Package 'rols'

March 26, 2024

Type Package

Title An R interface to the Ontology Lookup Service

Version 2.30.2

Description The rols package is an interface to the Ontology Lookup Service (OLS) to access and query hundred of ontolgies directly from R.

Depends methods

Imports httr2, jsonlite, utils, Biobase, BiocGenerics (>= 0.23.1)

Suggests GO.db, knitr (>= 1.1.0), BiocStyle (>= 2.5.19), testthat, lubridate, DT, rmarkdown,

biocViews ImmunoOncology, Software, Annotation, MassSpectrometry, GO

VignetteBuilder knitr

License GPL-2

Encoding UTF-8

URL http://lgatto.github.io/rols/

BugReports https://github.com/lgatto/rols/issues

Collate AllClasses.R AllGenerics.R utils.R Ontologies.R Terms.R cvparam.R OlsSearch.R Properties.R zzz.R

RoxygenNote 7.3.0

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

git_branch RELEASE_3_18

git_last_commit 8b7efef

git_last_commit_date 2024-02-16

Repository Bioconductor 3.18

Date/Publication 2024-03-25

Author Laurent Gatto [aut, cre], Tiage Chedraoui Silva [ctb], Andrew Clugston [ctb]

Maintainer Laurent Gatto <laurent.gatto@uclouvain.be>

R topics documented:

`CVParam` *Controlled Vocabulary*

Description

'CVParam' objects instantiate controlled vocabulary entries.

Usage

```
CVParam(label, name, accession, value, exact = TRUE)

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

## S4 method for signature 'CVParam'
rep(x, times)
```

Arguments

`label`	'character(1)' with the ontology label. If missing, a user-defined parameter is created.
`name`	'character(1)' with the name of the 'CVParam' to be constructed. This argument can be omitted if 'accession' is used and 'label' is not missing.
`accession`	'character(1)' with the accession of the 'CVParam' to be constructed. This argument can be omitted if 'name' is used. Ignored for user-defined instances.
`value`	'character(1)' with the value of the 'CVParam' o be constructed. This argument is optional.
`exact`	'logical(1)' defining whether the query to retrieve the 'accession' (when 'name' is used) should be an exact match.
`object`	'CVParam' object.
`x`	'CVParam' to be repeated.
`times`	'numeric(1)' defining the number of repetitions.

Methods

- ‘charIsCVParam(x)‘ checks if ‘x‘, a character of the form ‘"[ONTO, ACCESSION, NAME, VALUE]"‘, is a valid (possibly user-defined) ‘CVParam‘. ‘"ONTO"‘ is the ontology label (prefix), ‘"ACCESSION"‘ is the term accession number, ‘"NAME"‘ is the term’s name and ‘"VALUE"‘ is the value. Note that only the syntax validity is verified, not the semantics. See example below.

- ‘coerce(from = "CVParam", to = "character")‘ coerces ‘CVParam‘ ‘from‘ to a ‘character‘ of the following form: ‘[label, accession, name, value]‘. ‘as.character‘ is also defined.

- ‘coerce(from = "character", to = "CVParam")‘ coerces ‘character‘ ‘from‘ to a ‘CVParam‘. ‘as.CVParam‘ is also defined. If a ‘label‘ is absent, the ‘character‘ is converted to a User param, else, the ‘label‘ and ‘accession‘ are used to query the Ontology Lookup Service (see [OlsSearch()]). If a ‘name‘ is provided and does not match the retrieved name, a warning is thrown.

This function is vectorised; if the ‘from‘ character is of length greater than 1, then a list of ‘CVParam‘ is returned. The queries to the OLS are processed one-by-one, though.

Author(s)

Laurent Gatto

Examples

```
## User param
CVParam(name = "A user param", value = "the value")
## CVParam ESI from PSI's Mass Spectrometry ontology
Term("MS", "MS:1000073")
(esi <- CVParam(label = "MS", accession = "MS:1000073"))
class(esi)

## From a CVParam object to a character
cv <- as(esi, "character")
cv ## note the quotes

## From a character object to a CVParam
as(cv, "CVParam")
as("[MS, MS:1000073, , ]", "CVParam") ## no name
as("[MS, MS:1000073, ESI, ]", "CVParam") ## name does not match
as(c(cv, cv), "CVParam") ## more than 1 character

x <- c("[MS, MS:1000073, , ]", ## valid CV param
       "[, , Hello, world]",   ## valid User param
       "[this, one is, not, valid]", ## not valid
       "[ , , , ]") ## not valid

stopifnot(charIsCVParam(x) == c(TRUE, TRUE, FALSE, FALSE))

## A list of expected valid and non-valid entries
rols:::validCVchars
rols:::notvalidCVchars
```

OlsSearch *Querying OLS*

Description

Searching the Ontology Lookup Service is done first by creating an object ‘OlsSearch‘ using the ‘OlsSearch()‘ constructor. Query responses are then retrieved with the ‘olsSearch()‘ function.

Usage

```
OlsSearch(
  q,
  ontology = "",
  type = "",
  slim = "",
  fieldList = "",
  queryFields = "",
  exact = FALSE,
  groupField = FALSE,
  obsoletes = FALSE,
  local = TRUE,
  childrenOf = "",
  rows = 20L,
  start = 0L
)

olsSearch(object, all = FALSE)

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

olsRows(object)

olsRows(object) <- value

allRows(object)
```

Arguments

q	‘characher(1)‘ containing the search query.
ontology	‘character()‘ defining the ontology to be queried. Default is the empty character, to search all ontologies.
type	‘character()‘ restricting the search to an entity type, one of ‘"class"‘, ‘"property"‘, ‘"individual"‘ or ‘"ontology"‘.
slim	‘character()‘ restricts the search to an particular set of slims by name.

fieldList	‘character()‘ specifcies the fields to return. The defaults are iri, label, short_form, obo_id, ontology_name, ontology_prefix, description and type. Default is ‘"""‘ for all fields.
queryFields	‘character()‘ specifcies the fields to query, the defaults are label, synonym, description, short_form, obo_id, annotations, logical_description, iri. Default is ‘"""‘ for all fields.
exact	‘logical(1)‘ defining if exact matches should be returned. Default is ‘FALSE‘.
groupField	‘logical(1)‘, set to ‘TRUE‘rue to group results by unique id (IRI).
obsoletes	‘logical(1)‘ defining whether obsolete terms should be queried. Default is ‘FALSE‘.
local	‘character(1)‘, default is ‘FALSE‘. Set to ‘TRUE‘ to only return terms that are in a defining ontology e.g. only return matches to gene ontology terms in the gene ontology, and exclude ontologies where those terms are also referenced
childrenOf	‘character()‘ to restrict a search to children of a given term. Supply a list of IRI for the terms that you want to search under.
rows	‘integer(1)‘ defining the number of query returns. Default is 20L. Maximum number of values returned by the server is 1000. To retrieve the next results, set ‘start‘ 1000. See examle below.
start	‘integer(1)‘ defining the results page. number. Default is 0L.
object	‘OlsSeach‘ result object.
all	‘logical(1)‘ Should all rows be retrieved. Default is ‘FALSE‘. Can also be set in the queary object directly with ‘allRows()‘.
value	replacement value

Author(s)

Laurent Gatto

Examples

```
## Many results across all ontologies
OlsSearch(q = "trans-golgi")

## Exact matches
OlsSearch(q = "trans-golgi", exact = TRUE)

## Exact match in the gene ontology (go or GO) only
OlsSearch(q = "trans-golgi", exact = TRUE, ontology = "go")
OlsSearch(q = "trans-golgi", exact = TRUE, ontology = "GO")

## Exact match in the GO and Uberon
OlsSearch(q = "trans-golgi", exact = TRUE,
          ontology = c("GO", "Uberon"))

## Testing different ESI queries
OlsSearch(q = "electrospray", ontology = "MS")
OlsSearch(q = "ionization", ontology = "MS")
OlsSearch(q = "electrospray ionization", ontology = "MS")
```

```
OlsSearch(q = "electrospray ionization", ontology = "MS", exact=TRUE)

## Request 5 results instead of 20 (default)
OlsSearch(q = "plasma,membrane", ontology = "go", rows = 5)
## Same as above
OlsSearch(q = "plasma membrane", ontology = "go", rows = 5)

## or, once the object was created
(res <- OlsSearch(q = "plasma,membrane", ontology = "go"))
olsRows(res) <- 5
res

## all results
res <- allRows(res)
res

res <- OlsSearch(q = "trans-golgi", ontology = "go", rows = 5)
res
res <- olsSearch(res)
res
as(res, "data.frame")
trms <- as(res, "Terms")
trms
termPrefix(trms)
termId(trms)

## Setting rows and start parameters
tg1 <- OlsSearch(q = "trans-golgi", rows = 5, start = 0) |>
                 olsSearch() |>
                 as("data.frame")
tg2 <- OlsSearch(q = "trans-golgi", rows = 5, start = 5) |>
                 olsSearch() |>
                 as("data.frame")
tg3 <- OlsSearch(q = "trans-golgi", rows = 10, start = 0) |>
                 olsSearch() |>
                 as("data.frame")

## The two consecutive small results are identical
## to the larger on.
identical(rbind(tg1, tg2), tg3)
```

Ontologies *Ontologies*

Description

The rols package provides an interface to PRIDE's Ontology Lookup Servive (OLS) and can be used to query one or multiple ontologies, stored as 'Ontology' and 'Ontologies' instances, and containing various information as provided by OLS.

Usage

```
## S4 method for signature 'missing'
Ontologies(object)

## S4 method for signature 'character'
Ontology(object)

## S4 method for signature 'Ontology'
Ontology(object)

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

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

## S4 method for signature 'character'
olsVersion(object)

## S4 method for signature 'Ontology'
olsVersion(object)

## S4 method for signature 'Ontologies'
olsVersion(object)

## S4 method for signature 'character'
olsLoaded(object)

## S4 method for signature 'Ontology'
olsLoaded(object)

## S4 method for signature 'Ontologies'
olsLoaded(object)

## S4 method for signature 'Ontology'
olsLinks(object)

## S4 method for signature 'Ontology'
olsConfig(object)

## S4 method for signature 'character'
olsUpdated(object)

## S4 method for signature 'Ontology'
olsUpdated(object)

## S4 method for signature 'Ontologies'
olsUpdated(object)
```

```
## S4 method for signature 'character'
olsPrefix(object)

## S4 method for signature 'Ontology'
olsPrefix(object)

## S4 method for signature 'Ontologies'
olsPrefix(object)

## S4 method for signature 'character'
olsDesc(object)

## S4 method for signature 'Ontology'
olsDesc(object)

## S4 method for signature 'Ontologies'
olsDesc(object)

## S4 method for signature 'character'
olsTitle(object)

## S4 method for signature 'Ontology'
olsTitle(object)

## S4 method for signature 'Ontologies'
olsTitle(object)

## S4 method for signature 'character'
olsStatus(object)

## S4 method for signature 'Ontology'
olsStatus(object)

## S4 method for signature 'Ontologies'
olsStatus(object)

## S4 method for signature 'character'
olsNamespace(object)

## S4 method for signature 'Ontology'
olsNamespace(object)

## S4 method for signature 'Ontologies'
olsNamespace(object)

## S4 method for signature 'character'
ontologyUrl(object)
```

```
## S4 method for signature 'Ontology'
ontologyUrl(object)

## S4 method for signature 'Ontologies'
lapply(X, FUN, ...)

## S4 method for signature 'Ontologies'
x[i, j = "missing", drop = "missing"]

## S4 method for signature 'Ontologies'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'Ontologies'
length(x)
```

Arguments

object	an instance of class ‘Ontologies‘ or ‘Ontology‘. For some functions, a ontology identifier is applicable.
X	‘Ontologies‘ object.
FUN	a ‘function‘ to be applied to each ‘Ontology‘ element of ‘X‘.
...	additional arguments passed to ‘FUN‘.
x	an ‘Ontologies‘ object.
i	index of elecements to subset.
j	ignored.
drop	ignored.

Details

Ontologies are referred to by their namespace, which is lower case: the Gene Onology is "go", the Mass spectrometry ontology is "ms", etc. The ontologies also have prefixes, which are upper case: the Gene Onology prefix "GO", the Mass spectrometry ontology prefix "MS". One exception to this rule is the Drosophila Phenotype Ontology, whose namespace and prefix are "dpo" and "FBcv" respectively (there might be more). This is particularly confusing as the FlyBase Controlled Vocabulary has "fbcv" and "FBcv" as namespace and prefix respectively.

When using a character to initialise an ontology or query a term, "fbcv" (this is case insensitive) will refer to the the FlyBase Controlled Vocabulary. The the Drosophila Phenotype Ontology will have to be referred as "dpo" (also case insensitive).

Constructors

Objects can be created in multiple ways. The [Ontologies()] function will initialise all available ontolgies as an ‘Ontologies‘ object, while a call to [Ontology()] with an ontology namespace or prefix as argument will initialise the ontology of interest as an ‘Ontology‘ instance.

‘Ontolgies‘ instances can be subset with ‘[‘ and ‘[[‘ (using their namespace, see Details) and iterated over with ‘lapply‘. ‘Ontolgies‘ can be converted into a simple ‘data.frame‘ containing the ontology

prefixes, namespaces and titles using ‘as(., "data.frame")‘. ‘Ontologies‘ can also be coerced to lists of ‘Ontology‘ ojects with ‘as(., "list")‘.

Accessors

- ‘olsDesc(object = "Ontology")‘ returns the description of an ontology. Also works for ‘Ontologies‘ objects and a ‘character‘ describing an ontology namespace or prefix (see Details).

- ‘olsPrefix(object = "Ontology")‘ retruns the prefix of an ontology. Also works for ‘Ontologies‘ objects and a ‘character‘ describing an ontology namespace or prefix (see Details).

- ‘olsVersion(object = "Ontology")‘ returns the version of the ontology. Also works with an a ‘character‘ defining an ontology namespace or prefix (see Details) or an object of class ‘Ontologies‘, in which case it returns a list of versions.

- ‘olsLoaded(object = "Ontology")‘ returns the loading date of the ontology. Also works with a ‘character‘ containing the ontology namespace or prefix (see Details) or an object of class ‘Ontologies‘.

- ‘olsUpdated(object = "Ontology")‘ returns the update date of the ontology. Also works with a ‘character‘ containing the ontology namespace or prefix (see Details) or an object of class ‘Ontologies‘.

- ‘olsStatus(object = "Ontology")‘ returns the status of the ontology. Also works with a ‘character‘ containing the ontology namespace or prefix (see Details) or an object of class ‘Ontologies‘.

- ‘olsTitle(object = "Ontology")‘ returns the title of an ontology. Also works with a ‘character‘ containing the ontology namespace or prefix (see Details) or an object of class ‘Ontologies‘.

- ‘olsNamespace(object = "Ontology")‘ returns the namespace of an ontology. Also works with a ‘character‘ containing the ontology namespace or prefix (see Details) or an object of class ‘Ontologies‘.

- ‘olsLinks(object = "Ontology")‘ returns a named ‘character‘ with hyperlink to the ontology itself, and other associated concepts such as its terms.

- ‘olsConfig(object = "Ontology")‘ returns a list of additional unstructured, partly redundant information about the ontology.

- ‘ontologyUrl(object = "Ontology") return the hyperlink to the ontology itself. It can also be used with a ‘character‘ defining the namespace or prefix of an ontology, in which case it is created from the base OLS API URL.

Ontology terms

Once an ontology has been created an an ‘Ontology‘ instance, all its terms can be requested using the ‘Terms()‘ constructor. See [Terms()] for details.

Author(s)

Laurent Gatto

Examples

```
############################
## All ontologies
```

```
(onts <- Ontologies())

############################
## Alzheimer's Disease Ontology (ADO)
## 1. From the ontologies object
(ado1 <- onts[['ado']])
## 2. Create from its namespace
(ado2 <- Ontology('ado')) ## also works with ADO

all.equal(ado1, ado2)

olsVersion(ado1)
olsPrefix(ado1)
olsNamespace(ado1)
olsTitle(ado1)
olsDesc(ado1)
olsLinks(ado1)
str(olsConfig(ado1))
```

`Properties` *Term Properties*

Description

Properties (relationships) between terms can be queries for complete [Ontology()] objects and [Term()]/[Terms()] instances, and the results are stored as objects of class ‘Property‘ or ‘Properties‘.

Usage

```
## S4 method for signature 'Ontology'
Properties(object)

## S4 method for signature 'character'
Properties(object)

## S4 method for signature 'Term'
Properties(object)

## S4 method for signature 'Terms'
Properties(object)

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

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

## S4 method for signature 'Properties'
length(x)
```

Arguments

object	First input object.
x	A ‘Properties‘ object.

Examples

```
## Term properties
trm <- Term("uberon", "UBERON:0002107")
trm
Properties(trm)

## Ontology properties
Properties('ado')
```

Terms *Ontology Terms*

Description

The ‘Term‘ class describes an ontology term. A set of terms are instantiated as a ‘Terms‘ class.

Usage

```
## S4 method for signature 'character'
Terms(object, pagesize = 1000, obsolete = NULL)

## S4 method for signature 'Ontology'
Terms(object, pagesize = 1000, obsolete = NULL)

## S4 method for signature 'character'
Term(object, id)

## S4 method for signature 'Ontology'
Term(object, id)

children(object)

parents(object)

ancestors(object)

descendants(object)

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

## S4 method for signature 'Terms'
```

```
show(object)

## S4 method for signature 'Term'
termSynonym(object)

## S4 method for signature 'Terms'
termSynonym(object)

## S4 method for signature 'Term'
isObsolete(object)

## S4 method for signature 'Terms'
isObsolete(object)

## S4 method for signature 'Term'
isRoot(object)

## S4 method for signature 'Terms'
isRoot(object)

## S4 method for signature 'Term'
termLabel(object)

## S4 method for signature 'Terms'
termLabel(object)

## S4 method for signature 'Term'
termId(object)

## S4 method for signature 'Terms'
termId(object)

## S4 method for signature 'Term'
termLinks(object)

## S4 method for signature 'Term'
termPrefix(object)

## S4 method for signature 'Terms'
termPrefix(object)

## S4 method for signature 'Term'
termDesc(object)

## S4 method for signature 'Terms'
termDesc(object)

## S4 method for signature 'Term'
```

```
termOntology(object)

## S4 method for signature 'Terms'
termOntology(object)

## S4 method for signature 'Term'
termNamespace(object)

## S4 method for signature 'Terms'
termNamespace(object)

## S4 method for signature 'Terms'
length(x)

## S4 method for signature 'Terms'
unique(x)

## S4 method for signature 'Terms'
x[i, j = "missing", drop = "missing"]

## S4 method for signature 'Terms'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'Terms'
lapply(X, FUN, ...)

as.Term.data.frame(x)

as.Terms.data.frame(x)
```

Arguments

object	generally an instance of class ‘Terms‘ or ‘Term‘. In some cases, an ontology identifier is applicable.
pagesize	‘numeric(1)‘, converted to an integer, defining the response page size. Default is 1000.
obsolete	‘NULL‘ or ‘logical(1)‘ defining whether obsolete terms (‘TRUE‘), current terms (‘FALSE‘) or all (‘NULL‘, default) should be returned.
id	‘character(1)‘ with the term’s identifier.
x	a ‘Terms‘ object.
i	index of elecements to subset.
j	ignored.
drop	ignored.
X	‘Terms‘ object.
FUN	a ‘function‘ to be applied to each ‘Term‘ element of ‘X‘.
...	additional arguments passed to ‘FUN‘.

Contructors

Objects can be created using the ‘Term()‘ and ‘Terms()‘ constructers. The latter is used with an object of class ‘Ontology‘ or a ‘character‘ describing a valid ontology prefix to download and instantiate all terms of an ontology of interest. The former takes an ‘Ontology‘ object (or an ontology prefix) and a term identifer to instantiate that specific term.

For any given ‘Term‘ object, the ‘children‘, ‘parents‘, ‘ancestors‘ and ‘descendants‘ terms can be generated with the ‘children()‘, ‘parents()‘, ‘ancestor()‘ and ‘descendants()‘ function. ‘Terms‘ instances can be subset with ‘[‘ and ‘[[‘ and iterated over with ‘lapply‘.

Accessors

- ‘isObsolete(object = "Term")‘ returns a ‘TRUE‘ if the term is obsolete, ‘FALSE‘ otherwise. Also works on ‘Terms‘ instances.

- ‘isRoot(object = "Term")‘ returns a ‘TRUE‘ if the term is a root term, ‘FALSE‘ otherwise. Also works on ‘Terms‘ instances.

- ‘termDesc(object = "Term")‘ returns a ‘character‘ with the term’s description. Also works on ‘Terms‘ instances.

- ‘termId(object = "Term")‘ returns a ‘character‘ with the term’s identifier. Also works on ‘Terms‘ instances.

- ‘termLabel(object = "Term")‘ returns a ‘character‘ with the term’s label. Also works on ‘Terms‘ instances.

- ‘termNamespace(object = "Term")‘ returns a ‘character‘ with the term’s namespace. Also works on ‘Terms‘ instances.

- ‘termOntology(object = "Term")‘ returns a ‘character‘ with the term’s ontology (where it was retrieved from). Also works on ‘Terms‘ instances.

- ‘termPrefix(object = "Term")‘ returns a ‘character‘ with the term’s (ontology) prefix (where it was retrieved from). Also works on ‘Terms‘ instances.

- ‘termSynonym(object = "Term")‘ returns a ‘character‘ with the term’s synpnym(s). Also works on ‘Terms‘ instances.

- ‘termLinks(object = "Term")‘ returns a named ‘character‘ with hyperlink to/from the term.

Related terms

- ‘children(object = "Term")‘ returns a new ‘Terms‘ instance with the ‘object‘’s children or ‘NULL‘ if there are no children.

- ‘parents(object = "Term")‘ returns a new ‘Terms‘ instance with the ‘object‘’s parents or ‘NULL‘ if there are no parents.

- ‘ancestors(object = "Term")‘ returns a new ‘Terms‘ instance with the ‘object‘’s ancestors or ‘NULL‘ if there are no ancestors.

- ‘descendants(object = "Term")‘ returns a new ‘Terms‘ instance with the ‘object‘’s descendants or ‘NULL‘ if there are no descendants.

Coercion

- ‘as(x, "data.fram")‘ coerces a ‘Term‘ or ‘Terms‘ instance into a ‘data.frame‘ of length 1 (for the former) or length ‘length(x)‘ for the latter. The result will contain the following columns: id, label, description of the term(s), their ontology, whether they are obsolete, have children or are root node, the first synonym only, their iri and whether they are defining the ontology. Any missing value will be reported as ‘NA‘.

Author(s)

Laurent Gatto

Examples

```
## Alzheimer's Disease Ontology (ADO)
(adoterms <- Terms('ado'))

## Focus on squamous epithelium
(trm <- adoterms[["UBERON:0006914"]])

## Accessors
termLabel(trm)
head(termLabel(adoterms))
termId(trm)
termDesc(trm)
termOntology(trm)
termNamespace(trm)
termSynonym(trm) ## none

## Related terms
children(trm)
descendants(trm) ## includes child

parents(trm)
ancestors(trm) ## includes parent

## A single term from an ontology
Term("ado", "ADO:0000090")
```

Index