Package ‘OmaDB’

April 16, 2019

Title R wrapper for the OMA REST API

Version 1.2.2

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

License GPL-2

LazyData true

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

biocViews Software, ComparativeGenomics, FunctionalGenomics, Genetics, Annotation, GO, FunctionalPrediction

RoxygenNote 6.0.1.9000

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

git_branch RELEASE_3_8

git_last_commit 3696142

git_last_commit_date 2019-01-04

Date/Publication 2019-04-15

R topics documented:

`OmaDB-package` *OmaDB: A package for the orthology prediction data download from OMA database.*

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/oma/home/`.

OmaDB functions

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

- getData()
- getHOG()
- getGenomeAlignment()
- getTaxonomy()
- mapSequence()
- getAnnotation()
- getXref()

In addition to these, roma 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

`bulkRetrieve` *Bulk retrieve information for a list of proteins*

Description

The function to bulk retrieve information for a list of proteins.

Usage

```
bulkRetrieve(protein_list)
```

Arguments

`protein_list`	list of protein members

Value

a list of S3 objects

Examples

```
orthologs = getData(type="protein",id='YEAST58')$orthologs
bulkRetrieve(orthologs)
```

`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 roma protein objects 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(getData(type="protein",id="YEAST01"),getData(type="protein",id="YEAST03"))
annotations = formatTopGO(geneList,format="geneID2GO")
```

`getAnnotation` *Get GO annotation for a sequence Function*

Description

The function to obtain GO annotation for a given sequence.

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 containg the GO annotaition information linked to the query sequence

Examples

```
getAnnotation(query="MNDPSLLGYPNVGPQQQQQQQQQQHAGLLGKGTPNALQQQLHMNQLTGIPPPGLMNNSDVHTSSNNNSRQLLDQLANGNANMLNM
```

`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(getData("group","YEAST58"),'members')
```

getData *Get the Data Function*

Description

The function to obtain the information available for a single entry in the datase.

Usage

```
getData(type, id = NULL, 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

Examples

```
getData(type = "protein", id="YEAST00001")
getData(type = "group", id="YEAST00001")
```

getGenomeAlignment *Get Whole Genome Alignment Function*

Description

The function to obtain the list of orthologs for 2 whole genomes.

Usage

```
getGenomeAlignment(genome_id1, genome_id2, chr1 = NULL, chr2 = NULL,
  per_page = 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
per_page	the number of instances to be returned or 'all'. default is set to a 100.
rel_type	the pairs relationship type

Value

a dataframe containing information about both the entries in the orthologous pair and their relationship

Examples

```
getGenomeAlignment(genome_id1="YEAST",genome_id2="ASHGO")
getGenomeAlignment(genome_id1="YEAST",genome_id2="ASHGO",chr1="1")
```

`getHOG` *Get the HOG Data Function*

Description

The function to obtain the information available for a Hierarchical orthologous group entry in the datase.

Usage

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

Arguments

`id`	an identifier for the entry to be returned - either its id or one of its protein members
`level`	a specific level for the HOG to be restricted to - set to the root level by default. A taxonomic level can be identified by its full capitalised name e.g. "Fungi" or "Saccharomycetaceae".
`members`	boolean that when set to TRUE returns a dataframe containg the protein members at a given hog and/or level

Value

an object containing the JSON keys as attributes

Examples

```
getHOG(id = "YEAST590")
getHOG(id = "YEAST590",level="Saccharomycetaceae", members=TRUE)
```

`getInfo` *Get further information for a dataframe of members*

Description

The function to obtain further information from a dataframe containing a list of members.

Usage

```
getInfo(df, type, format = NULL)
```

Arguments

`df`	the dataframe or a list of dataframes containing the genomic range data of interest
`type`	the type of information to be retrieved
`format`	currently only relevant to type = ontologies where it can be set to either "geneID2GO" or "GO2geneID". Default is "geneID2GO"

Value

an list

Examples

```
sequences = getInfo(df = getData("group","YEAST58")['members'],type='sequences')
```

`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(getData("group","YEAST58"))
```

`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, newick = TRUE)
```

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")
```

`getTopGO` *Get the topGO Object function*

Description

The function to create a topGO object containing the GO annotations for the given protein list.

Usage

```
getTopGO(annotations, format, myInterestingGenes)
```

Arguments

`annotations`	list of GO annoatations obtained from the formatTopGO()
`format`	format for the data to be returned in - either 'GO2geneID' or 'geneID2GO'
`myInterestingGenes`	list of identifiers for the genes of interest or a dataframe containing them

Value

topGO object

Examples

```
geneList = list(getData(type="protein",id="YEAST58"),getData(type="protein",id="YEAST00059"))
annotations = formatTopGO(geneList,format="geneID2GO")
library(topGO)
getTopGO(annotations, myInterestingGenes = list("YEAST00058"), format = "geneID2GO")
```

---

getTree *Get the Tree Object*

Description

The function to obtain the tree object from newick.

Usage

```
getTree(newick)
```

Arguments

`newick`	the newick of interest.

Value

an tree object

Examples

```
taxonomy = getTaxonomy(root="Alveolata")
getTree(newick=taxonomy$newick)
```

---

getXref *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

```
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

Examples

```
getXref(pattern="MAL")
```

`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/`

`hog` *An example HOG object.*

Description

An object containing information for the HOG:0273533.1b.

Usage

```
hog
```

Format

An S3 object with 8 variables:

hog_id hog identifier

level the taxonomic level of this hog

levels_url url pointer to the hog information at a given level

members_url url pointer to the list of gene members for this hog

alternative_members a dataframe object containing the rest of the taxonomic levels in this hog

roothog_id the root taxonomic level of this hog

parent_hogs a dataframe containing information on the parent hogs to the current hogs

children_hogs a dataframe containing information on the children hogs to the current hogs ...

Source

https://omabrowser.org/api/hog/HOG:0273533.1b/

`mapSequence` *Map the Protein Sequence Function*

Description

The function to identify a sequence.

Usage

```
mapSequence(query, search, 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

```
mapSequence(query="MNDPSLLGYPNVGPQQQQQQQQQQHAGLLGKGTPNALQQQLHMNQLTGIPPPGLMNNSDVHTSSNNNSRQLLDQLANGNANMLNMNMD
mapSequence(search="mixed",query="NKLLQPTDFQQSHIAEASKSLVDCTKQALMEMADTLTDSKTAKKQQPTGDSTPSGTATNSAVSTPLTPKIELF
```

`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_isoform 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 aligment 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/`

`protein` *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

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/

---

resolveURL *Get the Further Information behind the URL Function*

Description

The function to obtain further information from a given url.

Usage

```
resolveURL(url_field)
```

Arguments

`url_field`	the url of interest

Value

a data.frame containing the information behind an URL

Examples

```
resolveURL(url_field="http://omadev.cs.ucl.ac.uk/api/protein/YEAST58/ontology/")
```

---

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

With 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=MNDPSLLGYPNVGPQQQQQQQQQQHAGLLGKGTPNALQQQLHMNQLTGIPPPG`

`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=MNDPSLLGYPNVGPQQQQQQQQQQHAGLLGKGTPNALQQQLHMNQLTGIPPP` `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

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

Arguments

`x`	object
`name`	attribute

Value

API response behind the URL

Index