Package 'Onassis'

April 12, 2018

Version 1.0.1

Date 2017-05-10

Title OnASSIs Ontology Annotation and Semantic SImilarity software

Author Eugenia Galeota

Maintainer Eugenia Galeota <eugenia.galeota@gmail.com>

Description
A package that allows the annotation of text with ontology terms (mainly from OBO ontologies) and the computation of semantic similarity measures based on the structure of the ontology between different annotated
samples.

License GPL-2

Depends R (>= 3.4), rJava, OnassisJavaLibs

Imports GEOmetadb, RSQLite, SRAdb, data.table, methods, tools, utils, AnnotationDbi

SystemRequirements Java (>= 1.8)

RoxygenNote 6.0.1

VignetteBuilder rmarkdown, knitr

Suggests BiocStyle, rmarkdown, knitr, htmltools, DT, org.Hs.eg.db, gplots, GenomicRanges

Encoding UTF-8

LazyData yes

biocViews Annotation, DataImport, Clustering, Network, Software, GeneTarget

NeedsCompilation no

R topics documented:

`annotate` `annotate`

Description

this function runs the entityfinder on given input

Usage

```
annotate(inputFileorDf, dictionary, options = NA, outDir = tempdir(),
  multipleDocs = FALSE)
```

Arguments

`inputFileorDf`	the file, directory, or data frame to annotate
`dictionary`	the Conceptmapper dictionary file
`options`	the options to run the finder
`outDir`	the directory where to store the output files from onassis
`multipleDocs`	TRUE if the input file is one but contains multiple documents

Value

dataframe of annotations

Examples

```
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')
myopts <- CMoptions()
paramValueIndex(myopts) <- 40
entities <- Onassis::annotate(readRDS(system.file('extdata', 'vignette_data',
 'GEO_human_chip.rds', package='Onassis')), options = myopts,
 dictionary=sample_dict)
```

annotateTissueDisease annotateTissueDisease

Description

annotateTissueDisease connects to the GEOmetadb to retrieve the metadata of the samples provided in the gsm_list parameter. A dictionary for tissues/cell lines is built rom the tissue_obo file provided. All the samples' metadata are annotated with tissue concepts from the tissue_obo and samples are clustered based on the semantic similarity (default measure) of the samples. In particular, the samples are clustered and then the function cutree is used to cut the clustering tree at the specified height. Tissue semantic sets are then created associating all the samples that are similar above the provided height_threshold. Within each semantic set, samples are annotated with disease concepts from disease_obo parameter. For each disease the function retrieves the columns of the score matrix.

Usage

```
annotateTissueDisease(geo_metadb_path, gsm_list, tissue_obo, disease_obo,
  outdir, height_threshold, score_matrix)
```

Arguments

geo_metadb_path	The full path of the directory where the GEOmetadb.sqlite file is stored
gsm_list	A list of GEO sample ids (GSMs) to annotate with tissue and disease concepts
tissue_obo	The obo ontology containing concepts to identify tissues/cell lines
disease_obo	The obo ontology containing concepts to identify diseases
outdir	The directory where the results will be stored
height_threshold	The percentage of clusters to merge based on the height of the dendrogram produced by the hclust method. Height_threshold is defined in the range [0, 1].
score_matrix	A matrix where rows represent units (GRanges or genes) and columns represent GSMs.

Value

A list of the tissue semantic sets defined by Onassis. For each tissue, a list of diseases and for each disease the columns of the score_matrix that were annotated with a given tissue and a given disease based

Examples

```
if(!file.exists(file.path(getwd(), 'GEOmetadb.sqlite'))){
    message('To run this example please copy GEOmetadb.sqlite in your current working directory')
     } else{
     geo_metadb_path <- getwd()
    score_matrix <- readRDS(system.file('extdata', 'score_matrix.rds', package='Onassis'))
     gsm_list <- colnames(score_matrix)
     tissue_obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
     disease_obo <- system.file('extdata', 'sample.do.obo', package='OnassisJavaLibs')
     outdir = getwd()
     height_threshold <- 0.4
```

```
result_list <- annotateTissueDisease(geo_metadb_path, gsm_list, tissue_obo, disease_obo, outdir, height_
 }
```

CMdictionary-class *CMdictionary class that stores a Conceptmapper dictionary*

Description

CMdictionary is a class that wraps a Conceptmapper ccp-nlp Java dictionary. Its methods allow the creation of a dictionary from OBO ontologies in OBO or OWL format. Different options to build the dictionary are available.

This constructor instantiates a dictionary object for Conceptmapper.

This function builds a dictionary for Conceptmapper.

Method dict_location

Method dict_location<-

Method dictInfo

Method dictInfo<-

Method dictRef

Method dictRef<-

Method dictTypes

Method buildDictionary

This method returns and sets the location of the dictionary.

dict_location<-

This method shows the list of details of the conceptmapper dictionary

This method sets the list of details of the conceptmapper dictionary.

This method retrieves the java reference the conceptmapper dictionary

This method retrieves the java reference to the conceptmapper file

This method builds a dictionary for Conceptmapper.

This method shows a lis of the pre-defined conceptmapper dictionary types

Usage

```
CMdictionary(dict_location = NA_character_, dictInfo = list(),
  dictRef = .jnull())

dictionary(inputFileOrDb = NULL, dictType = "OBO", outputdir = getwd(),
  synonymType = "ALL", taxID = 0)

dict_location(.Object)

dict_location(.Object) <- value

dictInfo(.Object)

dictInfo(.Object) <- value
```

```
dictRef(.Object)

dictRef(.Object) <- value

dictTypes(.Object)

buildDictionary(.Object, outputDir = tempdir(), dictType = "OBO",
  synonymType = "EXACT", inputFileOrDb = NULL, taxID = 0)

## S4 method for signature 'CMdictionary'
dict_location(.Object)

## S4 replacement method for signature 'CMdictionary'
dict_location(.Object) <- value

## S4 method for signature 'CMdictionary'
dictInfo(.Object)

## S4 replacement method for signature 'CMdictionary'
dictInfo(.Object) <- value

## S4 method for signature 'CMdictionary'
dictRef(.Object)

## S4 replacement method for signature 'CMdictionary'
dictRef(.Object) <- value


  ## S4 method for signature
## 'CMdictionary,character,character,character,character,numeric'
buildDictionary(.Object,
  outputDir = tempdir(), dictType = "OBO", synonymType = "EXACT",
  inputFileOrDb = NULL, taxID = 0)

## S4 method for signature 'CMdictionary'
dictTypes(.Object)
```

Arguments

dict_location	The path of the created dictionary file
dictInfo	Information about how dictionary has been created. It is a list with the following fields
dictRef	Reference to the java object representing the dictionary
inputFileOrDb	The local OBO/OWL ontology to be converted into an XML Conceptmapper dictionary. If NA is passed and the dicType parameter is not the generic OBO then the method tries to download the corresponding dictionary from the available repositories. For ENTREZ and TARGET dictionary types a file named gene_info.gz is automatically downloaded from ftp://ncbi.nlm.nih.gov/gene/data/gene_info.gz if not provided by the user. Alternatively an annotation package of the type Org.xx.eg.db from Bioconductor can be used. In

this case the gene ids and their alternative identifiers will be retrieved from the annotation database without the need of downloading a gene_info file.

`dictType` the type of input dictionary

OBO A dictionary that has been created by An OBO file

ENTREZ Entrez genes dictionary

TARGET Entrez genes dictionary, Histone marks and Histone modifications

CMDICT A previously created dictionary file in the Conceptmapper XML format

`outputdir` the directory where the XML conceptmapper dictionary will be stored. Defaults to the tmp system's directory

`synonymType` The type of synonyms to consider when building the dictionary for Conceptmapper. For further detail `http://owlcollab.github.io/oboformat/doc/obo-syntax.html`. Default: EXACT

EXACT

ALL

`taxID` the taxonomy identifier of the organism when the dictionary type is ENTREZ or TARGET. If 0 all the taxonomies will be included in the new dictionary.

`.Object` instance of class CMdictionary

`value` is the list of metadata associated with the dictionary, for example the Dictionary source OBO file, the dicitonary type (OBO, ENTREZ, TARGET) or any other information that the user needs to store.

`outputDir` the directory where the XML conceptmapper dictionary will be stored. Defaults to the tmp system's directory

Details

The following methods can be applied to CMdictionary

`dictTypes`
`buildDictionary`
`dictInfo`

Value

An object of type CMdictionary that can be used to create a dictionary

An object of type CMdictionary that can be used to annotate text with the `EntityFinder`.

The location of the dictionary

list of details about the dictionary

the dictionary with updated dictInfo field

java reference to the Conceptmapper dictionary

list of details about the dictionary

An object of type CMdictionary that can be used to annotate text with the `EntityFinder`.

the list of dictionary types available

Slots

`dict_location` The path of the created dictionary file

`dictRef` Reference to the java object representing the dictionary

`dictInfo` Information about how dictionary has been created. It is a list with the following fields

- Dictionary Type:The type of dictionary
  - OBOA dictionary that has been created by An OBO file
  - ENTREZEntrez genes dictionary
  - TARGETEntrez genes dictionary, Histone marks and Histone modifications
  - CMDICTA previously created dictionary file in the Conceptmapper XML format
- SynonymType: The type of synonyms to consider when building the dictionary for Conceptmapper. For further detail http://owlcollab.github.io/oboformat/doc/obo-syntax.html
  - EXACT
  - BROAD
  - NARROW
  - RELATED
  - ALL
- dictSource: The OBO/OWL dictionary file to convert to a Conceptmapper dictionary in case the type is OBO. The XML file in case the type is CMDICT
- taxID The NCBI taxon identifier for species to create the Entrez gene dictionary (e.g 9606 for Mus musculus)

Examples

```
dict <- new('CMdictionary')

sample_dict <- CMdictionary()


obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')

dictionary <- CMdictionary()
loc <- dict_location(dictionary)
dictionary <- CMdictionary()
dict_location(dictionary) <- getwd()

dictionary <- CMdictionary()
dictInfo(dictionary)
dictionary <- CMdictionary()
dictInfo(dictionary) <-
list(Dictionary_type =  'ENTREZ from OrgDb', Dictionary_source ='OrgDb')
dictionary <- CMdictionary()
dictRef(dictionary)
dictionary <- CMdictionary()
dict_file <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dictRef(dictionary) <- .jnew('java/io/File', dict_file)
dictionary <- CMdictionary()
## Not run:
#' ##This might take some time to download the dictionary
dict <- buildDictionary(dictionary, dictType = 'TARGET', inputFileOrDb='org.Hs.eg.db')
```

```
dict_file <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dictionary <- buildDictionary(dictionary, dictType='OBO', inputFileOrDb=dict_file)

## End(Not run)
dictionary <- CMdictionary()
dictTypes(dictionary)
```

CMoptions-class *CMoptions*

Description

CMoptions is a class that represents Conceptmapper configurations. It allows users to show and set the possible combinations of different parameters for Conceptmapper running.

This function shows the list of possible combinations of options to run the entity finder

Method arguments

Method arguments<-

Method listCombinations

Method paramValueIndex

Method paramValueIndex<-

This method retrieves the list of parameters currently set to run Conceptmapper

This method sets a list of parameters and updates the paramValueIndex to the correct value if necessary.

This method shows the list of options to run the Entity finder

This method retrieves all the possible parameter combination for Conceptmapper.

paramValueIndex An integer value to index the 576 parameter combinations

SearchStrategy The matching strategy for finding concepts in the input text

- CONTIGUOUS_MATCHLongets match of contiguous tokens within enclosing span
- SKIP_ANY_MATCHLongest match of not-necessarily contiguous tokens
- SKIP_ANY_MATCH_ALLOW_OVERLAPLongest match of not-necessarily contiguous tokens, overlapping matches are allowed

CaseMatch

- CASE_IGNOREFold everything to lowercase for matching
- CASE_INSENSITIVEFold only tokens with initial caps to lowercase
- CASE_FOLD_DIGITSFold all (and only) tokens with a digit
- CASE_SENSITIVEPerform no case folding

Stemmer

- BIOLEMMATIZER A stemmer specific for biomedical literature
- PORTER A stemmer that removes the commoner morphological and inflexional endings from words in English
- NONE No word stemming

StopWords

- PUBMED A list of stop words obtained analyzing Pubmed papers
- NONE No stop words

OrderIndependentLookup

- ON Ordering within span is ignored (i.e. 'Breast cancer' would equal 'Cancer breast')
- OFF Ordering is taken into consideration

FindAllMatches • YES All the matches within the span are found
• NO Only the longest match within the span will be returned

SynonymType • EXACT_ONLY Only exact synonyms are considered
• ALL All synonym types are included

This method retrieves the parameter combination index corresponding to a given parameter combination. The value of the paramValueIndex lays in the range [0:575]

This method sets the parameter combination for the given index. When the paramValueIndex is set to a given value the corresponding settings for the other parameters are automatically loaded.

Usage

```
CMoptions(arguments = list())

arguments(x)

arguments(x) <- value

listCombinations(x)

paramValueIndex(x)

paramValueIndex(x) <- value

## S4 method for signature 'CMoptions'
arguments(x)

## S4 replacement method for signature 'CMoptions'
arguments(x) <- value

## S4 method for signature 'CMoptions'
show(object)

## S4 method for signature 'CMoptions'
listCombinations(x)

## S4 method for signature 'CMoptions'
paramValueIndex(x)

## S4 replacement method for signature 'CMoptions'
paramValueIndex(x) <- value
```

Arguments

arguments	the list of parameters to set for conceptmapper
x	CMoptions instance
value	a list of arguments in the containing paramValueIndex (options), SearchStrategy, CaseMatch, Stemmer, Stopwords, OrderIndependentLookup, FindAllMatches and SynonymType parameters in the specified order.
object	CMoptions instance

Details

The following methods can be applied to CMoptions

`paramValueIndex`
`show`
`arguments`
`listCombinations`

Value

instance of the class CMoptions set to the default combination of values

list of parameters currently set

The updated CMoptions S4 object

the list of options

The data frame with all the possible parameter combinations

The paramValueIndex corresponding to the current options. If user has not changed the options this is set to 31

The updated CMoptions S4 object

Slots

`arguments` The following argument can be set to run Conceptmapper

paramValueIndex An integer value to index the 576 parameter combinations

SearchStrategy The matching strategy for finding concepts in the input text

- CONTIGUOUS_MATCHLongets match of contiguous tokens within enclosing span
- SKIP_ANY_MATCHLongest match of not-necessarily contiguous tokens
- SKIP_ANY_MATCH_ALLOW_OVERLAPLongest match of not-necessarily contiguous tokens, overlapping matches are allowed

CaseMatch • CASE_IGNOREFold everything to lowercase for matching

- CASE_INSENSITIVEFold only tokens with initial caps to lowercase
- CASE_FOLD_DIGITSFold all (and only) tokens with a digit
- CASE_SENSITIVEPerform no case folding

Stemmer • BIOLEMMATIZER A stemmer specific for biomedical literature

- PORTER A stemmer that removes the commoner morphological and inflexional endings from words in English
- NONE No word stemming

StopWords • PUBMED A list of stop words obtained analyzing Pubmed papers

- NONE No stop words

OrderIndependentLookup • ON Ordering within span is ignored (i.e. 'Breast cancer' would equal 'Cancer breast')

- OFF Ordering is taken into consideration

FindAllMatches • YES All the matches within the span are found

- NO Only the longest match within the span will be returned

SynonymType • EXACT_ONLY Only exact synonyms are considered

- ALL All synonym types are included

Examples

```
options <- new('CMoptions')
op <- CMoptions()
opts <- CMoptions()
arguments(opts)
opts <- CMoptions()
arguments(opts) <- as.list(c('CONTIGUOUS_MATCH', 'CASE_IGNORE',
 'BIOLEMMATIZER', 'NONE', 'OFF', 'YES', 'EXACT_ONLY'))
opt <- CMoptions()
show(opt)
opts <- CMoptions()
listCombinations(opts)
opts <- CMoptions()
paramValueIndex(opts)

opts <- CMoptions()
list_opts <- listCombinations(opts)

paramValueIndex(opts) <- 2
```

`connectToGEODB` `connectToGEODB`

Description

This method allows users to connect to the GEOmetadb downloaded. If no parameter is provided than the function retrieves the database in sqlite format and returns a connection to query the database

Usage

```
connectToGEODB(sqliteFileName = NULL, download = FALSE, destdir = getwd())
```

Arguments

`sqliteFileName`	optional SQLite file path of the SQLite database if already downloaded
`download`	If TRUE allow the automatic downloading of the database file.
`destdir`	optional destination directory

Value

A connection to the GEOmetadb

Examples

```
## Not run:
 geo_connection <- connectToGEODB(download=TRUE)

## End(Not run)
 if(file.exists('GEOmetadb.sqlite')){
    geo_con <- connectToGEODB()
 } else {
 message('Please provide GEOmetadb.sqlite file')
 }
```

connectToSRADB connectToSRADB

Description

This method allows users to connect to the SRAdb downloaded. If no parameter is provided than the function retrieves the database in sqlite format and return a connection to query the database

Usage

```
connectToSRADB(sqliteFileName = NULL)
```

Arguments

sqliteFileName	optional SQLite file path of the SQLite database if already downloadd

Value

A connection to the SRAdb

Examples

```
if(file.exists('SRAdb.sqlite')){
 sra_con <- connectToSRADB('SRAdb.sqlite')
}else{
 print('You need to download SRAdb.sqlite to run this example')
}
```

create_score_matrix createScoreMatrix

Description

This functions allows the creation of a score matrix for genomic regions overlapping the genomic regions provided as reference (e.g. promoter regions). Each entry of the matrix corresponds to 1 if the sample represented on the column overlaps the genomic interval represented on the row.

Usage

```
create_score_matrix(ref_granges, granges_list)
```

Arguments

ref_granges	An object of type GRanges to be considered as the reference GRanges
granges_list	A list of GRanges, one for each sample, to be mapped on the reference granges

Value

A logical score matrix where the number of rows corresponds to the number of different genomic intervals in the ref_granges object and each column is associated to a Grange in the granges_list. Score(i, j) is set to 1 if the j-th sample has a interval overlapping the one in the i-th row of the reference GRanges.

Examples

```
granges <- readRDS(system.file('extdata', 'sample_granges.rds', package='Onassis'))
ref_granges <- granges[[1]]

for(i in 2:length(granges)) {
 ref_granges <- GenomicRanges::union(ref_granges, granges[[i]])
}
score_mat <- create_score_matrix(ref_granges, granges)
```

`EntityFinder-class` *EntityFinder class to create a Conceptmapper instance*

Description

EntityFinder is a class that wraps a Conceptmapper pipeline using the CCP UIMA Type System https://github.com/UCDenver-ccp/ccp-nlp. The pipeline includes a sentence detector, offset tokenizer and retrieves concepts from dictionaries built from OBO/OWL formatted ontology files.

This function shows the list of possible combinations of options to run the entity finder

Method typeSystemRef

Method typeSystemRef<-

Method annotateDF

Method findEntities

This method sets the type system to the ccp-nlp one to run the EntityFinder

This method sets the type system to the ccp-nlp one to run the EntityFinder

This method finds concepts of a Conceptmapper Dictionary of type CMdictionary in a given directory or in a single pipe separated file containing a named document in each row, with a specified configuration of type CMoptions.

This method finds concepts of a Conceptmapper Dictionary of type CMdictionary of data contained in a data frame, with a specified configuration of type CMoptions.

Usage

```
EntityFinder(typeSystemRef = .jnull())

typeSystemRef(x)

typeSystemRef(x) <- value

annotateDF(object, descr_df, outDir = tempdir(), configOpt, cmDict)

findEntities(object, inputDirOrFile, multipleDocs = FALSE,
  outDir = tempdir(), configOpt, cmDict)

## S4 method for signature 'EntityFinder'
typeSystemRef(x)

## S4 replacement method for signature 'EntityFinder'
```

```
typeSystemRef(x) <- value


  ## S4 method for signature
## 'EntityFinder,character,logical,character,CMoptions,CMdictionary'
findEntities(object,
  inputDirOrFile, multipleDocs = FALSE, outDir = tempdir(), configOpt,
  cmDict)

## S4 method for signature 'EntityFinder,data.frame,character,CMoptions'
annotateDF(object,
  descr_df, outDir = tempdir(), configOpt, cmDict)
```

Arguments

typeSystemRef	the reference to the ccp-nlp type system
x	instance of the class EntityFinder
value	the java type system to detect concepts from ontologies.
object	instance of the class EntityFinder
descr_df	the table of text to annotate. The data frame should have identifiers in the first column and descriptions or text in the rest of the columns.
outDir	The directory where the Conceptmapper annotated files are stored. Default: the system tmp directory.
configOpt	Object of type CMoptions in which the parameters to run Conceptmapper are stored
cmDict	Object of type CMdictionary containing the reference to a previously created Conceptmapper dictionary. Alternatively the path to a Conceptmapper xml file can be passed.
inputDirOrFile	the directory where the files to annotate are stored or the text file to annotate. A single file containing in each row sample names, the | symbol and the description of the sample is also allowed.
multipleDocs	TRUE if a single file containing different text sources has been given as inputDirOrFile. FALSE if each text is in a separate file. Defaults to FALSE

Details

The following methods can be applied to EntityFinder

findEntities
buildDictionary

Value

instance of the class CMoptions set to the default combination of values

the reference to the Java type system currently set

The updated EntityFinder S4 object

A data frame of annotations containing the sample name, the id of the OBO concept, the corresponding name, the part of the text containing the annotation

A data frame of annotations containing the sample name, the id of the OBO concept, the corresponding name, the part of the text containing the annotation

Slots

typeSystemRef The reference to the Java object representing the type system

Examples

```
finder <- new('EntityFinder')
op <- CMoptions()
ef <- EntityFinder()
typeSystemRef(ef)
ef <- EntityFinder()
type_system_array_list <- .jnew('java/util/ArrayList')
ccp_nlp_type_system <- .jfield('edu/ucdenver/ccp/nlp/uima/util/TypeSystemUtil',
 name = 'CCP_TYPE_SYSTEM')
sentence_detector_type_system_str <- 'org.cleartk.token.type.Sentence'
conceptmapper_type_system <-
'edu.ucdenver.ccp.nlp.wrapper.conceptmapper.TypeSystem'
dictTerm <- 'analysis_engine.primitive.DictTerm'
tokenizer <- 'org.apache.uima.conceptMapper.support.tokenizer.TokenAnnotation'
vector_of_ts <- c(ccp_nlp_type_system, sentence_detector_type_system_str,
 conceptmapper_type_system, dictTerm, tokenizer)
type_system_description <-
J('org/uimafit/factory/TypeSystemDescriptionFactory')$createTypeSystemDescription(vector_of_ts)
typeSystemRef(ef) <- type_system_description
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')

opts <- CMoptions()
ef <- EntityFinder()
annotations <- findEntities(ef,
system.file('extdata', 'test_samples', 'test_samples.txt', package='Onassis'), multipleDocs=TRUE, outDir=ge
 configOpt=opts, cmDict=dict)

obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')
opts <- CMoptions()
ef <- EntityFinder()
methylation <- readRDS(system.file('extdata', 'vignette_data',
'GEOmethylation.rds', package='Onassis'))
annotations <- annotateDF(ef, methylation[1:10, ], getwd(), opts, dict)
```

---

experiment_types experiment_types

Description

This method retrierves the experiment types stored in GEOmetadb

Usage

```
experiment_types(GEOcon)
```

Arguments

GEOcon	connection to the SQLite GEOmetadb databse

Value

A character vector with all the possible experiment values

Examples

```
if(file.exists('GEOmetadb.sqlite')){
 geo_con <- connectToGEODB('GEOmetadb.sqlite')
 experiments <- experiment_types(geo_con)
}else{
 print('You need to download GEOmetadb.sqlite to run this example')
}
```

`filterTerms` `filterTerms`

Description

filterTerms allows users to remove a set of defined terms from the results of the annotation provided by Onassis

Usage

```
filterTerms(annotated_df, termlist = c())
```

Arguments

`annotated_df`	a data frame resulting from the annotation process of Onassis
`termlist`	the list of terms to be filtered out from the annotations

Value

a data frame with removed annotations

Examples

```
metadatafile <- readRDS(system.file('extdata', 'vignette_data',
'GEO_human_chip.rds', package='Onassis'))
healthy_gsms <- findHealthy(metadatafile)
```

findHealthy findHealthy

Description

findHealthy annotates as 'Healthy' the samples whose metadata matches with one of the elements of a list of sentence used to describe the normal disease state or healthy.

Usage

```
findHealthy(metadata_df)
```

Arguments

metadata_df	a data frame where the first column corresponds to the identifier of a sample #' and the other columns the metadata relative to the sample

Value

a data frame with healthy samples annotated as 'Healthy'

Examples

```
metadatafile <- readRDS(system.file('extdata', 'vignette_data',
'GEO_human_chip.rds', package='Onassis'))

healthy_gsms <- findHealthy(metadatafile)
```

getGEOMetadata getGEOMetadata

Description

This method retrierves the descriptive fields of the samples in GEO for a given experiment_type, organism or platform.

Usage

```
getGEOMetadata(geo_con, experiment_type = NA, organism = NA, gpl = NA)
```

Arguments

geo_con	connection to the SQLite GEOmetadb databse
experiment_type	The type of experiment. Allowed values can be obtained through the function experiment_types
organism	Optional type of organism. Allowed species can be obtained using the function organism_types. If no organism is passed as parameter the query will retrieve all the organisms
gpl	Optional platform identifier in case a platform based query has to be executed

Value

A data frame with the queried samples' metadata

Examples

```
if(file.exists('GEOmetadb.sqlite')){
 geo_con <- connectToGEODB('GEOmetadb.sqlite')
methilation <- getGEOMetadata(geo_con,
'Methylation profiling by high throughput sequencing', 'Homo sapiens')
expression <- getGEOMetadata(geo_con,
 'Expression profiling by array', 'Homo sapiens', 'GPL570')
}else{
 print('You need to download GEOmetadb.sqlite to run this example')
}
```

`getSRAMetadata` `getSRAMetadata`

Description

This method retrierves the descriptive fields of experiments and samples in SRAdb for a given library_strategy (experiment type), of a given organism and center name.

Usage

```
getSRAMetadata(sra_con, library_strategy, library_source = NA,
  taxon_id = NA, center_name = NA)
```

Arguments

`sra_con`	connection to the SQLite SRAdb databse
`library_strategy`	corresponding to the type of experiment. Allowed values can be obtained through the function `library_strategies`
`library_source`	corresponding to the material from the sample (GENOMIC, METAGENOMIC, TRANSCRIPTOMIC...)
`taxon_id`	Optional taxon id of the organism. If no taxon_id is passed as parameter the query will retrieve all the organisms
`center_name`	Optional center name identifier. Possible values can be obtained querying the database e.g. dbGetQuery(sra_con, 'select distinct center_name from sample')

Value

A data frame with the queried samples' metadata

Examples

```
if(file.exists('SRAdb.sqlite')){
   sra_con <- connectToSRADB('SRAdb.sqlite')
   geo_human_chip <- getSRAMetadata(sra_con, library_strategy='ChIP-Seq',
   library_source='GENOMIC', taxon_id=9606, center_name='GEO')
   bisulfite_seq <- getSRAMetadata(sra_con, library_strategy='Bisulfite-Seq',
    library_source='GENOMIC', taxon_id=9606, center_name='GEO')
 } else{
   print('You need to download the SRAdb.sqlite to run this example')
   }
```

---

library_sources library_sources

Description

This method retrierves the library source types stored in SRAdb

Usage

```
library_sources(SRAcon)
```

Arguments

SRAcon	connection to the SQLite SRAdb databse

Value

A character vector with all the possible library sources

Examples

```
if(file.exists('SRAdb.sqlite')){
   sra_con <- connectToSRADB('SRAdb.sqlite')
   sources <- library_sources(sra_con)
   head(sources)
   }else{
   print('You need to download the SRAdb.sqlite to run this example')
   }
```

---

library_strategies library_strategies

Description

This method retrierves the library strategy types stored in SRAdb

Usage

```
library_strategies(SRAcon)
```

Arguments

SRAcon	connection to the SQLite SRAdb databse

Value

A character vector with all the possible library strategy values

Examples

```
if(file.exists('SRAdb.sqlite')){
   sra_con <- connectToSRADB('SRAdb.sqlite')
   strategies <- library_strategies(sra_con)
   head(strategies)
 } else{
   print('You need to download the SRAdb.sqlite to run this example')
 }
```

OnASSiS *OnASSiS (Ontology Annotations and Semantic Similarity software)*

Description

OnASSiS (Ontology Annotations and Semantic Similarity software) is a package for the annotation of any given text with concepts from biomedical ontologies that also provides features to relate the concepts using semantic similarity metrics.

Details

OnASSiS package

OnASSiS (Ontology Annotations and Semantic Similarity software) is a package that uses Conceptmapper, an Apache UIMA (Unstructured Information Management Architecture) https://uima.apache.org/downloads/sandbox/ConceptMapperAnnotatorUserGuide/ConceptMapperAnnotatorUserGuide.html dictionary lookup tool to retrieve dictionary terms in a given text.

In particular a Conceptmapper wrapper specific for the biomedical domain, ccp-nlp, (https://github.com/UCDenver-ccp/ccp-nlp) has been personalized to retrieve concepts from OBO ontologies in a given text with different options.

The package also provides the possibility to annotate Gene Expression Omnibus (GEO) metadata for stored experiments and samples.

Different annotated sets of text can be then compared using semantic similarity metrics based on the structure of the biomedical ontologies. The semantic similarity module has been obtained using the Java slib (http://www.semantic-measures-library.org/sml/)

organism_types organism_types

Description

This method retrierves the allowed organisms in GEOmetadb

Usage

```
organism_types(geo_con)
```

Arguments

geo_con	connection to the SQLite GEOmetadb databse

Value

A character vector with all the possible organism values

Examples

```
if(file.exists('GEOmetadb.sqlite')){
 geo_con <- connectToGEODB('GEOmetadb.sqlite')
 species <- organism_types(geo_con)
}else{
 print('You need to download GEOmetadb.sqlite to run this example')
}
```

semanticdifference semanticdifference

Description

semanticdifference function allows to test, within a given tissue class, the difference between i) the healthy samples and one or more disease states and ii) if there is a significant difference between the different disease states for each unit represented in each row of a scorematrix containing as rows genomic units and as columns GEO sample ids.

Usage

```
semanticdifference(score_matrix, list_of_annotations, fun_name, test_type)
```

Arguments

score_matrix	A matrix where rows represent units (GRanges or genes) and columns #' represent GSMs.
list_of_annotations	A list of lists where the firt level elements represent a tissue/cell line semantic set, the second level elements represent disease semantic sets and each element of the list is a named score matrix with column names corresponding to sample (GSM) identifiers

fun_name — The name of a testing function to measure the differences between semantic classes. For the test_type parameter = 'pair' the semanticdifference function applies the test function to the tissue semantic states including 'Healthy' and one or or more diseases. In this case the function (e.g. wilcoxon.test) should take as input a couple of vectors and return as output the value of the of the statistic '$statistic' field and a p-value '$p.value' field. For the test_type = 'multiple' the function tests if there are differences between the semantic disease classes within the tissue class. The function (e.g. kruskal.test) takes as input a list of vectors (one for each disease semantic set) and returns the value of the statistic in the '$statistic' field and the corresponding p-value.

test_type — This value can be 'pair' or 'multiple'. In case it is set to pair the function to provide in fun_name requires the tissue semantic class to include at least two disease semantic classes of which one has to be 'Healthy' to compare diseases against healthy. For example the 'wilcoxon.test'. When set to multiple the fun_name provided should take as input a list of input vectors corresponding to the different disease conditions. A minimum of three is required. For example the 'kruskal.test' can be used.

Value

The semanticdifference function returns a list where each element corresponds to a tissue/cell line semantic state. The content of the list for 'multiple' test types a matrix of the test results and corresponding p.values.

Examples

```
granges <- readRDS(system.file('extdata', 'sample_granges.rds', package='Onassis'))
ref_granges <- granges[[1]]
for(i in 2:length(granges)) {
 ref_granges <- GenomicRanges::union(ref_granges, granges[[i]])
}
score_mat <- create_score_matrix(ref_granges, granges)
gsm_list <- names(granges)
list_of_annotations <- readRDS(system.file('extdata', 'list_of_annotations.rds', package='Onassis'))
fun_name = 'wilcox.test'
fun_type = 'pair'
sem_dif <- semanticdifference(score_mat, list_of_annotations, fun_name, fun_type)
```

---

showSimilarities showSimilarities

Description

This function shows the list of pairwise, information content and groupwise measures to compute the semantic similarities

Usage

```
showSimilarities()
```

Value

the list of options

Examples

```
measures <- showSimilarities()
```

`Similarity-class` *Similarity class to compute similarities between concepts in ontologies and samples annotated with different concepts* `Similarity-class`

Description

Similarity is a class that wraps some methods of the Java library slib http://www.semantic-measures-library.org/sml/. Starting from OBO ontologies it is possible to build semantic graphs that allow the computation of different similarity measures between concepts belonging to the same ontology, group of concepts, samples annotated with different ontology concepts. Further details about the graph based semantic similarity measures are available at http://www.semantic-measures-library.org/sml/index.php?q=doc_graph_based_advanced

this constructr initializes the Similarity class to compute the similarity between couple of terms, couple of samples, or group of terms

this function computes the similarity between couple of terms, couple of samples, or group of terms

Method similarityInstance

Method similarityInstance<-

Method icConfig

Method icConfig<-

Method pairwiseConfig

Method pairwiseConfig<-

Method pairwiseConfigRef

Method pairwiseConfigRef<-

Method groupwiseConfigRef

Method groupwiseConfigRef<-

Method groupConfig

Method groupConfig<-

Method ontology

Method ontology<-

Method showOpts

Method sim

Method groupsim

Method samplesim

Method multisim

This method retrieves the java object referencing the Similarity class

`similarityInstance<-`

This method retrieves the configuration of the intrinsic information content measure

This method sets the configuration of the intrinsic information content measure by taking as parameter the short flag associated to the information measure. To have details about the available short flags see the pairwiseConfig help

This method retrieves the reference to the Java configuration used to compute semantic similarities.

configures the pairwise java object to compute semantic similarity between two concepts of a given ontology, by passing as input the java reference to one of the allowed pairwise semantic similarity measures. For a complete list check the details section of the function pairwiseConfig.

This method shows the value of the pairwise configuration.

and configures the pairwise measure to compute semantic similarity between two concepts of a given ontology.To set the pairwise measure one of the available short flags described in details should be used.

This method shows the value of the groupwise configuration used to compute semantic similarities between groups of concepts.

Sets the groupwise measure to the reference of a groupwise measure to the semantic similarity between groups of concepts. For available meausres see the groupConfig function's details.

This method shows the value of the groupwise configuration used to compute semantic similarities between groups of concepts.

Sets the groupwise measure to compute the semantic similarity between groups of concepts. For available meausres use the method showOpts(sim).

This method shows a list of the possible measures to compute pairwise and groupwise semantic similarity between concepts

This method creates a semantic graph to compute semantic similarity between concepts. It takes as input an OBO ontology in RDF, OWL or OBO format.

This method shows the ontology.

This method computes the semantic similarity between two terms of a given ontology.

This method computes the semantic similarity between two groups of terms of a given ontology.

This method computes the semantic similarity between two named samples annotated with a group of ontolgy terms belonging to the same ontology

This method computes the semantic similarity between samples annotated with different ontology terms from different ontologies

Usage

```
Similarity(pairwiseConfig = NA_character_, icConfig = NA_character_,
  groupConfig = NA_character_)

similarity(ontologyFile, termlist1, termlist2, annotatedtab = NA,
  pairConf = c("resnik", "seco"), groupConf = "ui")

similarityInstance(object)

similarityInstance(object) <- value

icConfig(object)

icConfig(object) <- value

pairwiseConfig(object)
```

```
pairwiseConfig(object) <- value

pairwiseConfigRef(object)

pairwiseConfigRef(object) <- value

groupwiseConfigRef(object)

groupwiseConfigRef(object) <- value

groupConfig(object)

groupConfig(object) <- value

ontology(object)

ontology(object) <- value

showOpts(object)

sim(object, term1, term2)

groupsim(object, termList1, termList2)

samplesim(object, sample1, sample2, annotated_df)

multisim(similarities, annotations, sample1, sample2, aggregating_function)

## S4 method for signature 'Similarity'
similarityInstance(object)

## S4 replacement method for signature 'Similarity'
similarityInstance(object) <- value

## S4 method for signature 'Similarity'
icConfig(object)

## S4 replacement method for signature 'Similarity'
icConfig(object) <- value

## S4 method for signature 'Similarity'
pairwiseConfigRef(object)

## S4 replacement method for signature 'Similarity'
pairwiseConfigRef(object) <- value

## S4 method for signature 'Similarity'
pairwiseConfig(object)

## S4 replacement method for signature 'Similarity'
pairwiseConfig(object) <- value
```

```
## S4 method for signature 'Similarity'
groupwiseConfigRef(object)

## S4 replacement method for signature 'Similarity'
groupwiseConfigRef(object) <- value

## S4 method for signature 'Similarity'
groupConfig(object)

## S4 replacement method for signature 'Similarity'
groupConfig(object) <- value

## S4 method for signature 'Similarity'
showOpts(object)

## S4 replacement method for signature 'Similarity'
ontology(object) <- value

## S4 method for signature 'Similarity'
ontology(object)

## S4 method for signature 'Similarity,character,character'
sim(object, term1, term2)

## S4 method for signature 'Similarity,character,character'
groupsim(object, termList1,
  termList2)

## S4 method for signature 'Similarity,character,character,data.frame'
samplesim(object, sample1,
  sample2, annotated_df)

## S4 method for signature 'list,list,character,character'
multisim(similarities, annotations,
  sample1, sample2, aggregating_function = "mean")
```

Arguments

pairwiseConfig	the pairwise configuration
icConfig	the information content configuration
groupConfig	the groupwise configuration
ontologyFile	the file in OBO or RDF format to build the graph
termlist1	The URI or character vector of URIs belonging to the first set or the sample id of an annotated data frame
termlist2	The URI or character vector of URIs belonging to the second set or the sample id of the annotated data frame
annotatedtab	The data frame of annotations
pairConf	a configuration for the pairwise meausres. Defaults to resnik and seco.
groupConf	one of the allowed configurations for groupwise measures

object	Instance of the class Similarity-class
value	the reference to a java Similarity object
term1	The URI of the ontology term in the format http://purl.obolibrary.org/obo/Ontology_id (e.g 'http://purl.obolibrary.org/obo/CL_0000542')
term2	The URI of the ontology term
termList1	A vector of URIs of ontology terms in the format http://purl.obolibrary.org/obo/Ontology_id (e.g http://purl.obolibrary.org/obo/BTO_0004732)
termList2	A vector of URIs of ontology terms
sample1	A sample ID with its annotations available in a data frame
sample2	A sample ID with its annotations available in a data frame
annotated_df	data frame with annotations obtained using entityFinder. The data frame should have at least a column named 'sample_id' with the sample identifier and a column named 'term_url' with the URL of the ontology terms annotating the sample. The ontology terms must belong to the ontology loaded in the Similarity class.
similarities	a list of Similarity instances, one for each ontology used to annotate the data
annotations	a list of annotated data frames obtained using annotateDF or findEntities, one for each ontology
aggregating_function	A function used to aggregate the single similarities obtained from each ontology annotation. The function should be applied to a numeric vector. The default value is 'mean'

Details

The following methods can be applied to Similarity

ontology<-
pairwiseConfig
groupConfig
showOpts
sim
groupsim
samplesim

The following measures can be used to compute semantic similarities between two concepts.

- 'edge_rada_lca' : Computes the similarity of two concepts based on the shortest path linking the two concepts.
  $sim(u, v) = 1/sp(u, v)$
- 'edge_wupalmer': Computes the similarity of two concepts based on the depth of the concepts and the depth of their most specific common ancestor
  $sim(u, v) = depth(MSCA[u, v])/(depth(u) + depth(v))$
- 'edge_resnik': Computes the similarity of two concepts based on the shortest path between the concepts and the maximum depth of the taxonomy
  $(2 * max_depth - min_sp(u, v))/(2 * max_depth)$
  max_depth is the maximum depth in the ontology
  sp(u,v) is the shortest path legnth between u and v

- 'edge_leachod': Computes the similarity of two concepts based on the shortest path as Rada but also considering the depth of the ontology
  $sim(u, v) = -log((sp(u, v) + 1)/2 * max_depth)$
- 'edge_slimani': Computes the similarity of two concepts based on the depth of the most specific common ancesto and the max depth of the concepts
  $sim(u, v) = 2 * depth(MCA)/((depth(u) + depth(v) + 1) * pf))$
  depth(MCA) is the maximum depth of the most common ancestor of the concepts
  pf is a penalization factor used when concepts belong to the same hierarchy

The following measures require the specification of an additional meausre to compute the information content of nodes.

- 'lin': Computes the similarity between two concepts based on the information content of the two concepts and the information content of the most informative common ancestor of the two concepts
  $sim(u, v) = (2 * IC(MICA))/(IC(u) + IC(v))$
  IC(MICA) is the information content of the most informative common ancestor of u and v. MICA is the concept in the ancestors of both u and v that maximizes the Information Content measure.
- 'resnik': Computes the similarity between two concepts based on the information content of the most informative common ancestors of the compared concepts
  $sim(u, v) = IC(MICA)$
- 'schlicker': Computes the similarity between two concepts based on the information concent of the most informative common ancestor of the compared concepts and its probability of occurrence
  $sim(u, v) = (2 * IC(MICA))/(IC(u) + IC(v)) * (1 - Prob_M ICA)$
  Prob_MICA is the probability of occurrence of the most informative common ancestor of the compared concepts
- 'jaccard': Computes the similarity between two concepts based on the information content of the most informative common ancestor.
  $sim(u, v) = IC(MICA)/(IC(u) + IC(v) - IC(MICA))$ if the sum of the IC of the concepts is different from the IC of the MICA else sim(u, v) = 0.
- 'sim': This measure is based on `lin` similarity
  $sim(u, v) = lin(u, v) - (1 - (1/(1 + IC(MICA))))$
- 'jc_norm': Computes the similarity between two concepts based on the IC of the most informative ancestor of the concpets
  $sim(u, v) = 1 - (IC(u) + IC(v) - 2 * IC(MICA))/2$

Information content based measures require the configuration parameter for estimating concept specificity. Intrinsic estimation uses the topological properties of the taxonomic backbone of the semantic graph. There are different options:

- 'zhou': Intrinsic estimation of the specificity of the concepts based on their depth in the ontology.
  $IC(c) = k(1 - log(D(c))/log(|C|)) + (1 - k)(log(max(depth(x)))/log(depth_max))$
  k is a factor to adjust the weight of the two items of the equation
  D(c) is the number of hyponims of concept c
  |C| is the number of concepts in the ontology
  depth(c) is the maximum depth of concept c
  depth_max is the maximum depth in the ontology

- 'resnik_1995': Intrinsic estimation of the specificity of concepts based on the number of ancestors of the concept.
  $IC(c) = |A(c)|$
- 'seco'Intrinsic estimation of the specificity of the concepts based on the number of concepts they subsume.
  $IC(c) = 1 - (log(D(c)/log(|C|))$
  D(c) is the number of hyponims of concept c
  |C| is the number of concepts in the ontology
- 'sanchez': Intrinsic estimation of the specificity of the concepts based on the number of leaves and the number of subsumers of the concepts
  $IC(c) = -log(x/nb_leaves + 1)$ with $x = |leaves(c)|/|A(c)|$
  nb_leaves is the represents the number of leaves corresponding to the root node of the hierarchy
  leaves(c) is the number of leaves corresponding to the concept c
  |A(c)| is the number of concepts that subsume c
- 'anc_norm': Intrinsic estimation of the specificity of concepts based on the number of ancestors of a given concept normalized on the number of concepts in the ontology.
- 'depth_min_non_linear': Intrinsic estimation of the specificity of concepts based on their minimum depth.
- 'depth_max_non_linear': Intrinsic estimation of the specificity of concepts based on their maximum depth.

The following measures are indirect groupwise measures, meaning that they are used to aggregate individual pairwise measures.

- 'min': Minimum of the pairwise similarities of the concepts in the two groups
- 'average': Average of the pairwise similarities of the concepts in the two groups
- 'max': Max of the pairwise similarities of the concepts in the two groups
- 'bma': Best match average
- 'bmm': Best match max

Direct groupwise measures directly compare the sets of concepts considering the features of both sets.

- 'ui': Considers the intersection and the union of the set of ancestors of the two groups of concepts:
  $sim(group_u, group_v) = |intersection(A(group_u), A(group_v))|/|union(A(group_u), A(group_v))|)$
- 'nto_max': Normalized max Term Overlap, computes the groupwise semantic similarity considering the inclusive set of ancestors of the two groups of concepts.
  $sim(group_u, group_v) = |intersection(A(group_u), A(group_v))|/|max(|A(group_u)|, |A(group_v)|)|$
- 'lee': Computes the groupwise semantic similarity considering the inclusive set of ancestors of the two groups of conceps.
  $sim(group_u, group_v) = |union(A(group_u), A(group_v))|$
- 'lp': Computes the groupwise semantic similarity between two groups of concepts as the depth of the longest shared path from the root node
- 'gic': Computes the groupwise semantic similarity between two groups of concepts as the ration between the information content of the concepts in the intersection of the ancestors in the two groups and the information content of the concepts in the union of the ancestors in the two groups.
  $sim(group_u, group_v) = IC_intersection/IC_union$

- 'batet': Computes the groupwise semantic similarity between two groups of concepts considering the union and intersection of ancestors normalized on the number of concepts in the ontology.
  $sim(group_u, group_v) = |(union(A(group_u), A(group_v)) - intersection(A(group_u), A(group_v)))|/(|(union($
  $tot_concepts)$

Value

instance of the class Similarity to compute semantic similarities based on the configured measures

the similarity value based on the configured measures

The java reference to an object of class Similarity

The object of class Similarity with a new instance of the java Similarity class

The measure used to compute concepts' information content

The similarity object with the new information conten measure set

The reference to the pairwise configuration used to compute semantic similarity

The pairwise measure

instance of the Similarity class with the new pairwise option.

groupwise configured measure for the similarity object provided as input

instance of the Similarity class with the new grouprwise option.

groupwise configured measure for the similarity object provided as input

instance of the Similarity class with the new grouprwise option.

the list of pairwise, information content and groupwise measures to compute the semantic similarities

The Similarity object where 'ontology' slot refers to the Java graph created

Ontology object

the semantic similarity of the two provided concepts

the semantic similarity of the two provided groups of concepts

The semantic similarity between the samples sample1 and sample2

The aggregate semantic similarity between the samples sample1 and sample2

Slots

`similarityInstance` The Java reference to the Java Similarity class.

`pairwiseConfig` The list of measures used to compute the semantic similarity between two concpets in the same ontology.

`pairwiseConfigRef` The reference to the Java object of type CMconf corresponding to the pairwise configuration

`groupConfig` The groupwise configuration to compute the semantic similarity between groups of concepts.

`icConfig` The information content measure

`groupwiseConfigRef` The reference to the Java configuration object for the computation of semantic similarity between groups of concepts

`ontology` The ontology to compute semantic similarities

Examples

```
sim <- new('Similarity')

obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')
myopts <- new('CMoptions')
paramValueIndex(myopts) <- 40
term_list1 <- c('http://purl.obolibrary.org/obo/CL_0000000', 'http://purl.obolibrary.org/obo/CL_0000236')
term_list2 <- c('http://purl.obolibrary.org/obo/CL_0000542')
sim <- similarity(obo, term_list1, term_list2)

obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')
myopts <- CMoptions()
paramValueIndex(myopts) <- 40
term_list1 <- c('http://purl.obolibrary.org/obo/CL_0000000',
 'http://purl.obolibrary.org/obo/CL_0000236')
term_list2 <- c('http://purl.obolibrary.org/obo/CL_0000542')
sim <- similarity(obo, term_list1, term_list2)

sim <- Similarity()
similarityInstance(sim)
sim <- Similarity()
similarityInstance(sim) <-.jnew('iit/comp/epigen/nlp/similarity/Similarity')
sim <- Similarity()
icConfig(sim)
sim <- Similarity()
icConfig(sim) <- 'sanchez'
sim <- Similarity()
pairwiseConfigRef(sim)
sim <- Similarity()
pairwiseConfigRef(sim) <- c('resnik')
sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
pairwiseConfig(sim)
sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
pairwiseConfig(sim) <- 'edge_resnik'
 #The following configuration uses an information content based measure
 pairwiseConfig(sim) <- c('resnik', 'seco')
sim <- Similarity()
groupwiseConfigRef(sim)
sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
groupwiseConfigRef(sim) <- 'ui'
sim <- Similarity()
groupConfig(sim)
sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
groupConfig(sim) <- 'ui'

sim <- Similarity()
showOpts(sim)
```

```
sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo

sim <- Similarity()
ontology(sim)

sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
pairwiseConfig(sim) <- showOpts(sim)$pairwiseMeasures[9]
similarity <- sim(sim, 'http://purl.obolibrary.org/obo/CL_0000542',
'http://purl.obolibrary.org/obo/CL_0000236')

sim <- Similarity()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
pairwiseConfig(sim) <- showOpts(sim)$pairwiseMeasures[9]
groupConfig(sim) <- showOpts(sim)$groupwiseMeasures[3]
similarity <- groupsim(sim, c('http://purl.obolibrary.org/obo/CL_0000542',
 'http://purl.obolibrary.org/obo/CL_0000236'),
c('http://purl.obolibrary.org/obo/CL_0000000'))
similarity

sim <- Similarity()

pairwiseConfig(sim) <- showOpts(sim)$pairwiseMeasures[9]
groupConfig(sim) <- showOpts(sim)$groupwiseMeasures[3]
ef <- EntityFinder()
opts <- CMoptions()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
sample_dict <- dictionary(inputFileOrDb=obo, outputdir=getwd(), synonymType='ALL')
sra_chip_seq <- readRDS(system.file('extdata', 'vignette_data', 'GEO_human_chip.rds',
  package='Onassis'))
chipseq_dict_annot <- annotate(sra_chip_seq[1:20,c('sample_accession', 'title',
 'experiment_attribute', 'sample_attribute', 'description')], dictionary=sample_dict,
  options=opts)
s <- samplesim(sim, as.character(as.vector(chipseq_dict_annot$sample_id[1])),
as.character(as.vector(chipseq_dict_annot$sample_id[7])) , chipseq_dict_annot)
ef <- EntityFinder()

opts <- CMoptions()

cell_dict_file <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- dictionary(inputFileOrDb=cell_dict_file, outputdir=getwd(),
synonymType='ALL')
samples <- findEntities(ef, system.file('extdata', 'test_samples',
'test_samples.txt',
package='Onassis'), outDir=getwd(), multipleDocs=TRUE, configOpt=opts,
 cmDict=sample_dict)

d_dict_file <-  system.file('extdata', 'sample.do.obo', package='OnassisJavaLibs')
disease_dict <- dictionary(inputFileOrDb=d_dict_file,
outputdir=getwd(), synonymType='ALL')
disease <- findEntities(ef, system.file('extdata', 'test_samples',
```

```
'test_samples.txt', package='Onassis'),
 multipleDocs=TRUE, outDir=getwd(), configOpt=opts,
  cmDict=disease_dict)


cell_sim <- Similarity()
ontology(cell_sim) <- cell_dict_file

disease_sim <- Similarity()
ontology(disease_sim) <- d_dict_file

pairwiseConfig(cell_sim) <- showOpts(cell_sim)$pairwiseMeasures[9]
pairwiseConfig(disease_sim) <- showOpts(disease_sim)$pairwiseMeasures[9]
groupConfig(cell_sim) <- showOpts(cell_sim)$groupwiseMeasures[3]
groupConfig(disease_sim) <- showOpts(disease_sim)$groupwiseMeasures[3]
similarity <- multisim(list(cell_sim, disease_sim),
list(samples, disease),
as.character(as.vector(samples[1,1])),
as.character(as.vector(samples[5,1])), 'mean')
```

Index