

# Package ‘OmaDB’

October 16, 2018

**Title** R wrapper for the OMA REST API

**Version** 1.0.1

**Author** Klara Kaleb

**Maintainer** Klara Kaleb <klarakaleb@outlook.com>, Adrian Al-  
tenhoff <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

**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\_7

**git\_last\_commit** e902bdf

**git\_last\_commit\_date** 2018-06-29

**Date/Publication** 2018-10-15

## R topics documented:

OmaDB-package . . . . .	2
formatTopGO . . . . .	3
getAnnotation . . . . .	3
getAttribute . . . . .	4
getData . . . . .	4
getGenomeAlignment . . . . .	5
getGRanges . . . . .	5
getHOG . . . . .	6
getObjectAttributes . . . . .	6
getOntologies . . . . .	7
getSequences . . . . .	7

getTaxonomy . . . . .	8
getTopGO . . . . .	8
getXref . . . . .	9
group . . . . .	9
hog . . . . .	10
mapSequence . . . . .	10
orthologs . . . . .	11
pairs . . . . .	12
protein . . . . .	13
resolveURL . . . . .	14
sequence_annotation . . . . .	14
sequence_map . . . . .	15
taxonomy . . . . .	15
xref . . . . .	16

## Index 17

---

OmaDB-package	<i>OmaDB: A package for the orthology prediction data download from OMA database.</i>
---------------	---------------------------------------------------------------------------------------

---

## 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

---

formatTopGO	<i>Format the GO annotations data</i>
-------------	---------------------------------------

---

**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="YEAST58"),getData(type="protein",id="YEAST00059"))
annotations = formatTopGO(geneList,format="geneID2GO")
```

---

getAnnotation	<i>Get GO annotation for a sequence Function</i>
---------------	--------------------------------------------------

---

**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 containing the GO annotation information linked to the query sequence

**Examples**

```
getAnnotation(query="MNDPSLLGYPNVGPQQQQQQQQHAGLLGKGTNPALQQQLHMNQLTGIPPPGLMNSDVHTSSNNNSRQLLDQLANGNANMLNM")
```

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 dataset.

**Usage**

```
getData(type, id = 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 Roma" vignette.

**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)
```

**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

**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")
```

---

getGRanges      *Get the GRanges object*

---

**Description**

The function to obtain the GRanges object from a dataframe containing a list of members.

**Usage**

```
getGRanges(df)
```

**Arguments**

df	the dataframe or a list of dataframes containing the genomic range data of interest
----	-------------------------------------------------------------------------------------

**Value**

an GRanges object

**Examples**

```
gRanges = getGRanges(df = getData("group", "YEAST58")['members'])
```

---

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)
```

---

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"))
```

---

getOntologies	<i>Get the gene ontologies from members</i>
---------------	---------------------------------------------

---

**Description**

The function to obtain the gene ontologies from a dataframe of members.

**Usage**

```
getOntologies(df)
```

**Arguments**

df                    the dataframe or a list of dataframes containing the member proteins of interest

**Value**

a list containing the geneID2GO information

**Examples**

```
ontologies = getOntologies(df = getData("group", "YEAST58")['members'])
```

---

getSequences	<i>Get the sequences from members</i>
--------------	---------------------------------------

---

**Description**

The function to obtain the protein sequences from a dataframe of members.

**Usage**

```
getSequences(df)
```

**Arguments**

df                    the dataframe or a list of dataframes containing the member proteins of interest

**Value**

a list containing the AAString object for each member gene

**Examples**

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

---

getTaxonomy	<i>Get the Taxonomic tree function</i>
-------------	----------------------------------------

---

### 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)
```

### 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

### Value

an object containing the JSON keys as attributes

### Examples

```
getTaxonomy()
getTaxonomy(members="YEAST,ASHGO")
getTaxonomy(root="Alveolata")
```

---

getTopGO	<i>Get the topGO Object function</i>
----------	--------------------------------------

---

### 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 annotations 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")
```

---

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 AAStrng 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="MNDPSLLGYPNVGPQQQQQQQQHAGLLGKGTGNALQQQLHMNQLTGIPPPGLMNSDVHTSSNNNSRQLLDQLANGNANMLNMNM
mapSequence(search="mixed", query="NKLLQPTDFQQSHIAEASKSLVDCTKQALMEMADTLTDSKTAKKQPTGDSTPSGTATNSAVSTPLTPKIEL
```

---

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 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/>

---

protein	<i>An example protein object.</i>
---------	-----------------------------------

---

**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=MNDPSLLGYPNVGPQQQQQQQQHAGLLGKGTPNALQQQLHMNQLTGIPPPG>

---

sequence_map	<i>An example dataframe containing proteins identified from a given sequence.</i>
--------------	-----------------------------------------------------------------------------------

---

### 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=MNDPSLLGYPNVGPQQQQQQQQHAGLLGKGTPNALQQQLHMNQLTGIPPPG>

---

taxonomy	<i>An example newick format taxonomy object.</i>
----------	--------------------------------------------------

---

### 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>



# Index

## \*Topic **datasets**

- group, [9](#)
- hog, [10](#)
- orthologs, [11](#)
- pairs, [12](#)
- protein, [13](#)
- sequence\_annotation, [14](#)
- sequence\_map, [15](#)
- taxonomy, [15](#)
- xref, [16](#)

formatTopGO, [3](#)

- getAnnotation, [3](#)
- getAttribute, [4](#)
- getData, [4](#)
- getGenomeAlignment, [5](#)
- getGRanges, [5](#)
- getHOG, [6](#)
- getObjectAttributes, [6](#)
- getOntologies, [7](#)
- getSequences, [7](#)
- getTaxonomy, [8](#)
- getTopGO, [8](#)
- getXref, [9](#)
- group, [9](#)

hog, [10](#)

mapSequence, [10](#)

- OmaDB (OmaDB-package), [2](#)
- OmaDB-package, [2](#)
- orthologs, [11](#)

- pairs, [12](#)
- protein, [13](#)

resolveURL, [14](#)

- sequence\_annotation, [14](#)
- sequence\_map, [15](#)

taxonomy, [15](#)

xref, [16](#)