Package ‘ontoProc’

October 16, 2019

Title processing of ontologies of anatomy, cell lines, and so on

Description Support harvesting of diverse bioinformatic ontologies, making particular use of the ontologyIndex package on CRAN. We provide snapshots of key ontologies for terms about cells, cell lines, chemical compounds, and anatomy, to help analyze genome-scale experiments, particularly cell x compound screens. Another purpose is to strengthen development of compelling use cases for richer interfaces to emerging ontologies.

Version 1.6.0

Author Vince Carey `<stvjc@channing.harvard.edu>`

Imports Biobase, S4Vectors, methods, AnnotationDbi, stats, utils, shiny, graph, Rgraphviz, ontologyPlot, dplyr, magrittr

Suggests knitr, org.Hs.eg.db, org.Mm.eg.db, testthat, BiocStyle

Depends R (>= 3.5), ontologyIndex

Maintainer VJ Carey `<stvjc@channing.harvard.edu>`

License Artistic-2.0

LazyLoad yes

LazyData yes

biocViews Infrastructure, GO

RoxygenNote 6.1.1

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/ontoProc

git_branch RELEASE_3_9

git_last_commit 9cb53f1

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`allGOterms` *allGOterms: data.frame with ids and terms*

Description

allGOterms: data.frame with ids and terms

Usage

```
allGOterms
```

Format

data.frame instance

Source

This is a snapshot of all the terms available from GO.db (3.4.2), August 2017, using keys(GO.db, keytype="TERM").

Examples

```
data(allGOterms)
head(allGOterms)
```

`c,TermSet-method` *combine TermSet instances*

Description

combine TermSet instances

Usage

```
## S4 method for signature 'TermSet'
c(x, ...)
```

Arguments

`x`	TermSet instance
`...`	additional instances

Value

TermSet instance

`cellTypeToGO` *utilities for approximate matching of cell type terms to GO categories and annotations*

Description

utilities for approximate matching of cell type terms to GO categories and annotations

Usage

```
cellTypeToGO(celltypeString, gotab, ...)

cellTypeToGenes(celltypeString, gotab, orgDb, cols = c("ENSEMBL",
  "SYMBOL"), ...)
```

Arguments

`celltypeString`	character atom to be used to search GO terms using
`gotab`	a data.frame with columns GO (goids) and TERM (term strings) `agrep`
`...`	additional arguments to `agrep`
`orgDb`	instances of orgDb
`cols`	columns to be retrieved in select operation

Value

data.frame

data.frame

Note

Very primitive, uses agrep to try to find relevant terms.

Examples

```
data(allGOterms)
library(org.Hs.eg.db)
head(cellTypeToGO("serotonergic neuron", allGOterms))
head(cellTypeToGenes("serotonergic neuron", allGOterms, org.Hs.eg.db))
```

`cleanCLOnames` *obtain named character vector of terms from Cell Line Ontology, omitting obsolete and trailing 'cell'*

Description

obtain named character vector of terms from Cell Line Ontology, omitting obsolete and trailing 'cell'

Usage

```
cleanCLOnames()
```

Value

character()

Examples

```
cleanCLOnames()[1:10]
```

`CLfeats` *produce a data.frame of features relevant to a Cell Ontology class*

Description

produce a data.frame of features relevant to a Cell Ontology class

Usage

```
CLfeats(ont, tag = "CL:0001054")
```

Arguments

`ont`	instance of ontologyIndex ontology
`tag`	character(1) a CL: class tag

Value

a data.frame instance

Note

This function will look in the intersection_of and has_part, lacks_part components of the CL entry to find properties asserted of or inherited by the cell type identified in 'tag'

Examples

```
cl = getCellOnto()
pr = getPROnto()
go = getGeneOnto()
CLfeats(cl, tag="CL:0001054")
```

`ctmarks` *app to review molecular properties of cell types via cell ontology*

Description

app to review molecular properties of cell types via cell ontology

Usage

```
ctmarks(cl)
```

Arguments

`cl`	an import of a Cell Ontology (or extended Cell Ontology) in ontology_index form

Value

a data.frame with features for selected cell types

Note

Prototype of harvesting of cell ontology by searching has_part, has_plasma_membrane_part, intersection_of and allied ontology relationships. Uses shiny. Can perform better if getPROnto() and getGeneOnto() values are in .GlobalEnv as pr and go respectively.

`cyclicSigset` *as in Bakken et al. (2017 PMID 29322913) create gene signatures for k cell types, each of which fails to express all but one gene in a set of k genes*

Description

as in Bakken et al. (2017 PMID 29322913) create gene signatures for k cell types, each of which fails to express all but one gene in a set of k genes

Usage

```
cyclicSigset(idvec, conds = c("hasExp", "lacksExp"),
  tags = paste0("CL:X", 1:length(idvec)))
```

Arguments

`idvec`	character vector of identifiers, must have names() set to identify cells bearing genes
`conds`	character(2) tokens used to indicate condition to which signature element contributes
`tags`	character vector of cell-type identifiers; for Cell Ontology use CL: as prefix, one element for each element of idvec

Value

a long data.frame

Examples

```
sigels = c("CL:X01"="GRIK3", "CL:X02"="NTNG1", "CL:X03"="BAGE2",
        "CL:X04"="MC4R", "CL:X05"="PAX6", "CL:X06"="TSPAN12", "CL:X07"="hSHISA8",
     "CL:X08"="SNCG", "CL:X09"="ARHGEF28", "CL:X10"="EGF")
sigdf = cyclicSigset(sigels)
head(sigdf)
```

`demoApp` *demonstrate the use of makeSelectInput*

Description

demonstrate the use of makeSelectInput

Usage

```
demoApp()
```

Value

Run only for side effect of starting a shiny app.

Examples

```
  if (interactive()) {
require(shiny)
print(demoApp())
}
```

`dropStop` *dropStop is a utility for removing certain words from text data*

Description

dropStop is a utility for removing certain words from text data

Usage

```
dropStop(x, drop, lower = TRUE, splitby = " ")
```

Arguments

`x`	character vector of strings to be cleaned
`drop`	character vector of words to scrub
`lower`	logical, if TRUE, x converted with `tolower`
`splitby`	character, used with strsplit to tokenize `x`

Value

a list with one element per input string, split by " ", with elements in `drop` removed

Examples

```
data(minicorpus)
minicorpus[1:3]
dropStop(minicorpus)[1:3]
```

`fastGrep` *some fields of interest are lists, and grep per se should not be used – this function checks and uses grep within vapply when appropriate*

Description

some fields of interest are lists, and grep per se should not be used – this function checks and uses grep within vapply when appropriate

Usage

```
fastGrep(patt, onto, field, ...)
```

Arguments

`patt`	a regular expression whose presence in field should be checked
`onto`	an ontologyIndex instance
`field`	the ontologyIndex component to be searched
`...`	passed to grep

Value

logical vector indicating vector or list elements where a match is found

Examples

```
cheb = getChebiOnto()
ind = fastGrep("17-AAG", cheb, "synonym")
cheb$name[ind]
```

---

getCellOnto *load ontologies that may include non-ascii strings and therefore cannot be in data folder*

Description

load ontologies that may include non-ascii strings and therefore cannot be in data folder

Usage

```
getCellOnto(useNew = TRUE)

getCellLineOnto()

getEFOOnto()

getChebiLite()

getCellosaurusOnto()

getUBERON_NE()

getChebiOnto()

getOncotreeOnto()

getDiseaseOnto()

getGeneOnto()

getHCAOnto()

getPROnto()

getPATOnto()
```

Arguments

`useNew`	logical(1) only for getCellOnto if TRUE cell ontology of July 2018, otherwise use legacy

Value

instance of ontology_index (S3) from ontologyIndex

instance of ontology_index (S3) from ontologyIndex

Note

Provenance information is kept in the form of excerpts of top records in ‘dir(system.file("obo", package="ontoProc"), full=TRUE)‘

getChebiOnto loads ontoRda/chebi_full.rda

getOncotreeOnto loads ontoRda/oncotree.rda

getDiseaseOnto loads ontoRda/diseaseOnto.rda

getHCAOnto loads ontoRda/hcaOnto.rda produced from hcao.owl at https://github.com/HumanCellAtlas/ontology/releas
2/11/2019, python pronto was used to convert OWL to OBO.

getPROnto loads ontoRda/PRonto.rda, produced from http://purl.obolibrary.org/obo/pr.obo ’reasoned’ ontology from OBO foundry, 02-08-2019. In contrast to other ontologies, this is imported via get_OBO with ‘extract_tags=’minimal’‘.

getPATOnto loads ontoRda/patoOnto.rda, produced from https://raw.githubusercontent.com/pato-ontology/pato/master/pato.obo from OBO foundry, 02-08-2019.

Examples

```
co = getCellOnto(useNew=TRUE)
co
clo = getCellLineOnto()
length(clo$id)
che = getChebiLite()
length(che$id)
efo = getEFOOnto()
length(efo$id)
```

`humrna` *humrna: a data.frame of SRA metadata related to RNA-seq in humans*

Description

humrna: a data.frame of SRA metadata related to RNA-seq in humans

Usage

```
humrna
```

Format

data.frame

Note

arbitrarily chosen from RNA-seq studies for taxon 9606

Source

NCBI SRA

Examples

```
data(humrna)
names(humrna)
head(humrna[,1:5])
```

`improveNodes` *inject linefeeds for node names for graph, with textual annotation from ontology*

Description

inject linefeeds for node names for graph, with textual annotation from ontology

Usage

```
improveNodes(g, ont)
```

Arguments

`g`	graphNEL instance
`ont`	instance of ontology from ontologyIndex

`ldfToTerms` *use output of cyclicSigset to generate a series of character vectors constituting OBO terms*

Description

use output of cyclicSigset to generate a series of character vectors constituting OBO terms

Usage

```
ldfToTerms(ldf, propmap, sigels, prologMaker = function(id, ...)
  sprintf("id: %s", id))
```

Arguments

`ldf`	a 'long format' data.frame as created by cyclicSigset
`propmap`	a character vector with names of elements corresponding to 'abbreviated' relationship tokens and element values corresponding to full relationship-naming strings
`sigels`	a named character vector associating cell types (names) to genes expressed in a cyclic set, one element per type
`prologMaker`	a function with arguments (id, ...), in which id is character(1), that generates a vector of strings that will be used for each cell type-specific term.

Value

a character vector, strings can be concatenated to OBO

Note

ldfToTerms is not sufficiently general to produce terms for any reasonably populated long data frame/propmap combination, but it is a working example for the cyclic set context.

Examples

```
# a set of cell types -- names are cell type token, values are genes expressed in a
# cyclic set -- each cell type expresses exactly one gene in the set and fails to
# express all the other genes in the set.  See Figs 3 and 4 of Bakken et al [PMID 29322913].
sigels = c("CL:X01"="GRIK3", "CL:X02"="NTNG1", "CL:X03"="BAGE2",
        "CL:X04"="MC4R", "CL:X05"="PAX6", "CL:X06"="TSPAN12", "CL:X07"="hSHISA8",
        "CL:X08"="SNCG", "CL:X09"="ARHGEF28", "CL:X10"="EGF")
# create the associated long data frame
ldf = cyclicSigset(sigels)
# describe the abbreviations
pmap = c("hasExp"="has_expression_of", lacksExp="lacks_expression_of")

# now define the prolog for each cell type
makeIntnProlog = function(id, ...) {
# make type-specific prologs as key-value pairs
    c(
      sprintf("id: %s", id),
      sprintf("name: %s-expressing cortical layer 1 interneuron, human", ...),
    sprintf("def: '%s-expressing cortical layer 1 interneuron, human described via RNA-seq observations' [PMID
      "is_a: CL:0000099 ! interneuron",
      "intersection_of: CL:0000099 ! interneuron")
}
tms = ldfToTerms(ldf, pmap, sigels, makeIntnProlog)
cat(tms[[1]], sep="\n")
```

`liberalMap` *Produce a data.frame with a set of naive terms mapped to all matching ontology ids and their formal terms*

Description

Produce a data.frame with a set of naive terms mapped to all matching ontology ids and their formal terms

Usage

```
liberalMap(terms, onto, useAgrep = FALSE, ...)
```

Arguments

`terms`	character() vector, can use grep-compatible regular expressions
`onto`	an instance of ontologyIndex::ontology_index
`useAgrep`	logical(1) if TRUE, agrep will be used
`...`	passed to agrep if used

Value

a data.frame

Examples

```
cands = c("astrocyte$", "oligodendrocyte", "oligodendrocyte precursor",
   "neoplastic", "^neuron$", "^vascular", "badterm")
co = ontoProc::getCellOnto()
liberalMap(cands, co)
```

`makeSelectInput` *generate a selectInput control for an ontologyIndex slice*

Description

generate a selectInput control for an ontologyIndex slice

Usage

```
makeSelectInput(onto, term, type = "siblings", inputId, label,
  multiple = TRUE, ...)
```

Arguments

`onto`	ontologyIndex instance
`term`	character(1) term used as basis for term list option set in the control
`type`	character(1) 'siblings' or 'children', relationship to 'term' that the options will satisfy
`inputId`	character(1) for use in server
`label`	character(1) for labeling in ui
`multiple`	logical(1) passed to `selectInput`
`...`	additional parameters passed to `selectInput`

Value

a `selectInput` control

Examples

```
makeSelectInput
```

`make_graphNEL_from_ontology_plot` *obtain graphNEL from ontology_plot instance of ontologyPlot*

Description

obtain graphNEL from ontology_plot instance of ontologyPlot

Usage

```
make_graphNEL_from_ontology_plot(x)
```

Arguments

`x`	instance of S3 class ontology_plot

Value

instance of S4 graphNEL class

Examples

```
requireNamespace("Rgraphviz")
requireNamespace("graph")
cl = getCellOnto()
cl3k = c("CL:0000492", "CL:0001054", "CL:0000236", "CL:0000625",
   "CL:0000576", "CL:0000623", "CL:0000451", "CL:0000556")
p3k = ontologyPlot::onto_plot(cl, cl3k)
gnel = make_graphNEL_from_ontology_plot(p3k)
gnel = improveNodes(gnel, cl)
graph::graph.par(list(nodes=list(shape="plaintext", cex=.8)))
gnel = Rgraphviz::layoutGraph(gnel)
Rgraphviz::renderGraph(gnel)
```

`mapOneNaive` *use grep or agrep to find a match for a naive token into ontology*

Description

use grep or agrep to find a match for a naive token into ontology

Usage

```
mapOneNaive(naive, onto, useAgrep = FALSE, ...)
```

Arguments

`naive`	character(1)
`onto`	an instance of ontologyIndex::ontology_index
`useAgrep`	logical(1) if TRUE, agrep will be used
`...`	passed to agrep if used

Value

if a match is found, the result of grep/agrep with value=TRUE is returned; otherwise a named NA_character_ is returned

named vector, names are ontology identifiers, values are matched strings

Examples

```
co = ontoProc::getCellOnto()
mapOneNaive("astrocyte", co)
```

`minicorpus` *minicorpus: a vector of annotation strings found in 'study title' of SRA metadata.*

Description

minicorpus: a vector of annotation strings found in 'study title' of SRA metadata.

Usage

```
minicorpus
```

Format

character vector

Note

arbitrarily chosen from titles of RNA-seq studies for taxon 9606

Source

NCBI SRA

Examples

```
data(minicorpus)
head(minicorpus)
```

`nomenCheckup` *repair nomenclature mismatches (to curated term set) in a vector of terms*

Description

repair nomenclature mismatches (to curated term set) in a vector of terms

Usage

```
nomenCheckup(cand, namedOffic, n = 1, tagcolname = "tag", ...)
```

Arguments

`cand`	character vector of candidate terms
`namedOffic`	named character vector of curated terms, the names are regarded as tags, intended to be identifiers in curated ontologies
`n`	numeric(1) number of nearest neighbors to return
`tagcolname`	character(1) prefix used to name columns for tags in output
`...`	passed to `adist`

Value

a data.frame instance with 2n+1 columns (column 1 is candidate, remaining n pairs of columns are (term, tag) for n nearest neighbors as measured by `adist`.

Examples

```
candidates = c("JHH7", "HUT102", "HS739T", "NCIH716")
# the candidates are cell line names returned in the text dump from
# https://portals.broadinstitute.org/ccle/page?gene=AHR
# note that one must travel to the third nearest neighbor
# to find the match (and tag) for Hs 739.T
# in this example, we compare to cell line names in Cell Line Ontology
nomenCheckup(candidates, cleanCLOnames(), n=3, tagcolname="clo")
```

`onto_plot2` *high-level use of graph/Rgraphviz for rendering ontology relations*

Description

high-level use of graph/Rgraphviz for rendering ontology relations

Usage

```
onto_plot2(ont, terms2use, cex = 0.8, ...)
```

Arguments

`ont`	instance of ontology from ontologyIndex
`terms2use`	character vector
`cex`	numeric(1) defaults to .8, supplied to Rgraphviz::graph.par
`...`	passed to onto_plot of ontologyPlot

Examples

```
cl = getCellOnto()
cl3k = c("CL:0000492", "CL:0001054", "CL:0000236", "CL:0000625",
   "CL:0000576", "CL:0000623", "CL:0000451", "CL:0000556")
onto_plot2(cl, cl3k)
```

`packDesc2019` *packDesc2019: overview of ontoProc resources*

Description

packDesc2019: overview of ontoProc resources

Usage

```
packDesc2019
```

Format

data.frame instance

Note

Brief survey of functions available to load serialized ontology_index instances imported from OBO.

Examples

```
head(packDesc2019)
```

`PROSYM` *PROSYM: HGNC symbol synonyms for PR (protein ontology) entries identified in Cell Ontology*

Description

PROSYM: HGNC symbol synonyms for PR (protein ontology) entries identified in Cell Ontology

Usage

```
PROSYM
```

Format

data.frame instance

Note

This is a snapshot of the synonyms component of an extract_tags='everything' import of PR. The 'EXACT.*PRO-short.*:DNx' pattern is used to retrieve HGNC symbols. See ?getPROnto for more provenance information.

Source

OBO Foundry

Examples

```
data(PROSYM)
head(PROSYM)
```

`recognizedPredicates` *enumerate ontological relationships used in ontoProc utilities*

Description

enumerate ontological relationships used in ontoProc utilities

Usage

```
recognizedPredicates()
```

Value

character vector, names of elements are abbreviated tokens that may be used in code

Examples

```
head(recognizedPredicates())
```

`secLevGen` *simple generation of children of 'choices' given as terms, returned as TermSet*

Description

simple generation of children of 'choices' given as terms, returned as TermSet

Usage

```
secLevGen(choices, ont)
```

Arguments

`choices`	vector of terms
`ont`	instance of ontology_index (S3) from ontologyIndex package

Value

TermSet instance

Examples

```
efoOnto = getEFOOnto()
secLevGen( "disease", efoOnto )
```

`selectFromMap` *select a set of elements from a term ’map’ and return a contribution to a data.frame*

Description

select a set of elements from a term ’map’ and return a contribution to a data.frame

Usage

```
selectFromMap(namedvec, index)
```

Arguments

`namedvec`	named character vector, as returned from `mapOneNaive`
`index`	numeric() or integer(), typically of length one

Value

a data.frame; if `index` does not inherit from `numeric`, a data.frame of one row with columns ’ontoid’ and ’term’ populated with `NA_character_` is returned, otherwise a similarly named data.frame is returned with contents from the selected elements of namedvec

Examples

```
co = ontoProc::getCellOnto()
mast = mapOneNaive("astrocyte", co)
selectFromMap(mast, 1)
```

`seur3kTab` *tabulate the basic outcome of PBMC 3K tutorial of Seurat*

Description

tabulate the basic outcome of PBMC 3K tutorial of Seurat

Usage

```
seur3kTab()
```

Value

a data.frame

Examples

```
seur3kTab()
```

`siblings_TAG` *generate a TermSet with siblings of a given term, excluding that term by default*

Description

generate a TermSet with siblings of a given term, excluding that term by default

acquire the label of an ontology subject tag

acquire the labels of children of an ontology subject tag

Usage

```
siblings_TAG(Tagstring = "EFO:1001209", ontology, justSibs = TRUE)

label_TAG(Tagstring = "EFO:0000311", ontology)

children_TAG(Tagstring = "EFO:1001209", ontology)
```

Arguments

`Tagstring`	a character(1) that identifies a term
`ontology`	instance of ontology_index (S3) from ontologyIndex
`justSibs`	character(1)

Value

TermSet instance

character(1)

TermSet instance

Note

for `label_TAG`, `Tagstring` may be a vector

Examples

```
efoOnto = getEFOOnto()
siblings_TAG( "EFO:1001209", efoOnto )
efoOnto = getEFOOnto()
label_TAG( "EFO:0000311", efoOnto )
efoOnto = getEFOOnto()
children_TAG( ontology = efoOnto )
```

`stopWords` *stopWords: vector of stop words from xpo6.com*

Description

stopWords: vector of stop words from xpo6.com

Usage

```
stopWords
```

Format

character vector

Note

"Stop words" are english words that are assumed to contribute limited semantic value in the analysis of free text.

Source

`http://xpo6.com/list-of-english-stop-words/`

Examples

```
data(stopWords)
head(stopWords)
```

`sym2CellOnto` *use Cell Ontology and Protein Ontology to identify cell-type defining conditions in which a given gene is named*

Description

use Cell Ontology and Protein Ontology to identify cell-type defining conditions in which a given gene is named

Usage

```
sym2CellOnto(sym, cl, pr)
```

Arguments

`sym`	gene symbol, must be used in protein ontology as a PRO:DNx exact match token
`cl`	result of getCellOnto()
`pr`	result of getPROnto()

Value

DataFrame if any hits are found. A field 'cond' abbreviates the identified conditions: (has/lacks)PMP (plasma membrane part) (hi/lo)PMAmt (plasma membrane amount), (has/lacks)Part.

Note

Currently just checks for *plasma_membrane_part, *plasma_membrane_amount, and *Part conditions.

Examples

```
if (!exists("cl")) cl = getCellOnto()
if (!exists("pr")) pr = getPROnto()
sym2CellOnto("ITGAM", cl, pr)
sym2CellOnto("FOXP3", cl, pr)
```

`TermSet-class` *manage ontological data with tags and a DataFrame instance*

Description

manage ontological data with tags and a DataFrame instance

abbreviated display for TermSet instances

Usage

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

Arguments

`object`	instance of TermSet class

Value

instance of TermSet

Examples

```
efoOnto = getEFOOnto()
defsibs = siblings_TAG("EFO:1001209", efoOnto)
class(defsibs)
defsibs
```

Index