ontoTools
November 11, 2009

R topics documented:

accessMat .......................... 1
A.csr ................................ 2
buildGOgraph ........................ 2
compoundGraph-class ............. 3
depthStruct ........................ 4
GDL_NCIThesaurus .................. 5
gomfAmat ........................... 5
litOnto ............................. 6
LLGOMFcp ......................... 6
namedSparse-class ............... 7
ontology-class ................... 8
OOC-class .......................... 9
ooMapLL2GOMFdemo ............... 10
otkvEnv2namedSparse ............ 10
rootedDAG-class ................. 11
semsim ............................ 11
SGDIvocab ......................... 13
STMA ............................... 14
taggedHierNomenclature-class .. 15
toDot-methods .................... 16

Index 17

accessMat .......................... matrix utilities for ontoTools.

Description

A variety of matrix utilities used in ontoTools

Usage

accessMat(object)

Arguments

object  object
buildGOgraph

**Author(s)**
Vince Carey <stvjc@channing.harvard.edu>

---

### A.csr

**Description**
demonstration sparse matrix matrix.csr form

**Usage**
data(A.csr)

**Format**
The format is: list() - attr(*, "ra")= num [1:27] 2.894 -0.610 -0.714 0.546 0.901 ...
- attr(*, "ja")= int [1:27] 3 4 5 2 4 1 3 4 5 3 ...
- attr(*, "ia")= int [1:11] 1 4 6 10 11 14 18 21 22 26 ...
- attr(*, "dimension")= int [1:2] 10 5
attr(*, "class")= atomic [1:1] matrix.csr
attr(*, "package")= chr ".GlobalEnv"

**Source**
SparseM library matrix.csr example generated it

---

### buildGOgraph

**Description**
build graphNEL corresponding to bioc GO environment

**Usage**
buildGOgraph(useenv=GOMFPARENTS)

**Arguments**

useenv
useenv: environment to be used

**Details**
all GO MF tags are nodes, edges drawn from node to parent

**Value**
graphNEL instance

**Author(s)**
Vince Carey <stvjc@channing.harvard.edu>
See Also

Biograph package

Examples

```r
# takes a while; trivial invocation
```

**compoundGraph-class**

Class "compoundGraph" list representation of multiple graph::graph objects

**Objects from the Class**

Objects can be created by calls of the form `new("compoundGraph", ...)`.

**Slots**

- `grList`: Object of class "list" list of graph::graph objects
- `between`: Object of class "list" list of node-to-node connections across graphs

**Methods**

- `adjMat` signature(cg = "compoundGraph"): ...
- `between` signature(object = "compoundGraph"): ...
- `grList` signature(object = "compoundGraph"): ...
- `toDot` signature(G = "compoundGraph", outDotFile = "character", renderList = "list", optList = "missing"): ...
- `toDot` signature(G = "compoundGraph", outDotFile = "character", renderList = "list", optList = "list"): ...
- `toDot` signature(G = "compoundGraph", outDotFile = "missing", renderList = "list", optList = "missing"): ...

**Note**

Should be supplanted by Rgraphviz facilities before too long.

**Author(s)**

VJ Carey <stvjc@channing.harvard.edu>

**References**

~put references to the literature/web site here ~

**See Also**

`Rgraphviz::subgraph`
depthStruct  

**tools for manipulating depth concepts for rooted DAGs**

**Description**

return a list of environments giving mapping from node name to rooted DAG depth and from depth to vector of names of nodes at that depth.

**Usage**

```r
depthStruct(rg)
ontoDepth(rg)
DMdepth(g, maxd)
```

**Arguments**

- `rg`: instance of class rootedDAG
- `g`: instance of class depth
- `maxd`: maxd: bound on depth to be measured

**Value**

depthStruct: a list of two environments (see examples).

**Note**

ontoDepth is the workhorse for depthStruct. DMdepth is a function that works on a plain graph, creating the 'daughter matrix' and computing depths.

**Author(s)**

Vince Carey <stvjc@channing.harvard.edu>

**Examples**

```r
data(litOnto)
print(litOnto)
g1 <- new("rootedDAG", DAG=litOnto, root="A")
o1 <- new("ontology", name="demo", version="0.1", rDAG=g1)
print(ds <- depthStruct(g1))
ds$tag2depth("A")
ds$tag2depth("H")
ds$depth2tag(2)
```
Description

Structures for working with formal nomenclatures

Usage

\begin{verbatim}
data(GDI_NCIThesaurus)
parents(term, nom)
children(term, nom)
getDefs(term, nom)
\end{verbatim}

Arguments

- `term`: character string, term
- `nom`: instance of class `taggedHierNomenclature`

Details

DAG-structured nomenclatures are in wide use. For any term, one can seek parents (generalizations) or children (specializations). These resources prototype tools for dealing with such structures, including provenance information.

Author(s)

Vince Carey <stvjc@channing.harvard.edu>

Examples

\begin{verbatim}
data(GDI_NCIThesaurus)
parents("Mesna", GDI_NCIThesaurus)
parents("Actinomycin_Antibiotic", GDI_NCIThesaurus)
\end{verbatim}

\begin{verbatim}
gomfAmat 
\end{verbatim}

\begin{verbatim}
 sparse matrix representing accessibilities of terms in GO MF graph; graph also documented here
\end{verbatim}

Description

sparse matrix representing accessibilities of terms in GO MF graph

Usage

\begin{verbatim}
data(gomfAmat); data(goMFgraphDemo)
\end{verbatim}
Format

The format is: list() - attr(*, "Dimnames")=List of 2 ..: chr[1 : 5399] "GO : 0000005" "GO : 0000006" "GO : 0000007" "GO : 0000008" .... : chr [1:5399] "GO:0000005" "GO:0000006" "GO:0000007" "GO:0000008" ... - attr(*, "mat")=list() ..- attr(*, "ra")=num [1:33263] 0 1 1 1 1 1 1 1 1 ... - attr(*, "ia")=int [1:5399] 1 2 3 4 5 6 ... - attr(*, "class")= atomic [1:1] matrix.csr ..- attr(*, "package")= chr ".GlobalEnv"

Source

built from bioconductor graph, GO and ontoTools package tools

litOnto: graph illustrating the ontology concept; litObj: matrix illustrating the object-ontology mapping

Description

litOnto: graph illustrating the ontology concept; litObj: matrix illustrating the object-ontology mapping

Usage

data(litOnto)

Format

The format is: list() - attr(*, "nodes")= chr [1:12] "A" "B" "C" "D" ... - attr(*, "edgeL")=List of 12 ..: Listof1.... edge: NULL ..: Listof1.... edges: int 1 ..: Listof1.... edges: int 1 ..: Listof1.... edges: int 2 ..: Listof1.... edges: int 2 ..: Listof1.... edges: int 3 ..: Listof1.... edges: int 3 ..: Listof1.... edges: int 1 ..: Listof1.... edges: int 1 ..: Listof1.... edges: int 7 - attr(*, "edgemode")= chr "directed" - attr(*, "class")= chr "graphNEL"

LLGOMFcp

resources saved for computation of concept probabilities for GO MF terms applied to human LocusLink entries

Description

concept probabilities for GO MF terms applied to human LocusLink entries

Usage

data(LLGOMFcp)
namedSparse-class

Format
The format is: Named num [1:5399] 0 0 0 0 0 ... - attr(*, "names")= chr [1:5399] "GO:0000005" "GO:0000006" "GO:0000007" "GO:0000008" ...

Source
derived from Bioconductor packages humanLLMappings and GO

namedSparse-class Class "namedSparse" adds margin names to sparse matrices

Description
manages margin names for sparse matrices

Objects from the Class
Objects can be created by calls of the form `new("namedSparse", ...)` These are S4 objects that include a SparseM::matrix.csr, associated dimnames in the customary form, and two lists of name-to-index mapping environments (for row and column name resolution, with forward (name to index) and backward (index to name) mapping).

Slots
Dimnames: Object of class "list" ordinary dimnames matrix metadata
mat: Object of class "matrix.csr" sparse matrix

Methods
Arith signature(e1 = "namedSparse", e2 = "namedSparse"): ...
show signature(object = "namedSparse"): ...

Note
A constructor `makeNamedSparse` is illustrated in the example.

Author(s)
VJ Carey <stvjc@channing.harvard.edu>

Examples
data(A.csr)
namedA <- mkNS(A.csr) # installs default dimnames R(1:nrow), C(1:ncol)
print(namedA)
print(cclSums(namedA))
dimnames(namedA) <- list(paste("A",1:10,sep=""),paste("B",1:5,sep=""))
print(namedA)
print(namedA %*% t(namedA))
ontology-class

Class "ontology" wraps a rooted DAG with some ontology metadata

Description

instances of class ontology are used to represent things like Gene Ontology

Objects from the Class

Objects can be created by calls of the form `new("ontology", ...)`. This simply possesses information on name and version of ontology.

Slots

name: Object of class "character" name of ontology
version: Object of class "character" version tag
rDAG: Object of class "rootedDAG" the rooted DAG representing the terminology hierarchy

Methods

accessMat signature(object = "ontology"): returns square matrix with 1 in element r,c if term corresponding to r can be reached from term corresponding to c
name signature(x = "ontology"): access name
OVersion signature(x = "ontology"): access version
rDAG signature(x = "ontology"): access the rooted DAG
show signature(object = "ontology"): concise report

Note

This class was written to deal with ontologies that are representable as rooted DAGs. It is not clear that this is a good use of the term 'ontology', which has broader implications. However this does work for Gene Ontology.

Author(s)

~~who you are~~

Examples

data(litOnto)
print(litOnto)
g1 <- new("rootedDAG", DAG=litOnto, root="A")
o1 <- new("ontology", name="demo", version="0.1",
rDAG=g1)
# can also use
o1b <- makeOntology(name="demo", version="0.1",
graph=litOnto, root="A")
show(o1)
print(accessMat(o1))
print(OVersion(o1))
Description

Object that binds ontology (structured vocabulary) with an object-term map.

Objects from the Class

Objects can be created by calls of the form `new("OOC", ...)`.

Slots

- `ontology`: Object of class "ontology" instance of `ontoTools::ontology`
- `OOmap`: Object of class "namedSparse" `SparseM::matrix.csr` bound with dimnames facilities

Methods

- `coverageMat` signature(x = "OOC"): return a namedSparse incidence matrix with r,c element indicating whether term c covers object r
- `ontology` signature(x = "OOC"): accessor
- `OOmap` signature(x = "OOC"): accessor
- `show` signature(object = "OOC"): concise printer

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

data(litOnto)
g1 <- new("rootedDAG", DAG=litOnto, root="A")
o1 <- new("ontology", name="demo", version="0.1", rDAG=g1)
kvlist <- list(W="E", X="K", Y="B", Z=c("D","G"))
litMap <- otkvList2namedSparse( names(kvlist), LETTERS[1:12], kvlist )
print(litMap)
ooc1 <- makeOOC( o1, litMap )
show(ooc1)
print(coverageMat(ooc1))
# note the following will be slow with large OOCs
print(conceptProbs(ooc1))
# for larger OOCs it is useful to precompute the accessibility
# matrix of the ontology and the map from objects to terms -- these
# can be supplied as additional arguments to conceptProbs
Description

object-term mapping for human locuslink entries and GO MF

Usage

data(ooMapLL2GOMFdemo)

Format

The format is: list() - attr(*, "Dimnames")=List of 2 ..: chr[1: 10776]"1""10""100""1000"..... : chr [1:5399] "GO:0000005" "GO:0000006" "GO:0000007" "GO:0000008" ... - attr(*, "mat")=
list() - attr(*, "ra")= num [1:19679] 0 1 1 1 1 1 1 1 1 .... - attr(*, "ja")= int [1:19679] 1 1842
541 3528 3753 485 3799 1594 1804 1098 ... - attr(*, "ia")= int [1:10777] 1 3 6 8 10 13 14 17
23 28 ... - attr(*, "dimension")= int [1:2] 10776 5399 ... - attr(*, "class")= chr "matrix.csr" - attr(*,
"rowindex")=List of 2 ..n2i : length0 < environment > .. i2n:length 0 <environment> - attr(*,
"colindex")=List of 2 ..n2i : length0 < environment > .. i2n:length 0 <environment> - attr(*,
"class")= chr "namedSparse"

Source

bioconductor GO, humanLLMapping and ontoTools otkv tools.

Description

obtain sparse matrix representation of key-value structures

Usage

otkvEnv2namedSparse(obs, tms, otkvEnv)

Arguments

obs: vector of object tags
tms: vector of terms to which objects are mapped
otkvEnv: environment encoding the key-value mapping
otkvlist: list encoding the key-value mapping

Author(s)

Vince Carey <stvjc@channing.harvard.edu>
Examples

```
okvList2namedSparse(c("A","B","D","E"), letters[1:7],
    list("A"=c("a","b"), "B"=c("b","d"), "E"="c"))
```

rootedDAG-class  Class "rootedDAG"

Description

wraps a graph that can be shown to be a DAG and has a root (one node with no ancestor)

Objects from the Class

Objects can be created by calls of the form `new("rootedDAG", ...)`. Does not extend `graph` but probably should.

Slots

- **root**: Object of class "character" name of root
- **DAG**: Object of class "graph" DAG

Methods

- **DAG** signature(x = "rootedDAG")": extract the graph
- **getMatrix** signature(g = "rootedDAG", type = "character", mode = "character")":
  Currently only `child2parent` can be used for type, meaning that row corresponds to child, column corresponds to ancestor and mat[row,column] is 1 if node corresponding to row is a child of node corresponding to ancestor. Type can be `sparse` (return sparse representation) or `dense`.
- **root** signature(x = "rootedDAG")": extract name of root

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

semsim  Compute semantic similarity measure for terms in an object-ontology complex

Description

Compute semantic similarity measure for terms in an object-ontology complex

Usage

```
semsim(c1, c2, ooc, acc=NULL, pc=NULL)
conceptProbs(ooc, acc=NULL, inds=NULL)
subsumers(c1, c2, ont, acc=NULL)
pms(c1, c2, ooc, acc=NULL, pc=NULL)
usageCount(map, acc, inds)
```
Arguments

- **c1, c2**: "character" terms to be compared
- **ooc**: an object of class "OOC": object-ontology complex
- **ont**: an object of class "ontology": annotated rooted DAG
- **acc**: an object of class "ontology": annotated rooted DAG
- **pc**: optional vector of concept probabilities, if pre-computed
- **inds**: vector of numeric indices, row indices of object-ontology map to be processed

Details

For large ontologies, computation of the term accessibility relationships and term probabilities can be costly. Once these are computed to support one semsim calculation, they should be saved. The acc and pc parameters allow use of this saved information.

Value


Author(s)

Vince Carey <stvjc@channing.harvard.edu>

References

PW Lord et al, Bioinformatics, 19(10)2003:1275

Examples

```r
# we are given a graph of GOMF and the OOmap between LL and GOMF
# derived from humanLLMappings and stored as data resources in
# ontoTools -- these will have to be updated regularly
# data(goMFgraph.1.15)
data(LL2GOMFooMap.1.15)
# build the rooted DAG, the ontology, and the OOC objects
# gomfrDAG <- new("rootedDAG", root="GO:0003674", DAG=goMFgraph.1.15)
GOMFonto <- new("ontology", name="GOMF", version="bioc GO 1.15", rDAG=gomfrDAG)
LLGOMFOOC <- makeOOC(GOMFonto, LL2GOMFooMap.1.15)
# we are given the accessibility matrix for the GO MF graph as a
data resource, and we can compute some term probabilities
# data(goMFamat.1.15)
pc <- conceptProbs(LLGOMFOOC, goMFamat.1.15, inds=1:20)
# now we will get a sample of GO MF terms and compute the
# semantic similarities of pairs of terms in the sample
```
# data(LL2GOMFcp.1.15) # full set of precomputed concept probabilities
library(GO.db)
library(Biobase)
library(combinat)
library(annotate)
GO() # get the GO environments
GOtags <- ls(GOTERM)
GOlabs <- mget(GOtags, GOTERM, ifnotfound=NA)
GOMFtags <- GOtags[sapply(GOlabs,Ontology)=="MF"]
GOMFtags <- GOMFtags[!is.na(GOMFtags)]
GOMFtermObs <- mget(GOMFtags, env=GOTERM)
GOMFterms <- sapply(GOMFtermObs, Term)
ntags <- length(GOMFtags)
if (any(duplicated(GOMFterms))){
dups <- (1:ntags)[duplicated(GOMFterms)]
GOMFterms[dups] <- paste(GOMFterms[dups],".2",sep="")
}
#names(GOMFterms) <- GOMFtags
set.seed(1234)
# does not lead to common samples across platforms...
st <- sample(names(GOMFterms),size=50) # take the sample
st <- intersect(st, names(LL2GOMFcp.1.15))[1:10] # use only those terms available in bioc
# thus ...
pst <- combn(st,2) # get a matrix with the pairs of terms in columns
bad = c(4L, 12L, 19L, 25L, 31L, 32L, 33L, 34L, 35L) # can't use 8640
pst <- pst[,-bad]
npst <- ncol(pst)
ss <- rep(NA,npst)
for (i in 1:npst) # compute semantic similarities
{cat(i)
 ss[i] <- semsim( pst[1,i], pst[2,i], ooc=LLGOMFOOC, acc=goMFamat.1.15, pc=LL2GOMFcp.1.15)
}
print(summary(ss))
top <- (1:npst)[ss==max(ss,na.rm=TRUE)][1] # index of the most similar pair
# note -- must come to an understanding of the NAs
print( GOMFterms[ as.character(pst[,top]) ] )
pen <- (1:npst)[ss[1:npst]==max(ss[1:npst,na.rm=TRUE])][1] # second most similar
print( GOMFterms[ as.character(pst[,pen]) ] )
Details
Currently unites some english terms with a formal tag to NCI Metathesaurus. Additional content to be added, with bridges to NCI EVS.

Value
This is an instance of a `taggedHierNomenclature` derived from an informal specification of terms about breast cancer provided by S. Ramaswamy.

Author(s)
Vince Carey <stvjc@channing.harvard.edu>

Examples

data(SGDIvocab)
SGDIvocab
grep("differ", SGDIvocab)
getTerms(SGDIvocab)

---

STMA

*Vocabulary from statistics theory and methods abstracts*

Description
Vocabulary from statistics theory and methods abstracts

Usage
data(STMA)

Author(s)
Vince Carey <stvjc@channing.harvard.edu>

Examples
data(STMA)
grep("exponential", STMA)
parents("Exponential families", STMA)
children("PROBABILITY DISTRIBUTIONS", STMA)
Class "taggedHierNomenclature"~

**Description**

representation of a DAG-structured nomenclature

**Objects from the Class**

Objects can be created by calls of the form `new("taggedHierNomenclature", ...)`. See example

**Slots**

- `tags`: Object of class "character"; formal tags, often semantically opaque
- `parents`: Object of class "character"; terms regarded as generalizations of the given term
- `delim`: Object of class "character"; the parent strings are decomposed using this delimiter
- `rootToken`: Object of class "character"; token used to indicate root of DAG
- `name`: Object of class "character" name of nomenclature
- `provenance`: Object of class "provStruct" information on origins of vocabulary
- `inMappings`: Object of class "character" list of mappings in which the term is employed
- `terms`: Object of class "character" actual subject matter terms being organized
- `definitions`: Object of class "character" verbal definitions of terms

**Extends**

Class "nomenclature", directly.

**Methods**

- `children` signature(term = "character", nom = "taggedHierNomenclature"): ...
- `parents` signature(term = "character", nom = "taggedHierNomenclature"): ...
- `show` signature(object = "taggedHierNomenclature"): ...

**Author(s)**

VJ Carey <stvjc@channing.harvard.edu>

**See Also**

[GDI_NCIThesaurus](#)

**Examples**

data(GDI_NCIThesaurus)
GDI_NCIThesaurus
Methods for Function toDot in Package ‘ontoTools’ – should be replaced by Rgraphviz facilities soon

Description

These methods write graphviz dot language for various graph structures encountered with ontologies. This activity should be moved to Rgraphviz ASAP, but there are aspects of representation and portability that need to be resolved.

Methods

```r
G = "graphNEL", outDotFile = "character", renderList = "list", optList = "list" create dot language description of graph
G = "graphNEL", outDotFile = "character", renderList = "missing", optList = "missing" create dot language description of graph
G = "graphNEL", outDotFile = "character", renderList = "missing", optList = "list" create dot language description of graph
G = "graphNEL", outDotFile = "missing", renderList = "missing", optList = "missing" create dot language description of graph
G = "graphNEL", outDotFile = "missing", renderList = "missing", optList = "list" create dot language description of graph
G = "graphNEL", outDotFile = "missing", renderList = "list", optList = "missing" create dot language description of graph
G = "graphNEL", outDotFile = "missing", renderList = "list", optList = "list" create dot language description of graph
G = "compoundGraph", outDotFile = "character", renderList = "list", optList = "missing" create dot language description of graph
G = "compoundGraph", outDotFile = "character", renderList = "list", optList = "list" create dot language description of graph
G = "compoundGraph", outDotFile = "missing", renderList = "list", optList = "missing" create dot language description of graph
```

Examples

```r
example(randomGraph)
tmp <- tempfile()
toDot( g1, tmp )
readLines(tmp)
unlink(tmp)
```
Index

*Topic classes*
compoundGraph-class, 3
namedSparse-class, 7
ontology-class, 8
OOC-class, 9
rootedDAG-class, 11
taggedHierNomenclature-class, 15

*Topic datasets*
gomfAmat, 5
litOnto, 6
LLGOMFcp, 6
ooMapLL2GOMFdemo, 10

*Topic methods*
toDot-methods, 16

toDot-methods, 16

*Topic models*
A.csr, 1
accessMat, 1
buildGOgraph, 2
depthStruct, 4
GDI_NCIThesaurus, 5
otkvEnv2namedSparse, 10
semsim, 11
SGDIvocab, 13
STMA, 14

+, namedSparse, namedSparse-method
(namedSparse-class), 7
[-, namedSparse-method
(namedSparse-class), 7
%*%, namedSparse, namedSparse-method
(namedSparse-class), 7
%+% (namedSparse-class), 7

A.csr, 1
accessMat, 1
accessMat, ontology-method
(ontology-class), 8
adjMat, compoundGraph-method
(compoundGraph-class), 3
adjMat, compoundGraph-method
(compoundGraph-class), 3
AMN (namedSparse-class), 7
annoSource-class (SGDIvocab), 13
annotationResource-class
(SGDIvocab), 13
Arith, namedSparse, namedSparse-method
(namedSparse-class), 7
as.matrix, namedSparse-method
(namedSparse-class), 7
as.matrix.ok (namedSparse-class), 7
between (compoundGraph-class), 3
between, compoundGraph-method
(compoundGraph-class), 3
buildGOgraph, 2
child2parentMatDense (accessMat), 1
child2parentMatSparse
(accessMat), 1
children (GDI_NCIThesaurus), 5
children, character, taggedHierNomenclature-method
(taggedHierNomenclature-class), 15
colinds (namedSparse-class), 7
colinds, namedSparse, character-method
(namedSparse-class), 7
colSums (namedSparse-class), 7
colSums, namedSparse, missing, missing-method
(namedSparse-class), 7
colSumsSp (namedSparse-class), 7
compoundGraph-class, 3
conceptProbs (semsim), 11
coverageMat (OOC-class), 9
coverageMat, OOC-method
(OOC-class), 9
coverMat (accessMat), 1

dag (rootedDAG-class), 11
dag, rootedDAG-method
(rootedDAG-class), 11
daughterMat (accessMat), 1
daughterSpMat (accessMat), 1
depthStruct, 4
dimnames, namedSparse-method
(namedSparse-class), 7
dimnames<-, namedSparse, list-method
(namedSparse-class), 7
INDEX

DMdepth (depthStruct), 4
exns2 (namedSparse-class), 7
exptArchive-class (SGDIvocab), 13
exptSample-class (SGDIvocab), 13
GDI_NCIThesaurus, 5, 15
GDILabel (GDI_NCIThesaurus), 5
GDILabel-class (SGDIvocab), 13
GDontology (GDI_NCIThesaurus), 5
GDontology-class (SGDIvocab), 13
GDIplatform-class (SGDIvocab), 13
getDefs (GDI_NCIThesaurus), 5
getDefs, character, nomenclature-method
(GDI_NCIThesaurus), 5
getMatrix (accessMat), 1
getMatrix, rootedDAG, character, character-method
(rootedDAG-class), 11
getTerms (GDI_NCIThesaurus), 5
getTerms, taggedHierNomenclature-method
(SGDIvocab), 13
go1.15DAG (LLGOMFcp), 6
GOMF1.15 (LLGOMFcp), 6
gomfAmat, 5
gomfAmat.1.15 (LLGOMFcp), 6
gomfAmat.1.4 (LLGOMFcp), 6
gomfGraph.1.15 (LLGOMFcp), 6
gomfGraph.1.4 (LLGOMFcp), 6
gomfGraphDemo (gomfAmat), 5
grep, character, taggedHierNomenclature-method
(SGDIvocab), 13
grep, character, taggedHierNomenclature-method
(SGDIvocab), 13
grList (compoundGraph-class), 3
grList, compoundGraph-method
(compoundGraph-class), 3
Iyer517 (ooMapLL2GOMFdemo), 10
litObj (litOnto), 6
litOnto, 6
LL2GOMFCp.1.15 (LLGOMFcp), 6
LL2GOMFCp.1.4 (LLGOMFcp), 6
LL2GOMFOcc1.15 (LLGOMFcp), 6
LL2GOMFocMap.1.15 (LLGOMFcp), 6
LL2GOMFocMap.1.4 (LLGOMFcp), 6
LLGOMFCp, 6
makeNamedSparse
(namedSparse-class), 7
makeOntology (ontology-class), 8
makeOOC (OOC-class), 9
makeSparseZero
(namedSparse-class), 7
mapNamesInds (namedSparse-class), 7
mat (namedSparse-class), 7
mat, namedSparse-method
(namedSparse-class), 7
maxval (namedSparse-class), 7
maxval, matrix.csr-method
(namedSparse-class), 7
mkNS (namedSparse-class), 7
name (ontology-class), 8
name, ontology-method
(ontology-class), 8
namedSparse-class, 7
ncol, namedSparse-method
(namedSparse-class), 7
newadj (accessMat), 1
nomenclature-class (SGDIvocab), 13
prow, namedSparse-method
(namedSparse-class), 7
nsparse (namedSparse-class), 7
ontoDepth (depthStruct), 4
ontology (ontology-class), 8
ontology, OOC-method (OOC-class), 9
ontology-class, 8
OOC-class, 9
OOCmap (OOC-class), 9
OOCmap, OOC-method (OOC-class), 9
OOmap, OOmap, OOC-method (OOC-class), 9
OVersion (ontology-class), 8
OVersion, ontology-method
(ontology-class), 8
parents (GDI_NCIThesaurus), 5
parents, character, taggedHierNomenclature-method
(taggedHierNomenclature-class), 15
pms (sensim), 11
provStruct (GDI_NCIThesaurus), 5
provStruct-class (SGDIvocab), 13
rDAG (ontology-class), 8
rDAG, ontology-method
(ontology-class), 8
revArcs (accessMat), 1
root (rootedDAG-class), 11
root, rootedDAG-method
(rootedDAG-class), 11
rootedDAG-class, 11
INDEX

rowinds (namedSparse-class), 7
rowinds, namedSparse, character-method (namedSparse-class), 7
rowSums (namedSparse-class), 7
rowSums, namedSparse, missing, missing-method (namedSparse-class), 7
rowSumsSp (namedSparse-class), 7
simsim, 11
SGDlvocab, 13
show, exptArchive-method (SGDlvocab), 13
show, exptSample-method (SGDlvocab), 13
show, GDIplatform-method (SGDlvocab), 13
show, namedSparse-method (namedSparse-class), 7
show, nomenclature-method (GDI_NCIThesaurus), 5
show, ontology-method (ontology-class), 8
show, OOC-method (OOC-class), 9
show, provStruct-method (SGDlvocab), 13
show, symMapping-method (SGDlvocab), 13
show, taggedHierNomenclature-method (taggedHierNomenclature-class), 15
STMA, 14
subsumers (semsim), 11
sumSp (namedSparse-class), 7
sumSpSlow (namedSparse-class), 7
symMapping-class (SGDlvocab), 13
t, namedSparse-method (namedSparse-class), 7
taggedHierNomenclature, 5, 14
taggedHierNomenclature (taggedHierNomenclature-class), 15
taggedHierNomenclature-class, 15
thesList (GDI_NCIThesaurus), 5
toDot (toDot-methods), 16
toDot, compoundGraph, character, list, list-method (compoundGraph-class), 3
toDot, compoundGraph, character, list, missing-method (compoundGraph-class), 3
toDot, compoundGraph, missing, list, missing-method (compoundGraph-class), 3
toDot, graphNEL, character, list, list-method (toDot-methods), 16
toDot, graphNEL, character, missing, list, list-method (toDot-methods), 16
toDot, graphNEL, character, missing, missing-method (toDot-methods), 16
toDot, graphNEL, missing, character, missing-method (toDot-methods), 16
toDot, graphNEL, missing, list, list-method (toDot-methods), 16
toDot, graphNEL, missing, list, missing-method (toDot-methods), 16
toDot, graphNEL, missing, missing, list-method (toDot-methods), 16
toDot, graphNEL, missing, missing, missing-method (toDot-methods), 16
toDot-methods, 16
typedSymMapping-class (SGDlvocab), 13
usageCount (semsim), 11