Package 'MGRASTer'

February 19, 2015

Type Package

Title API Client for the MG-RAST Server of the US DOE KBase

Version 0.9

Depends R (>= 3.0), utils

Imports

Suggests RJSONIO

URL https://github.com/braithwaite/MGRASTer/

Date 2014-07-28

Description Convenience Functions for R Language Access to the v.1 API of the MG-RAST Metagenome Annotation Server, part of the US Department of Energy (DOE) Systems Biology Knowledge Base (KBase).

License BSD_2_clause + file LICENSE

Copyright University of Chicago

LazyLoad yes

LazyData yes

LazyDataCompression xz

Collate source.R

Author Daniel T. Braithwaite [aut, cre]

Maintainer Daniel T. Braithwaite <contact.dtb@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-02 07:29:38

R topics documented:

call.MGRAST 2
doc.MGRAST 5
MGRASTAPI 7

Index **9**

`call.MGRAST` *Interface to MG-RAST API*

Description

An interface to the MG-RAST API enabling convenient access in the R environment to provided resources, as well as facilitating export.

Usage

```
call.MGRAST(resource, request, ..., args=NULL, destfile=NULL, parse=is.null(destfile),
  verify=parse, bugs=c("ignore", "warn", "stop", "ask", "report"), quiet=TRUE,
  timeout=300, issue=TRUE)
parse.MGRAST(call.url)
```

Arguments

`resource`	name of API resource (string)
`request`	name of request, valid for the specified resource (string)
`...`	required and/or optional parameters for the call
`args`	further parameters (`list`)
`destfile`	file to save the retrieved resource (string)
`parse`	attempt JSON parsing of result? (`logical`)
`verify`	check integrity of the returned object? (`logical`)
`bugs`	reporting action for API-side problems – not used
`quiet`	print debugging messages? (`logical`)
`timeout`	number of seconds to allow for the call (single `integer`)
`issue`	issue the call, or only return the constructed URL? (`logical`)
`call.url`	URL for an API resource (string)

Details

This documentation assumes familiarity with the MG-RAST API, which is described elsewhere.

`resource` and `request` are required for all API calls. Optional and required parameters may be given in `...` and must be named. In general, parameters should be length-one `character` vectors, which is what is meant by "string", above. Regardless, parameters are coerced by `as.character`. Parameters may also be given with `args`, which must be a named list. Partial `resource`, `request`, and parameter names are matched wherever possible, as are partial parameter values tied to a controlled vocabulary.

Parameters named `id` are specially handled with some helpful scrubbing. In particular, prefixes `mgl`, `mgm`, `mgp`, and `mgs` are added as necessary, and multiple `id`s given in a vector are separated.

For certain resource-request combinations, `destfile` is mandatory. For all others, it is optional. When `verify=TRUE` a check is made that all documented components for a resource-request are actually received.

Setting `quiet=FALSE` can help shed light on unexpected results.

`parse.MGRAST()` tokenizes a valid API URL. If `x` is a URL validly requesting an API resource, then `do.call(call.MGRAST, parse.MGRAST(x))` will retrieve that resource.

Value

For `call.MGRAST()`, a `list` mirroring the JSON structure of the received resource (invisibly). But if `parse=FALSE`, then the resource, uninterpreted. For non-JSON resources, `parse` is ignored.

But if `destfile` is not `NULL` and the resource was successfully written to `destfile`, then `destfile`. If `parse=TRUE` and relevant, the file is written in .rda format, otherwise raw (usually meaning, as legible text).

Only a URL for the specified resource, if `issue=FALSE`.

For `parse.MGRAST()`, a named `list` including `resource`, `request`, and further parameters for the indicated API call.

Author(s)

Daniel T. Braithwaite

References

http://metagenomics.anl.gov
http://api.metagenomics.anl.gov
http://www.json.org

See Also

doc.MGRAST, MGRASTAPI

Examples

```
## Not run:
##  The calls below are adapted from the MG-RAST API documentation pages.
##  An example is provided for most resource types.
##  Note that resources are returned invisibly by call.MGRAST().
##  These examples are "not run" because they take too long.

##
##  Use of call.MGRAST(), parse.MGRAST(), and do.call() together

parse.MGRAST ("http://api.metagenomics.anl.gov/download/mgm4447943.3?stage=650")
ll <- list (id=4447943.3, stage=650)
call.MGRAST ("down", "set", args=ll)

print (tt <- tempfile())
parse.MGRAST ("http://api.metagenomics.anl.gov/download/mgm4447943.3?file=350.1")
```

```
call.MGRAST ("down", "inst", id=4447943.3, file=350.1, destfile=tt)
parse.MGRAST (call.MGRAST (issue=FALSE, "down", "inst", id=4447943.3, file=350.1,
destfile=tt))
unlink (tt)

parse.MGRAST ("http://api.metagenomics.anl.gov/library?limit=20&order=name")
call.MGRAST ("lib", "query", lim=20, ord="name")
do.call (call.MGRAST,
parse.MGRAST ("http://api.metagenomics.anl.gov/library?limit=20&order=name"))

##
##  accessing the "m5nr" resource

call.MGRAST ("m5", "ont", source="Sub", min="level3")
call.MGRAST ("m5", "tax", filter="Bacteroidetes", filter_lev="phylum", min="genus")
call.MGRAST ("m5", "md5", id="000821a2e2f63df1a3873e4b280002a8", source="InterPro")
call.MGRAST ("m5", "func", text="sulfatase", source="GenBank")
call.MGRAST ("m5", "seq", text="MAGENHQWQGSIL", source="TrEMBL")

##
##  parsing URLs for the calls above

parse.MGRAST (
"http://api.metagenomics.anl.gov/m5nr/ontology?source=Subsystems&min_level=level3")
parse.MGRAST (paste0 ("http://api.metagenomics.anl.gov/m5nr/taxonomy",
"?filter=Bacteroidetes&filter_level=phylum&min_level=genus"))
parse.MGRAST (paste0 ("http://api.metagenomics.anl.gov/m5nr/md5/",
"000821a2e2f63df1a3873e4b280002a8?source=InterPro"))
parse.MGRAST ("http://api.metagenomics.anl.gov/m5nr/function/sulfatase?source=GenBank")
parse.MGRAST (
"http://api.metagenomics.anl.gov/m5nr/sequence/MAGENHQWQGSIL?source=TrEMBL")

##
##  annotation data via the "matrix" resource,
##  and different ways to provide arguments

xx <- c (4447943.3, 4447192.3, 4447102.3, 4447103.3)
yy <- "4447943.3 4447192.3 4447102.3 4447103.3"
aa <- list (group="level3", source="Sub", res="ab", ident=80,
filter_lev="phylum", filter="Firmicutes")
call.MGRAST ("matrix", "org", id=xx, group="family", source="Ref", result="abund", eval=15)
call.MGRAST ("matrix", "func", id=xx, args=aa)
call.MGRAST ("matrix", "feat", id=yy, source="KEGG", result="ev", len=25)

##
##  parsing URLs for the calls above

parse.MGRAST (paste0 (
"http://api.metagenomics.anl.gov/matrix/organism",
"?id=mgm4447943.3&id=mgm4447192.3&id=mgm4447102.3&id=mgm4447103.3",
"&group_level=family&source=RefSeq&result_type=abundance&evalue=15"))
parse.MGRAST (paste0 (
"http://api.metagenomics.anl.gov/matrix/function",
```

```
"?id=mgm4447943.3&id=mgm4447192.3&id=mgm4447102.3&id=mgm4447103.3",
"&group_level=level3&source=Subsystems&result_type=abundance&identity=80",
"&filter_level=phylum&filter=Firmicutes"))
parse.MGRAST (paste0 (
"http://api.metagenomics.anl.gov/matrix/feature",
"?id=mgm4447943.3&id=mgm4447192.3&id=mgm4447102.3&id=mgm4447103.3",
"&source=KEGG&result_type=evalue&length=25"))

##
##  examples for still other resources

parse.MGRAST ("http://api.metagenomics.anl.gov/metadata/export/mgp128")
call.MGRAST ("metadata", "exp", id=128)

parse.MGRAST (
"http://api.metagenomics.anl.gov/metagenome/mgm4447943.3?verbosity=metadata")
call.MGRAST ("metagenome","inst", id=4447943.3, verb="meta")

parse.MGRAST ("http://api.metagenomics.anl.gov/project?limit=20&order=name")
call.MGRAST ("proj", "query", lim=20, ord="name")

parse.MGRAST ("http://api.metagenomics.anl.gov/sample/mgs25823?verbosity=full")
call.MGRAST ("samp", "inst", id=25823, verb="full")

##
##  finally, examples for annotation downloads (they take substantial time)

print (tt <- tempfile())
call.MGRAST ("annot", "seq", id=4447943.3, eval=10, type="org", source="Swiss",
destfile=tt)
call.MGRAST ("annot", "sim", id=4447943.3, ident=80, type="fu", source="KO",
destfile=tt)
unlink (tt)

## End(Not run)
```

---

doc.MGRAST *Documentation for MG-RAST API*

Description

A utility to flexibly inspect the documentation tree of the MG-RAST API.

Usage

```
doc.MGRAST(depth=1, stratum=NULL, head=NULL, ...)
```

Arguments

depth	show this many levels (single integer)
stratum	show all subtrees matching this name (string)
head	show elements beginning only here (character)
...	additional arguments passed to str()

Details

This documentation assumes familiarity with the MG-RAST API, which is described elsewhere.

The MG-RAST API is locally represented with nested lists that this function helps to explore. It is a specialized version of str(). Usage is best understood through the examples.

A length-one character vector is meant by "string", above.

Value

None. Output is printed to the screen, as with str().

Author(s)

Daniel T. Braithwaite

References

http://metagenomics.anl.gov
http://api.metagenomics.anl.gov
http://www.json.org

See Also

str, call.MGRAST, parse.MGRAST, MGRASTAPI

Examples

```
##  list resources, then list all requests of all resources
doc.MGRAST()
doc.MGRAST(2)

##  show detail for matrix resource (partial matching works)
doc.MGRAST (head="matrix")
doc.MGRAST (2, head="mat")

##  show options for all requests
doc.MGRAST (stratum="options")
doc.MGRAST (2, "options")

##  show options in detail for a specific request
doc.MGRAST (3, head=c("annot","seq","param","opt"))

##  show return values of all m5nr requests in detail
doc.MGRAST (5, "attributes.data", "m5nr", nchar.max=30)
```

MGRASTAPI *Local Representation of MG-RAST API*

Description

The session copy of the documentation tree of the MG-RAST API, and tools for dynamic update.

Usage

```
.MGRAST
load.MGRAST(ff=API.file())
build.MGRAST(ff=API.filename)
```

Arguments

ff	a file name (string or NULL)

Details

This documentation assumes familiarity with the MG-RAST API, which is described elsewhere.

The environment .MGRAST contains a nested list structure API built from the documentation-qua-specification of the MG-RAST API published at http://api.metagenomics.anl.gov. A prebuilt version of this tree is distributed with the MGRASTer package, which the routines call.MGRAST() and parse.MGRAST() consult.

build.MGRAST() builds a new, accurate-up-to-the-minute copy of the documentation tree, which may be desirable to do after a change or patch is implemented in the API. However, to take effect, a new copy must be assigned into .MGRAST (as in the examples).

load.MGRAST() returns the tree as saved in a specified file, or the version of the tree actually in use.

Value

For build.MGRAST(), the file name ff, or the new documentation tree itself when ff=NULL. For load.MGRAST(), the copy of the API documentation tree stored in ff (by default the prepackaged copy), but the copy active in the current session when ff=NULL.

Author(s)

Daniel T. Braithwaite

References

http://metagenomics.anl.gov
http://api.metagenomics.anl.gov
http://www.json.org

See Also

doc.MGRAST, call.MGRAST, parse.MGRAST

Examples

```
## Not run:
##  get API currently in use
API <- get ("API", .MGRAST)

##  compare to API distributed with the package
identical (API, load.MGRAST())

##  build and save a new (possibly updated) copy
build.MGRAST ("my_copy.rda")

##  load it
API <- load.MGRAST ("my_copy.rda")

##  put it into effect
assign ("API", API, .MGRAST)

## End(Not run)
```

Index

.MGRAST *(*MGRASTAPI*)*, 7

build.MGRAST *(*MGRASTAPI*)*, 7

call.MGRAST, 2, *6*, *8*

doc.MGRAST, *3*, 5, *8*

load.MGRAST *(*MGRASTAPI*)*, 7

MGRASTAPI, *3*, *6*, 7

parse.MGRAST, *6*, *8*
parse.MGRAST *(*call.MGRAST*)*, 2

str, *6*