Package ‘OmaDB’

March 21, 2024

Title  R wrapper for the OMA REST API
Version  2.18.0
Author  Klara Kaleb
Maintainer  Klara Kaleb <klara.kaleb18@ic.ac.uk>, Adrian Altenhoff <adrian.altenhoff@inf.ethz.ch>
Description  A package for the orthology prediction data download from OMA database.
Depends  R (>= 3.5), httr (>= 1.2.1), plyr(>= 1.8.4)
Imports  utils, ape, Biostrings, GenomicRanges, IRanges, methods, topGO, jsonlite
URL  https://github.com/DessimozLab/OmaDB
BugReports  https://github.com/DessimozLab/OmaDB/issues
Encoding  UTF-8
License  GPL-3
LazyData  true
Suggests  knitr, rmarkdown, testthat
VignetteBuilder  knitr
biocViews  Software, ComparativeGenomics, FunctionalGenomics, Genetics, Annotation, GO, FunctionalPrediction
RoxygenNote  7.1.1
git_url  https://git.bioconductor.org/packages/OmaDB
git_branch  RELEASE_3_18
git_last_commit  88c9a3a
git_last_commit_date  2023-10-24
Repository  Bioconductor 3.18
Date/Publication  2024-03-20
OmaDB-package

Description

OmaDB is a wrapper for the REST API for the Orthologous MAtrix project (OMA) which is a database for the inference of orthologs among complete genomes. For more details on the OMA project, see https://omabrowser.org/.
OmaDB functions

The package contains a range of functions that are used to query the database. Some of the main functions are listed below:

- getProtein()
- getHOG()
- getOMAGroup()
- getGenomePairs()
- getTaxonomy()
- mapSequence()
- annotateSequence()
- searchProtein()

In addition to these, OmaDB features a range of functions that are used to format the retrieved data into some commonly used Bioconductor objects using packages such as GenomicRanges, Biostrings, topGO and ggtree. Some of them are listed below:

- formatTopGO()
- getGRanges()

The above functions are described in more detail in the package vignette’s listed below:

- Get started with OmaDB
- Exploring Hierarchical orthologous groups with OmaDB
- Exploring Taxonomic trees with OmaDB
- Sequence Analysis with OmaDB

---

**annotateSequence**

*Map GO annotation to a sequence that is not available in the OMA Browser*

**Description**

This function obtain Gene Ontology annotation for a given sequence that does not need to exist in the OMA Browser so far. The query sequence will analysed and a fast homology detection approach based on kmers will be used to detect the closest sequences in OMA. GO annotations for these top hits will be used to annotated the query sequence.

**Usage**

`annotateSequence(query)`
Arguments
query the sequence to be annotated, it can be either a string or an AAString object from the Biostrings package

Value
a data.frame containing the GO annotation information of the most similar protein to the query sequence

Examples
annotateSequence(query='MNDPSLLGYPNVGPQQQQQHQAGLLGKGTPNALQQLHMQNHQLGPPGMLNSSDVHTSSNNSRQLLDQLANGLNLNMLNNNNNNNNNNNNNNNMGSGVMMNASTAAVNSIG ... SYINSAVYELINTGRVQIHQNGRDFGYMSEKNFSHNLALKSSYNCIGELPFTNFTPSFTDVIDYIWFSTHALRVRGLLGEVDPEYVSKFIGFPNDKFPSDHIPLLARFEFMKTNTGSKKV

formatTopGO
Format the GO annotations data

Description
The function to create a list of GO annotations that is compatible with topGO from protein objects in roma

Usage
formatTopGO(geneList, format)

Arguments
geneList the list of OmaDB protein objects or a dataframe of ontologies to be included in the analysis - this is where the GO annotations are extracted from.
format format for the data to be returned in - either 'GO2geneID' or 'geneID2GO'

Value
a list containing the GO2geneID or geneID2GO information

Examples
geneList = list(getProtein(id='YEAST01'),getProtein(id='YEAST03'))
annotations = formatTopGO(geneList,format='geneID2GO')
getAnnotation-deprecated

Get GO annotation for a sequence Function

Description
This function should no longer be used. Use instead annotateSequence.

Usage
getAnnotation(query)

Arguments
query the sequence to be annotated, it can be either a string or an AAString object from the Biostrings package

Value
a data.frame containing the GO annotation information linked to the query sequence

See Also
OmaDB-deprecated

getAttribute

Get the value for the Object Attribute

Description
The function to obtain the value for an object attribute.

Usage
getAttribute(obj, attribute)

Arguments
obj the object of interest
attribute the attribute of interest

Value
an value for a given object attribute

Examples
members = getAttribute(getOMAGroup(id = 'YEAST58'), 'members')
Description

The function to obtain the information available for a single entry in the database. This function should no longer be used. It has been divided into several functions: Use the following functions instead.

- `getProtein` to obtain proteins (former `type=protein`)
- `getGenome` to obtain genomes (former `type=genome`)
- `getOMAGroup` to obtain genomes (former `type=group`)

Usage

gData(type, id, attribute = NULL)

Arguments

- `type` the type for the entry to be returned - either protein, genome or group
- `id` an identifier for the entry to be returned. For more information, see the 'Get started with OmaDB' vignette.
- `attribute` an extra attribute

Value

an object containing the JSON keys as attributes

See Also

OmaDB-deprecated

Description

This function obtains the basic information for one specific genome available on the OMA Browser, or - if no id is provided - a dataframe with all available genomes.

Usage

gGetGenome(id = NULL, attribute = NULL)
getGenomeAlignment-deprecated

**Arguments**

- id: A genome identifier. By default, all available genomes will be returned.
- attribute: An extra attribute to be returned (proteins)

**Details**

Ids can be either the scientific name of a species, the NCBI taxonomy id or the UniProtKB mnemonic species code.

The optional argument attribute can be used to directly load the proteins belonging to the genome. Alternatively, you can access the proteins attribute of the result which will transparently load the proteins from the OMA Browser.

**Value**

an object containing the JSON keys as attributes or a dataframe

**Examples**

getGenome()
getGenome(id='HUMAN')
getGenome(id=9606)
getGenome(id='HUMAN',attribute='proteins')

getGenomeAlignment-deprecated

*Get Whole Genome Alignment Function*

**Description**

This function should no longer be used. Use instead getGenomePairs.

**Usage**

getGenomeAlignment(genome1, genome2, chr1 = NULL, chr2 = NULL, rel_type = NULL)

**Arguments**

- genome_id1: an identifier for the first genome, which can be either its taxon id or UniProt species code
- genome_id2: an an identifier for the second genome, which can be either its taxon id or UniProt species code
- chr1: the chromosome of interest for the first genome
- chr2: the chromosome of interest for the second genome
- rel_type: the pairs relationship type
- per_page: the number of instances to be returned or 'all'. default is set to a 100.
getGenomePairs

Description

This function retrieves the pairwise relations among two genomes from the OMA Browser database. The relations are orthologs in case the genomes are different and "close paralogs" and "homoeologs" in case they are the same.

Usage

getGenomePairs(genome_id1, genome_id2, chr1 = NULL, chr2 = NULL, rel_type = NULL, ...)

Arguments

gene_id1: an identifier for the first genome, which can be either its taxon id or UniProt species code.
gene_id2: an identifier for the second genome, which can be either its taxon id or UniProt species code.
chr1: the chromosome of interest for the first genome.
chr2: the chromosome of interest for the second genome.
rel_type: the pairs relationship type.
...
qwargs

Details

By using the parameters chr1 and chr2, one can limit the relations to a certain chromosome for one or both genomes. The id of the chromosome corresponds to the chromosome ids from the getGenome result.

The rel_type parameter further limits the returned relations to a specific subtype of orthologs (i.e. "1:1", "1:n", "m:1", "m:n") or - within a genome to either "close paralogs" or "homeologs".

Value

A dataframe containing information about both the entries in the orthologous pair and their relationship.
**getHOG**

**Examples**

```r
getGenomePairs(genome_id1='YEAST', genome_id2='ASHGO')
```

---

**getHOG**

*Retrieve a HOG from the OMA Browser*

**Description**

The function retrieves a specific Hierarchical Orthologous Group (HOG) from the OMA Browser database. A HOG is a set of genes that have all descended from a single ancestral gene at a specific taxonomic level.

**Usage**

```r
getHOG(id, level = NULL, members = FALSE)
```

**Arguments**

- **id**
  - an identifier for the HOG to be returned - either its HOG ID or a protein id.
- **level**
  - a specific level for the HOG to be restricted to. level can either be 'root', or the name of a taxonomic level that is part of the HOG, e.g. 'Fungi'. By default it will retrieve the deepest level of the most specific subhog for the given ID.
- **members**
  - boolean that when set to TRUE returns a dataframe containing the protein members at a given hog level.

**Details**

A HOG can be identified by its member proteins and a taxonomic level, or a HOG ID. As a taxonomic level, you can use either 'root' to retrieve the HOG at its deepest level, or the name of NCBI taxonomy level, or leave it out in which case the deepest level that doesn’t include a duplication node is used.

The function either returns a single hog object or a list of hog objects. The later happens if the HOG ID you provide has already split into several sub-hogs at the level you indicate.

**Value**

an object containing HOG attributes, or a list of those

**Examples**

```r
getHOG(id = 'YEAST590')
getHOG(id = 'YEAST590', level='root')
getHOG(id = 'YEAST590', level='Saccharomycetaceae', members=TRUE)
```
getLocus

Get loci for a given list of proteins

Description
Function to obtain loci in genomic range format for a given list of proteins

Usage
getLocus(proteins)

Arguments
proteins the dataframe or a list of dataframes containing the protein data of interest. this can either be the members df or a list of protein ids.

Value
genomic range object from the GenomicRanges package in Bioconductor

Examples

loci = getLocus(proteins = getOMAGroup('YEAST58')['members'])

getObjectAttributes

Get the Object Attributes

Description
The function to obtain the attributes and their data types for the object created.

Usage
getObjectAttributes(obj)

Arguments
obj the object of interest

Value
an list of object attributes and their data classes

Examples

attributes = getObjectAttributes(getOMAGroup(id = 'YEAST58'))
getOMAGroup

Retrieves an OMA Group from the OMA Browser

Description

This function obtains an OMA Group from the OMA Browser database. An OMA Group is defined to be a clique of proteins that are all orthologous to each other, i.e. they are all related through speciation events only. An OMA Group can thus by definition not contain any inparalogs. It is a very stringent orthology grouping approach. OMA Groups are mostly useful to infer phylogenetic species tree where they can be used as marker genes.

Usage

getOMAGroup(id, attribute = NULL)

Arguments

id	An identifier for the group. See above for possible types of IDs.
attribute	an extra attribute to be returned (close_groups)

Details

Retrieving an OMA Group can be done using a group nr as id, its fingerprint (a 7mer AA sequence which is unique to proteins in that group), a member protein id or any sequence pattern that is unique to the group.

Value

an object containing the JSON keys as attributes or a dataframe

Examples

getOMAGroup(id='58')
getOMAGroup(id='P12345')
getOMAGroup(id='NNRRGRI')
getOMAGroup(id='58', attribute='close_groups')

getProtein

Retrieves a protein from the OMA Browser

Description

This function enables to retrieve information on one or several proteins from the OMA Browser database.
Usage

getProtein(id, attribute = NULL)

Arguments

id Identifier(s) for the entry or entries to be returned. a character string if single entry or a vector if multiple.
attribute Instead of the protein, return the attribute property of the protein. Attribute needs to be one of ’domains’, ’orthologs’, ’gene_ontology’, ’locus’, or ’homoeologs’.

Details

In its simplest form the function returns the base data of the query protein. The query protein can be selected with any unique id, for example with a UniProtKB accession (P12345), an OMA id (YEAST00012), or a RefSeq id (NP_001226). To retrieve more than one protein, you should pass a vector of IDs.

Non-scalar properties of proteins such as their domains, GO annotations, orthologs or homeologs will get loaded upon accessing them, or if you only need this information you can set the attribute parameter to the property name and retrieve this information directly.

Value

An object containing the JSON keys as attributes or a dataframe containing the non-scalar protein property.

See Also

For non-unique non-unique IDs or partial ID lookup, use searchProtein instead.

Examples

getProtein(id='YEAST00001')
getProtein(id='YEAST00001', attribute='orthologs')
getProtein(id=c('YEAST00001','YEAST00002','YEAST00012'))
getProtein(id=c('YEAST00001','YEAST00002','YEAST00012'), attribute='gene_ontology')

getTaxonomy  
Get the Taxonomic tree function

Description

The function to obtain the taxonomic tree from the database in the newick format that can be plugged into phylo.io for visualisation.

Usage

getTaxonomy(root = NULL, members = NULL, newick = TRUE)
getTopGO

Arguments

- root: optional parameter, the root of the node of interest
- members: optional parameter, list of member ncbi taxon or UniProt IDs that should be included in the induced taxonomy.
- newick: optional parameter, boolean default set to TRUE

Value

an object containing the JSON keys as attributes

Examples

getTaxonomy()
getTaxonomy(members='YEAST,ASHGO')
getTaxonomy(root='Alveolata')

geneList = list(getProtein(id='YEAST58'),getProtein(id='YEAST00059'))
annotations = formatTopGO(geneList,format='geneID2GO')
library(topGO)
getTopGO(annotations, foregroundGenes = list('YEAST00058'), format = 'geneID2GO', ontology = 'BP')
getTree  

Get the Tree Object

Description
A convenience function to obtain a tree object from newick tree, essentially wraps read.tree from the ape package.

Usage
getTree(newick)

Arguments
newick The newick tree to be instantiated.

Value
a tree object

Examples
taxonomy = getTaxonomy(root='Alveolata')
getTree(newick=taxonomy$newick)

getVersion  

Get the API and database version function

Description
The function to obtain the API and database version that the package is using.

Usage
getVersion()

Value
S3 object

Examples
getVersion()
getXref-deprecated

Get the CrossReferences in the OMA database for a pattern

Description
This function is should no longer be used. Use instead searchProtein.

Usage
getXref(pattern)

Arguments
pattern the pattern to query the OMA database with - needs to be at least 3 characters long

Value
a data.frame containing information on the cross references for a given pattern

See Also
OmaDB-deprecated

group An example OMA group object.

Description
An object containing information for the OMA group number 737636.

Usage
group

Format
An S3 object with 4 variables:
group_nr group number, not stable across releases
fingerprint fingerprint of the oma group, stable across releases
related_groups url to the endpoint containing the list of oma groups that share some of the orthologs with this oma group
members list of protein members of this oma group ...

Source
https://omabrowser.org/api/group/YEAST58/
mapSequence

Description

Map the Protein Sequence Function

Usage

mapSequence(query, search = NULL, full_length = FALSE)
**Arguments**

- **query**: the sequence to be searched, it can be either a string or an AAString object from the Biostrings package.
- **search**: argument to choose search strategy. Can be set to 'exact', 'approximate' or 'mixed'. Defaults to 'mixed', meaning first tries to find exact match. If no target can be found, uses approximate search strategy to identify query sequence in database.
- **full_length**: a boolean indicating whether or not for exact matches, the query sequence must be matching the full target sequence. By default, a partial exact match is also reported as exact match.

**Value**

a data.frame containing the information of matches for the query sequence

**Examples**

```r
mapSequence(query = 'MNDPSLLGYPNVGQQQQQQHQAGLLKGTNALQQQHLMNQLTGIPPPGLMNNSVHSSLNSQRLDLQLANGNAMLNMMDNNNNNNNNNNNNNGGGSGVMMNASTAAVNSIG ... SYINSAVYELINTGRVQIHQEGNGRDFGYMSEKNFSHNLALKSSYNCIGELPFTNFTPSFTDVIDYIWFSTHALRVRGLLGEVDPEYVSKFIGFPNDKFPSDHIPLLARFEFMKTNTGSKKV')
mapSequence(search = 'mixed', query = 'NKLLQPTDFQSHIAEASKSLVDCTKQALMEMADTLTDSKTAKKQQTGDTPSGTATNSAVSTPLTPKIELFANGH')
```

---

**Description**

These functions are provided for compatibility with older versions of **OmaDB** only, and will be defunct at the next release.

**Usage**

- `getXref(pattern)`
- `getAnnotation(query)`
- `getGenomeAlignment(genome1, genome2, chr1 = NULL, chr2 = NULL, rel_type = NULL)`
- `getData(type, id, attribute = NULL)`

**Details**

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- `getXref`

  For `getXref`, use `searchProtein`. 

---

OmaDB-deprecated  Deprecated functions in package OmaDB.
orthologs

getAnnotation

For getAnnotation, use annotateSequence.

geneAlignment

For getGenomeAlignment, use getGenomePairs.

data

For getData, use getProtein, getGenome, getOMAGroup.

| orthologs | An example orthologs object. |

**Description**

A dataframe containing information for the orthologs of protein YEAST00058.

**Usage**

orthologs

**Format**

A dataframe object with 15 variables:

- entry_nr entry number of the ortholog
- omaid oma identifier of the ortholog
- canonicalid canonicalid of the ortholog
- sequence_md5 sequence_md5 of the ortholog
- oma_group oma_group of the ortholog
- oma_hog_id hog id of the ortholog
- chromosome chromosomal location of the ortholog
- locus.start start locus of the ortholog
- locus.end end locus of the ortholog
- locus.strand locus strand of the ortholog
- is_main_isofom true/false
- rel_type relationship type of the ortholog to the gene
- distance ortholog distance
- score ortholog score ...

**Source**

https://omabrowser.org/api/protein/YEAST00058/orthologs
pairs

An example genome alignment object.

Description
A dataframe containing information for the whole genome alignment of YEAST and ASHGO.

Usage
pairs

Format
A dataframe object with 12 variables for each member of the pair, as well some 3 additional variables:

- **entry_nr**  entry number of the ortholog
- **omaid**  oma identifier of the ortholog
- **canonicalid**  canonicalid of the ortholog
- **sequence_md5**  sequence_md5 of the ortholog
- **oma_group**  oma_group of the ortholog
- **oma_hog_id**  hog id of the ortholog
- **chromosome**  chromosomal location of the ortholog
- **locus.start**  start locus of the ortholog
- **locus.end**  end locus of the ortholog
- **locus.strand**  locus strand of the ortholog
- **is_main_isoform**  true/false
- **rel_type**  relationship type of the ortholog to the gene
- **distance**  ortholog distance
- **score**  ortholog score ...

Source
https://omabrowser.org/api/pairs/YEAST/ASHGO/
An example protein object.

**Description**

An object containing information for the YEAST00058 protein.

**Usage**

`protein`

**Format**

A S3 object with 23 variables:

- `entry_nr` entry number of the protein
- `entry_url` url pointer to the protein
- `omaid` oma identifier of the protein
- `canonicalid` canonicalid of the protein
- `sequence_md5` sequence_md5 of the protein
- `oma_group` oma_group of the protein
- `oma_hog_id` hog id of the protein
- `chromosome` chromosomal location of the protein
- `locus` GRanges object with the locus information for the protein
- `is_main_isoform` true/false
- `roothog_id` root taxonomic level of the relevant hog
- `roothog_id` taxonomic levels of the hog in which the protein is present
- `sequence_length` length of the protein sequence
- `sequence` AAString of the protein sequence
- `cdna` DNAString of the protein sequence
- `domains` url pointer to the list of protein domains
- `xref` url pointer to the list of protein cross references
- `orthologs` url pointer to the list of protein orthologs
- `homeologs` url pointer to the list of protein homeologs
- `gene_ontology` url pointer to the list of protein GO ontologies
- `oma_group_url` url pointer to the protein oma group
- `oma_hog_members` url pointer to the protein hog members
- `alternative_isoforms_urls` list of url pointers to the protein isoforms ...

**Source**

[https://omabrowser.org/api/protein/6633022/](https://omabrowser.org/api/protein/6633022/)
**resolveURL**

*Load data for a given url from the OMA Browser API.*

**Description**

This function is usually not needed by users. In most circumstances an attribute containing a URL is automatically loaded when accessed. However, in case the data is transformed into a dataframe, this will no longer be true, in which case one can access the data behind this attribute using this function.

**Usage**

`resolveURL(url)`

**Arguments**

- `url` The url of interest

**Value**

a data.frame containing the information behind an URL

**Examples**

```r
resolveURL('http://omabrowser.org/api/protein/YEAST58/gene_ontology/
```

---

**searchProtein**

*Get the CrossReferences in the OMA database for a pattern*

**Description**

The function to list all the crossreferences that match a certain defined pattern.

**Usage**

`searchProtein(pattern)`

**Arguments**

- `pattern` the pattern to query the OMA database with - needs to be at least 3 characters long

**Value**

a data.frame containing information on the cross references for a given pattern
Examples

searchProtein(pattern='MAL')

sequence_annotation  An example dataframe containing GO annotations identified from a given sequence.

Description

An example dataframe containing GO annotations identified from a given sequence.

Usage

sequence_annotation

Format

A dataframe with 13 variables:

- **Qualifier**: qualifier of the annotation
- **GO_ID**: GO term for the annotation
- **GO term for the annotation**
- **Evidence**: evidence for the annotation
- **Date**: date
- **DB_Object_Type**: identified object type
- **DB_Object_Name**: identified object name
- **Aspect**: aspect
- **Assigned_By**: assignment of the annotation
- **GO_name**: GO term name
- **DB**: database
- **DB.Reference**: database reference
- **Synonym**: synonym ...

Source

https://omabrowser.org/api/function/?query=MNDPSLLGYPNVGPQQQQQQQQQHQAGLLGTPNALQQLHMNQLTGIPPPGLMNNSDVHTSSNNNSRQLLDQLANGN ... SANIFKYDFLTRLYLNGNSLTELPAEIKNLSNLRVLDLSHNRLTSLPAELGSCFQLKYFYFFDNMVTTLPWEFGNLCNLQFLGVEGNPLEKQFLKILTEKSVTGLIFYLRDNRPEIPLPHER
**sequence_map**

An example dataframe containing proteins identified from a given sequence.

**Description**

An example dataframe containing proteins identified from a given sequence.

**Usage**

sequence_map

**Format**

A dataframe with 3 variables:

- **query**: sequence that was queried
- **identified_by**: type of identification
- **targets**: list of protein targets identified ...

**Source**

https://omabrowser.org/api/sequences/?query=MNDPSLLGYPNVGQQQQQQQQQHQAGLLGKPANALQQQQHMLQNLGTIPGLMPM

---

**setAPI**

*Set the url to the OMA Browser API*

**Description**

Function to set the base url to the OMA Browser API. If no url is specified, the default OMA Browser API url is used.

**Usage**

setAPI(url)

**Arguments**

- **url**: Base url to the API
taxonomy

An example newick format taxonomy object.

Description
An example newick format taxonomy object.

Usage
taxonomy

Format
An S3 with 2 variables:

- **root_taxon** sequence that was queried
- **newick** taxonomy newick ...

Source
https://omabrowser.org/api/taxonomy/Alveolata/?type=newick

xref

An example xref object.

Description
An example xref object.

Usage
xref

Format
A dataframe with 8 variables:

- **xref** cross reference
- **source** source of the cross reference
- **entry_nr** oma database entry number
- **oma_id** oma id of the cross reference
- **genome.code** genome_id of the cross reference
- **genome.taxon_id** taxon_id of the cross reference
- **genome.species** species of the cross reference
- **genome.genome_url** genome url pointer of the cross reference ...
Source

https://omabrowser.org/api/xref/?search=MAL

$.omadb_obj

Resolve URLs automatically when accessed

Description

The function to obtain further information from a given url.

Usage

```r
## S3 method for class 'omadb_obj'
x$name
```

Arguments

- `x`: object
- `name`: attribute

Value

API response behind the URL
# Index

**datasets**
- group, 15
- hog, 16
- orthologs, 18
- pairs, 19
- protein, 20
- sequence_annotation, 22
- sequence_map, 23
- taxonomy, 24
- xref, 24

**internal**
- getAnnotation-deprecated, 5
- getData-deprecated, 6
- getGenomeAlignment-deprecated, 7
- getXref-deprecated, 15
- OmaDB-deprecated, 17
- $.omadb_obj, 25

annotateSequence, 3, 5, 18

formatTopGO, 4

getAnnotation (OmaDB-deprecated), 17
getAnnotation-deprecated, 5
getAttribute, 5
getData (OmaDB-deprecated), 17
getData-deprecated, 6
getGenome, 6, 6, 8, 18
getGenomeAlignment (OmaDB-deprecated),
    17
getGenomeAlignment-deprecated, 7
getGenomePairs, 7, 8, 18
getHOG, 9
getLocus, 10
getObjectAttributes, 10
getOMAGroup, 6, 11, 18
getProtein, 6, 11, 18
getTaxonomy, 12
getTopGO, 13
getTree, 14

getVersion, 14
getXref (OmaDB-deprecated), 17
getXref-deprecated, 15

hog, 16

mapSequence, 16

OmaDB (OmaDB-package), 2
OmaDB-deprecated, 17
OmaDB-package, 2
orthologs, 18

pairs, 19

protein, 20

resolveURL, 21

searchProtein, 12, 15, 17, 21
sequence_annotation, 22
sequence_map, 23
setAPI, 23

taxonomy, 24

xref, 24