

# Package ‘Onassis’

March 30, 2021

**Version** 1.12.0

**Date** 2020-09-01

**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 (>= 4.0), rJava, OnassisJavaLibs

**Imports** GEOmetadb, RSQLite, data.table, methods, tools, utils, AnnotationDbi, RCurl, stats, DT, data.table, knitr, Rtsne, dendextend, clusteval, ggplot2, ggfortify

**SystemRequirements** Java (>= 1.8)

**RoxygenNote** 7.0.2

**VignetteBuilder** knitr

**Suggests** BiocStyle, rmarkdown, htmltools, org.Hs.eg.db, gplots, GenomicRanges, kableExtra

**Encoding** UTF-8

**LazyData** yes

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

**git\_url** <https://git.bioconductor.org/packages/Onassis>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** f8644e7

**git\_last\_commit\_date** 2020-10-27

**Date/Publication** 2021-03-29

**R topics documented:**

annotate	3
annotateDF	5
CaseMatch	6
CMdictionary	6
CMdictionary-class	8
CMoptions	9
CMoptions-class	10
collapse	11
compare	12
connectToGEODB	14
dictInfo	15
dictionary	16
dictRef	16
dictTypes	17
dict_location	18
entities	18
EntityFinder	19
EntityFinder-class	20
experiment_types	20
filterconcepts	21
filterTerms	22
FindAllMatches	22
findEntities	23
findHealthy	24
getGEOMetadata	24
groupConfig	25
groupsim	27
groupwiseConfigRef	27
icConfig	28
listCMoptions	29
listSimilarities	30
mergeonassis	30
multisim	31
OnASSiS	32
Onassis-class	33
ontology	34
OrderIndependentLookup	35
organism_types	35
pairsim	36
pairwiseConfig	37
pairwiseConfigRef	39
paramValueIndex	40
samplesim	41
scores	42
SearchStrategy	43
sim	44
simil	44
Similarity	45
Similarity-class	46
similarityInstance	47

Stemmer . . . . .	48
StopWords . . . . .	48
SynonymType . . . . .	49
typeSystemRef . . . . .	50

**Index** **52**

annotate                      annotate

**Description**

This method annotates the entities contained in a data frame with the concepts from a specific dictionary.

**Usage**

```

annotate(input = NA, dictType = NA, dictionary = NA, ...)

## S4 method for signature 'data.frame,character,character'
annotate(input,
  dictType = "OBO", dictionary = NA, dictoutdir = getwd(),
  d_synonymtype = "EXACT", taxID = 0, annot_out = getwd(),
  paramValueIndex = NA, SearchStrategy = "CONTIGUOUS_MATCH",
  CaseMatch = "CASE_INSENSITIVE", Stemmer = "NONE",
  StopWords = "NONE", OrderIndependentLookup = "ON",
  FindAllMatches = "YES", e_synonymtype = "ALL",
  multipleDocs = FALSE, disease = FALSE)

```

**Arguments**

- input                      A data frame where the first column is the ID of the sample or document to annotate
- 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
- dictionary                      The local OBO/OWL ontology to be converted into an XML Conceptmapper dictionary or the URL to download the file. If NA is passed and the dicType parameter is not the default 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 can be automatically downloaded from [ftp://ncbi.nlm.nih.gov/gene/data/gene\\_info.gz](ftp://ncbi.nlm.nih.gov/gene/data/gene_info.gz) if its path is not provided by the user in this parameter. Alternatively an annotation package of the type Org.xx.eg.db from Bioconductor can be used. In this case the gene identifiers and their alternative names will be retrieved from the annotation database without the need of downloading a gene\_info file.
- ...                              Optional parameters

dictoutdir	Optional parameter to specify the location where the Conceptmapper dictionary file will be stored. Defaults to current working directory.
d_synonymtype	Optional parameter to specify the type of synonyms to consider when building the dictionary for Conceptmapper. For further detail <a href="http://owollcollab.github.io/oboformat/doc/obo-syntax.html">http://owollcollab.github.io/oboformat/doc/obo-syntax.html</a> . Default: EXACT <b>EXACT</b> <b>ALL</b>
taxID	the taxonomy identifier of the organism when the dictType = 'ENTREZ' or 'TARGET' and the dictionary parameter refers to a gene_info.gz file. If 0 all the taxonomies will be included in the new dictionary.
annot_out	The path of the output directory where Conceptmapper annotation files will be stored
paramValueIndex	An integer value to index the 576 parameter combinations
SearchStrategy	The matching strategy for finding concepts in the input text <ul style="list-style-type: none"> <li>• CONTIGUOUS_MATCH Longest match of contiguous tokens within enclosing span</li> <li>• SKIP_ANY_MATCH Longest match of not-necessarily contiguous tokens</li> <li>• SKIP_ANY_MATCH_ALLOW_OVERLAP Longest match of not-necessarily contiguous tokens, overlapping matches are allowed</li> </ul>
CaseMatch	<ul style="list-style-type: none"> <li>• CASE_IGNORE Fold everything to lowercase for matching</li> <li>• CASE_INSENSITIVE Fold only tokens with initial caps to lowercase</li> <li>• CASE_FOLD_DIGITS Fold all (and only) tokens with a digit</li> <li>• CASE_SENSITIVE Perform no case folding</li> </ul>
Stemmer	<ul style="list-style-type: none"> <li>• BIOLEMMATIZER A stemmer specific for biomedical literature</li> <li>• PORTER A stemmer that removes the commoner morphological and inflexional endings from words in English</li> <li>• NONE No word stemming</li> </ul>
StopWords	<ul style="list-style-type: none"> <li>• PUBMED A list of stop words obtained analyzing Pubmed papers</li> <li>• NONE No stop words</li> </ul>
OrderIndependentLookup	<ul style="list-style-type: none"> <li>• ON Ordering within span is ignored (i.e. 'Breast cancer' would equal 'Cancer breast')</li> <li>• OFF Ordering is taken into consideration</li> </ul>
FindAllMatches	<ul style="list-style-type: none"> <li>• YES All the matches within the span are found</li> <li>• NO Only the longest match within the span will be returned</li> </ul>
e_synonymtype	The type of synonyms for the EntityFinder <ul style="list-style-type: none"> <li>• EXACT_ONLY Only exact synonyms are considered</li> <li>• ALL All synonym types are included</li> </ul>
multipleDocs	TRUE when multiple documents are loaded from a single file with each row representing a document. The file should have two columns. The first for the unique document identifier and the second for the textual descriptions
disease	A logical value set to TRUE if the annotation requires the 'Healthy' condition to be found.

**Value**

instance of class [Onassis-class](#) with annotated entities

### Examples

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

obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
onassis_results <- annotate(geo_chip, 'OBO', dictionary=obo)
entities <- entities(onassis_results)
entities <- entities[sample(nrow(entities), 30),]
```

---

annotateDF	annotateDF
------------	------------

---

### Description

Method to find concepts from a Conceptmapper Dictionary of type [CMdictionary](#) contained in a given data frame, with a specified configuration of type [CMoptions](#). This is a method of the [EntityFinder-class](#)

### Usage

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

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

### Arguments

object	Instance of class <a href="#">EntityFinder-class</a>
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 output directory
configOpt	instance of class <a href="#">CMoptions-class</a>
cmDict	Object of type <a href="#">CMdictionary-class</a> containing the reference to a previously created Conceptmapper dictionary. Alternatively the path to a Conceptmapper xml file can be passed.

### Value

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

### Examples

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

---

CaseMatch

CaseMatch

---

### Description

Method to get and set the CaseMatch parameter of [CMoptions](#) instances

### Usage

```
CaseMatch(x)

CaseMatch(x) <- value

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

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

### Arguments

x	instance of class <a href="#">CMoptions</a>
value	One of the following : <ul style="list-style-type: none"> <li>• <code>CASE_IGNORE</code> Fold everything to lowercase for matching</li> <li>• <code>CASE_INSENSITIVE</code> Fold only tokens with initial caps to lowercase</li> <li>• <code>CASE_FOLD_DIGITS</code> Fold all (and only) tokens with a digit</li> <li>• <code>CASE_SENSITIVE</code> Perform no case folding</li> </ul>

### Value

The CaseMatch corresponding to the current options when used as get, the new [CMoptions](#) object with updated parameters when used to set.

### Examples

```
opts <- CMoptions()
CaseMatch(opts)
opts <- CMoptions()
CaseMatch(opts) <- 'CASE_SENSITIVE'
```

---

CMdictionary

CMdictionary

---

### Description

Constructor method for creating instances of class [CMdictionary-class](#). The created Conceptmap-per dictionary will be stored as an XML file in the file system.

**Usage**

```
CMdictionary(inputFileOrDb = NULL, dictType = "OBO",
             outputDir = tempdir(), synonymType = "EXACT", taxID = 0,
             outputDirOp = TRUE)
```

**Arguments**

inputFileOrDb	The local OBO/OWL ontology to be converted into an XML Conceptmapper dictionary or the URL of a OBO/OWL file. If inputFileOrDb is NA and the dictType 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 will be automatically downloaded from <a href="ftp://ncbi.nlm.nih.gov/gene/data/gene_info.gz">ftp://ncbi.nlm.nih.gov/gene/data/gene_info.gz</a> if a valid path is not provided by the user. Alternatively the name of an annotation package of the type Org.xx.eg.db from Bioconductor can be used. In this case the gene unique identifiers 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 <b>OBO</b> A dictionary that has been created by An OBO file <b>ENTREZ</b> Entrez genes dictionary <b>TARGET</b> Entrez genes dictionary, Histone marks and Histone modifications <b>CMDICT</b> 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 <a href="http://owlcollab.github.io/oboformat/doc/obo-syntax.html">http://owlcollab.github.io/oboformat/doc/obo-syntax.html</a> . Default: EXACT <b>EXACT</b> <b>ALL</b>
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.
outputDirOp	set to TRUE to clean the directory before creating the dictionary

**Value**

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

**Examples**

```
## Not run:
#' ##This might take some time to download the dictionary
dict <- CMdictionary(dictType = 'TARGET', inputFileOrDb='org.Hs.eg.db')

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

## End(Not run)
```

---

CMdictionary-class      *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.

### Details

The following methods can be applied to CMdictionary instances

```
dict_location
dict_location<-
dictInfo
dictInfo<-
dictRef
dictRef<-
```

To show the available dictionary types use the function `dictTypes`

### 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://owllcollab.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')
```



COptions

COptions

**Description**

This constructor creates an object of type COptions

**Usage**

```
COptions(SearchStrategy = "CONTIGUOUS_MATCH",
         CaseMatch = "CASE_INSENSITIVE", Stemmer = "NONE",
         StopWords = "NONE", OrderIndependentLookup = "ON",
         FindAllMatches = "YES", SynonymType = "ALL", paramValueIndex = NA)
```

**Arguments**

SearchStrategy	The matching strategy for finding concepts in the input text <ul style="list-style-type: none"> <li>CONTIGUOUS_MATCH Longest match of contiguous tokens within enclosing span</li> <li>SKIP_ANY_MATCH Longest match of not-necessarily contiguous tokens</li> <li>SKIP_ANY_MATCH_ALLOW_OVERLAP Longest match of not-necessarily contiguous tokens, overlapping matches are allowed</li> </ul>
CaseMatch	<ul style="list-style-type: none"> <li>CASE_IGNORE Fold everything to lowercase for matching</li> <li>CASE_INSENSITIVE Fold only tokens with initial caps to lowercase</li> <li>CASE_FOLD_DIGITS Fold all (and only) tokens with a digit</li> <li>CASE_SENSITIVE Perform no case folding</li> </ul>
Stemmer	<ul style="list-style-type: none"> <li>BIOLEMMATIZER A stemmer specific for biomedical literature</li> <li>PORTER A stemmer that removes the commoner morphological and inflexional endings from words in English</li> <li>NONE No word stemming</li> </ul>
StopWords	<ul style="list-style-type: none"> <li>PUBMED A list of stop words obtained analyzing Pubmed papers</li> <li>NONE No stop words</li> </ul>
OrderIndependentLookup	<ul style="list-style-type: none"> <li>ON Ordering within span is ignored (i.e. 'Breast cancer' would equal 'Cancer breast')</li> <li>OFF Ordering is taken into consideration</li> </ul>
FindAllMatches	<ul style="list-style-type: none"> <li>YES All the matches within the span are found</li> <li>NO Only the longest match within the span will be returned</li> </ul>
SynonymType	<ul style="list-style-type: none"> <li>EXACT_ONLY Only exact synonyms are considered</li> <li>ALL All synonym types are included</li> </ul>
paramValueIndex	An integer value to index the 576 parameter combinations

**Value**

instance of the class COptions set to the default combination of parameters

**Examples**

```
op <- COptions()
```

---

COptions-class	<i>Class to set the options to run the EntityFinder</i>
----------------	---

---

### Description

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

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

### Usage

```
## S4 method for signature 'COptions'
show(object)
```

### Arguments

object            COptions instance

### Details

The following methods can be applied to COptions

```
show
paramValueIndex
paramValueIndex<-
SearchStrategy
SearchStrategy<-
CaseMatch
CaseMatch<-
Stemmer
Stemmer<-
StopWords
StopWords<-
OrderIndependentLookup
OrderIndependentLookup
FindAllMatches
FindAllMatches<-
SynonymType
SynonymType<-
```

### Value

the list of options

### Slots

paramValueIndex An integer value to index the 576 parameter combinations

SearchStrategy The matching strategy for finding concepts in the input text

- CONTIGUOUS\_MATCH Longest match of contiguous tokens within enclosing span
- SKIP\_ANY\_MATCH Longest match of not-necessarily contiguous tokens

- SKIP\_ANY\_MATCH\_ALLOW\_OVERLAP Longest match of not-necessarily contiguous tokens, overlapping matches are allowed
- CaseMatch
  - CASE\_IGNORE Fold everything to lowercase for matching
  - CASE\_INSENSITIVE Fold only tokens with initial caps to lowercase
  - CASE\_FOLD\_DIGITS Fold all (and only) tokens with a digit
  - CASE\_SENSITIVE Perform 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('COptions')
opt <- COptions()
show(opt)
```

---

collapse

collapse

---

### Description

This method collapses semantic states in an *Onassis* object.

### Usage

```
collapse(onassis = NA, simil_thresh)
```

```
## S4 method for signature 'Onassis'
collapse(onassis = NA, simil_thresh)
```

### Arguments

`onassis` instance of class [Onassis-class](#)  
`simil_thresh` the semantic similarity threshold to use to merge similar semantic sets

### Value

a new object of class [Onassis-class](#) with collapsed annotations for the entities and a new similarity matrix of similarities between newly created semantic sets

**Examples**

```

geo_chip <- readRDS(system.file('extdata', 'vignette_data',
'GEO_human_chip.rds', package='Onassis'))
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
onassis_results <- annotate(geo_chip, 'OBO', dictionary=obo)
entities <- entities(onassis_results)
entities(onassis_results) <- entities[sample(nrow(entities), 15),]
onassis_results <- sim(onassis_results)
collapsed_onassis <- collapse(onassis_results, 0.9)

```

---

compare

compare

---

**Description**

This method compares lists of scores (e.g. gene expression values, gene copy numbers, binding factors intensity) associated to the annotated entities (samples) of an Onassis object according to the semantic sets obtained from the annotation step. For Onassis objects annotated with a single ontology the method applies a test function to determine differences between subsets of the scores in one semantic set compared to subsets of scores in any other semantic set. For Onassis objects containing annotated entities with two ontologies, the first ensemble of semantic sets (e.g. cell lines) will be used as the main container to organize samples and the comparisons will be carried out between the entities belonging to different semantic sets of the second ontology (e.g. disease) within each semantic set defined from the first ontology.

**Usage**

```

compare(onassis, ...)

## S4 method for signature 'Onassis'
compare(onassis, score_matrix = NA, by = "row",
        fun_name = "wilcox.test", fun_args = list(), padj = FALSE)

```

**Arguments**

onassis	instance of class <a href="#">Onassis-class</a>
...	Optional parameters
score_matrix	a matrix of scores containing on the rows genomic units and on the columns the samples annotated in the entities
by	'row' if the test refers to single genomic units in multiple conditions, 'col' if the test compares all the genomic units across different conditions. Defaults to row
fun_name	name of the test to apply
fun_args	list of arguments needed by the function
padj	TRUE if multiple test correction is needed

## Details

The entities slot of an Onassis object can contain annotations with concepts from one ontology or two ontologies. The function `compare` separates these two scenarios.

**ONE ONTOLOGY CASE:** The entities (samples) are assigned to their corresponding semantic set (e.g. cell lines) A score matrix should have as rows genomic units (gene names, genomic regions, mutation identifiers...) and as columns the identifiers of all the entities annotated and belonging to semantic sets found in the `sample_id` field. The score for each entry of the matrix can be any biological measurement (gene expression RPKMs, peak intensity, copy numbers... ) Importantly, each semantic set can contain a different number of annotated entities (samples) and thus a single vector of scores (in case only 1 entity belongs to 1 semantic set) or a subset of the columns of the scores matrix If by is "row" then the function provided as `fun_name` will be applied to compare, genomic unit by genomic unit (rows of the score matrix), all the possible couples of semantic sets. The function can be any multiple statistical test function taking as parameters: - two numeric vectors of potentially different lengths (for genomic unit in row i, n samples in semantic set 1 and m samples in semantic set 2) - other optional arguments needed by the function passed as a list in `fun_args` The function should always provide as result a "statistic" and "p.value" In case the `padj` is set to TRUE the bonferroni correction will be applied for multiple tests If by is "col" then the function provided as `fun_name` will be applied to the couples of semantic sets considering all the possible values for the genomic units. In this case the test function should take as parameters - two matrices with potentially different number of columns - other optional arguments needed by the function can be passed as `fun_args` **TWO ONTOLOGIES CASE:** The comparisons in this case will be carried out considering for each semantic set defined from the primary ontology, all the possible couples of semantic sets generated from the second ontology (e.g. for each cell line, different diseases) within the first ontology semantic sets

## Value

The results of the comparison between semantic classes. In case of one ontology a square matrix whose rows and columns contain the semantic sets. For each couple of semantic sets *i* and *j* if by is 'row' the element in the row *i* and column *j* is a data frame with biological entities (genes, regions..) and for each entity, the results of the statistics function and p.value. For multiple test functions if `padj = TRUE` then also the Bonferroni correction will be reported. For each couple of semantic sets *i* and *j* if by is 'col' the element at row *i* and column *j* is a couple of values with the result of the statistic and corresponding p.value In case of two ontologies the result will be a list named with the first level semantic sets and each element of the list will be the result of the comparisons between second level semantic sets within first level semantic sets. Depending on the `by` parameter, the elements of the list will be matrices or vectors as for the one ontology case.

## Examples

```
#Loading ChIP-seq data
geo_chip <- readRDS(system.file('extdata', 'vignette_data', 'GEO_human_chip.rds', package='Onassis'))
#Sampling 30 samples
geo_chip <- geo_chip[sample(nrow(geo_chip), 30) ,]
# Loading the obo ontology for cell lines
obo1 <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
# Loading the obo ontology for diseases
obo2 <- system.file('extdata', 'sample.do.obo', package='OnassisJavaLibs')
#Annotating cell lines
onassis_results1 <- annotate(geo_chip, 'OBO', dictionary=obo1)
#Annotating diseases
onassis_results2 <- annotate(geo_chip, 'OBO', dictionary=obo2)
# Creating a score matrix
```

```

n <- length(unique(geo_chip$sample_accession))
m <- 50
score_matrix <- matrix(sample(0:1, m * n, replace = TRUE), m, n)
colnames(score_matrix) <- unique(geo_chip$sample_accession)
rownames(score_matrix) <- paste0('gene_', seq(1, m, 1))
# Merging the annotations from the two ontologies in a single object
my_onassis <- mergeonassis(onassis1 = onassis_results1, onassis2 = onassis_results2)
scores(my_onassis) <- score_matrix
# Comparing the scores associated to samples belonging to cell line semantic sets
# By default the wilcox.test will be applied to each of the 50 genes
scores(onassis_results1) <- score_matrix
gene_by_gene_cell_differences <- compare(onassis_results1)
head(gene_by_gene_cell_differences[1,2])
# Applying the same wilcox.test but obtaining a multiple test correction
gene_by_gene_cell_differences_ADJ <- compare(onassis_results1, padj=TRUE)
head(gene_by_gene_cell_differences_ADJ[1,2])
# Comparing disease genes across all the tissue semantic sets with wilcox.test and applying Bonferroni correction
gene_by_gene_disease_differences <- compare(my_onassis, by='row', padj=TRUE, fun_name='wilcox.test')
# Comparing diseases in cell line semantic sets
disease_differences <- compare(my_onassis, by='col', fun_name='wilcox.test')
# Using a personalized function
mykruskal <- function(x, y, params){
  kruskal.test(as.list(c(x, y)))}
cell_differences_personalized <- compare(onassis_results1, by='col', fun_name='mykruskal')

```

---

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(sqliteFilePath = NULL, download = FALSE,
  destdir = getwd())

```

## Arguments

sqliteFilePath optional SQLite full file path of the SQLite database if already downloaded  
download If TRUE allow the automatic downloading of the database file.  
destdir optional destination directory. Current working directory is the default

## 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')
}
```

---

dictInfo

*Get and Set the dicInfo slot for the [CMdictionary](#) class*

---

**Description**

Method to get and set a list of info about the dictionary

**Usage**

```
dictInfo(.Object)

dictInfo(.Object) <- value

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

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

**Arguments**

.Object	An instance of class <a href="#">CMdictionary</a>
value	list of details about the dictionary

**Value**

list of details about the dictionary  
object of class [CMdictionary](#)

**Examples**

```
dictionary <- new('CMdictionary')
dictInfo(dictionary)
dictionary <- new('CMdictionary')
dictInfo(dictionary) <-
list(Dictionary_type = 'ENTREZ from OrgDb', Dictionary_source = 'OrgDb')
```

---

dictionary	dictionary<-
------------	--------------

---

### Description

Method to get and set the dictionary slot of the class [Onassis-class](#)

### Usage

```
dictionary(object)
```

```
dictionary(object) <- value
```

```
## S4 replacement method for signature 'Onassis'
dictionary(object) <- value
```

```
## S4 method for signature 'Onassis'
dictionary(object)
```

### Arguments

object	instance of class <a href="#">Onassis-class</a>
value	the path of an OBO file

### Value

The path of the dictionary in case of get, the instance of Onassis with new dictionary in case of set

### Examples

```
onassis <- Onassis()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dictionary(onassis) <- obo
o <- Onassis()
dictionary(o)
```

---

dictRef	dictRef
---------	---------

---

### Description

This method retrieves and sets the java reference the conceptmapper dictionary



**Usage**

```
dictRef(.Object)

dictRef(.Object) <- value

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

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

**Arguments**

.Object	An instance of class <a href="#">CMdictionary</a>
value	the reference of a XML Conceptmapper dictionary file already created

**Value**

java reference to the Conceptmapper dictionary

**Examples**

```
dictionary <- new('CMdictionary')
dictRef(dictionary)
dictionary <- new('CMdictionary')
dict_file <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dictRef(dictionary) <- .jnew('java/io/File', dict_file)
```

---

dictTypes

dictTypes

---

**Description**

A function to show the pre-defined Conceptmapper dictionary types wwith their descriptions

**Usage**

```
dictTypes()
```

**Value**

list of types that can be used to create a Conceptmapper dictionary

**Examples**

```
dictTypes()
```

---

dict_location	dict_location
---------------	---------------

---

**Description**

A method to get and set the location of [CMdictionary](#)

**Usage**

```
dict_location(.Object)

dict_location(.Object) <- value

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

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

**Arguments**

.Object	An instance of class <a href="#">CMdictionary</a>
value	the path of the new dictionary

**Value**

the path of the dictionary in the file system

**Examples**

```
dictionary <- new('CMdictionary')
loc <- dict_location(dictionary)
dictionary <- new('CMdictionary')
dict_location(dictionary) <- getwd()
```

---

entities	entities<-
----------	------------

---

**Description**

Method to get and set the entities slot of the class [Onassis-class](#)

**Usage**

```
entities(object)

entities(object) <- value

## S4 replacement method for signature 'Onassis'
entities(object) <- value
```

```
## S4 method for signature 'Onassis'
entities(object)
```

### Arguments

object            instance of class [Onassis-class](#)  
value             a data frame with annotated entities

### Value

the entities of the Onassis object in case of get and the Onassis object with new entities in case of set

### Examples

```
onassis <- Onassis()
entities(onassis) <- data.frame()

o <- Onassis()
entities(o)
```

---

EntityFinder	EntityFinder
--------------	--------------

---

### Description

this function creates instances of the class [EntityFinder](#)

### Usage

```
EntityFinder(input, dictionary, options = NA, outDir = tempdir(),
multipleDocs = FALSE)
```

### Arguments

input            the file, directory, or data frame with the text to annotate  
dictionary       A dictionary of the type [CMDictionary](#) or the path to an already created Conceptmapper XML file  
options          an object of class [COptions](#). If NA, the default configuration will be set.  
outDir           the directory where the annotated files will be stored  
multipleDocs    TRUE when multiple documents are loaded from a single file with each row representing a document. The file should have two columns. The first for the unique document identifier and the second for the textual descriptions

### Value

dataframe of annotations

**Examples**

```

obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- CMdictionary(input=obo, outputDir=getwd(), synonymType='ALL')
myopts <- CMOptions()
paramValueIndex(myopts) <- 40
entities <- EntityFinder(input=readRDS(system.file('extdata', 'vignette_data', 'GEO_human_chip.rds', package=

```

---

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.

**Details**

The following methods can be applied to EntityFinder

[findEntities](#)  
[annotateDF](#)

The methods can be automatically called using the function [EntityFinder](#)

**Slots**

typeSystemRef The reference to the Java object representing the type system

**Examples**

```
finder <- new('EntityFinder')
```

---

experiment\_types      experiment\_types

---

**Description**

This method retrieves the experiment types stored in GEOmetadb

**Usage**

```
experiment_types(GEOcon)
```

**Arguments**

GEOcon                  connection to the SQLite GEOmetadb database

**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')
}
```

---

filterconcepts	filterconcepts
----------------	----------------

---

## Description

This method filters unwanted concepts from the entities of an [Onassis-class](#) object

## Usage

```
filterconcepts(onassis, concepts_to_filter)

## S4 method for signature 'Onassis'
filterconcepts(onassis, concepts_to_filter = c())
```

## Arguments

`onassis` An object of class [Onassis-class](#) with already annotated entities  
`concepts_to_filter` A vector with unwanted concepts

## Value

the instance of [Onassis-class](#) with filtered entities

## Examples

```
geo_chip <- readRDS(system.file('extdata', 'vignette_data',
  'GEO_human_chip.rds', package='Onassis'))
geo_chip <- geo_chip[sample(nrow(geo_chip), 15) ,]
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
onassis_results <- annotate(geo_chip, 'OBO', dictionary=obo)
filtered_onassis <- filterconcepts(onassis_results, c('cell'))
```

---

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)
```

---

FindAllMatches	<i>Method FindAllMatches</i>
----------------	------------------------------

---

**Description**

Method to get and set the FindAllMatches parameter

**Usage**

```
FindAllMatches(x)
```

```
FindAllMatches(x) <- value
```

```
## S4 method for signature 'COptions'
FindAllMatches(x)
```

```
## S4 replacement method for signature 'COptions'
FindAllMatches(x) <- value
```

**Arguments**

x	instance of class <a href="#">COptions</a>
value	<ul style="list-style-type: none"> <li>• YES All the matches within the span are found</li> <li>• NO Only the longest match within the span will be returned</li> </ul>

**Value**

The FindAllMatches corresponding to the current options when used as get, the new COptions object with updated parameters when used to set.

**Examples**

```
opts <- COptions()
FindAllMatches(opts)
opts <- COptions()
FindAllMatches(opts) <- 'YES'
```

---

findEntities	<i>Method findEntities</i>
--------------	----------------------------

---

**Description**

This method finds concepts from 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 [COptions](#). This is a method of the [EntityFinder-class](#)

**Usage**

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

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

**Arguments**

object	instance of the class EntityFinder
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
outDir	The directory where the Conceptmapper annotated files are stored. Default: the system tmp directory.
configOpt	Object of type COptions in which the parameters to run Conceptmapper are stored
cmDict	Instance of class <a href="#">CMdictionary</a> or the file path of an already created CMdictionary

**Value**

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

**Examples**

```

obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
dict <- CMdictionary(inputFileOrDb=obo, outputDir=getwd(), synonymType='ALL')

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

```

---

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 retrieves 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 <code>experiment_types</code>
organism	Optional type of organism. Allowed species can be obtained using the function <code>organism_types</code> . 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')
  methylation <- 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')
}
```

---

groupConfig	groupConfig
-------------	-------------

---

**Description**

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 `listSimilarities()`.

**Usage**

```
groupConfig(object)

groupConfig(object) <- value

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

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

**Arguments**

object	instance of class <a href="#">Similarity-class</a>
value	See details

## Details

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:

$$\text{sim}(\text{group}_u, \text{group}_v) = |\text{intersection}(A(\text{group}_u), A(\text{group}_v))| / |\text{union}(A(\text{group}_u), A(\text{group}_v))|$$

- 'nto\_max': Normalized max Term Overlap, computes the groupwise semantic similarity considering the inclusive set of ancestors of the two groups of concepts.

$$\text{sim}(\text{group}_u, \text{group}_v) = |\text{intersection}(A(\text{group}_u), A(\text{group}_v))| / \max(|A(\text{group}_u)|, |A(\text{group}_v)|)$$

- 'lee': Computes the groupwise semantic similarity considering the inclusive set of ancestors of the two groups of concepts.

$$\text{sim}(\text{group}_u, \text{group}_v) = |\text{union}(A(\text{group}_u), A(\text{group}_v))|$$

- '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.

$$\text{sim}(\text{group}_u, \text{group}_v) = IC_{\text{intersection}} / IC_{\text{union}}$$

- '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.

$$\text{sim}(\text{group}_u, \text{group}_v) = (|\text{union}(A(\text{group}_u), A(\text{group}_v))| - |\text{intersection}(A(\text{group}_u), A(\text{group}_v))|) / (|\text{union}(A(\text{group}_u), A(\text{group}_v))|)$$

## Value

groupwise configured measure for the similarity object provided as input instance of the Similarity class with the new groupwise option.

## Examples

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

---

groupsim	<i>Method groupsim</i>
----------	------------------------

---

### Description

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

### Usage

```
groupsim(object, termList1, termList2)

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

### Arguments

object	instance of class <a href="#">Similarity-class</a>
termList1	A vector of URIs of ontology terms in the format <a href="http://purl.obolibrary.org/obo/Ontology_id">http://purl.obolibrary.org/obo/Ontology_id</a> (e.g <a href="http://purl.obolibrary.org/obo/BTO_0004732">http://purl.obolibrary.org/obo/BTO_0004732</a> )
termList2	A vector of URIs of ontology terms

### Value

the semantic similarity of the two provided groups of concepts

### Examples

```
sim <- new('Similarity')
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
pairwiseConfig(sim) <- listSimilarities()$pairwiseMeasures[9]
groupConfig(sim) <- listSimilarities()$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
```

---

groupwiseConfigRef	groupwiseConfigRef
--------------------	--------------------

---

### Description

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 measures see the groupConfig function's details.

**Usage**

```

groupwiseConfigRef(object)

groupwiseConfigRef(object) <- value

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

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

```

**Arguments**

object            instance of class [Similarity-class](#)  
value             one of the groupwise measures. See [groupConfig](#)

**Value**

groupwise configured measure for the similarity object provided as input  
instance of the Similarity class with the new groupwise option.

**Examples**

```

sim <- new('Similarity')
groupwiseConfigRef(sim)
sim <- new('Similarity')
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
groupwiseConfigRef(sim) <- 'ui'

```

---

icConfig

icConfig

---

**Description**

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

**Usage**

```

icConfig(object)

icConfig(object) <- value

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

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

```

**Arguments**

object	instance of class <a href="#">Similarity-class</a>
value	the information content measure selected.

**Value**

The measure used to compute concepts' information content  
 The similarity object with the new information conten measure set

**Examples**

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

---

listCMOptions	listCMOptions
---------------	---------------

---

**Description**

This method retrieves all the possible parameters combinations for Conceptmapper.

**paramValueIndex** An integer value to index the 576 parameter combinations

**SearchStrategy** The matching strategy for finding concepts in the input text

- CONTIGUOUS\_MATCH Longests match of contiguous tokens within enclosing span
- SKIP\_ANY\_MATCH Longest match of not-necessarily contiguous tokens
- SKIP\_ANY\_MATCH\_ALLOW\_OVERLAP Longest match of not-necessarily contiguous tokens, overlapping matches are allowed

**CaseMatch** • CASE\_IGNORE Fold everything to lowercase for matching

- CASE\_INSENSITIVE Fold only tokens with initial caps to lowercase
- CASE\_FOLD\_DIGITS Fold all (and only) tokens with a digit
- CASE\_SENSITIVE Perform 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

**Usage**

```
listCMOptions()
```

**Value**

The data frame with all the possible parameter combinations

**Examples**

```
o <- listCMOptions()
```

---

listSimilarities	listSimilarities
------------------	------------------

---

**Description**

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

**Usage**

```
listSimilarities()
```

**Value**

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

**Examples**

```
s <- listSimilarities()
```

---

mergeonassis	mergeonassis
--------------	--------------

---

**Description**

This method unifies the entities of two Onassis objects

**Usage**

```
mergeonassis(onassis1 = NA, onassis2 = NA)
```

```
## S4 method for signature 'Onassis,Onassis'
mergeonassis(onassis1 = NA, onassis2 = NA)
```

**Arguments**

onassis1	instance of class <a href="#">Onassis-class</a>
onassis2	instance of class <a href="#">Onassis-class</a>

**Value**

new object of type `Onassis-class` with merged entities

**Examples**

```
geo_chip <- readRDS(system.file('extdata', 'vignette_data',
  'GEO_human_chip.rds', package='Onassis'))
geo_chip <- geo_chip[sample(nrow(geo_chip), 15) ,]
obo1 <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
obo2 <- system.file('extdata', 'sample.do.obo', package='OnassisJavaLibs')
onassis_results1 <- annotate(geo_chip, 'OBO', dictionary=obo1)
onassis_results2 <- annotate(geo_chip, 'OBO', dictionary=obo2)
onassis_results <- mergeonassis(onassis_results1, onassis_results2)
```

---

multisim

multisim

---

**Description**

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

**Usage**

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

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

**Arguments**

similarities	a list of Similarity instances, one for each ontology used to annotate the data
annotations	a list of annotated data frames obtained using <code>annotateDF</code> or <code>findEntities</code> , one for each ontology
sample1	the name of a sample in annotations
sample2	the name of a sample in annotations
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'

**Value**

The aggregate semantic similarity between the samples `sample1` and `sample2`

**Examples**

```

ef <- new('EntityFinder')

opts <- COptions()
cell_dict_file <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- CMdictionary(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 <- CMdictionary(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 <- new('Similarity')
ontology(cell_sim) <- cell_dict_file

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

pairwiseConfig(cell_sim) <- listSimilarities()$pairwiseMeasures[9]
pairwiseConfig(disease_sim) <- listSimilarities()$pairwiseMeasures[9]
groupConfig(cell_sim) <- listSimilarities()$groupwiseMeasures[3]
groupConfig(disease_sim) <- listSimilarities()$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')

```

---

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.

This constructor instantiates an Onassis object.

**Usage**

```

Onassis(dictionary = NA_character_, entities = data.frame(),
similarity = matrix(), scores = matrix())

```

**Arguments**

dictionary	The path of the dictionary file
entities	a data frame to store entities



similarity	A matrix of the similarities between entities
scores	The result of comparisons of the elements in the entities

### 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/>)

### Value

An object of type Onassis that can be used to analyze metadata

### Examples

```
onassis <- Onassis()
```

---

Onassis-class

*Onassis-class*

---

### Description

Onassis is a container class for annotating samples metadata with concepts from dictionaries/ontologies, creating semantic sets of unique annotations, computing the distances between different semantic sets and eventually comparing the different identified conditions.

### Details

The following methods can be applied to Onassis

[annotate](#)  
[collapse](#)  
[compare](#)  
[dictionary](#)  
[simil](#)  
[entities](#)  
[scores](#)  
[sim](#)

**Slots**

dictionary One or more input dictionaries to annotate samples metadata  
 entities a data frame containing the result of the annotation of the input with ontology terms  
 similarity A matrix of the similarities between the entries in the entities slot  
 scores An optional score matrix containing genomic units on the rows (genes, regions) and on the columns the elements on the rows of the entities slot

---

ontology	ontology<-
----------	------------

---

**Description**

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.

**Usage**

```
ontology(object)

ontology(object) <- value

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

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

**Arguments**

object	instance of class <a href="#">Similarity-class</a>
value	The path of an ontology file

**Value**

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

**Examples**

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

---

```
OrderIndependentLookup
      OrderIndependentLookup
```

---

**Description**

Method to get and set the OrderIndependentLookup parameter

**Usage**

```
OrderIndependentLookup(x)

OrderIndependentLookup(x) <- value

## S4 method for signature 'COptions'
OrderIndependentLookup(x)

## S4 replacement method for signature 'COptions'
OrderIndependentLookup(x) <- value
```

**Arguments**

x	instance of class <a href="#">COptions</a>
value	<ul style="list-style-type: none"> <li>• ON Ordering within span is ignored (i.e. 'Breast cancer' would equal 'Cancer breast')</li> <li>• OFF Ordering is taken into consideration</li> </ul>

**Value**

The OrderIndependentLookup corresponding to the current options when used as get, the new COptions object with updated parameters when used to set.

**Examples**

```
opts <- COptions()
OrderIndependentLookup(opts)

opts <- COptions()
OrderIndependentLookup(opts) <- 'ON'
```

---

```
organism_types      organism_types
```

---

**Description**

This method retrieves the allowed organisms in GEOmetadb

**Usage**

```
organism_types(geo_con)
```

**Arguments**

geo\_con                    connection to the SQLite GEOmetadb database

**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')
}
```

---

pairsim

pairsim

---

**Description**

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

**Usage**

```
pairsim(object, term1, term2)
```

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

**Arguments**

object                    instance of class [Similarity-class](#)

term1                    The URI of the ontology term in the format [http://purl.obolibrary.org/obo/Ontology\\_id](http://purl.obolibrary.org/obo/Ontology_id) (e.g 'http://purl.obolibrary.org/obo/CL\_0000542')

term2                    The URI of the ontology term

**Value**

the semantic similarity of the two provided concepts

**Examples**

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

---

```
pairwiseConfig      pairwiseConfig
```

---

### Description

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.

### Usage

```
pairwiseConfig(object)

pairwiseConfig(object) <- value

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

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

### Arguments

object	instance of class <a href="#">Similarity-class</a>
value	See details

### Details

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 length 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 ancestor 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 measure 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 content of the most informative common ancestor of the compared concepts and its probability of occurrence  

$$sim(u, v) = (2 * IC(MICA)) / (IC(u) + IC(v)) * (1 - Prob_{MICA})$$
 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 concepts  

$$sim(u, v) = 1 - (IC(u) + IC(v) - 2 * IC(MICA)) / 2$$

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.  

$$IC(c) = k(1 - \log(D(c)) / \log(|C|)) + (1 - k)(\log(\max(\text{depth}(x))) / \log(\text{depth}_{max}))$$
 k is a factor to adjust the weight of the two items of the equation  
 D(c) is the number of hyponyms 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 hyponyms 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.

### Value

The pairwise measure

instance of the Similarity class with the new pairwise option.

### Examples

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

---

pairwiseConfigRef      pairwiseConfigRef

---

### Description

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.

### Usage

```
pairwiseConfigRef(object)

pairwiseConfigRef(object) <- value

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

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

### Arguments

object            instance of class [Similarity-class](#)  
value             See details

### Value

The reference to the pairwise configuration used to compute semantic similarity

### Examples

```
sim <- new('Similarity')
pairwiseConfigRef(sim)
sim <- new('Similarity')
pairwiseConfigRef(sim) <- c('lin')
```

---

paramValueIndex	<i>Method paramValueIndex</i>
-----------------	-------------------------------

---

### Description

Method to get and set the parameter combination index corresponding to a given parameter combination. The value of the paramValueIndex lays in the range [0:575]

### Usage

```
paramValueIndex(x)

paramValueIndex(x) <- value

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

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

### Arguments

x                instance of class [COptions](#)  
value            Index corresponding to a given parameter combination

### Value

The paramValueIndex corresponding to the current options when used as get, the new COptions object with updated parameters when used to set.



**Examples**

```

opts <- COptions()
paramValueIndex(opts)

opts <- COptions()
paramValueIndex(opts) <- 2

```

---

samplesim	samplesim
-----------	-----------

---

**Description**

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

**Usage**

```

samplesim(object, sample1, sample2, annotated_df)

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

```

**Arguments**

object	instance of class <a href="#">Similarity-class</a>
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.

**Value**

The semantic similarity between the samples sample1 and sample2

**Examples**

```

sim <- new('Similarity')

pairwiseConfig(sim) <- listSimilarities()$pairwiseMeasures[9]
groupConfig(sim) <- listSimilarities()$groupwiseMeasures[3]
ef <- new('EntityFinder')
opts <- COptions()
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
ontology(sim) <- obo
sample_dict <- CMdictionary(inputFileOrDb=obo, outputDir=getwd(), synonymType='ALL')
sra_chip_seq <- readRDS(system.file('extdata', 'vignette_data', 'GEO_human_chip.rds',
  package='Onassis'))
chipseq_dict_annot <- EntityFinder(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)
```

---

scores

scores<-

---

### Description

This method gets and sets the scores slot of a class [Onassis-class](#)

### Usage

```
scores(object)
```

```
scores(object) <- value
```

```
## S4 replacement method for signature 'Onassis'
scores(object) <- value
```

```
## S4 method for signature 'Onassis'
scores(object)
```

### Arguments

object            instance of class [Onassis-class](#)

value            a matrix of scores

### Value

the matrix of scores in case of get and the new [Onassis-class](#) object in case of set

### Examples

```
onassis <- Onassis()
scores(onassis) <- matrix()
```

```
o <- Onassis()
scores(o)
```

---

SearchStrategy	<i>Method SearchStrategy</i>
----------------	------------------------------

---

## Description

Method to get and set the SearchStrategy parameter

## Usage

```
SearchStrategy(x)

SearchStrategy(x) <- value

## S4 method for signature 'COptions'
SearchStrategy(x)

## S4 replacement method for signature 'COptions'
SearchStrategy(x) <- value
```

## Arguments

x	instance of class <a href="#">COptions</a>
value	The matching strategy for finding concepts in the input text <ul style="list-style-type: none"><li>• <b>CONTIGUOUS_MATCH</b> Longest match of contiguous tokens within enclosing span</li><li>• <b>SKIP_ANY_MATCH</b> Longest match of not-necessarily contiguous tokens</li><li>• <b>SKIP_ANY_MATCH_ALLOW_OVERLAP</b> Longest match of not-necessarily contiguous tokens, overlapping matches are allowed</li></ul>

## Value

The SearchStrategy corresponding to the current options when used as get, the new COptions object with updated parameters when used to set.

## Examples

```
opts <- COptions()
SearchStrategy(opts)

opts <- COptions()
SearchStrategy(opts) <- 'SKIP_ANY_MATCH_ALLOW_OVERLAP'
```

---

sim	sim
-----	-----

---

### Description

This method computes the similarities of the entities annotated in a object fo class [Onassis-class](#).

### Usage

```
sim(onassis = NA, ...)

## S4 method for signature 'Onassis'
sim(onassis, iconf = "sanchez", pairconf = "lin",
    groupconf = "bma")
```

### Arguments

onassis	instance of class <a href="#">Onassis-class</a>
...	Optional parameters
iconf	the information content measure see <a href="#">pairwiseConfig</a> help for details
pairconf	the pairwise measure to compute semantic similarity between single concepts. See <a href="#">pariwiseConfig</a> help for details
groupconf	the groupwise measure to compute semantic similarity between groups of concepts. See <a href="#">groupConfig</a> help for details

### Value

an instance of [Onassis-class](#) with computed similarities

### Examples

```
geo_chip <- readRDS(system.file('extdata', 'vignette_data', 'GEO_human_chip.rds', package='Onassis'))
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
onassis_results <- annotate(geo_chip, 'OBO', dictionary=obo)
entities <- entities(onassis_results)
entities(onassis_results) <- entities[sample(nrow(entities), 30),]
onassis_results <- sim(onassis_results)
```

---

simil	simil<-
-------	---------

---

### Description

Method to get and set the similarity slot of the class [Onassis-class](#)

**Usage**

```

simil(object)

simil(object) <- value

## S4 replacement method for signature 'Onassis'
simil(object) <- value

## S4 method for signature 'Onassis'
simil(object)

```

**Arguments**

object            instance of class [Onassis-class](#)  
value             a matrix of similarities between the entities of an object of class [Onassis-class](#)

**Value**

the similarity matrix of an object of type [Onassis-class](#) in case of get, the new [Onassis-class](#) instance in case of set

**Examples**

```

onassis <- Onassis()
simil(onassis) <- matrix()

o <- Onassis()
simil(o)

```

---

Similarity

Similarity

---

**Description**

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

**Usage**

```

Similarity(ontology, termlist1, termlist2, annotatedtab = NA,
           icConf = "seco", pairConf = "lin", groupConf = "ui")

```

**Arguments**

ontology            The ontology file to create the DAG to compute similarities  
termlist1            The single concept or vector of concepts in the first set or the name of a sample if annotatedtab contains an annotated table  
termlist2            The single concept or vector of concepts in the second set or the name of a sample if annotatedtab contains an annotated table  
annotatedtab        The table of annotation of samples or entities

icConf            the information content configuration  
 pairConf        the pairwise configuration  
 groupConf      the groupwise configuration

### Value

Measure of the similarity bewtween the concepts passed as input

### Examples

```
obo <- system.file('extdata', 'sample.cs.obo', package='OnassisJavaLibs')
sample_dict <- CMdictionary(input=obo, outputDir=getwd(), synonymType='ALL')
myopts <- CMOptions(paramValueIndex=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)
```

---

Similarity-class	<i>Similarity class to compute similarities between concepts in ontologies and samples annotated with different concepts</i>
------------------	--

---

### Description

Similarity is a class that wraps some methods of the Java library <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](http://www.semantic-measures-library.org/sml/index.php?q=doc_graph_based_advanced)

### Details

The following methods can be applied to Similarity

```
ontology<-
pairwiseConfig
groupConfig
sim
groupsim
samplesim
```

To see the available similarity measures run the function `listSimilarities`

### Slots

similarityInstance The Java reference to the Java Similarity class.

pairwiseConfig The list of measures used to compute the semantic similarity between two concepts 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')
```

---

```
similarityInstance    similarityInstance
```

---

### Description

This method retrieves the java object referencing the Similarity class

### Usage

```
similarityInstance(object)

similarityInstance(object) <- value

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

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

### Arguments

object	Instance of the class <a href="#">Similarity-class</a>
value	the reference to a java Similarity object

### Value

The java reference to an object of class Similarity  
 The object of class Similarity with a new instance of the java Similarity class

### Examples

```
sim <- new('Similarity')
similarityInstance(sim)
sim <- new('Similarity')
similarityInstance(sim) <- .jnew('iit/comp/epigen/nlp/similarity/Similarity')
```

---

Stemmer	<i>Method Stemmer</i>
---------	-----------------------

---

### Description

Method to get and set the Stemmer parameter

### Usage

```
Stemmer(x)

Stemmer(x) <- value

## S4 method for signature 'COptions'
Stemmer(x)

## S4 replacement method for signature 'COptions'
Stemmer(x) <- value
```

### Arguments

x	instance of class <a href="#">COptions</a>
value	<ul style="list-style-type: none"> <li>• BIOLEMMATIZER A stemmer specific for biomedical literature</li> <li>• PORTER A stemmer that removes the commoner morphological and inflexional endings from words in English</li> <li>• NONE No word stemming</li> </ul>

### Value

The Stemmer corresponding to the current options when used as get, the new COptions object with updated parameters when used to set.

### Examples

```
opts <- COptions()
Stemmer(opts)

opts <- COptions()
Stemmer(opts) <- 'PORTER'
```

---

StopWords	<i>Method StopWords</i>
-----------	-------------------------

---

### Description

Method to get and set the StopWords parameter



**Usage**

```

StopWords(x)

StopWords(x) <- value

## S4 method for signature 'COptions'
StopWords(x)

## S4 replacement method for signature 'COptions'
StopWords(x) <- value

```

**Arguments**

x                    instance of class `COptions`

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

**Value**

The StopWords corresponding to the current options when used as get, the new COptions object with updated parameters when used to set.

**Examples**

```

opts <- COptions()
StopWords(opts)

opts <- COptions()
StopWords(opts) <- 'NONE'

```

---

SynonymType

*Method SynonymType*


---

**Description**

Method to get and set the SynonymType parameter

**Usage**

```

SynonymType(x)

SynonymType(x) <- value

## S4 method for signature 'COptions'
SynonymType(x)

## S4 replacement method for signature 'COptions'
SynonymType(x) <- value

```

**Arguments**

- x                    instance of class `COptions`
- value                • EXACT\_ONLY Only exact synonyms are considered  
                       • ALL All synonym types are included

**Value**

The `SynonymType` corresponding to the current options when used as get, the new `COptions` object with updated parameters when used to set.

**Examples**

```
opts <- COptions()
SynonymType(opts)

opts <- COptions()
SynonymType(opts) <- 'ALL'
```

---

typeSystemRef	<i>Method typeSystemRef</i>
---------------	-----------------------------

---

**Description**

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

**Usage**

```
typeSystemRef(x)

typeSystemRef(x) <- value

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

## S4 replacement method for signature 'EntityFinder'
typeSystemRef(x) <- value
```

**Arguments**

- x                    instance of the class `EntityFinder`
- value                the java type system to detect concepts from ontologies.

**Value**

the reference to the Java type system currently set  
 The updated EntityFinder S4 object

**Examples**

```
ef <- new('EntityFinder')
typeSystemRef(ef)
ef <- new('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
```

# Index

annotate, [3](#), [33](#)  
annotate, data.frame, character, character-method [16](#)  
    (annotate), [3](#)  
annotateDF, [5](#), [20](#)  
annotateDF, EntityFinder, data.frame, character, COptions-method (annotateDF), [5](#)

CaseMatch, [6](#), [10](#)  
CaseMatch, COptions-method (CaseMatch), [6](#)  
CaseMatch<- (CaseMatch), [6](#)  
CaseMatch<-, COptions-method (CaseMatch), [6](#)  
CMdictionary, [5](#), [6](#), [15](#), [17–19](#), [23](#)  
CMdictionary-class, [8](#)  
COptions, [5](#), [6](#), [9](#), [19](#), [22](#), [23](#), [35](#), [40](#), [43](#), [48–50](#)  
COptions-class, [10](#)  
collapse, [11](#), [33](#)  
collapse, Onassis-method (collapse), [11](#)  
compare, [12](#), [33](#)  
compare, Onassis-method (compare), [12](#)  
connectToGEODB, [14](#)  
connectToGEODB, (connectToGEODB), [14](#)

dict\_location, [8](#), [18](#)  
dict\_location, CMdictionary-method (dict\_location), [18](#)  
dict\_location<- (dict\_location), [18](#)  
dict\_location<-, CMdictionary-method (dict\_location), [18](#)  
dictInfo, [8](#), [15](#)  
dictInfo, CMdictionary-method (dictInfo), [15](#)  
dictInfo<- (dictInfo), [15](#)  
dictInfo<-, CMdictionary-method (dictInfo), [15](#)  
dictionary, [16](#), [33](#)  
dictionary, Onassis-method (dictionary), [16](#)  
dictionary<- (dictionary), [16](#)  
dictionary<-, Onassis-method (dictionary), [16](#)  
dictRef, [8](#), [16](#)  
dictRef, CMdictionary-method (dictRef), [16](#)  
dictRef<- (dictRef), [16](#)  
dictRef<-, CMdictionary-method (dictRef), [16](#)  
dictTypes, [8](#), [17](#)

entities, [18](#), [33](#)  
entities, Onassis-method (entities), [18](#)  
entities<- (entities), [18](#)  
entities<-, Onassis-method (entities), [18](#)  
EntityFinder, [7](#), [19](#), [19](#), [20](#)  
EntityFinder-class, [20](#)  
experiment\_types, [20](#)

filterconcepts, [21](#)  
filterconcepts, Onassis-method (filterconcepts), [21](#)  
filterTerms, [22](#)  
FindAllMatches, [10](#), [22](#)  
FindAllMatches, COptions-method (FindAllMatches), [22](#)  
FindAllMatches<- (FindAllMatches), [22](#)  
FindAllMatches<-, COptions-method (FindAllMatches), [22](#)  
findEntities, [20](#), [23](#)  
findEntities, EntityFinder, character, logical, character, COptions-method (findEntities), [23](#)  
findHealthy, [24](#)

GEOHandler-function (connectToGEODB), [14](#)  
getGEOmetadata, [24](#)  
groupConfig, [25](#), [28](#), [46](#)  
groupConfig, Similarity-method (groupConfig), [25](#)  
groupConfig<- (groupConfig), [25](#)  
groupConfig<-, Similarity-method (groupConfig), [25](#)  
groupsim, [27](#), [46](#)  
groupsim, Similarity, character, character-method (groupsim), [27](#)  
groupwiseConfigRef, [27](#)  
groupwiseConfigRef, Similarity-method (groupwiseConfigRef), [27](#)

- groupwiseConfigRef<-  
    (groupwiseConfigRef), 27
- groupwiseConfigRef<- , Similarity-method  
    (groupwiseConfigRef), 27
- icConfig, 28
- icConfig, Similarity-method (icConfig),  
    28
- icConfig<- (icConfig), 28
- icConfig<- , Similarity-method  
    (icConfig), 28
- listCMOptions, 29
- listSimilarities, 30, 46
- mergeonassis, 30
- mergeonassis, Onassis, Onassis-method  
    (mergeonassis), 30
- multisim, 31
- multisim, list, list, character, character-method  
    (multisim), 31
- OnASSiS, 32
- Onassis (OnASSiS), 32
- Onassis-class, 33
- OnASSiS-package (OnASSiS), 32
- Onassis-package (OnASSiS), 32
- ontology, 34
- ontology, Similarity-method (ontology),  
    34
- ontology<- (ontology), 34
- ontology<- , Similarity-method  
    (ontology), 34
- OrderIndependentLookup, 10, 35
- OrderIndependentLookup, CMOptions-method  
    (OrderIndependentLookup), 35
- OrderIndependentLookup<-  
    (OrderIndependentLookup), 35
- OrderIndependentLookup<- , CMOptions-method  
    (OrderIndependentLookup), 35
- organism\_types, 35
- pairsim, 36
- pairsim, Similarity, character, character-method  
    (pairsim), 36
- pairwiseConfig, 28, 37, 46
- pairwiseConfig, Similarity-method  
    (pairwiseConfig), 37
- pairwiseConfig<- (pairwiseConfig), 37
- pairwiseConfig<- , Similarity-method  
    (pairwiseConfig), 37
- pairwiseConfigRef, 39
- pairwiseConfigRef, Similarity-method  
    (pairwiseConfigRef), 39
- pairwiseConfigRef<-  
    (pairwiseConfigRef), 39
- pairwiseConfigRef<- , Similarity-method  
    (pairwiseConfigRef), 39
- paramValueIndex, 10, 40
- paramValueIndex, CMOptions-method  
    (paramValueIndex), 40
- paramValueIndex<- (paramValueIndex), 40
- paramValueIndex<- , CMOptions-method  
    (paramValueIndex), 40
- samplesim, 41, 46
- samplesim, Similarity, character, character, data.frame-method  
    (samplesim), 41
- scores, 33, 42
- scores, Onassis-method (scores), 42
- scores<- (scores), 42
- scores<- , Onassis-method (scores), 42
- SearchStrategy, 10, 43
- SearchStrategy, CMOptions-method  
    (SearchStrategy), 43
- SearchStrategy<- (SearchStrategy), 43
- SearchStrategy<- , CMOptions-method  
    (SearchStrategy), 43
- show, 10
- show, CMOptions-method  
    (CMOptions-class), 10
- sim, 33, 44, 46
- sim, Onassis-method (sim), 44
- simil, 33, 44
- simil, Onassis-method (simil), 44
- simil<- (simil), 44
- simil<- , Onassis-method (simil), 44
- Similarity, 45
- Similarity-class, 46
- similarityInstance, 47
- similarityInstance, Similarity-method  
    (similarityInstance), 47
- similarityInstance<-  
    (similarityInstance), 47
- similarityInstance<- , Similarity-method  
    (similarityInstance), 47
- Stemmer, 10, 48
- Stemmer, CMOptions-method (Stemmer), 48
- Stemmer<- (Stemmer), 48
- Stemmer<- , CMOptions-method (Stemmer), 48
- StopWords, 10, 48
- StopWords, CMOptions-method (StopWords),  
    48
- StopWords<- (StopWords), 48
- StopWords<- , CMOptions-method  
    (StopWords), 48
- SynonymType, 10, 49

SynonymType, COptions-method  
    (SynonymType), [49](#)  
SynonymType<- (SynonymType), [49](#)  
SynonymType<- , COptions-method  
    (SynonymType), [49](#)

typeSystemRef, [50](#)  
typeSystemRef, EntityFinder-method  
    (typeSystemRef), [50](#)  
typeSystemRef<- (typeSystemRef), [50](#)  
typeSystemRef<- , EntityFinder-method  
    (typeSystemRef), [50](#)