotope experiments.
- `all.isos`: Include and calculate all isotopes? (if FALSE, only takes the 100/main isotope).

See Also

SQLite fn as.data.table, rbindlist, fwrite, fread, dbWriteTable, pbapply, check_chemform
checkAdductRule

Check for combined adduct rules

Description

Sometimes multiple rules apply - this function checks if they all apply as noted in the rule table.

Usage

checkAdductRule(adduct_rule_scores, adduct_table)

Arguments

adduct_rule_scores
  Scores from countAdductRuleMatches.

adduct_table
  Adduct table

Value

Table with TRUE/FALSE for each structure and adduct

See Also

as.data.table

Examples

iatom = smiles.to.iatom(c('OC[C@H]1OC(O)[C@H](O)[C@H](O)[C@H]1O'))
data(adduct_rules)
data(adducts)
addScore <- countAdductRuleMatches(iatom, adduct_rules = adduct_rules)
checkAdductRule(addScore, adduct_table = adducts)
chemspiderInfo

Find more info through ChemSpider.

Description
Takes ChemSpider CIDs and finds name, SMILES, citations on pubmed/references.

Usage
chemspiderInfo(ids, maxn = 100, apikey)

Arguments
- ids: Character vector of ChemSpider IDs.
- maxn: Max ids per batch (batch search is used), Default: 100
- apikey: ChemSpider API key

Value
Data table with match results

See Also
fn POST, add_headers read_json

cleanDB

Uniformize database and remove invalid formulas/SMILES

Description
This is a wrapper function to take a 'raw' input data table with compound information, uniformize the SMILES

Usage
cleanDB(db.formatted, cl, silent = TRUE, blocksize, smitype = "Canonical")

Arguments
- db.formatted: Data table with columns 'compoundname, structure, baseformula, charge, description'
- cl: parallel::makeCluster object for multithreading
- silent: Suppress warnings?
- blocksize: How many compounds to process per 'block'? Higher number means bigger memory spikes, but faster processing time.
- smitype: SMILES format, Default: 'Canonical'
Value

Data table with SMILES in the correct format, and charge/formula re-generated from said SMILES if available.

See Also

clusterApply pbapply check_chemform rbindlist

Examples

```r
## Not run: myDB = build.LMDB(tempdir())
## Not run: cleanedDB = cleanDB(myDB$db, cl = 0, blocksize = 10)

countAdductRuleMatches

Check which structures are OK according to given adduct rules

description

Calculate 'rules' for all compounds (requires iatom-ization)

Usage

countAdductRuleMatches(iatoms, adduct_rules)

Arguments

iatoms Iatomcontainers with compounds
adduct_rules Adduct rule table (default is data(adduct_rules))

Value

Table with all structures and if they pass the rules given for each adduct

See Also

matches.get.total.formal.charge

Examples

iatom = smiles.to.iatom(c('O[C@H]1OC(O)[C@H](O)[C@H](O)[C@H]1'))
data(adduct_rules)
addScore <- countAdductRuleMatches(iatom, adduct_rules = adduct_rules)
doAdduct

**Generate adduct for given structure**

**Description**

Takes in formula, an adduct of interest, and returns adduct formulas and charges.

**Usage**

```r
doAdduct(structure, formula, charge, adduct_table, query_adduct)
```

**Arguments**

- `structure`: SMILES structure
- `formula`: Molecular formula
- `charge`: Initial charge
- `adduct_table`: Adduct table
- `query_adduct`: Adduct 'Name' of interest

**Value**

Table with adducts of this compound

**See Also**

`check_chemform, mergeform, check_ded, subform, multiform`

**Examples**

```r
data(adduct_rules)
data(adducts)
structure = 'OC[C@H]1OC(0)[C@H](0)[C@H](0)[C@H]1O'
doAdduct(structure = structure, formula="C6H12O6", charge=0, adduct_table=adducts, query_adduct="[M+H]1+")
```

---

**doBT**

**Run Biotransformer on SMILES.**

**Description**

To predict enzymatic metabolite changes, you can enable Biotransformer processing in the base database creation. This runs the process on a given SMILES string.
doIsotopes

Usage

```r
doBT(
  smis = c("CCNC1=NC(=NC(=N1)Cl)NC(C)C"),
  jarloc, 
  opts = "cyp450:2; phaseII:1",
  cl = 0,
  help = FALSE
)
```

Arguments

- `smis` (SMILES string character vector)
- `jarloc` (Location of Biotransformer .jar file).
- `opts` (-q command line options).
- `cl` (parallel::makeCluster object).
- `help` (Show help for opts only?)

Value

Data table with metabolites of given SMILES and which reaction it pertains.

See Also

`fread`, `rbindlist`, `pblapply`

Examples

```r
## Not run: myFolder = tempdir()
## Not run: btLoc = downloadBT(outfolder = myFolder)
## Not run: doBT(smis = c("CCNC1=NC(=NC(=N1)Cl)NC(C)C"),
##                  jarloc = btLoc,
##                  opts = "cyp450:2; phaseII:1")
## End(Not run)
```

doIsotopes

Generate isotopes for given formula

Description

Takes in formula and returns isotope pattern m/z values.

Usage

```r
doIsotopes(formula, charge, count.isos = F)
```
```
downloadBT

Arguments

  formula     Molecular formula
  charge      Final charge
  count.isos  Add columns for amounts of 2H, 13C, 15N atoms? Useful for heavy isotope experiments.

Value

  Table with isotopes of this molecular formula

See Also

  isopattern

Examples

  doIsotopes(formula="C6H12O6", charge=0)

```

```
downloadBT

Download biotransformer jar file.

Description

  To predict enzymatic metabolite changes, you can enable Biotransformer processing in the base database creation. This downloads the necessary JAR file from the developer page.

Usage

  downloadBT(outfolder)

Arguments

  outfolder     In which folder do you want to save the .jar file?

Value

  File location

See Also

  download.file, unzip

Examples

  ## Not run: jarloc = downloadBT(outfolder = tempdir())
```
**filterFormula**

Apply seven golden rules to a vector of formulas

**Description**

Returns formulas that pass the user-given rules.

**Usage**

```r
filterFormula(formulas, rules = c("senior", "lewis", "hc", "chnops", "nops"))
```

**Arguments**

- `formulas`  Molecular formulas
- `rules`     Rules to apply. Default: c("senior", "lewis", "hc", "chnops", "nops")

**Value**

Vector of formulas that pass rules

**See Also**

`check_chemform`, `rbindlist`, `str_match`

**Examples**

```r
filterFormula("C6H12O6")
```

---

**getFormula**

Find possible formulas for a given m/z

**Description**

Using m/z and isotope distributions for each element, find possible molecular formulas within allowed error margin

**Usage**

```r
generateFormula(
  mz,
  add_name,
  adducts,
  ppm,
  elements = c("C", "H", "N", "O", "P", "S")
)
```
getPredicted

Arguments

- **mz**: M/z of interest
- **add_name**: Which adducts to consider
- **adducts**: Full adduct table (data(adducts) loads it into memory)
- **ppm**: Allowed error margin in parts per million

Value

Table with found formulas, their adduct and isotope percentage.

See Also

- `str_match`
- `initializeCHNOPS`

Examples

```r
getFormula(170, "[M+H]1+", adducts, 3)
```

---

**getPredicted**

*Get predicted formulas and adducts from m/z value*

Description

Wrapper function to predict formulas and then consider adducts as well.

Usage

```r
generated_predicted <- function(
    mz, ppm = 2, mode = "positive", rules = c("senior", "lewis", "hc", "chnops", "nops"),
    elements = c("C", "H", "N", "O", "P", "S"),
    search = c("PubChem", "ChemSpider"),
    detailed = TRUE, calc_adducts = adducts[Ion_mode == mode, ]$Name,
    adduct_table = adducts
)
```
Arguments

- **mz**: M/z of interest
- **ppm**: Error margin in parts per million, Default: 2
- **mode**: M/z found in positive or negative mode?, Default: 'positive'
- **rules**: Which golden rules to apply?, Default: c("senior", "lewis", "hc", "chnops", "nops")
- **elements**: Which elements to consider?, Default: c("C", "H", "N", "O", "P", "S")
- **search**: Check the found formulas on PubChem or ChemSpider?, Default: c("PubChem", "ChemSpider")
- **detailed**: Look up details like description etc. if hit found? Makes things slower!, Default: TRUE
- **calc_adducts**: Which adducts to consider?, Default: adducts[Ion_mode == mode,]$Name
- **adduct_table**: Adduct table to use, referred to by 'calc_adducts'. Allows use for custom 'adducts' such as in-source fragments etc.

Value

Table of found matches and associated info

See Also

rbindlist check_chemform

Examples

```r
## Not run: getPredicted(mz = 170, ppm = 2, mode = "positive",
rule = c("hc", "chnops", "nops"),
elements = c("C","H","O"))
## End(Not run)
```

iatom.to.charge

Get formal charge from iatomcontainer

Description

This function takes iatomcontainer object and returns the formal charge.

Usage

```r
iatom.to.charge(iatoms, silent = TRUE)
```

Arguments

- **iatoms**: list of rcdk iatomcontainers
- **silent**: suppress warnings?, Default: TRUE
Value

Character vector of formal charges per iatomcontainer.

See Also

get.total.formal.charge jcall

Examples

iatom = smiles.to.iatom(c('OC[C@H]1OC(O)[C@H](O)[C@H](O)[C@H]1O'))
ch = iatom.to.charge(iatom)

iatom.to.formula Get molecular formula from iatomcontainer

Description

This function takes iatomcontainer object and returns the molecular formula.

Usage

iatom.to.formula(iatoms, silent = TRUE)

Arguments

iatoms list of rcdk iatomcontainers
silent suppress warnings?, Default: TRUE

Value

Character vector of formulas per iatomcontainer.

See Also

get.mol2formula jcall

Examples

iatom = smiles.to.iatom(c('OC[C@H]1OC(O)[C@H](O)[C@H](O)[C@H]1O'))
form = iatom.to.formula(iatom)
**iatom.to.smiles**  
*Get SMILES from iatom container*

**Description**  
This function takes an rcdk iatomcontainer and returns SMILES.

**Usage**  
```r
iatom.to.smiles(iatoms, smitype = "Canonical", silent = TRUE)
```

**Arguments**  
- **iatoms**: list of iatomcontainers  
- **smitype**: Which type of SMILES to export?, Default: 'Canonical'  
- **silent**: Suppress warnings?, Default: TRUE

**Value**  
character vector of SMILES in the chosen format

**See Also**  
`get.smiles`, `smiles.flavors`, `jcall`

**Examples**  
```r
iatom = smiles.to.iatom(c(  
   "OC[\(C@H\)]1OC(\(O\))[C@H\](\(O\))[C@H\](\(O\))[C@H\]1O")  
))
smi = iatom.to.smiles(iatom)
```

---

**is.empty**  
*Is this item 'empty'?*

**Description**  
Checks if given object is either NULL, NA, or just whitespace.

**Usage**  
```r
is.empty(item)
```

**Arguments**  
- **item**: object to check
isotopes

Value

TRUE or FALSE

Examples

is.empty(NA)

<table>
<thead>
<tr>
<th>isotopes</th>
<th>Weights of all isotopes of all elements used</th>
</tr>
</thead>
</table>

Description

Sourced from 'enviPat' package

Usage

isotopes

Format

An object of class data.frame with 308 rows and 5 columns.

<table>
<thead>
<tr>
<th>lmdb</th>
<th>LMDB</th>
</tr>
</thead>
</table>

Description

LMDB database, included with permission from the DB creators.

Usage

lmdb

Format

An object of class data.table (inherits from data.frame) with 1070 rows and 6 columns.
openBaseDB

Create/open and prepare SQLite database

Description
Create/open and prepare SQLite database

Usage
openBaseDB(outfolder, dbname)

Arguments
outfolder Which folder are you building your databases in?
dbname What is the name of the database? (exclude .db)

Value
Nothing, writes SQLITE database to outfolder

See Also
SQLite dbExecute

pubChemInfo
Find additional info on a PubChem ID.

Description
Takes PubChem ID and finds name, formula, smiles, charge

Usage
pubChemInfo(ids, maxn = 30)

Arguments
ids Vector of identifiers.
maxn Compounds searched per batch search, Default: 30

Value
Table with additional info on PubChem IDs
removeDB

See Also

rrbindlist

Examples

pubChemInfo(c(5793))

removeDB

Remove database

Description

Removes database from disk completely.

Usage

removeDB(outfolder, dbname)

Arguments

outfolder
Which folder are you building your databases in?
dbname
What database do you want to remove? (exclude .db suffix)

Value

Nothing, removes database from disk

removeFailedStructures

Remove structures where isotope generation failed.

Description

Sometimes if a run crashes, or a structure is bugged somehow, it is still registered as 'done' in the extended database and cannot be redone. This function removes these structures. Warning: slow!

Usage

removeFailedStructures(outfolder, ext.dbname = "extended")

Arguments

outfolder
Which folder are your databases in?
ext.dbname
Extended database name (without .db suffix), Default: 'extended'

See Also

SQLite
revertAdduct  Break formula apart into adduct and main formula

Description
Used to use the formula creation function and consider adducts at the same time.

Usage
revertAdduct(formula, add_name, adduct_table = adducts)

Arguments
formula  Formula of interest
add_name  Adduct names to consider ('Name' column of adduct table)
adduct_table  Full adduct table, Default: adducts

Value
Table with formula, adduct, isotope

See Also
check_ded, subform, mergeform, multiform

Examples
revertAdduct("C6H13O6", "[M+H]1+")

sdfStream.joanna  Adjusted sdfStream version for databases that store their compounds in SDF format

Description
Adjusted from existing function to extract the columns needed for MetaDBparse database format
searchCMMR

Usage

sdfStream joanna(
  input,
  output,
  append = FALSE,
  fct,
  Nlines = 10000,
  startline = 1,
  restartNlines = 10000,
  silent = FALSE,
  ...
)

Arguments

input input SDF file
output output CSV file to write to
append Append to existing CSV file or start anew?, Default: FALSE
fct Function to apply to each object to get the wanted columns
Nlines Lines to read in one go?, Default: 10000
startline Start at line, Default: 1
restartNlines Restart after x lines, Default: 10000
silent Suppress warnings?, Default: FALSE
...
Other arguments

Value

Nothing, writes csv version of given SDF files to disk

searchCMMR  Search CMMR

Description

Queries Ceu Mass Mediator through their API

Usage

searchCMMR(
  cmm_url = "http://ceumass.eps.uspceu.es/mediator/api/v3/batch",
  metabolites_type = "all-except-peptides",
  databases = "["all-except-mine"]",
  masses_mode = "mz",
  ion_mode = "positive",
)
adducts = switch(ion_mode, positive =
    "[\"M+H\", \"M+2H\", \"M+Na\", \"M+K\", \"M+NH4\", \"M+H-H2O\"]",
    negative = "[\"M-H\", \"M+Cl\", \"M+FA-H\", \"M-H-H2O\"]")

tolerance = 10,
tolerance_mode = "ppm",
unique_mz
)

Arguments

cmm_url API base url, Default: 'http://ceumass.eps.uspceu.es/mediator/api/v3/batch'
metabolites_type Which type of metabolites to consider?, Default: 'all-except-peptides'
databases Which databases to consider?, Default: ['all-except-mine']
masses_mode Format of input compound, Default: 'mz'
ion_mode Which ion mode was the compound found in?, Default: 'positive'
adducts Adducts to be considered, Default: switch(ion_mode, positive = "[\"M+H\", \\"M+2H\", \"M+Na\", \"M+K\", \"M+NH4\", \"M+H-H2O\"]", negative = "[\"M-H\", \"M+Cl\", \"M+FA-H\", \"M-H-H2O\"]")
tolerance Error margin, units of 'tolerance_mode', Default: 10
tolerance_mode Mode of error margin, Default: 'ppm'
unique_mz M/z(s) to use in query

Value

Data table with match results

See Also

create_batch_body POST, content_type, content fromJSON progress_bar rbindlist

Examples

searchCMMR(unique_mz = "170.09240307", ion_mode = "positive")

searchFormula Find matches based on molecular formula

Description

Goes through database of choice (base database) and retrieves hits that have the molecular formula of interest.

Usage

searchFormula(formula, charge, outfolder, base.dbname)
Arguments

- `formula`: Molecular formula (should be checked by `enviPat::check_chemform` first!
- `charge`: Charge of formula
- `outfolder`: Which folder are your databases stored in?
- `base.dbname`: Base database name (without .db suffix)

Value

Data table with compounds with this molecular formula and the other available information

See Also

- SQLite `rbindlist`

Examples

```r
## Not run: myFolder = tempdir()
## Not run: buildBaseDB(outfolder = myFolder, "lmdb", test = TRUE)
## Not run: searchFormula(formula = c("C7H11N3O2"), charge = 0,
outfolder = myFolder, base.dbname = c("lmdb"))
## End(Not run)
```
### searchMZ

**Find matches for m/z value in given database**

**Value**

Data table with match results.

**See Also**

pbapply, as.data.table, rbindlist, check_chemform, fn, str_extract, str_match, readHTMLTable, list.clean, read_json, POST, add_headers, content_type, content, GET, fromJSON

**Examples**

```r
## Not run: searchFormulaWeb(formulas = c("C6H12O6"),
search = c("pubchem", "chemspider",
"knapsack", "supernatural2",
"chemidplus"), detailed = TRUE)
## End(Not run)
```

---

**Description**

This function takes user m/z, ppm error, base database and the extended database to return hits of interest.

**Usage**

```r
searchMZ(
  mzs, ionmodes, outfolder, base.dbname, ppm,
  ext.dbname = "extended", append = FALSE
)
```

**Arguments**

- `mzs` Vector of m/z values
- `ionmodes` Vector of pos/negative mode for each m/z value
- `outfolder` Which folder are your databases stored in?
- `base.dbname` Which base database do you want to retrieve info from? (without .db suffix)
- `ppm` Parts per million accepted error range
- `ext.dbname` Name of extended database (without .db suffix), Default: 'extended'
- `append` Use this when searching multiple base databases, so only one result table is created, Default: FALSE
searchMZonline

Value

Data table with match results

See Also

SQLite rbindlist dbExecute, dbGetQuery, dbDisconnect fn

Examples

```r
## Not run: myFolder = tempdir()
## Not run: buildBaseDB(outfolder = myFolder, "lmdb", test = TRUE)
## Not run: file.remove(file.path(myFolder, "extended.db"))
## Not run: data(adducts)
## Not run: data(adduct_rules)
## Not run: buildExtDB(outfolder = myFolder, base.dbname = "lmdb",
silent=FALSE, adduct_table = adducts, adduct_rules = adduct_rules)
## End(Not run)
## Not run: searchMZ(c("104.3519421"), "positive", outfolder = myFolder, "lmdb", ppm = 3)
```

searchMZonline

Find m/z matches with CMMR, ChemSpider or PubChem

Description

Wrapper function for all online searches.

Usage

```r
searchMZonline(
  mz = 178.1219,
  mode = "positive",
  adducts,
  ppm = 2,
  which_db = "cmmr",
  apikey = ""
)
```

Arguments

- `mz` M/z of interest, Default: 178.1219
- `mode` Is m/z positive or negative mode?, Default: ‘positive’
- `adducts` Which adducts will you consider (for cmmr only)
- `ppm`Allowed error margin in parts per million, Default: 2
- `which_db` Which online database do you want to search?, Default: ‘cmmr’
- `apikey`ChemSpider API key. Only required if searching in ChemSpider.
**searchRev**

**Value**

Table with match information

**See Also**

pbapply

**Examples**

```r
## Not run: searchMZonline(mz = 170.09240307, mode = "positive", which_db = "cmmr")
```

---

**searchRev**  
*Reverse searching*

**Description**

Takes a SMILES structure and finds m/z values for all adducts and isotopes matching that structure.

**Usage**

```r
searchRev(structure, ext.dbname = "extended", outfolder)
```

**Arguments**

- `structure`: SMILES structure string
- `ext.dbname`: Name of extended database (without .db), Default: 'extended'
- `outfolder`: Which folder are your databases in?

**Value**

Data table with m/z values, additionally molecular formula, charge, adduct, isotope

**See Also**

SQLite

**Examples**

```r
## Not run: myFolder = tempdir()
## Not run: buildBaseDB(outfolder = myFolder, "lmdb", test = TRUE)
## Not run: file.remove(file.path(myFolder, "extended.db"))
## Not run: data(adducts)
## Not run: data(adduct_rules)
## Not run: buildExtDB(outfolder = myFolder, base.dbname = "lmdb", silent=FALSE, adduct_table = adducts, adduct_rules = adduct_rules)
## End(Not run)
## Not run: searchRev("O=C(O)C(N)CC=1N=CN(Cl)C", outfolder = myFolder)
```
**showAllBase**

*Show all compounds in base db*

**Description**

Shows all compounds in this base database as data table.

**Usage**

```
showAllBase(outfolder, base.dbname)
```

**Arguments**

- `outfolder` Which folder is your database in?
- `base.dbname` Which base database do you want to explore? (exclude .db suffix)

**Details**

This may be quite memory consuming for larger databases!!

**Value**

Data table with whole database.

**See Also**

SQLite

**Examples**

```r
## Not run: myFolder = tempdir()
## Not run: buildBaseDB(outfolder = myFolder, "lmdb", test = TRUE)
## Not run: showAllBase(outfolder = myFolder, "lmdb")
```

---

**smiles.to.iatom**

*Get iatom containers from SMILES*

**Description**

FUNCTION_DESCRIPTION

**Usage**

```
## S3 method for class 'to.iatom'
smiles(smiles, silent = TRUE, cl = 0)
```
Arguments

- **smiles**: character vector of smiles
- **silent**: suppress warnings?, Default: TRUE
- **cl**: parallel::makeCluster object for multithreading, Default: 0

Value

- iatom containers for use in rcdk package

See Also

- jcall

Examples

```r
smiles.to.iatom(c('OC[C@H]1OC(0)[C@H](O)[C@H](O)[C@H](O)1'))
```

---

**writeDB**  
*Write table to SQLite database*

Description

Simple wrapper - take data table or data frame and append it to the given table in the given database.

Usage

```r
writeDB(conn, table, tblname)
```

Arguments

- **conn**: Database connection (from DBI::dbConnect or similar)
- **table**: Data frame or data table
- **tblname**: SQLITE table name to append to

Value

- Nothing, writes table to SQLITE database

See Also

- dbWriteTable
**Index**

* datasets
  - adduct_rules, 4
  - adducts, 3
  - isotopes, 39
  - lmdb, 39

add_headers, 29, 46
adduct_rules, 4
adducts, 3
as.data.table, 6, 8, 12, 17, 19, 22, 24–28, 46

build.ANPDB, 4
build.BLOODEXPOSOME, 5
build.BMDB, 5
build.CHEBI, 6
build.DIMEDB, 7
build.DRUGBANK, 7
build.ECMDB, 8
build.EXPOSOMEEXPLORER, 9
build.FOODB, 9
build.HMDB, 10
build.KEGG, 11
build.LIPIDMAPS, 11
build.LMDB, 12
build.MACONDA, 13
build.MASSBANK, 13
build.MCDB, 14
build.METABOLIGHTS, 15
build.METABOLICSWORKBENCH
  (build.REACTOME), 19
build.METACYC, 15
build.mVOC, 16
build.PAMDB, 17
build.PHARMGKB, 17
build.PHENOLEXPLORER, 18
build.REACTOME, 19
build.RESPECT, 19
build.RMDB, 20
build.SMPDB, 21
build.STOFF, 21

build.T3DB, 22
build.VMH, 23
build.WIKIDATA, 23
build.WIKIPATHWAYS, 24
build.YMDB, 25
buildBaseDB, 25
buildExtDB, 27

capitalize, 7, 9
cast, 7
check_chemform, 27, 30, 31, 34, 36, 46
check_ded, 31, 42
checkAdductRule, 28
chemspiderInfo, 29
cleanDB, 29
clusterApply, 30
connections, 6, 10, 14
content, 23, 44, 46
content_type, 23, 44, 46
countAdductRuleMatches, 30
create_batch_body, 44
datablock, 6, 12, 25
datablock2ma, 6, 12, 25
dbDisconnect, 13, 26, 47
dbExecute, 40, 47
dbGetQuery, 47
dbWriteTable, 27, 50
doAdduct, 31
doBT, 31
doIsotopes, 32
download.file, 4–10, 12–15, 17, 18, 20–22,
  25, 33
downloadBT, 33

filterFormula, 34
fn, 13, 27, 29, 46, 47
fread, 4, 7, 9, 10, 12, 13, 16, 18, 21, 22,
  25–27, 32
fromJSON, 8, 44, 46
fwrite, 27
GET, 23, 46
get.mol2formula, 37
get.smiles, 11, 38
get.total.formal.charge, 30, 37
getFormula, 34
genNodeSet, 16
getPredicted, 35
getURL, 6, 8–12, 14, 16–18, 20–25
html_nodes, 12, 19, 24
html_text, 12
iatom.to.charge, 36
iatom.to.formula, 37
iatom.to.smiles, 38
initializeCHNOPS, 35
is.empty, 38
isopattern, 33
isotopes, 39
jcall, 37, 38, 50
keggFind, 11
keggGet, 11
list.clean, 46
lmdb, 39
load.molecules, 11
matches, 30
mergeform, 31, 42
multiform, 31, 42
openBaseDB, 40
pbapply, 5, 7, 9, 11–16, 20, 21, 23, 25, 27, 30, 46, 48
pblapply, 32
pboptions, 6, 8, 10, 14
POST, 29, 44, 46
progress_bar, 44
pubChemInfo, 40
query_wikidata, 24
rbindlist, 8, 10–12, 14–16, 20, 21, 23, 25, 27, 30, 32, 34, 36, 41, 44–47
read_excel, 22
read_html, 19, 24
read_json, 5, 15, 29, 46
read_xlsx, 5, 17, 18
read_xml, 12
readHTMLTable, 8, 10, 14, 16, 46
removeDB, 41
removeFailedStructures, 41
revertAdduct, 42
sdfStream.joanna, 42
searchCMMR, 43
searchFormula, 44
searchFormulaWeb, 45
searchMZ, 46
searchMZonline, 47
searchRev, 48
showAllBase, 49
smiles.flavors, 38
smiles.to.iatom, 49
SPARQL, 24
SQLite, 13, 27, 40, 41, 45, 47–49
str_extract, 46
str_match, 6, 8–14, 16–18, 20–23, 25, 34, 35, 46
stri_detect, 12
subform, 31, 42
untar, 10
unzip, 6–10, 12–14, 18, 20–22, 25, 33
writeDB, 50
xmlAttrs, 16
xmlEventParse, 6, 8, 10, 14
xmlToList, 8, 15
xmlValue, 6, 8, 10, 14