Package: MGnifyR (via r-universe)

June 24, 2024

Type Package

Version 1.1.0

Title R interface to EBI MGnify metagenomics resource

Description Utility package to facilitate integration and analysis of EBI MGnify data in R. The package can be used to import microbial data for instance into TreeSummarizedExperiment (TreeSE). In TreeSE format, the data is directly compatible with miaverse framework.

biocViews Infrastructure, DataImport, Metagenomics

License Artistic-2.0 | file LICENSE

Encoding UTF-8

Depends R (>= 4.3.0), MultiAssayExperiment, TreeSummarizedExperiment, SummarizedExperiment, BiocGenerics

Imports mia, ape, dplyr, httr, methods, plyr, reshape2, S4Vectors, urltools, utils, tidyjson

Suggests biomformat, broom, ggplot2, knitr, rmarkdown, testthat, xml2, BiocStyle, miaViz, vegan, scater, phyloseq, magick

URL https://github.com/EBI-Metagenomics/MGnifyR

BugReports https://github.com/EBI-Metagenomics/MGnifyR/issues

VignetteBuilder knitr

RoxygenNote 7.3.0

Collate 'utils.R' 'MgnifyClient.R' 'AllGenerics.R' 'AllClasses.R' 'AllAccessors.R' 'MGnifyR.R' 'deprecate.R' 'doQuery.R' 'getData.R' 'getFile.R' 'getMetadata.R' 'getResult.R' 'searchAnalysis.R'

Repository https://bioc.r-universe.dev

RemoteUrl https://github.com/bioc/MGnifyR

RemoteRef HEAD

RemoteSha e8c773f606083e3466b025f16c29da07e20a4e02

Contents

`MGnifyR-package` `MGnifyR` *Package.*

Description

`MGnifyR` implements an interface to the EBI MGnify database. See the vignette for a general introduction to this package. About MGnify for general MGnify information, and API documentation for details about the JSONAPI implementation.

Author(s)

Maintainer: Tuomas Borman <tuomas.v.borman@utu.fi> (ORCID)

Authors:

- Ben Allen <ben.allen@ncl.ac.uk>
- Leo Lahti <leo.lahti@iki.fi> (ORCID)

See Also

TreeSummarizedExperiment class

databaseUrl *MgnifyClient accessors and mutators*

Description

MgnifyClient accessors and mutators

Usage

```
databaseUrl(x)

authTok(x)

useCache(x)

cacheDir(x)

showWarnings(x)

clearCache(x)

verbose(x)

databaseUrl(x) <- value

authTok(x) <- value

useCache(x) <- value

cacheDir(x) <- value

showWarnings(x) <- value

clearCache(x) <- value

verbose(x) <- value

## S4 method for signature 'MgnifyClient'
databaseUrl(x)

## S4 method for signature 'MgnifyClient'
authTok(x)

## S4 method for signature 'MgnifyClient'
useCache(x)

## S4 method for signature 'MgnifyClient'
```

```
cacheDir(x)

## S4 method for signature 'MgnifyClient'
showWarnings(x)

## S4 method for signature 'MgnifyClient'
clearCache(x)

## S4 method for signature 'MgnifyClient'
verbose(x)

## S4 replacement method for signature 'MgnifyClient'
databaseUrl(x) <- value

## S4 replacement method for signature 'MgnifyClient'
authTok(x) <- value

## S4 replacement method for signature 'MgnifyClient'
useCache(x) <- value

## S4 replacement method for signature 'MgnifyClient'
cacheDir(x) <- value

## S4 replacement method for signature 'MgnifyClient'
showWarnings(x) <- value

## S4 replacement method for signature 'MgnifyClient'
clearCache(x) <- value

## S4 replacement method for signature 'MgnifyClient'
verbose(x) <- value
```

Arguments

x	A MgnifyClient object.
value	A value to be added to a certain slot.

Details

These functions are for fetching and mutating slots of MgnifyClient object.

Value

A value of MgnifyClient object or nothing.

Examples

```
mg <- MgnifyClient()
```

```
databaseUrl(mg)
showWarnings(mg) <- FALSE
```

---

deprecate *These functions will be deprecated. Please use other functions instead.*

Description

These functions will be deprecated. Please use other functions instead.

Usage

```
mgnify_client(
  username = NULL,
  password = NULL,
  usecache = FALSE,
  cache_dir = NULL,
  warnings = FALSE,
  use_memcache = FALSE,
  ...
)

mgnify_query(
  client,
  qtype = "samples",
  accession = NULL,
  asDataFrame = TRUE,
  maxhits = 200,
  usecache = FALSE,
  ...
)

mgnify_analyses_from_samples(client, accession, usecache = TRUE, ...)

mgnify_analyses_from_studies(client, accession, usecache = TRUE, ...)

mgnify_get_download_urls(
  client,
  accessions,
  accession_type,
  usecache = TRUE,
  ...
)

mgnify_download(
  client,
```

```
  url,
  file = NULL,
  read_func = NULL,
  usecache = TRUE,
  Debug = FALSE,
  ...
)

mgnify_get_analyses_results(
  client = NULL,
  accessions,
  retrievelist = c(),
  compact_results = TRUE,
  usecache = TRUE,
  bulk_dl = FALSE,
  ...
)

mgnify_get_analyses_phyloseq(
  client = NULL,
  accessions,
  usecache = TRUE,
  returnLists = FALSE,
  tax_SU = "SSU",
  get_tree = FALSE,
  ...
)

mgnify_get_analyses_metadata(client, accessions, usecache = TRUE, ...)

mgnify_retrieve_json(
  client,
  path = "biomes",
  complete_url = NULL,
  qopts = NULL,
  maxhits = 200,
  usecache = FALSE,
  Debug = FALSE,
  ...
)
```

Arguments

username	-
password	-
usecache	-
cache_dir	-
warnings	-

use_memcache -

... -

client -

qtype -

accession -

asDataFrame -

maxhits -

accessions -

accession_type -

url -

file -

read_func -

Debug -

retrievelist -

compact_results -

bulk_dl -

returnLists -

tax_SU -

get_tree -

path -

complete_url -

qopts -

Value

-

---

doQuery *Search MGnify database for studies, samples, runs, analyses, biomes, assemblies, and genomes.*

Description

Search MGnify database for studies, samples, runs, analyses, biomes, assemblies, and genomes.

Usage

```
doQuery(x, ...)

## S4 method for signature 'MgnifyClient'
doQuery(
  x,
  type = "studies",
  accession = NULL,
  as.df = TRUE,
  max.hits = 200,
  ...
)
```

Arguments

x	A MgnifyClient object.
...	Remaining parameter key/value pairs may be supplied to filter the returned values. Available options differ between types. See discussion above for details.
type	A single character value specifying the type of objects to query. Must be one of the following options: studies, samples, runs, analyses, biomes, assemblies, super-studies, experiment-types, pipelines, pipeline-tools, publications, genomes, genome-search, genome-search/gather, genome-catalogues, genomeset, cogs, kegg-modules, kegg-classes, antismash-geneclusters, annotations/go-terms, annotations/interpro-identifiers, annotations/kegg-modules, annotations/pfam-entries, annotations/kegg-orthologs, annotations/genome-properties, annotations/antismash-gene-annotations/organisms, or mydata. (By default: type = "studies")
accession	A single character value or a vector of character values specifying MGnify accession identifiers (of type type) or NULL. When NULL, all results defined by other parameters are retrieved. (By default: accession = NULL)
as.df	A single boolean value specifying whether to return the results as a data.frame or leave as a nested list. In most cases, as.df = TRUE will make the most sense. (By default: as.df = TRUE)
max.hits	A single integer value specifying the maximum number of results to return or FALSE. The actual number of results will actually be higher than max.hits, as clipping only occurs on pagination page boundaries. To disable the limit, set max.hits = NULL. (By default: max.hits = 200)

Details

doQuery is a flexible query function, harnessing the "full" power of the JSONAPI MGnify search filters. Search results may be filtered by metadata value, associated study/sample/analyse etc.

See [Api browser](https://www.ebi.ac.uk/metagenomics/api/v1/) for information on MGnify database filters. You can find help on customizing queries from [here](https://emg-docs.readthedocs.io/en/latest/api.html#customising-queries).

For example the following filters are available:

- **studies**: accession, biome_name, lineage, centre_name, include

- **samples**: accession, experiment_type, biome_name, lineage, geo_loc_name, latitude_gte, latitude_lte, longitude_gte, longitude_lte, species, instrument_model, instrument_platform, metadata_key, metadata_value_gte, metadata_value_lte, metadata_value, environment_material, environment_feature, study_accession, include
- **runs**: accession, experiment_type, biome_name, lineage, species, instrument_platform, instrument_model, metdata_key, metadata_value_gte, metadata_value_lte, metadata_value, sample_accession, study_accession, include
- **analyses**: biome_name, lineage, experiment_type, species, sample_accession, pipeline_version
- **biomes**: depth_gte, depth_lte
- **assemblies**: depth_gte, depth_lte

Unfortunately it appears that in some cases, some of these filters don't work as expected, so it is important to check the results returned match up with what's expected. Even more unfortunately if there's an error in the parameter specification, the query will run as if no filter parameters were present at all. Thus the result will appear superficially correct but will infact correspond to something completely different. This behaviour will hopefully be fixed in future incarnations of the MGnifyR or JSONAPI, but for now users should double check returned values.

It is currently not possible to combine queries of the same type in a single call (for example to search for samples *between* latitude). However, it is possible to run multiple queries and combine the results using set operations in R to get the desired behaviour.

Value

A nested list or data.frame containing the results of the query.

Examples

```
mg <- MgnifyClient(useCache = FALSE)

# Get a list of studies from the Agricultural Wastewater :
agwaste_studies <- doQuery(
    mg, "studies", biome_name="Agricultural wastewater"
    )

## Not run:
# Get all samples from a particular study
samps <- doQuery(mg, "samples", accession="MGYS00004521")

# Search polar samples
samps_np <- doQuery(mg, "samples", latitude_gte=66, max.hits=10)
samps_sp <- doQuery(mg, "samples", latitude_lte=-66, max.hits=10)

# Search studies that have studied drinking water
tbl <- doQuery(
    mg,
    type = "studies",
    biome_name = "root:Environmental:Aquatic:Freshwater:Drinking water",
    max.hits = 10)

## End(Not run)
```

getData *Versatile function to retrieve raw results*

Description

Versatile function to retrieve raw results

Usage

```
getData(x, ...)

## S4 method for signature 'MgnifyClient'
getData(x, type, accession.type = NULL, accession = NULL, as.df = TRUE, ...)
```

Arguments

x	A MgnifyClient object.
...	optional arguments fed to internal functions.
type	A single character value specifying the type of data retrieve. Must be one of the following options: studies, samples, runs, analyses, biomes, assemblies, super-studies, experiment-types, pipelines, pipeline-tools, publications, genomes, genome-search, genome-search/gather, genome-catalogues, genomeset, cogs, kegg-modules, kegg-classes, antismash-geneclusters, annotations/go-terms, annotations/interpro-identifiers, annotations/kegg-modules, annotations/pfam-entries, annotations/kegg-orthologs, annotations/genome-properties, annotations/antismash-gene-annotations/organisms, or mydata.
accession.type	A single character value specifying type of accession IDs (accession). Must be specified when accession is specified. (By default: accession.type = NULL)
accession	A single character value or a vector of character values specifying accession IDs to return results for. (By default: accession = NULL)
as.df	A single boolean value specifying whether to return the results as a data.frame or leave as a nested list. (By default: as.df = TRUE)

Details

This function returns data from MGnify database. Compared to getResult, this function allows more flexible framework for fetching the data. However, there are drawbacks: for counts data, getResult returns optimally structured data container which is easier for downstream analysis. getData returns raw data from the database. However, if you want to retrieve data on pipelines or publications, for instance, getResult is not suitable for it, and getData can be utilized instead.

Value

data.frame or list

See Also

getResult

Examples

```
# Create a client object
mg <- MgnifyClient(useCache = FALSE)

# Find kegg modules for certain analysis
df <- getData(
    mg, type = "kegg-modules",
    accession = "MGYA00642773", accession.type = "analyses")
```

`getFile` *Download any MGnify files, also including processed reads and identified protein sequences*

Description

Download any MGnify files, also including processed reads and identified protein sequences

Listing files available for download

Usage

```
getFile(x, ...)

searchFile(x, ...)

## S4 method for signature 'MgnifyClient'
getFile(x, url, file = NULL, read.func = NULL, ...)

## S4 method for signature 'MgnifyClient'
searchFile(
  x,
  accession,
  type = c("studies", "samples", "analyses", "assemblies", "genomes", "run"),
  ...
)
```

Arguments

`x`	A `MgnifyClient` object.
`...`	Additional arguments; not used currently.
`url`	A single character value specifying the url address of the file we wish to download.

file	A single character value or NULL specifying an optional local filename to use for saving the file. If NULL (default), MGNify local cache settings will be used. If the file is intended to be processed in a separate program, it may be sensible to provide a meaningful file, rather than having to hunt through the cache folders. If file is NULL *and* useCache(client) is FALSE, the read.func parameter must be supplied or the file will be downloaded and then deleted. (By default: file = NULL)
read.func	A function specifying an optional function to process the downloaded file and return the results, rather than relying on post processing. The primary use-case for this parameter is when local disk space is limited and downloaded files can be quickly processed and discarded. The function should take a single parameter, the downloaded filename, and may return any valid R object. (By default: read.func = NULL)
accession	A single character value or a vector of character values specifying accession IDs to return results for.
type	A single character value specifying the type of objects to query. Must be one of the following options: analysis, samples, studies, assembly, genome or run. (By default: type = "samples")

Details

getFile is a convenient wrapper round generic the URL downloading functionality in R, taking care of things like local caching and authentication.

THe function is a wrapper function allowing easy enumeration of downloads available for a given accession (or list thereof). Returns a single data.frame containing all available downloads and associated metadata, including the url location and description. This can then be filtered to extract the urls of interest, before actually retrieving the files using fetFile()

Value

Either the local filename of the downloaded file, be it either the location in the MGNifyR cache or file. If read.func is used, its result will be returned.

data.frame containing all discovered downloads. If multiple accessions are queried, the accessions column may to filter the results - since rownames are not set (and wouldn;'t make sense as each query will return multiple items)

Examples

```
# Make a client object
mg <- MgnifyClient(useCache = FALSE)

# Create a vector of accession ids - these happen to be \code{analysis}
# accessions
accession_vect <- c("MGYA00563876", "MGYA00563877")
downloads <- searchFile(mg, accession_vect, "analyses")

# Filter to find the urls of 16S encoding sequences
url_list <- downloads[
```

```
    downloads$attributes.description.label == "Contigs encoding SSU rRNA",
    "download_url"]

# Example 1:
# Download the first file
supplied_filename <- getFile(
    mg, url_list[[1]], file="SSU_file.fasta.gz")

## Not run:
# Example 2:
# Just use local caching
cached_filename <- getFile(mg, url_list[[2]])

# Example 3:
# Using read.func to open the reads with readDNAStringSet from
# \code{biostrings}. Without retaining on disk
dna_seqs <- getFile(
    mg, url_list[[3]], read.func = readDNAStringSet)

## End(Not run)

# Make a client object
mg <- MgnifyClient(useCache = TRUE)
# Create a vector of accession ids - these happen to be \code{analysis}
# accessions
accession_vect <- c(
    "MGYA00563876", "MGYA00563877", "MGYA00563878",
    "MGYA00563879", "MGYA00563880" )
downloads <- searchFile(mg, accession_vect, "analyses")
```

`getMetadata` *Get all Study, Sample and Analysis metadata for the supplied analyses accessions*

Description

Get all Study, Sample and Analysis metadata for the supplied analyses accessions

Usage

```
getMetadata(x, ...)

## S4 method for signature 'MgnifyClient'
getMetadata(x, accession, ...)
```

Arguments

`x`	A `MgnifyClient` object.

...	Optional arguments; not currently used.
accession	A single character value or a vector of analysis accession IDs specifying accessions to retrieve data for.

Details

The function retrieves all associated study, sample and analysis metadata attributes as a list of analyses accessions.

Value

data.frame of metadata for each analysis in the accession list.

Examples

```
# Create a client object
mg <- MgnifyClient(useCache = FALSE)

# Download all associated study/sample and analysis metadata
accession_list <- c("MGYA00377505")
meta_dataframe <- getMetadata(mg, accession_list)
```

---

getResult *Get microbial and/or functional profiling data for a list of accessions*

Description

Get microbial and/or functional profiling data for a list of accessions

Usage

```
getResult(x, ...)

## S4 method for signature 'MgnifyClient'
getResult(
  x,
  accession,
  get.taxa = TRUE,
  get.func = TRUE,
  output = "TreeSE",
  ...
)
```

Arguments

x	A MgnifyClient object.
...	optional arguments: • **taxa.su** A single character value specifying which taxa subunit results should be selected? Currently, taxonomy assignments in the MGnify pipelines rely on rRNA matches to existing databases (GreenGenes and SILVA), with later pipelines checking both the SSU and LSU portions of the rRNA sequence. taxa.su allows then selection of either the Small subunit (SSU) or Large subunit results in the final TreeSummarizedExperiment object. Older pipeline versions do not report results for both subunits, and thus for some accessions this value will have no effect. • **get.tree** A single boolean value specifying whether to include available phylogenetic trees in the TreeSummarizedExperiment object. (By default: get.tree = TRUE) • **as.df** A single boolean value enabled when output = "list". The argument specifies whether return functional data as a named list (one entry per element in the output list) of data.frames, with each data.frame containing results for all requested accessions. If FALSE, The function returns a list of lists, each element consisting of results for a single accession. (By default: as.df = TRUE) • **bulk.dl** A single boolean value specifying should MGnifyR attempt to speed things up by downloading relevant studies TSV results and only extracting the required columns, rather than using the JSONAPI interface. When getting results where multiple accessions share the same study, this option may result in significantly faster processing. However, there appear to be (quite a few) cases in the database where the TSV result columns do NOT match the expected accession names. This will hopefully be fixed in the future, but for now bulk.dl defaults to TRUE. When it does work, it can be orders of magnitude more efficient. (By default: buld_dl = TRUE)
accession	A single character value or a vector of character values specifying accession IDs to return results for.
get.taxa	A boolean value specifying whether to retrieve metagenomic data. (By default: get.taxa = TRUE)
get.func	A boolean value or a single character value or a vector character values specifying functional analysis types to retrieve. If get.func = TRUE, all available functional datatypes are retrieved, and if FALSE, functional data is not retrieved. The current list of available types is "antismash-gene-clusters", "go-slim", "go-terms", "interpro-identifiers", "taxonomy", "taxonomy-itsonedb", "taxonomy-itsunite", "taxonomy-lsu", and "taxonomy-ssu". Note that depending on the particular analysis type, pipeline version etc., not all functional results will be available. (By default: get.func = TRUE)
output	A single character value specifying the format of an output. Must be one of the following options: "TreeSE", "list", or "phyloseq". (By default: output = "TreeSE")

Details

Given a set of analysis accessions and collection of annotation types, the function queries the MGNify API and returns the results. This function is convenient for retrieving highly structured (analysis vs counts) data on certain instances. For example, BIOM files are downloaded automatically. If you want just to retrieve raw data from the database, see `getData`.

Value

If only metagenomic data is retrieved, the result is returned in `TreeSummarizedExperiment` object by default. The result can also be returned as a `phyloseq` object or as a list of `data.frames`. Note that `phyloseq` object can include only one phylogenetic tree meaning that some taxa might be lost when data is subsetted based on tree.

When functional data is retrieved in addition to metagenomic data, the result is returned as a `MultiAssayExperiment` object. Other options are a list containing `phyloseq` object and `data.frames` or just `data.frames`.

Functional data can be returned as a `MultiAssayExperiment` object or as a list of `data.frames`.

See Also

`getData`

Examples

```
# Create a client object
mg <- MgnifyClient(useCache = FALSE)

# Get OTU tables as TreeSE
accession_list <- c("MGYA00377505")
tse <- getResult(mg, accession_list, get.func=FALSE, get.taxa=TRUE)

## Not run:
# Get functional data along with OTU tables as MAE
mae <- getResult(mg, accession_list, get.func=TRUE, get.taxa=TRUE)

# Get same data as list
list <- getResult(
    mg, accession_list, get.func=TRUE, get.taxa=TRUE, output = "list",
    as.df = TRUE, use.cache = TRUE)

## End(Not run)
```

`MgnifyClient` *Constructor for creating a MgnifyClient object to allow the access to MGnify database.*

Description

Constructor for creating a MgnifyClient object to allow the access to MGnify database.

A MgnifyClient object

Usage

```
MgnifyClient(
  username = NULL,
  password = NULL,
  useCache = FALSE,
  cacheDir = tempdir(),
  showWarnings = FALSE,
  verbose = TRUE,
  clearCache = FALSE,
  ...
)
```

Arguments

username	A single character value specifying an optional username for authentication. (By default: username = NULL)
password	A single character value specifying an optional password for authentication. (By default: password = NULL)
useCache	A single boolean value specifying whether to enable on-disk caching of results during this session. In most use cases should be TRUE. (By default: useCache = FALSE)
cacheDir	A single character value specifying a folder to contain the local cache. Note that cached files are persistent, so the cache directory may be reused between sessions, taking advantage of previously downloaded results. The directory will be created if it doesn't exist already. (By default: cacheDir = tempdir())
showWarnings	A single boolean value specifying whether to print warnings during invocation of some MGnifyR functions. (By default: showWarnings = FALSE)
verbose	A single boolean value specifying whether to print extra output during invocation of some MGnifyR functions. (By default: verbose = FALSE)
clearCache	A single boolean value specifying whether to clear the cache. (By default: clearCache = FALSE)
...	optional arguments: • **url** A single character value specifying an url address of the database. (By default: url = "https://www.ebi.ac.uk/metagenomics/api/v1")

Details

All functions in the MGnifyR package take a MgnifyClient object as their first argument. While not essential to querying the raw MGnify API (which is exposed as relative standard JSONAPI), the object allows the simple handling of both user authentication and access to private data, and local on-disk caching of results.

An object that are required by functions of MGnifyR package.

Value

A MgnifyClient object.

Slots

`databaseUrl` A single character value specifying an URL address of database.

`authTok` A single character value specifying authentication token.

`useCache` A single boolean value specifying whether to use cache.

`cacheDir` A single character value specifying cache directory.

`showWarnings` A single boolean value specifying whether to show warnings.

`clearCache` A single boolean value specifying whether to clear cache.

`verbose` A single boolean value specifying whether to show messages.

Constructor

See `MgnifyClient` for constructor.

Accessor

See `MgnifyClient-accessors` for accessor functions.

Examples

```
my_client <- MgnifyClient(
    useCache = TRUE, cacheDir = "/scratch/MGnify_cache_location"
    )

## Not run:
# Use username and password to get access to non-public data
my_client <- MgnifyClient(
    username = "Webin-1122334", password = "SecretPassword",
    useCache = TRUE, cacheDir = "/scratch/MGnify_cache_location"
    )

## End(Not run)
```

`searchAnalysis` *Look up analysis accession IDs for one or more study or sample accessions*

Description

Look up analysis accession IDs for one or more study or sample accessions

Usage

```
searchAnalysis(x, ...)

## S4 method for signature 'MgnifyClient'
searchAnalysis(x, type, accession, ...)
```

Arguments

x	A MgnifyClient object.
...	Optional arguments; not currently used.
type	A single character value specifying a type of accession IDs specified by accession. Must be "studies" or "samples".
accession	A single character value or a vector of character values specifying study or sample accession IDs that are used to retrieve analyses IDs.

Details

Retrieve analysis accession IDs associated with the supplied study or sample accession.

Value

vector of analysis accession IDs.

Examples

```
# Create a client object
mg <- MgnifyClient(useCache = FALSE)

# Retrieve analysis ids from study MGYS00005058
result <- searchAnalysis(mg, "studies", c("MGYS00005058"))

## Not run:
# Retrieve all analysis ids from samples
result <- searchAnalysis(
    mg, "samples", c("SRS4392730", "SRS4392743"))

## End(Not run)
```

Index