Package 'epos'

November 4, 2020

Type Package

Title Epilepsy Ontologies' Similarities

Version 0.1.3

Author Bernd Mueller

Maintainer Bernd Mueller <bernd.mueller@zbmed.de>

Description Analysis and visualization of similarities between epilepsy ontologies based on text mining results by comparing ranked lists of co-occurring drug terms in the corpus of LIVIVO. The ranked result lists of neurological drug terms co-occurring with named entities from the epilepsy ontologies EpSO, ESSO, and EPILONT are aggregated in order to generate two different results: an overview table of drugs that are relevant to epilepsy created with the method createNeuroTable, and a plot of tanimoto similarity coefficients between the aggregated list of drug terms against the list of drug terms from each of the ontologies created with the method createTanimotoBaseline(). The alignment of the Top-K Ranked Lists is conducted using the R-package TopKLists <https://cran.r-project.org/package=TopKLists>. The source data to create the ranked lists of drug names is produced using the text mining workflows described in Mueller, Bernd and Hagelstein, Alexandra (2016) <doi:10.4126/FRL01-006408558> and Mueller, Bernd et al. (2017) <doi:10.1007/978-3-319-58694-6_22>.

Depends R (>= 3.6.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports hash, ggplot2, testthat, gridExtra, TopKLists, stringr, xtable, mongolite

Suggests knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2020-11-04 22:40:03 UTC

R topics documented:

`calcCosine` *Calculate the cosine similarity metric for two lists a and b*

Description

Calculate the cosine similarity metric for two lists a and b

Usage

```
calcCosine(a, b)
```

Arguments

a	list with elements that should be of same type as in list b
b	list with elements

Value

co list with length of set b containing the cosine similarity coefficient at each position

Examples

```
calcCosine(c(1,2), c(2,3))
```

---

calcDice *Calculate the dice similarity metric for two lists a and b*

Description

Calculate the dice similarity metric for two lists a and b

Usage

```
calcDice(a, b)
```

Arguments

a	list with elements that should be of same type as in list b
b	list with elements

Value

di list with length of set b containing the dice similarity coefficient at each list element

Examples

```
calcDice(c(1,2), c(2,3))
```

calcDSEA *Calculate dsea scores of one list in comparison to reference list*

Description

Calculate dsea scores of one list in comparison to reference list

Usage

```
calcDSEA(alist, N)
```

Arguments

alist	list of drug names to be used for calculating dsea
N	numeric value with maximum length of lists for dsea calculation

Value

list with dsea scores

Examples

```
calcDSEA(c("Valproic acid", "Lamotrigine", "Ketamin"), 3)
```

calcEnrichment *Calculate enrichment of one list in comparison to reference list*

Description

Calculate enrichment of one list in comparison to reference list

Usage

```
calcEnrichment(alist)
```

Arguments

alist	the list to compare

Value

list with calculated enrichment used for plotting

Examples

```
a <- calcEnrichment(c("Clobazam","Oxcarbazepine"))
```

calcJaccard *Calculate the jaccard coefficient for two lists a and b*

Description

Calculate the jaccard coefficient for two lists a and b

Usage

```
calcJaccard(a, b)
```

Arguments

a	list with elements that should be of same type as in list b
b	list with elements

Value

ja list with length of set b containing the jaccard similarity coefficient for each list element

Examples

```
calcJaccard(c(1,2), c(2,3))
```

cosine *Calculate cosine similarity metric*

Description

Calculate cosine similarity metric

Usage

```
cosine(ainterb, lengtha, lengthb)
```

Arguments

ainterb	integer value with number of intersecting elements between set a and b
lengtha	integer value with the number of items in set a
lengthb	integer value with the number of items in set b

Value

cosine double vlaue with the cosine similarity coefficient

Examples

```
cosine(1,3,4)
```

createBaseTable *Main function to call everything and produce the results*

Description

Main function to call everything and produce the results

Usage

```
createBaseTable(coocepso, coocesso, coocepi)
```

Arguments

coocepso	list of drug names sorted by frequency co-occuring with EpSO
coocesso	list of drug names sorted by frequency co-occuring with ESSO
coocepi	list of drug names sorted by frequency co-occuring with EPILONT

Value

result table containin the aggregated list of drug terms and their associations

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
utils::data(rawDrugNamesCoOcESSO, package="epos")
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
createBaseTable(coocepso = rawDrugNamesCoOcEpSO[1:150],
  coocesso=rawDrugNamesCoOcESSO[1:150],
  coocepi=rawDrugNamesCoOcEPILONT[1:150])
```

createDashVectorForATC *Creates a vector with an X at each position where a drug from the druglist matches the ATC class list slatc*

Description

Creates a vector with an X at each position where a drug from the druglist matches the ATC class list slatc

Usage

```
createDashVectorForATC(druglist, atchashda, atchashsec, slatc)
```

Arguments

druglist	list of drug names
atchashda	hash retrieved from readAtcMapIntoHashMapDrugNamesAtcCodes
atchashsec	hash retrieved from readSecondLevelATC
slatc	list of ATC classes

Value

list with crosses if the drug in druglist matches at the position of the ATC class in slatc

Examples

```
## Not run:
createDashVectorForATC(druglist, atchashda, atchashsec, slatc)

## End(Not run)
```

---

createJaccardPlotDBMeSH

Creates the plot for all jaccard coefficients amongst the three epilepsy ontologies

---

Description

Creates the plot for all jaccard coefficients amongst the three epilepsy ontologies

Usage

```
createJaccardPlotDBMeSH(jmeshepso, jmeshesso, jmeshepi)
```

Arguments

jmeshepso	list containing jaccard coefficients between mesh and epso for increasing k
jmeshesso	list containing jaccard coefficients between mesh and esso for increasing k
jmeshepi	list containing jaccard coefficients between mesh and epi for increasing k

Value

jaccardepilepsyplot the ggplot object

Examples

```
## Not run:
jaccardepilepsyplot <- createJaccardPlotAll(jaccardepso, jaccardesso)

## End(Not run)
```

createJaccardPlotMeSHFive *Creates the plot for all jaccard coefficients amongst the three epilepsy ontologies*

Description

Creates the plot for all jaccard coefficients amongst the three epilepsy ontologies

Usage

```
createJaccardPlotMeSHFive(
  jmeshepso,
  jmeshesso,
  jmeshepi,
  jmeshepilepsyand,
  jmeshepilepsyor
)
```

Arguments

jmeshepso	list of jaccard coefficients between mesh and epso for increasing k
jmeshesso	list of jaccard coefficients between mesh and esso for increasing k
jmeshepi	list of jaccard coefficients between mesh and epi for increasing k
jmeshepilepsyand	list of jaccard coefficients between mesh and the intersection of epso, esso, and epi for increasing k
jmeshepilepsyor	list of jaccard coefficients between mesh and the union of epso, esso, and epi for increasing k

Value

jaccardepilepsyplot the ggplot object

Examples

```
## Not run:
jaccardepilepsyplot <- createJaccardPlotAll(jaccardepso, jaccardesso)

## End(Not run)
```

createNeuroTable *Create the final resulting data frame*

Description

Create the final resulting data frame

Usage

```
createNeuroTable(atchashda, atchashsec, dneuromaxk)
```

Arguments

atchashda	hashmap retrieved from readAtcMapIntoHashMapDrugNamesAtcCodes
atchashsec	hashmap retrieved from readSecondLevelATC
dneuromaxk	data frame containing columns for each intersection, ATC class, and reference list

Value

data frame containing drug names with additional columns listing association to ATC classes

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
utils::data(rawDrugNamesCoOcESSO, package="epos")
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
utils::data(rawDrugNamesCoOcEPISEM, package="epos")
utils::data(rawDrugNamesCoOcFENICS, package="epos")
atchashda <-
readAtcMapIntoHashMapDrugNamesAtcCodes(
  system.file("extdata", "db-atc.map", package = "epos"), "\t")
atchashaa <-
  readAtcMapIntoHashMapAtcCodesAtcNames(
    system.file("extdata", "db-atc.map", package = "epos"), "\t")
atchashsec <-
  readSecondLevelATC(
    system.file("extdata", "atc-secondlevel.map", package = "epos"), "\t")
epso <- genDictListFromRawFreq(rawDrugNamesCoOcEpSO)
neuroepso <- filterNeuroDrugs(epso, atchashda)
esso <- genDictListFromRawFreq(rawDrugNamesCoOcESSO)
neuroesso   <- filterNeuroDrugs(esso, atchashda)
epi <- genDictListFromRawFreq(rawDrugNamesCoOcEPILONT)
neuroepi    <- filterNeuroDrugs(epi, atchashda)
episem <- genDictListFromRawFreq(rawDrugNamesCoOcEPISEM)
neuroepisem <- filterNeuroDrugs(episem, atchashda)
fenics <- genDictListFromRawFreq(rawDrugNamesCoOcFENICS)
neurofenics <- filterNeuroDrugs(fenics, atchashda)
mx <- max(
```

```
    c(length(neuroepso), length(neuroesso), length(neuroepi),
      length(neuroepisem), length(neurofenics)))
dneuro <-
  data.frame(EpSO = c(neuroepso, rep("", (mx-length(neuroepso)))),
             ESSO = c(neuroesso, rep("", (mx-length(neuroesso)))),
             EPILONT = c(neuroepi, rep("", (mx-length(neuroepi)))),
             EPISEM = c(neuroepisem, rep("", (mx-length(neuroepisem)))),
             FENICS = c(neurofenics, rep("", (mx-length(neurofenics)))))
dneuromaxk <- TopKLists::calculate.maxK(dneuro, L=5, d=5, v=10)
neurotable <- createNeuroTable(atchashda, atchashsec, dneuromaxk)
```

`createTanimotoBaseline` *Creates the plot for all jaccard coefficients amongst the three epilepsy ontologies*

Description

Creates the plot for all jaccard coefficients amongst the three epilepsy ontologies

Usage

```
createTanimotoBaseline(neuroepso, neuroesso, neuroepi, dneuromaxk)
```

Arguments

`neuroepso`	list of neuro drug names co-occurring with epso
`neuroesso`	list of neuro drug names co-occurring with esso
`neuroepi`	list of neuro drug names co-occurring with epi
`dneuromaxk`	object returned from TopKLists::calculate.maxk

Value

jaccardepilepsyplot the ggplot object

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
utils::data(rawDrugNamesCoOcESSO, package="epos")
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
atchashda <-
  readAtcMapIntoHashMapDrugNamesAtcCodes(
    system.file("extdata", "db-atc.map", package = "epos"), "\t")
tepso <- genDictListFromRawFreq(rawDrugNamesCoOcEpSO[1:150])
neuroepso <- filterNeuroDrugs(tepso, atchashda)
utils::data(rawDrugNamesCoOcESSO, package="epos")
tesso <- genDictListFromRawFreq(rawDrugNamesCoOcESSO[1:150])
neuroesso <- filterNeuroDrugs(tesso, atchashda)
```

```
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
tepi <- genDictListFromRawFreq(rawDrugNamesCoOcEPILONT[1:150])
neuroepi <- filterNeuroDrugs(tepi, atchashda)
dneuro <-
  data.frame(EpSO = neuroepso[1:9],
             ESSO = neuroesso[1:9],
             EPILONT = neuroepi[1:9])
dneuromaxk <- TopKLists::calculate.maxK(dneuro, 3, 5, 10)
tanimotobaseline <- createTanimotoBaseline(neuroepso, neuroesso, neuroepi, dneuromaxk)
```

`dice` *Calculate dice similarity metric*

Description

Calculate dice similarity metric

Usage

```
dice(ainterb, lengtha, lengthb)
```

Arguments

`ainterb`	integer value with number of intersecting elements between set a and b
`lengtha`	integer value with the number of items in set a
`lengthb`	integer value with the number of items in set b

Value

dice double vlaue with the dice similarity coefficient

Examples

```
dice(1, 3, 4)
```

`doFullPlot` *Does the full plot on one page*

Description

Does the full plot on one page

Usage

```
doFullPlot(
  cosinemeshplot,
  cosinedrugbankplot,
  cosineepilepsyplot,
  dicemeshplot,
  dicedrugbankplot,
  diceepilepsyplot,
  jaccardmeshplot,
  jaccarddrugbankplot,
  jaccardepilepsyplot
)
```

Arguments

cosinemeshplot	plot with cosine coefficients against MeSH
cosinedrugbankplot	plot with cosine coefficients against DrugBank
cosineepilepsyplot	plot with cosine coefficients of Epilepsy Ontologies
dicemeshplot	plot with dice coefficients against MeSH
dicedrugbankplot	plot with dice coefficients against DrugBank
diceepilepsyplot	plot with dice coefficients of Epilepsy Ontologies
jaccardmeshplot	plot with jaccard coefficients against MeSH
jaccarddrugbankplot	plot with jaccard coefficients against DrugBank
jaccardepilepsyplot	plot with jaccard coefficients of Epilepsy Ontologies

Value

full

Examples

```
## Not run:
full <- doFullPlot (cosinemeshplot,
                    cosinedrugbankplot,
                    cosineepilepsyplot,
                    dicemeshplot,
                    dicedrugbankplot,
                    diceepilepsyplot,
                    jaccardmeshplot,
                    jaccarddrugbankplot,
                    jaccardepilepsyplot)
```

```
## End(Not run)
```

`filterApprovedDrugs` *Filter a given list of drug names for having an ATC code, if not they are dropped*

Description

Filter a given list of drug names for having an ATC code, if not they are dropped

Usage

```
filterApprovedDrugs(druglist, atchashda)
```

Arguments

`druglist`	a list of drug names
`atchashda`	a hash containing the drug names as keys

Value

approveddrugs a hash filtered for having an ATC code

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
atchashda <-
  readAtcMapIntoHashMapDrugNamesAtcCodes(
  system.file("extdata", "db-atc.map", package = "epos"), "\t")
tepso <- genDictListFromRawFreq(rawDrugNamesCoOcEpSO)
filterApprovedDrugs(tepso, atchashda)
```

`filterNeuroDrugs` *Filter a given list of drug names for having an ATC code starting with N indicating to be a drug for the Nervous System*

Description

Filter a given list of drug names for having an ATC code starting with N indicating to be a drug for the Nervous System

Usage

```
filterNeuroDrugs(druglist, atchashda)
```

Arguments

druglist	a list of drug names
atchashda	a hash containing the drug names as keys

Value

neurodrugs a hash filtered for having an ATC code starting with N

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
atchashda <-
  readAtcMapIntoHashMapDrugNamesAtcCodes(
  system.file("extdata", "db-atc.map", package = "epos"), "\t")
tepso <- genDictListFromRawFreq(rawDrugNamesCoOcEpSO)
nepso <- filterNeuroDrugs(tepso, atchashda)
```

---

genDictListFromRawFreq

Clears object that was loaded from harddrive into a list of terms sorted by frequency

---

Description

Clears object that was loaded from harddrive into a list of terms sorted by frequency

Clears object that was loaded from harddrive into a list of terms sorted by frequency

Usage

```
genDictListFromRawFreq(topfreqdictraw)

genDictListFromRawFreq(topfreqdictraw)
```

Arguments

topfreqdictraw	list with terms from a dictionary sorted by frequency

Value

a sorted list of terms

a sorted list of terms

Examples

```
## Not run:
genDictListFromRawFreq(epi)

## End(Not run)
utils::data(rawDrugNamesCoOcEpSO, package="epos")
genDictListFromRawFreq(rawDrugNamesCoOcEpSO)
```

`getRefAll` *Retrieve the list of drugs from the union of all reference lists*

Description

Retrieve the list of drugs from the union of all reference lists

Usage

```
getRefAll()
```

Value

list of drugs from all reference lists

Examples

```
d <- getRefAll()
```

`getTermMatrix` *Receives a sorted hashmap with found entities from a dictionary*

Description

Receives a sorted hashmap with found entities from a dictionary

Usage

```
getTermMatrix(dictionary, database, collection)
```

Arguments

`dictionary`	Character vector that is the name of a dicitonary having pre-calculated stats. This can be MeSH, DrugBank, Agrovoc, EpSO, ESSO, or EPILONT
`database`	the name of the MongoDB database to be used
`collection`	the name of the MongoDB collection to be used

Value

a sorted hashmap containing all found entities from the respective dictionaries with frequencies

Examples

```
## Not run:
mesh <- getTermMatrix("MeSH")

## End(Not run)
```

`jaccard` *Calculate jaccard similarity metric for two sets a and b*

Description

Calculate jaccard similarity metric for two sets a and b

Usage

```
jaccard(ainterb, aunionb, lengtha, lengthb)
```

Arguments

`ainterb`	integer value with number of intersecting elements between set a and b
`aunionb`	integer value with number of union elements between set a and b
`lengtha`	length of set a
`lengthb`	length of set b

Value

jac double value with the jaccard similarity coefficient

Examples

```
jaccard(1,3, 2, 3)
```

plotDSEA *Plotting functions for DSEA lists*

Description

Plotting functions for DSEA lists

Usage

```
plotDSEA(dsepso, dsesso, dsepi, dsepisem, dsfenics, dsspace, k)
```

Arguments

dsepso	list with enrichment for EpSO
dsesso	list with enrichment for ESSO
dsepi	list with enrichment for EPILONT
dsepisem	list with enrichment for EPISEM
dsfenics	list with enrichment for FENICS
dsspace	list with enrichment for the combined ranked list
k	numeric value for the length to be plotted

Value

the plot object

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
utils::data(rawDrugNamesCoOcESSO, package="epos")
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
utils::data(rawDrugNamesCoOcEPISEM, package="epos")
utils::data(rawDrugNamesCoOcFENICS, package="epos")
atchashda <-
    readAtcMapIntoHashMapDrugNamesAtcCodes(
        system.file("extdata", "db-atc.map", package = "epos"), "\t")
epso <- genDictListFromRawFreq(rawDrugNamesCoOcEpSO)
neuroepso <- filterNeuroDrugs(epso, atchashda)
esso <- genDictListFromRawFreq(rawDrugNamesCoOcESSO)
neuroesso   <- filterNeuroDrugs(esso, atchashda)
epi <- genDictListFromRawFreq(rawDrugNamesCoOcEPILONT)
neuroepi    <- filterNeuroDrugs(epi, atchashda)
episem <- genDictListFromRawFreq(rawDrugNamesCoOcEPISEM)
neuroepisem <- filterNeuroDrugs(episem, atchashda)
fenics <- genDictListFromRawFreq(rawDrugNamesCoOcFENICS)
neurofenics <- filterNeuroDrugs(fenics, atchashda)
mx <- max(
    c(length(neuroepso), length(neuroesso), length(neuroepi),
```

```
      length(neuroepisem), length(neurofenics)))
dneuro <-
  data.frame(EpSO = c(neuroepso, rep("", (mx-length(neuroepso)))),
             ESSO = c(neuroesso, rep("", (mx-length(neuroesso)))),
             EPILONT = c(neuroepi, rep("", (mx-length(neuroepi)))),
             EPISEM = c(neuroepisem, rep("", (mx-length(neuroepisem)))),
             FENICS = c(neurofenics, rep("", (mx-length(neurofenics)))))
dneuromaxk <- TopKLists::calculate.maxK(dneuro, L=5, d=5, v=10)
neurospace <- as.character(dneuromaxk$topkspace)
dsepso <- calcDSEA(neuroepso, mx)
dsesso <- calcDSEA(neuroesso, mx)
dsepi <- calcDSEA(neuroepi, mx)
dsepisem <- calcDSEA(neuroepisem, mx)
dsfenics <- calcDSEA(neurofenics, mx)
dsspace  <-  calcDSEA (neurospace, mx)
p <- plotDSEA(dsepso, dsesso, dsepi, dsepisem, dsfenics, dsspace, dneuromaxk$maxK)
```

`plotEnrichment` *Plotting functions for enrichment lists*

Description

Plotting functions for enrichment lists

Usage

```
plotEnrichment(enepso, enesso, enepi, enepisem, enfenics, enspace, k)
```

Arguments

`enepso`	list with enrichment for EpSO
`enesso`	list with enrichment for ESSO
`enepi`	list with enrichment for EPILONT
`enepisem`	list with enrichment for EPISEM
`enfenics`	list with enrichment for FENICS
`enspace`	list with enrichment for the combined ranked list
`k`	numeric value for the length to be plotted

Value

the plot object

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
utils::data(rawDrugNamesCoOcESSO, package="epos")
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
utils::data(rawDrugNamesCoOcEPISEM, package="epos")
utils::data(rawDrugNamesCoOcFENICS, package="epos")
atchashda <-
readAtcMapIntoHashMapDrugNamesAtcCodes(
  system.file("extdata", "db-atc.map", package = "epos"), "\t")
atchashaa <-
  readAtcMapIntoHashMapAtcCodesAtcNames(
    system.file("extdata", "db-atc.map", package = "epos"), "\t")
atchashsec <-
  readSecondLevelATC(
    system.file("extdata", "atc-secondlevel.map", package = "epos"), "\t")
epso <- genDictListFromRawFreq(rawDrugNamesCoOcEpSO)
neuroepso <- filterNeuroDrugs(epso, atchashda)
esso <- genDictListFromRawFreq(rawDrugNamesCoOcESSO)
neuroesso   <- filterNeuroDrugs(esso, atchashda)
epi <- genDictListFromRawFreq(rawDrugNamesCoOcEPILONT)
neuroepi    <- filterNeuroDrugs(epi, atchashda)
episem <- genDictListFromRawFreq(rawDrugNamesCoOcEPISEM)
neuroepisem <- filterNeuroDrugs(episem, atchashda)
fenics <- genDictListFromRawFreq(rawDrugNamesCoOcFENICS)
neurofenics <- filterNeuroDrugs(fenics, atchashda)
mx <- max(
    c(length(neuroepso), length(neuroesso), length(neuroepi),
      length(neuroepisem), length(neurofenics)))
dneuro <-
  data.frame(EpSO = c(neuroepso, rep("", (mx-length(neuroepso)))),
             ESSO = c(neuroesso, rep("", (mx-length(neuroesso)))),
             EPILONT = c(neuroepi, rep("", (mx-length(neuroepi)))),
             EPISEM = c(neuroepisem, rep("", (mx-length(neuroepisem)))),
             FENICS = c(neurofenics, rep("", (mx-length(neurofenics)))))
dneuromaxk <- TopKLists::calculate.maxK(dneuro, L=5, d=5, v=10)
neurospace <- as.character(dneuromaxk$topkspace)
enepso <- calcEnrichment(neuroepso)
enesso <- calcEnrichment(neuroesso)
enepi <- calcEnrichment(neuroepi)
enepisem <- calcEnrichment(neuroepisem)
enfenics <- calcEnrichment(neurofenics)
enspace <- calcEnrichment (neurospace)
p <- plotEnrichment(enepso, enesso, enepi, enepisem, enfenics, enspace, dneuromaxk$maxK)
```

`rawDrugNamesCoOcEPILONT`

List drug terms with their frequency co-occurring with terms from the EPILONT ontology in publications since 2015 from the BioASQ 2020 corpus.

Description

List drug terms with their frequency co-occurring with terms from the EPILONT ontology in publications since 2015 from the BioASQ 2020 corpus.

Usage

```
rawDrugNamesCoOcEPILONT
```

Format

A named list of drug term frequencies

Source

The text mining workflows for data generation are described in Mueller, Bernd and Hagelstein, Alexandra (2016) <doi:10.4126/FRL01-006408558>, Mueller, Bernd et al. (2017) <doi:10.1007/978-3-319-58694-6_22>, and Mueller, Bernd and Rebholz-Schuhmann, Dietrich (2020) <doi:10.1007/978-3-030-43887-6_52>. The source data set for generating the data co-occurrence lists is the BioASQ 2020 corpus filtered for documents published since the year 2015. The source ontology for the creation of the dictionary is the Epilepsy Ontology (EPILONT) from https://bioportal.bioontology.org/ontologies/EPILONT

Examples

```
utils::data(rawDrugNamesCoOcEPILONT, package="epos")
```

---

rawDrugNamesCoOcEPISEM

List drug terms with their frequency co-occurring with terms from the EPISEM ontology in publications since 2015 from the BioASQ 2020 corpus.

Description

List drug terms with their frequency co-occurring with terms from the EPISEM ontology in publications since 2015 from the BioASQ 2020 corpus.

Usage

```
rawDrugNamesCoOcEPISEM
```

Format

A named list of drug term frequencies

Source

The text mining workflows for data generation are described in Mueller, Bernd and Hagelstein, Alexandra (2016) <doi:10.4126/FRL01-006408558>, Mueller, Bernd et al. (2017) <doi:10.1007/978-3-319-58694-6_22>, and Mueller, Bernd and Rebholz-Schuhmann, Dietrich (2020) <doi:10.1007/978-3-030-43887-6_52>. The source data set for generating the data co-occurrence lists is the BioASQ 2020 corpus filtered for documents published since the year 2015. The source ontology for the creation of the dictionary is the Epilepsy Semiology Ontology (EPISEM) from https://bioportal.bioontology.org/ontologies/EPIS

Examples

```
utils::data(rawDrugNamesCoOcEPISEM, package="epos")
```

---

rawDrugNamesCoOcEpSO *List drug terms with their frequency co-occurring with terms from the EpSO ontology in publications since 2015 from the BioASQ 2020 corpus.*

Description

List drug terms with their frequency co-occurring with terms from the EpSO ontology in publications since 2015 from the BioASQ 2020 corpus.

Usage

```
rawDrugNamesCoOcEpSO
```

Format

A named list of drug term frequencies

Source

The text mining workflows for data generation are described in Mueller, Bernd and Hagelstein, Alexandra (2016) <doi:10.4126/FRL01-006408558>, Mueller, Bernd et al. (2017) <doi:10.1007/978-3-319-58694-6_22>, and Mueller, Bernd and Rebholz-Schuhmann, Dietrich (2020) <doi:10.1007/978-3-030-43887-6_52>. The source data set for generating the data co-occurrence lists is the BioASQ 2020 corpus filtered for documents published since the year 2015. The source ontology for the creation of the dictionary is the Epilepsy and Seizure Ontology (EpSO) from https://bioportal.bioontology.org/ontologies/EPSO

Examples

```
utils::data(rawDrugNamesCoOcEpSO, package="epos")
```

rawDrugNamesCoOcESSO *List drug terms with their frequency co-occurring with terms from the ESSO ontology in publications since 2015 from the BioASQ 2020 corpus.*

Description

List drug terms with their frequency co-occurring with terms from the ESSO ontology in publications since 2015 from the BioASQ 2020 corpus.

Usage

```
rawDrugNamesCoOcESSO
```

Format

An object of class `numeric` of length 4991.

Source

The text mining workflows for data generation are described in Mueller, Bernd and Hagelstein, Alexandra (2016) <doi:10.4126/FRL01-006408558>, Mueller, Bernd et al. (2017) <doi:10.1007/978-3-319-58694-6_22>, and Mueller, Bernd and Rebholz-Schuhmann, Dietrich (2020) <doi:10.1007/978-3-030-43887-6_52>. The source data set for generating the data co-occurrence lists is the BioASQ 2020 corpus filtered for documents published since the year 2015. The source ontology for the creation of the dictionary is Epilepsy Syndrome Seizure Ontology (ESSO) from https://bioportal.bioontology.org/ontologies/ESS

Examples

```
utils::data(rawDrugNamesCoOcESSO, package="epos")
```

rawDrugNamesCoOcFENICS *List drug terms with their frequency co-occurring with terms from the FENICS ontology in publications since 2015 from the BioASQ 2020 corpus.*

Description

List drug terms with their frequency co-occurring with terms from the FENICS ontology in publications since 2015 from the BioASQ 2020 corpus.

Usage

```
rawDrugNamesCoOcFENICS
```

Format

A named list of drug term frequencies

Source

The text mining workflows for data generation are described in Mueller, Bernd and Hagelstein, Alexandra (2016) <doi:10.4126/FRL01-006408558>, Mueller, Bernd et al. (2017) <doi:10.1007/978-3-319-58694-6_22>, and Mueller, Bernd and Rebholz-Schuhmann, Dietrich (2020) <doi:10.1007/978-3-030-43887-6_52>. The source data set for generating the data co-occurrence lists is the BioASQ 2020 corpus filtered for documents published since the year 2015. The source ontology for the creation of the dictionary is the Functional Epilepsy Nomenclature for Ion Channels (FENICS) from https://bioportal.bioontology.org/ontologies/FENICS

Examples

```
utils::data(rawDrugNamesCoOcFENICS, package="epos")
```

---

readAtcMapIntoHashMapAtcCodesAtcNames

Processes the input file db-atc.map to form a HashMap containing the drug names with ATC codes

---

Description

Processes the input file db-atc.map to form a HashMap containing the drug names with ATC codes

Usage

```
readAtcMapIntoHashMapAtcCodesAtcNames(filename, seperator)
```

Arguments

filename	character vector with the file name of the file db-atc.map
seperator	character vector with the seperator used within the map-file

Value

atchashaa hash with atc codes as keys and atc names as values

Examples

```
atchashaa <-
  readAtcMapIntoHashMapAtcCodesAtcNames(
    system.file("extdata", "db-atc.map", package = "epos"), "\t")
```

`readAtcMapIntoHashMapDrugNamesAtcCodes` *Processes the input file db-atc.map to form a HashMap containing the drug names with ATC codes*

Description

Processes the input file db-atc.map to form a HashMap containing the drug names with ATC codes

Usage

```
readAtcMapIntoHashMapDrugNamesAtcCodes(filename, seperator)
```

Arguments

`filename`	character vector with the file name of the file db-atc.map
`seperator`	character vector with the seperator used within the map-file

Value

atchashda hash with drug names as keys and atc codes as values

Examples

```
atchashda <- readAtcMapIntoHashMapDrugNamesAtcCodes(
  system.file("extdata", "db-atc.map", package = "epos"), "\t")
```

`readSecondLevelATC` *Read the second level ATC classes from the file atc-secondlevel.map*

Description

Read the second level ATC classes from the file atc-secondlevel.map

Usage

```
readSecondLevelATC(filename, seperator)
```

Arguments

`filename`	the file name that is supposed to be atc-secondlevel.map
`seperator`	the csv file delimiter

Value

atchashsec a hash with second level ATC classes as keys and their names as values

Examples

```
atchashsec <-
  readSecondLevelATC(
  system.file("extdata", "atc-secondlevel.map", package = "epos"), "\t")
```

Index