ChemClass

December 20, 2023

Type Package

Title Places chemicala into unique classes

Version 0.1.0

Author Ricahrd Judson

Maintainer Richard Judson <judson.richard@epa.gov>

Description Chemcials are placed into unique classes, under high level
categories that include drugs, pesticides, exper-driven classe3s, colors and ClassyFire

License GPL3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

R topics documented:

`cdk.prep` *Prepare the CDK descriptors for all of the SMILES*

Description

Prepare the CDK descriptors for all of the SMILES

Usage

```
cdk.prep()
```

`classify` *Run the classyfireR service for a series of batches. The input is a file with 2 columns: dtxsid and smiles. Each SMILES is sent to the classyfireR server and a classification is returned if successful. The service is buggy, so several hacks are built into this function to handle the different error conditions of the server. THe most vexing problem is that the server can go into an infinite loop without a return. These are indicated by "Request failed [xxx]" messages. These are mostly solved with a withTimeout condition. These errors seem to be caused by bad SMILES. The Dashboard return a single space when a SMILES is missing and these will cause this error. There other problematic bad SMILES, including ones with a # character (triple bond). Still need to find all bad types and filter them out. Linear alkanes (CCCC...CCC) are not classified and need to be classified by hand.*

Description

Manual fixes so far: 1. empty strings filtered from initial set and never tried now 2. Triple bonds ("#") replaced with double bond ("="). THis should not affect the classification At the end of each batch of chemicals a file is written out, allowing one to save results along the way and restart if needed. All of these files need to be concatenated together and chemicals with missing classification rerun at least one more time.

Usage

```
classify(
  batchstart = 1,
  batchsize = 100,
  maxtry = 2,
  sleepinterval.batch = 10,
  sleepinterval.chemical = 5,
  time.limit = 10
)
```

Arguments

`batchstart`	The index of the batch of chemicals to run
`batchsize`	The number of chemicals in a batch
`maxtry`	The service will sometimes fail but can pass on a later call This parameter is the number of tries to do before giving up
`sleepinterval.batch`	The number of seconds to wait at the end of a batch. This was an attempt to solve a problem that looked like refusal by the server to handle too many requests in a time period. Probably not needed
`sleepinterval.chemical`	The number of seconds to wait at the end of a chemical. This was an attempt to solve a problem that looked like refusal by the server to handle too many requests in a time period. Probably not needed
`time.limit`	ClassyFire R will sometimes go into an infinite loop on the server size without returning. This function will stop waiting for a return after this interval (in seconds) and move to the next chemical

Details

The output file includes the chemical name as well as the dtxsid. To add this, a file with the DSSTox inventory is read in.

`classify.concat` *Concatenates all of the classyfire output files*

Description

Concatenates all of the classyfire output files

Usage

```
classify.concat(batchset = 2)
```

Arguments

`batchsset`	The batch version for their directory

`clasyfire.prep` *Prepare the ClassyFire results as an RData file*

Description

Prepare the ClassyFire results as an RData file

Usage

```
clasyfire.prep()
```

`contains` *Find out if one string contains another*

Description

Find out if one string contains another

Usage

```
contains(x, query, verbose = F)
```

Arguments

`x`	The string to be searched in
`query`	the second string
`verbose`	if TRUE, the two strings are printed

Value

if x contains query, return TRUE, FALSE otherwise

`export.toxvaldb.chems` *Export the relevant chemciasl from toxval*

Description

Export the relevant chemciasl from toxval

Usage

```
export.toxvaldb.chems(toxval.db = "res_toxval_v94")
```

Arguments

`toxval.db`	Database version
`source`	The source to be updated

Value

Write a file with the results

`find.chems.by.class` *find chemicals that look like pesticides or drugs*

Description

find chemicals that look like pesticides or drugs

Usage

```
find.chems.by.class(cutoff = 0.6)
```

`find.chems.by.class.mapper` *map the putative classes for the new drugs and pesticides*

Description

map the putative classes for the new drugs and pesticides

Usage

```
find.chems.by.class.mapper()
```

`fix.casrn` *Fix a CASRN that has one of several problems*

Description

Fix a CASRN that has one of several problems

Usage

```
fix.casrn(casrn, cname = "", verbose = F)
```

Arguments

`casrn`	Input CASRN to be fixed
`cname`	An optional chemical name
`verbose`	if TRUE, print hte input values

Value

the fixed CASRN

`fix_smiles` *corrects all smiles list*

Description

‘fix_smiles()‘ corrects all smiles list

Usage

```
fix_smiles(smile_file = "inst/extdata/POD_chemical_SMILES.xlsx")
```

Arguments

`smile_file`	The location and file name for the exported excel file created. Null means do not save the file.

`get.drugs.from.refchemdb` *Get the Drugbank data from RefChemDB*

Description

Get the Drugbank data from RefChemDB

Usage

```
get.drugs.from.refchemdb()
```

`hello` *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`httrVsChemClass` *Analyze the HTTr data vs the chemical categories*

Description

Analyze the HTTr data vs the chemical categories

Usage

```
httrVsChemClass(dataset)
```

`httrVsChemClassChisq` *Analyze the HTTr data vs the chemical categories Calculate chisq stats of chemclass vs signature*

Description

Analyze the HTTr data vs the chemical categories Calculate chisq stats of chemclass vs signature

Usage

```
httrVsChemClassChisq(dataset = "MCF7 Screen", cutoff = 0)
```

`httrVsChemClassHeatmap` *Heatmap of the HTTr hit rates vs the chemical categories*

Description

Heatmap of the HTTr hit rates vs the chemical categories

Usage

```
httrVsChemClassHeatmap(to.file = F, nmin = 3, color.cut = 400)
```

`httrVsChemClassHitClusters` *Cluster chemicals by their hit patterns*

Description

Cluster chemicals by their hit patterns

Usage

```
httrVsChemClassHitClusters(dataset = "MCF7 Screen")
```

httrVsChemClassHitClusters.matcher

Find chemicals that are close to chemicals in a class

Description

process to get here httrVsChemClassHitClusters.step3.boxplot httrVsChemClassHitClusters.step3 httrVsChemClassHitClusters.step2 httrVsChemClassHitClusters

Usage

```
httrVsChemClassHitClusters.matcher(
  dataset = "MCF7 Screen",
  class = "Bisphenol",
  cutoff = 20
)
```

httrVsChemClassHitClusters.step2

Cluster chemicals by their hit patterns

Description

Cluster chemicals by their hit patterns

Usage

```
httrVsChemClassHitClusters.step2(dataset = "MCF7 Screen", nmax = 1000)
```

httrVsChemClassHitClusters.step3

Cluster chemicals by their hit patterns

Description

Cluster chemicals by their hit patterns

Usage

```
httrVsChemClassHitClusters.step3(dataset = "MCF7 Screen", nmax = 1000)
```

`httrVsChemClassHitClusters.step3.boxplot` *Box plot of the nearest chemcial of the same class*

Description

process to get here httrVsChemClassHitClusters.step3.boxplot httrVsChemClassHitClusters.step3 httrVsChemClassHitClusters.step2 httrVsChemClassHitClusters

Usage

```
httrVsChemClassHitClusters.step3.boxplot(
  to.file = F,
  dataset = "MCF7 Screen",
  nmax = 1000
)
```

`httrVsChemClassHitPatterns` *Analyze the HTTr data vs the chemical categories Run httrVsChemClassHitClusters to generate the input data file*

Description

Analyze the HTTr data vs the chemical categories Run httrVsChemClassHitClusters to generate the input data file

Usage

```
httrVsChemClassHitPatterns(
  to.file = F,
  dataset = "MCF7 Screen",
  classlist = c("Estrogen", "Bisphenol")
)
```

`httrVsChemClassMerge` *Merge HTTr hit rates vs the chemical categories*

Description

Merge HTTr hit rates vs the chemical categories

Usage

```
httrVsChemClassMerge()
```

httrVsChemClassPlot *Plot the HTTr data vs the chemical categories*

Description

Plot the HTTr data vs the chemical categories

Usage

```
httrVsChemClassPlot(to.file = F, nmin = 3, dataset = "MCF7 Screen")
```

httrVsChemPromiscuousGeneTargetBoxplot *Analyze the HTTr data vs the chemical categories Run httrVsChemClassHitClusters to generate the input data file*

Description

Analyze the HTTr data vs the chemical categories Run httrVsChemClassHitClusters to generate the input data file

Usage

```
httrVsChemPromiscuousGeneTargetBoxplot(to.file = F, dataset = "MCF7 Screen")
```

metal.finder *Find the metal containing compounds*

Description

Find the metal containing compounds

Usage

```
metal.finder()
```

nearest.neighbors *Prepare the CDK descriptors for all of the SMILES*

Description

Prepare the CDK descriptors for all of the SMILES

Usage

```
nearest.neighbors(class.prefix = "Antibiotic")
```

`nearest.neighbors.rectangle` *Prepare the CDK descriptors for all of the SMILES*

Description

Prepare the CDK descriptors for all of the SMILES

Usage

```
nearest.neighbors.rectangle(class.prefix = "Pharmaceutical unknown MOA")
```

`oasis.classes` *Get the chemical classes for OASIS*

Description

Get the chemical classes for OASIS

Usage

```
oasis.classes()
```

`printCurrentFunction` *Print the name of the current function*

Description

Print the name of the current function

Usage

```
printCurrentFunction(comment.string = NA)
```

Arguments

`comment.string`	An optinal string to be printed

runQuery *Runs a database query and returns a result set*

Description

Runs a database query and returns a result set

Usage

```
runQuery(query, db, do.halt = T, verbose = F)
```

Arguments

`query`	a properly formatted SQL query as a string
`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information

setDBConn *set SQL connection to the database*

Description

set SQL connection to the database

Usage

```
setDBConn(
  server = "ccte-mysql-res.epa.gov",
  user = "rjudson",
  password = NA,
  port = -1
)
```

Arguments

`server`	SQL server on which relevant database lives
`user`	SQL username to access database
`password`	SQL password corresponding to username

TxT *Calculate several statistics on a 2 x 2 matrix*

Description

Calculate several statistics on a 2 x 2 matrix

Usage

```
TxT(
  tp,
  fp,
  fn,
  tn,
  do.p = TRUE,
  chemclass = NA,
  signature = NA,
  gene_target = NA
)
```

Arguments

`tp`	number of true positives
`fp`	number of false positives
`fn`	number of false negatives
`tn`	number of true negatives
`do.p`	if TRUE, calculate an exact p-value
`rowname`	if not NA, add a column to the output with this rowname Returns: a list of the results a: TP b: FP c: FN d: TN sens: sensitivity spec: specificity ba: Balanced Accuracy accuracy: Accuracy relative.risk: Relative Risk odds.ratio: Odds Ratio or.ci.lwr: lower confidence interval of the Odds Ratio or.ci.upr: upper confidence interval of the Odds Ratio ppv: Positive Predictive Value npv: Negative Predictive Value p.value: Chi-squared p-value F1: 2TP/(2TP+FP+FN) sval: All of the results as a tab-delimited string title: the title of the results as a tab-delimited string mat: The results as a 1-row data frame @export

use.class.counts *Get the counts by class*

Description

Get the counts by class

Usage

```
## S3 method for class 'class.counts'
use()
```

`use.classes` *Add the main use classes*

Description

Add the main use classes

Usage

```
## S3 method for class 'classes'
use()
```

Index