Expanding the Orthologous Matrix (OMA) programmatic interfaces: REST API and the *OmaDB* packages for R and Python [version 1; referees: 1 approved, 1 approved with reservations]

Klara Kaleb¹, Alex Warwick Vesztrocy¹,², Adrian Altenhoff²,³, Christophe Dessimoz ¹,²,⁴-⁶

¹Centre for Life’s Origins and Evolution, Department of Genetics, Evolution and Environment, University College London, London, WC1E 6BT, UK
²Swiss Institute of Bioinformatics, Lausanne, Switzerland
³Department of Computer Science, ETH Zurich, Zurich, Switzerland
⁴Department of Computer Science, University College London, London, WC1E 6BT, Switzerland
⁵Department of Computational Biology, University of Lausanne, Lausanne, 1015, Switzerland
⁶Center for Integrative Genomics, University of Lausanne, Lausanne, 1015, Switzerland

Abstract
The Orthologous Matrix (OMA) is a well-established resource to identify orthologs among many genomes. Here, we present two recent additions to its programmatic interface, namely a REST API, and user-friendly R and Python packages called *OmaDB*. These should further facilitate the incorporation of OMA data into computational scripts and pipelines. The REST API can be freely accessed at https://omabrowser.org/api. The R OmaDB package is available as part of Bioconductor at http://bioconductor.org/packages/OmaDB/, and the omadb Python package is available from the Python Package Index (PyPI) at https://pypi.org/project/omadb/.

Keywords
orthologs, paralogs, hierarchical orthologous groups, comparative genomics, orthologous matrix, oma, API, R, python, REST, bioconductor

This article is included in the Bioconductor gateway.

This article is included in the RPackage gateway.
Introduction

Orthologs are pairs of protein coding genes that have common ancestry and have diverged due to speciation events\(^1\). The detection of orthologs is of fundamental importance in many fields in biology, such as comparative genomics, as it allows us to propagate existing biological knowledge to ever growing newly sequenced data\(^2\)\(^3\).

The Orthologous Matrix (OMA) is a method and resource for the inference of orthologs among complete genomes\(^4\). The OMA database (https://omabrowser.org) features broad scope and size with currently over 2,100 species from all three domains of life.

The OMA browser has supported multiple ways of exporting the underlying data from its beginning. Users can download data either via bulk archives or interactively through the browser—using where possible standard file formats, such as FASTA, OrthoXML\(^5\), or PhyloXML\(^6\). For programmatic access, early OMA database releases offered an Application Programming Interface (API) in the form of the Simple Object Access Protocol (SOAP). However, the complexity and limited adoption of SOAP has prompted us to recently switch to the simpler, faster, and more widely used Representational State Transfer (REST) protocol for the OMA API\(^4\). Here, we provide a description of this new OMA REST API.

Furthermore, the R environment is widely used in bioinformatics due to its flexibility as a high-level scripting language, statistical capabilities, and numerous bioinformatics libraries. In particular, the Bioconductor open source framework contains over 2,000 packages to facilitate either access to or manipulation of biological data. This motivated us to develop the OmaDB Bioconductor package which provides a more idiomatic and user-friendly access to OMA data in R implemented on top of the REST API.

Finally, to also enable Python users to easily interact with the database, we have developed a similar package in that language, compliant with the conventions and with support of typical complementary Python packages as outlined below.

Methods

We start by describing the OMA REST API, before moving on to detail the OmaDB Bioconductor package, and finally outline the omadb Python package.

OMA REST API

The REST framework is an API architectural style that is based on URLs and HTTP protocol methods. It was designed to be stateless and thus is context independent. That is, it does not save data internally between the HTTP requests which minimises server-side application state, thus easing parallelism. The combination of the HTTP and JSON data formats makes it particularly suitable for web applications and easily supported by most programming languages.

Since the backend of the OMA browser is almost fully based on Python and its frontend is supported by the Django web framework\(^8\), we have opted to use the Django Rest Framework (DRF) to implement a REST API in our latest release\(^4\). Most API calls require querying the OMA database, stored in HDF5\(^9\), using a custom Python library (“pyoma”). The query results are serialised in the format requested by the user — typically JSON.

Most data available through the OMA browser is now also accessible via the API. This includes individual genes and their attributes such as protein or cDNA sequences, cross-references, pairwise orthologs, hierarchical orthologous groups\(^10\), as well as species trees and the corresponding taxonomy. The API documentation as well as the interactive interface can be found at https://omabrowser.org/api/docs (Figure 1).

OmaDB Bioconductor package

To facilitate simplified access to the API and downstream analyses in the R environment, we have also developed an API wrapper package in R, now available in Bioconductor\(^7\) (http://bioconductor.org/packages/OmaDB/). This allowed for abstraction of the server interface, eliminating the need to know structure of the database or the URL endpoints to access the required data.

The package consists of a collection of functions that import OMA data into R friendly objects, namely S3 objects and data frames—depending on the query supplied. Due to the volume of the data available, some selected object attributes are at first given as URL endpoints. However, these are automatically loaded upon accession. OmaDB also facilitates further downstream analyses with other Bioconductor packages, such as GO enrichment analysis with topGO\(^11\), sequence analysis with BioStrings\(^12\), phylogenetic analyses using ggtree\(^13\) or gene locus analyses with the help of GenomicRanges\(^14\).
The open source code is hosted at https://github.com/DessimozLab/OmaDB/. The package requires R version >= 3.6 and Bioconductor version >= 3.9, as well as a stable internet connection.

### Package Installation

```r
if (!requireNamespace("BiocManager"))
  install.packages("BiocManager")
BiocManager::install("OmaDB")
# load the package
library(OmaDB)
```

**omadb Python package**

For Python users, we provide an analogous package also named *omadb*. Results are supplied to users as a hybrid attribute-dictionary object. As such, both attribute and key-based access is possible. Where the URL of a further API call is listed in a response, this has been designed to be automatically requested for the user.

For data that can be represented as a table, the *pandas* package[^11] is supported. HOGs can be analysed or displayed using the *pyham* library[^15]. Trees are retrievable as *DendroPy*[^17] or *ETE*[^18] Tree objects. Gene Ontology enrichment analyses are possible through the use of the *goatools* package[^19].

The open source code is hosted at https://github.com/DessimozLab/pyomadb/. The package requires Python >=3.6, as well as a stable internet connection. It is also available to download from PyPI, installable using pip.

### Package Installation

```bash
# Install in shell, using pip
$ pip install omadb

# In Python, load the package
>>> from omadb import Client
# Initialise the client
>>> c = Client()
```

---

[^11]: https://pandas.pydata.org/
[^15]: https://github.com/parisotlab/pyham/
[^17]: https://www.dendropy.org/
[^18]: https://ete-soft.github.io/
[^19]: https://github.com/davidruan/goatools/

---

**Figure 1.** Showcase of the OMA REST API documentation page, with an example of the interactive query and response.
Results
We provide six illustrative examples in R. The first shows a direct call to the REST API, while the other five
showcase the OmaDB R library. These examples are also available as a Jupyter notebook as part of the
OmaDB R code repository. We have also provided analogous examples in Python, also in the form of a Jupyter
notebook, included in its code repository—with the exception of Example 6, which uses a package only available
in R.

Example 1 - Simply accessing the API, in R, via URLs
One way to access the API is to directly send a request using httr in R. This approach requires the user to know
the URL of the API endpoint, as well as the URL of the API function of interest. Some additional processing
steps of the resultant response is usually needed. A simple example to retrieve information on the P53_RAT
protein is provided below.

```r
library(httr)
url <- "https://omabrowser.org/api/protein/P53_RAT/"
response <- GET(url)
response_content_list <- httr::content(response, as = "parsed")
```

Example 2 - Using a sequence to find its gene family (Hierarchical Orthologous Group) and function via
gene ontologies
Below is a simple workflow using the OmaDB package to annotate a given protein sequence, using the
mapSequence() function.

```r
library(OmaDB)
sequence <- 'MKLVFLVLLFLGALGLCLAGRRRSVQWCAVSQPEATKCFQWQRNMRRKVRGPPVSCIKRDSPIQCIQA
IARENADAATLDGGFIYEAGLAPYKLRPVAEVYGETQRPRTTHYAYAVAVVKKKGSGFQLNELOGLKSCHTGTLRRTA
GWCPVPITLGIFPLNWTGPPEIEAAVARFSPASCVFGADKQFPNLCRLCAGTGENKCAFSQEPYFSYSGA
FKCLRDRGAGDVFAFIRESTTVFEDLSEAEEDYELLPDNTKPKDFFKHARVFPSHAVARSVNGKEDAI
WNNLLRQAQERKFKDSFKQFLSGPSQKDLFLKDSAIIFSRVPPRIDSGLYLSGHTYFTAIQNLRKSEEEVA
ARRARVWCVACGEQELRKCNQWGSGLSEGSVCTCSSASTTEDCIALVKGKADCMLSDGGYVYTAGKCGLVPL
AENYKQSOSSDPCNPQVDVPVEVYAVVRRSBDTSLTVNSVKGKSSCNATAVRTEAGWNIPMLLNFNTGSC
KFDEYFSQSCAPSDDPSRNLACIGIDGEGKENCVSNERYGTYGAFRCLAENAGDVAFVKDVTVQLNND
GNNEAWAKDLKLFALLCFLDGRKPKVTBSCSACLAMAPNHAVVSREDKVRKLQVLLHQQAKFGNRNGSDC
PDKFCLPQSETKNNLFFNDNTECLHLGKHTYEKLYGPQVYGAGTNLNKKSTPSLLEACEFLRK'
seq.annotation <- mapSequence(sequence)
```

In this example, the sequence mapping identified one target sequence. From the seq.annotation object further
information can be obtained as follows:

```r
seq.annotation$targets[[1]]$canonicalid # 'TRFL_HUMAN'
```

Thus, our sequence is human lactotransferrin (also known as lactoferrin). Lactotransferrin is one of four
subfamilies of transferrins in mammals.

To investigate the evolutionary history of genes more precisely, we turn to Hierarchical Orthologous Groups
(HOGs)—sets of genes which have descended from a single common ancestral gene within a taxonomic range
of interest. For an introduction to HOGs, we refer the interested reader to the following short video: https://
youtu.be/5p5x5gxziZA.

By knowing the ID of the HOG to which our sequence belongs, we can obtain a list of all the HOG members
(i.e. all genes in the HOG), as follows:

```r
hog_id <- seq.annotation$targets[[1]]$oma_hog_id # 'HOG:0413862.1a.1b'
hog <- getHOG(id = hog_id, members = TRUE, level = 'Mammalia')
```

Note that it is also possible to query information on a HOG using the ID of one of its members. Therefore the
below will produce the same output.
hog <- getHOG(id = 'TRFL_HUMAN', members = TRUE, level = 'Mammalia')

We can easily retrieve the Gene Ontology (GO) terms\(^{23}\) that are associated to each of the members using OmaDB.

go_annotations <- getProtein(hog$members$omaid,
attribute = 'gene_ontology'
)

The resultant list of GO terms per gene is in the “geneID2GO” format by default, which is used by the topGO\(^{11}\) package.

To compare the function of lactotransferrins with their paralogous counterparts, we can retrieve a background set consisting of all members of the transferring HOG defined at the root of the eukaryotes

bgHOG <- getHOG(id = 'TR_HUMAN', members = TRUE, level = 'Eukaryota')
bgAnnnot <- getProtein(bgHOG$members$omaid, attribute = 'gene_ontology')

We can now construct a topGO object using the getTopGO function as seen below. Note that the background set of terms is set by getTopGO to all terms appearing in the list of annotations. This may not be appropriate in all cases—the choice of background set requires careful consideration\(^{24}\).

bgAnnnotFormatted = formatTopGO(bgAnnnot, format = 'geneID2GO')

library(topGO)
myGO <- getTopGO(annotations = bgAnnnotFormatted, format = 'geneID2GO',
foregroundGenes = hog$members$entry_nr, ontology = 'BP')
myRes <- runTest(myGO, algorithm = 'classic', statistic = 'fisher')
print(GenTable(myGO, myRes))

As the output in Table 1 indicates, several enriched terms in the mammalian lactotransferrin are related to bone formation, consistent with previous reports in the literature (e.g. 25). So is the role of lactotransferrin in antimicrobial activity (e.g. 26).

### Example 3 - Taxonomic tree visualisation

The taxonomic data obtained using the OmaDB package can easily be plugged into ggtree\(^{13}\) for phylogenetic tree visualisation. First, the tree is obtained using the getTaxonomy() function. In this example, the tree is rooted at the Hominoida taxonomic level. The default format of the object returned is newick.

tax <- getTaxonomy(root = 'Hominoida')

<table>
<thead>
<tr>
<th>GO.ID</th>
<th>Term</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0001501</td>
<td>skeletal system development</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0001503</td>
<td>ossification</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0001649</td>
<td>osteoblast differentiation</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0001816</td>
<td>cytokine production</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0001817</td>
<td>regulation of cytokine production</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0001818</td>
<td>negative regulation of cytokine production</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0002237</td>
<td>response to molecule of bacterial origin</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0002682</td>
<td>regulation of immune system process</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0002683</td>
<td>negative regulation of immune system process</td>
<td>&lt;1e-30</td>
</tr>
<tr>
<td>GO:0002761</td>
<td>regulation of myeloid leukocyte differentiation</td>
<td>&lt;1e-30</td>
</tr>
</tbody>
</table>
The resultant object can directly be used to build a phylogenetic tree using the ggtree package as below:

```r
library(ggtree)
tree <- getTree(tax$newick)
mytree <- ggtree(tree)
```

The tree can be further annotated using species silhouettes from PhyloPic (http://phylopic.org/). This functionality is already enabled within the ggtree package and just requires obtaining the relevant image codes. The workflow to produce Figure 2 is below.

```r
library(rphylopic)
labels <- tree$tip.label
labelsFormatted <- sapply(labels, FUN = function(x) 
gsub("_", " ", x, fixed = TRUE))
ids <- sapply(labelsFormatted, FUN = function(x)
  name_search(x)$canonicalName[1,1])
images <- sapply( as.character(ids), FUN = function(x)
  tryCatch(name_images(x)$same[[1]]$uid, error = 
  function(w) name_images(x)$supertaxa[[1]]$uid) )
d <- data.frame(label = labels, images = as.character(images))
library(dplyr)
library(ggimage)
mytree %+% d + geom_tiplab(aes(image = images), geom = 'phylopic',
  offset = 2.3, color = 'steelblue') + geom_tiplab(offset = 0.3) +
  ggplot2::xlim(0, 7)
```

**Figure 2.** Species taxonomy tree obtained using example 3.
Example 4 - Visualising the distribution of PAM distances in the taxonomic space

To obtain all orthologous pairs between two genomes, we can use the getGenomePairs() function. To limit server load, the resultant response is paginated and by default only returns the first page, capped at 100 entries. This is easily adjustable by setting the ‘per_page’ parameter to either the number of orthologs required or simply to ‘all’.

In this example, we compare the distribution of PAM distances (Point accepted mutations; 27) between orthologs of two species-pairs, namely human-dog and human-mouse. First, we request the required data:

```r
mouse_id = getGenome(id='Mus musculus')$taxon_id
human_id = getGenome(id='Homo sapiens')$taxon_id
dog_id = getGenome(id='Canis lupus familiaris')$taxon_id

human_mouse <- getGenomePairs(genome_id1 = human_id,
genome_id2 = mouse_id, rel_type = '1:1')
human_dog <- getGenomePairs(genome_id1 = human_id,
genome_id2 = dog_id, rel_type = '1:1')
```

We can then bind the two resultant data frames and plot the results (Figure 3), as so:

```r
human_mouse$Species <- 'Mus musculus'
human_dog$Species <- 'Canis lupus familiaris'
all_pairs <- rbind(human_mouse, human_dog)
all_pairs$Species <- as.factor(all_pairs$Species)
library(ggplot2)
g <- ggplot(all_pairs, aes(x = distance, fill = Species)) +
  geom_density(alpha = 0.5) +
  xlab('evolutionary distance [PAM]') +
  theme(legend.position = 'bottom', panel.grid.major =
    element_blank(), panel.grid.minor = element_blank(),
    panel.background = element_blank(), axis.line = element_line(colour =
    'black'))
print(g)
```

![Figure 3](image)

**Figure 3.** Distribution of evolutionary distances (in PAM units; 27) human-dog (red) and human-mouse (blue) pairs, obtained using example 4.
The two-sample Kolmogorov-Smirnov test can be performed on the two distributions, using the command:

```r
ks.test(human_dog$distance, human_mouse$distance)
```

This returns p-value < 2.2e-16. The median distance between dog and human is shorter than that of mouse and human (8.8 vs. 11.8). This is consistent with previous observations that the rodent has a longer branch than humans and carnivores, and due to carnivores, this is consistent with their shorter generation time.

**Example 5 - Annotating protein sequences not present in OMA**

Although the OMA database currently analyses over 2,100 genomes, many more have been sequenced, and the gap keeps on widening. It is nevertheless possible to use OMA to infer the function of custom protein sequences through a fast approximate search against all sequences in OMA.

```r
# Our mystery sequence is cystic fibrosis transmembrane conductance
# regulator in the Emperor penguin (UniProt ID: A0A087RGQ1_APTFO)
mySeq <-
'FFFLLRWTKPILRKGYRRLESLDIQIPSADSADNLSEKLEREWDRELATSKKKPKLINALRRCFFWKMYGILYLGEVTKSVQPLLGLRISAYDPFNSDRSIAYLIALGCLLFIVRLLIHPAIFGLHHIGMQRIAMFSLIYYKIKLL
LSRVLDS1GSTQVLISLSNLNLKTFDEGLALAHFWQVMLQAVALLMLDDLFLEASFGSLAFMLVIALLFAQWLQGMR
MKYRNKRAGIKNERLVTSEIELINQSVKAYCWEDAMEKIESIRETELKLTRKAAVYFNSASSAFSGSVFVVFLAV
LPYAVIKGIIKRLLTRFICVIRLMTVRQFGSQTWYDISGAINDKQLLEKKEYSLKNLTTGTVKDTAF
WDEGLGFLPEVKANQENNKSAPTSDNLFSNPFLHASVLPQDINFKERQGQLAVSVGSTGAIRTSDLMILMLEFLS
QGRKLHSGRIF8PSFQVTFEGKINNIFGFVSYEDEYRYVSIVACQLEEDISIFKFPDKXLLGVDGIGSISGGQARI
SLARAVYFADVLIDLSFPGHLIDPEETKEFESCVCLMKAMLNTRKLVTSKKEHLKIAKILLHECGFYFGTFSELQ
QPDPFESLGLMFDSFQFSAENRNILTETLRFSIEGEGTGSNEIKQSFQTSFNDXKRNKSIIINPLNASRKF
SVQRNMRQGNYGEDNGFDPERFLSVPDLQGDFVLRLSSMLNTDHLQGRRQQVLMGTSVNFGPSNKS
TTRFTKMSMFQTNLSLEIDITYRTLSRDSLVDITDEINEELEDKCEFDTDAEAMGTTTVNTYRYYTHKMNFLVLIL
CVTVPFLVEAVASLAGMLFLKQALKANTTTQSENSTSDKPPVIVTVTSSYYIIYIYVGAOLTMLANGIFLGPLVHTLI
TVSKTLHQQMKVHALHAPSTNSKAGGMNLRFSDKTDAVLDDLPLTVFDPQILILIIVIGAVTIVSFLQYPYFAS
PVIAAFILLRAYPLHTSQQLKISREPSIFPHTLTVLGLWRAFRRQPYETLHFKLNLHTANWFYLSTLRL
FQMREIMFVVSFVAVAFISVTGTDGSGKVGHIILTAMNIMTGQLWAVNSSIDVSDLMRSVGRIFKIPDMPTEEMKN
IKPHPKNNQFSADLVNREHRKEKKNWSFGQTVKDLTAKYESGGAALVENIFISISQSVQRVGLRTGSTKGSTFLA
FLRLLNTEDGIDQDGVSSTTVSVQQRAKFAGPVQPKVIFSGTFMNLDPYQWDEINKVAEEVLKSVIEQFGFQ
LDVFLVLDGGCVLSSHGHKLNCMLARVLSKAKILLLEPSALDHPVTSOIVRTKHKANFCTVNLSEHHRELAMEQCR
FLVIEDKLRQYESIQKLNEKSSRFQASONHADRLLKLPLVHHRNNSSRKPRFKITALQEETEEVEEQETRL'
```

This results in 54 GO annotations. By comparison, this sequence has merely 15 GO annotations in UniProt-GOA— all of which are also predicted by this method in OMA.

**Example 6 - Combining OmaDB with BgeeDB for gene expression**

We go back to the lactotransferrin gene family from Example 2. We can use OmaDB in conjunction with the BgeeDB Bioconductor package to retrieve expression data from the Bgee database as follows.

```r
BiocManager::install("BgeeDB")
library(BgeeDB)
# Bgee uses Ensembl gene IDs, obtainable using OmaDB's cross-references.
trfl_xrefs <- getProtein(id='TRFL_HUMAN')$xref
trfl_ens_id <- subset(trfl_xrefs, source == 'Ensembl Gene')$xref
# The Ensembl gene IDs need to be without version suffix
trfl_ens_id <- strsplit(trfl_ens_id,'.',fixed=TRUE)[[1]][1]
my_stage <- 'UBERON:0034920' # Infant stage
bgee.expr <- Bgee$new(species='Homo_sapiens')
expr.data <- loadTopAnatData(bgee.expr, stage = my_stage)
gegene.tissue.ids <-
  unlist(expr.data$gene2anatomy[trfl_ens_id], use.names = F)
tissues <- expr.data$organ.names
print(tissues[tissues$ID %in% gene.expr.tissue.ids,])
```

Among the tissues in which lactotransferrin is expressed according to Bgee (Table 2), we note the bone marrow and the palpebral conjunctiva (the eyelid inner surface). This is consistent with the aforementioned involvement of lactotransferrin in bone formation and anti-microbial activity.
Further tutorials on the OmaDB package can be found in the accompanying vignettes:

```r
browseVignettes('OmaDB')
```

## Discussion and outlook

Orthology is used for various purposes, such as species tree inference, gene evolution dynamic, or protein function prediction. The retrieval of orthologs is thus typically just the starting point of a larger analysis. Therefore, this overhaul and expansion of the OMA programmatic interface will facilitate the incorporation of OMA data in such larger analyses.

Our R package will continue to be maintained in line with the biannual Bioconductor releases. Further work to improve the package includes improvement in performance. For example, the responses are currently fully loaded into an R object of choice which, depending on the response size, may create some time lag in the response. We will also continue to update the package and API to incorporate new functionalities of OMA, such as support for local synteny which is currently under development.

Likewise, we will also maintain and further develop the Python package. In particular, we will explore the possibility of further integration with the BioPython library.

More generally, in OMA we will keep supporting the various ways of accessing the underlying data, including the interactive web browser and flat files in a variety of formats. The REST API is also complemented by a new SPARQL interface that enables highly specific queries, as well as federated queries over multiple resources. However, the query language is more complex.

We very much welcome feedback and questions from the community. We also highly appreciate contributions to the code in the form of pull requests. Our preferred channel for support is the BioStar website, where we monitor all posts with keyword “oma”.

### Software availability

Please note that this manuscript uses version 2.0 of the OmaDB R package, which is in the development version of Bioconductor (v.3.9). Until the release of Bioconductor v.3.9 in Spring 2019, there are two possible ways of installing it:

1. Install the development version of R (v.3.6) — required for Bioconductor v.3.9 — and install OmaDB using the command:
   ```
   BiocManager::install('OmaDB', version = 'devel')
   ```

2. Install OmaDB 2.0 directly from the github repo using the devtools R package:
   ```
   install.packages('devtools')
   library(devtools)
   install_github('dessimozlab/omadb')
   ```

---

**Table 2.** Human tissues in which lactotransferrin is expressed in infant stage, according to the Bgee database version 14 (output of Example 6).

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>UBERON:0001812</td>
<td>palpebral conjunctiva</td>
</tr>
<tr>
<td>UBERON:0000178</td>
<td>blood</td>
</tr>
<tr>
<td>UBERON:0002371</td>
<td>bone marrow</td>
</tr>
<tr>
<td>UBERON:0001154</td>
<td>vermiform appendix</td>
</tr>
<tr>
<td>UBERON:0002084</td>
<td>heart left ventricle</td>
</tr>
</tbody>
</table>
REST API available from: https://omabrowser.org/api
Documentation available from: https://omabrowser.org/api/docs

R OmaDB package available from: http://bioconductor.org/packages/OmaDB/
Source code available from: https://github.com/DessimozLab/OmaDB/
Archived source code as at time of publication: http://doi.org/10.5281/zenodo.2530250
License: LGPL-3

omadb Python package available from: https://pypi.org/project/omadb/
Source code available from: https://github.com/DessimozLab/pyomadb/
Archived source code as at time of publication: http://doi.org/10.5281/zenodo.2530253
License: GPL-2

Source code available from: https://github.com/DessimozLab/OmaDB/
REST API available from: https://omabrowser.org/api/api
Documentation available from: https://omabrowser.org/api/docs

Grant information
We acknowledge support by Swiss National Science Foundation grant 150654, UK BBSRC grant BB/M015009/1, the Swiss State Secretariat for Education, Research and Innovation (SERI), as well as a UCL Genetics, Evolution and Environment Departmental Summer Bursary (to KK).

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements
We thank Natasha Glover for helpful feedback on the manuscript, and Frédéric Bastian for help on the example involving BgeeDB.

References


Kaleb et al. present two packages, namely R/Bioconductor OmoDB and python omadb, that allow users to query and use data from the Orthologous Matrix database. The article is well written and the authors provide 6 examples that convincingly demonstrate the usefulness and reach of their work.

I have a couple of comments and suggestions below, presented in chronological order. The only serious one is a request for the authors to describe the outputs in their examples a bit more (see below for details), to facilitate the adoption for users that wouldn't be familiar with R.

- In https://omabrowser.org/api/docs, the pagination example has a typo. The genomes should be replaced with genome:

...  
$ "https://omabrowser.org/api/genomes/?page=2"  
HTTP/1.1 404 Not Found  
Server: nginx  
Date: Mon, 11 Feb 2019 06:04:30 GMT  
Content-Type: text/html; charset=utf-8  
Connection: keep-alive  
X-Frame-Options: SAMEORIGIN  
Vary: Cookie  
Set-Cookie: __utmmobile=d41d8cd98f00b204e9800998ecf8427e; expires=Wed, 10-Feb-2021 06:04:30 UTC; Path=/  
Set-Cookie: sessionid=9zb42ljib7apkubebl1e1t742i5p6f3a6; expires=Mon, 25-Feb-2019 06:04:30 GMT; HttpOnly; Max-Age=1209600; Path=/  

$ curl -I "https://omabrowser.org/api/genome/?page=2"  
HTTP/1.1 200 OK  
Server: nginx  
Date: Mon, 11 Feb 2019 06:04:32 GMT  
Content-Type: application/json  
Connection: keep-alive  
Link: ; rel="first", ; rel="prev", ; rel="next", ; rel="last"  
X-Total-Count: 2198  
Vary: Accept, Cookie  
Allow: GET, HEAD, OPTIONS  
X-Frame-Options: SAMEORIGIN
In the introduction, the authors explain that 'Most data available through the OMA browser is now accessible via the API'. I think it would be useful to know what data isn't available and whether the browser and REST API would ever be equivalent in terms of data served. This might be partly addressed later, in the discussion, where the authors mention 'support for local synteny'. Some additional details would be useful to redirect users to the appropriate interface. Similarly, it would be useful to know if the R and python packages provide access to the same data, or if differences also exist there.

I didn't see mention of the R and python packages on the OmaDB web page. This would be a useful addition for visitors.

In the Bioconductor package section, the authors explain that data is provided in 'R friendly objects, namely S3 objects and data frames'. I would suggest to rephrase this and only refer to objects, as S4 objects are also returned and the nature of the technical class system is probably not necessary in the frame of this document.

Regarding the R package, I would suggest to add URL and BugReports fields in the packages DESCRIPTION file. This helps users find the GitHub repository and report issues. I also noted that in the 'getting started' vignette, it looks like some section a missing a space after the section markup. I have send a pull request fixing these and some other minor issue.

Note that the html and R version of the vignette shouldn't be included in the package source.

In the python package section, the authors mention that this package is named 'omadb'. I would argue that the packages have different names, as programming languages are case sensitive and suggest to drop the also to avoid any confusion.

In the first sentence of the result section, authors should replace R library by R package, as they are referring to their package, not the location where the package is being installed (the library).

In general, it would be very useful for the authors to describe the different outputs they have. I am not expecting the authors to provide full details of the REST API responses, but describing how the results match the text would be important. For example, in example 1, they only show how to produce the 'response_content_list' response. Here, it would be useful to explain that this R list directly maps the REST json message, and point to the specific documentation entry point. Such an explanation motivates the example in the text and helps users, that aren't familiar with REST, to understand the relation between the server and the package.

Similarly in example 2, the authors create the `seq_annotation` variable and mention that only one target sequence was identified. Here, it would be useful to show that `length(seq_annotation$targets)` is equal to 1, to back their claim, to indicate how users can verify the number of targets, and motivate the use of the first list index in later code chunks.

Still in example 2, the authors query and extract the hog members. These data are however already present in the first output of that example, under `seq_annotation$targets[[1]]$oma_hog_members`. It would be useful to explain why the authors send a second query to obtain that data and clarify whether `oma_hog_members` is always equivalent to calling `getHOG` and `getProtein`.

When trying to reproduce the code, I first failed to run the code chunks calling `getProtein`. Later, the authors clarify the software requirements in more details. It would however be useful to briefly mention, early on in the Results section, what version was used for the examples.
In example 5, I would suggest to update to new function name, as `getAnnotations` is expected to be deprecated in the next release, especially as the new version of the package is anyway required for the `getProtein` function.

```r
> myAnnotations <- getAnnotation(mysterySeq)
Warning message:
'getAnnotation' is deprecated.
Use 'annotateSequence' instead.
See help("Deprecated")
```

Another example where an explanation of the output is important is example 5. The authors call `myAnnotations <- getAnnotations(mysterySeq)` and then refer to 54 GO annotation results. In repeating their analysis, I obtain a data frame with 55 observations (see below). It is unclear whether I have a different result, if one observation should be dropped, or if my output is completely wrong (was I even expecting a data frame?).

```r
> dim(myAnnotations)
[1] 55 13
```

In general, given the nature of the package, i.e. that it accesses an online repository that is (or can be) updated regularly, results may change, this also explaining why I may have different results.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

**Competing Interests:** No competing interests were disclosed.

**Referee Expertise:** Computational biology and bioinformatics, research software, reproducible research, omics.
I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Bastian Greshake Tzovaras 1, Ngoc-Vinh Tran 2

1 Environmental Genomics and Systems Biology Division, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA
2 Department for Applied Bioinformatics, Institute of Cell Biology and Neuroscience, Goethe University, Frankfurt am Main, Germany

The authors describe a new REST API as an interface to the well-established Orthologous Matrix database. As identifying and evaluating orthologs is a central step in many biological analyses, an easy way to query the over 2,100 species in OMA is highly valuable. To further facilitate querying the data through their API, the authors present packages for R and Python. The API is well documented on the OMA website and the R package comes with vignettes describing different use cases. The manuscript presented here focuses on the OmaDB R-package and showcases some of its functions.

Being somewhat "ahead of its time", the R package as described in the manuscript requires both the development version of R (v3.6) and Bioconductor (v3.9). The package installation instructions at the beginning of the manuscript only glances over it, more complete instructions are only found in the Software availability section at the end.

We recommend including more explicit warnings/instructions about the required versions at the beginning, otherwise potential users might be confused when trying to follow along with the examples given in the manuscript (As happened to us and it took us some time to figure out what's going on).

While the Python package is not extensively discussed in this manuscript, the authors provide a Binder that can be used to reproduce the same analyses using Python. We recommend putting a link to it (https://mybinder.org/v2/gh/DessimozLab/pyomadb/master?filepath=examples%2Fpyomadb-examples.ipynb) in the manuscript, to help users with taking up the Python library.

We welcome the switch from the SOAP API to a more modern REST implementation and the provided packages to interface with the API will be valuable for a lot of researchers working with orthologs.

Is the rationale for developing the new software tool clearly explained?  
Yes

Is the description of the software tool technically sound?  
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?  
Yes
Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool? 
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article? 
Yes

**Competing Interests:** No competing interests were disclosed.

**Referee Expertise:** bioinformatics, evolutionary biology

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com