Package 'HoloFoodR'

April 9, 2025

Type Package

Version 1.1.3

Title R interface to EBI HoloFood resource

Description Utility package to facilitate integration and analysis of EBI HoloFood data in R. This package streamlines access to the resource, allowing for direct loading of data into formats optimized for downstream analytics.

biocViews Software, Infrastructure, DataImport, Microbiome, MicrobiomeData

License Artistic-2.0 | file LICENSE

Encoding UTF-8

Depends R(>= 4.4.0), MultiAssayExperiment, TreeSummarizedExperiment

Imports dplyr, httr2, jsonlite, S4Vectors, stats, SummarizedExperiment, utils

Suggests BiocStyle, DT, ggh4x, ggsignif, knitr, MGnifyR, mia, miaViz, MOFA2, patchwork, reticulate, rmarkdown, scater, shadowtext, testthat, UpSetR

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

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

VignetteBuilder knitr

RoxygenNote 7.3.2

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

git_branch devel

git_last_commit 077f835

git_last_commit_date 2025-04-03

Repository Bioconductor 3.21

Date/Publication 2025-04-09

Author Tuomas Borman [aut, cre] (ORCID: <https://orcid.org/0000-0002-8563-8884>), Artur Sannikov [aut] (ORCID: <https://orcid.org/0000-0001-7765-123X>), Leo Lahti [aut] (ORCID: <https://orcid.org/0000-0001-5537-637X>)

Maintainer Tuomas Borman <tuomas.v.borman@utu.fi>

Contents

`addMGnify` *Add results from MGnifyR to HoloFoodR results*

Description

Add results from MGnifyR to HoloFoodR results

Usage

```
addMGnify(x, y, ...)

## S4 method for signature 'SummarizedExperiment,MultiAssayExperiment'
addMGnify(
  x,
  y,
  exp.name1 = "metagenomic",
  exp.name2 = "metagenomic_amplicon",
  replace = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment,SummarizedExperiment'
addMGnify(x, y, ...)

## S4 method for signature 'SummarizedExperiment,ANY'
addMGnify(x, y, id.col1 = "sample_biosample", id.col2 = "accession", ...)
```

Arguments

`x`	`SummarizedExperiment`. Results from `MGnifyR::getResult()`.
`y`	`MultiAssayExperiment` or `SummarizedExperiment` or `data.frame`-like table. Results from `HoloFoodR::getResult()` or sample metadata from it.
`...`	optional arguments not used currently.
`exp.name1`	`Character scalar`. Specifies the name of experiment that will be added to `y`. (Default: `"metagenomic"`)

exp.name2	Character scalar. Specifies the name of experiment from HoloFoodR results. This experiment is used to match IDs with MGnify data. (Default: "metagenomic_amplicon")
replace	Logical scalar. Whether to replace the template experiment. (Default: TRUE)
id.col1	Character scalar. Specifies the name of column from colData(x) that includes HoloFood identifiers. (Default: "sample_biosample")
id.col2	Character scalar. Specifies the name of column from colData(y[[exp.name2]]) that includes HoloFood identifiers. (Default: "accession")

Details

Metagenomic data is found in MGnify rather than HoloFoodR, and the two databases use different sample identifiers. However, MGnify's sample metadata includes references to the identifiers used in the HoloFood database, making it straightforward to convert sample IDs for alignment with HoloFood data. Despite this, HoloFood contains additional metadata not available in MGnify. Moreover, integrating data into a MultiAssayExperiment while maintaining accurate sample and system matches can be challenging.

This function is designed to simplify these tasks, enabling seamless integration of MGnify data with HoloFood data after retrieval from the database. You need only to input the returned data from MGnifyR::getResult() and HoloFoodR::getResult() functions.

Value

MultiAssayExperiment

Examples

```
## Not run:
# Get data from HoloFood database
mae <- HoloFoodR::getResult(
    salmon_sample_ids,
    use.cache = TRUE
)

# Get data from MGnify database
mg <- MgnifyClient(
    useCache = TRUE,
    cacheDir = ".MGnifyR_cache"
)
tse <- MGnifyR::getResult(
    mg,
    accession = mgnify_analyses_ids,
    get.func = FALSE
)

# Add MGnify data to HoloFood data
mae <- addMGnify(tse, mae)

## End(Not run)
```

`doQuery` *Search HoloFood database for animals, genome catalogues, samples, or viral catalogues*

Description

Search HoloFood database for animals, genome catalogues, samples, or viral catalogues

Usage

```
doQuery(type, flatten = TRUE, ...)
```

Arguments

`type`	`Character scalar` specifying the type of data to query. Must be one of the following options: `"animals"`, `"genome-catalogues"`, `"samples"` or `"viral-catalogues"`.
`flatten`	`Logical scalar` specifying whether to flatten the resulting `data.frame`. This means that columns with multiple values are separated to multiple columns. (Default: `TRUE`)
`...`	optional arguments: • **max.hits** `NULL` or `integer scalar` specifying the maximum number of results to fetch. When NULL, all results are fetched. (Default: `NULL`) • **spread.sample.types** `Logical scalar` specifying whether to create spread sample types column of animals data. In animals data, sample types column might have multiple values that might be hard to explore. This argument specifies whether to create presence/absence table from sample types. (Default: `TRUE`) • **use.cache** `Logical scalar` specifying whether to use cache. (Default: `FALSE`) • **cache.dir** `Character scalar` specifying cache directory. (Default: `tempdir()`) • **clear.cache** `Logical scalar` specifying whether to remove and clear cache (Default: `FALSE`)

Details

`doQuery` is a flexible query function which can be utilized to search available animals, genome catalogues, samples, or viral catalogues. Search results can be filtered; for example, animals can be filtered based on available samples. See [Api browser](https://www.holofooddata.org/api/docs) for information on filters. You can find help on customizing queries from [here](https://emg-docs.readthedocs.io/en/latest/api.html#customising-queries).

Value

`data.frame`

Examples

```
# Find animals results. The maximum amount of results is 100. Use filter
# so that only chicken is searched. (See details on customizing queries)
res <- doQuery("animals", max.hits = 100, system = "chicken")
head(res)
```

`getData` *Get data from HoloFood database*

Description

Get data from HoloFood database

Usage

```
getData(
  type = NULL,
  accession.type = NULL,
  accession = NULL,
  flatten = FALSE,
  ...
)
```

Arguments

`type`	`NULL` or `character` `scalar` specifying the type of data to query. Must be one of the following options: `"analysis-summaries"`, `"animals"`, `"genome-catalogues"`, `"samples"`, `"sample_metadata_markers"` or `"viral-catalogues"`. When genome or viral catalogues is fetched by their accession ID, the type can also be `"genomes"` or `"fragments"`. (Default: `NULL`)
`accession.type`	`NULL` or `character` `scalar` specifying the type of accession IDs. Must be one of the following options: `"animals"`, `"genome-catalogues"`, `"samples"` or `"viral-catalogues"`. (Default: `NULL`)
`accession`	`NULL` or `character` `vector` specifying the accession IDs of type `accession.type`. (Default: `NULL`)
`flatten`	`Logical` `scalar` specifying whether to flatten the resulting `data.frame`. This means that columns with multiple values are separated to multiple columns. (Default: `FALSE`)
`...`	optional arguments: • **max.hits** `NULL` or `integer` `scalar` specifying the maximum number of results to fetch. When NULL, all results are fetched. (Default: `NULL`) • **use.cache** `Logical` `scalar` specifying whether to use cache (Default: `FALSE`) • **cache.dir** `Character` `scalar` specifying cache directory. (Default: `tempdir()`) • **clear.cache** `Logical` `scalar` specifying whether to remove and clear cache (Default: `FALSE`)

Details

With `getData`, you can fetch data from the database. Compared to `getResult`, this function is more flexible since it can fetch any kind of data from the database. However, this function returns the data without further wrangling as `list` or `data.frame` which are not optimized format for fetching data on samples.

Search results can be filtered; for example, animals can be filtered based on available samples. See [Api browser](https://www.holofooddata.org/api/docs) for information on filters. You can find help on customizing queries from [here](https://emg-docs.readthedocs.io/en/latest/api.html#customising-queries).

Value

`list` or `data.frame`

See Also

`getResult`

Examples

```
# Find genome catalogues
catalogues <- getData(type = "genome-catalogues")
head(catalogues)

# Find genomes based on certain genome catalogue iD
res <- getData(
    type = "genomes", accession.type = "genome-catalogues",
    accession = catalogues[1, "id"], max.hits = 100)
# See the data.
head(res)
# It includes for instance summary of the CAZy
# (Carbohydrate-Active enZymes) annotations as a counts per category
cazy <- res[ , grepl("annotations.cazy", colnames(res)), drop = FALSE]
head(cazy)
# Moreover, it includes a sample list. This sample list represents a
# collection of samples where the MAG was identified. Thr data has also the
# completeness of MAG in a sample.
head(res[ c("metadata.Sample_accession", "metadata.Completeness")])
```

`getMetaboLights` *Get metabolomic data from MetaboLights database*

Description

Get metabolomic data from MetaboLights database

Usage

```
getMetaboLights(study.id, output = "list", ...)

getMetaboLightsFile(study.id, file, ...)
```

Arguments

study.id	character vector specifying the study identifier of data that is going to be fetched from the MetaboLights database.
output	character scalar specifying output format. Must be "list", "TreeSE" (TreeSummarizedExperiment) or "SE" (SummarizedExperiment). (Default: "list")
...	optional arguments: • **cache.dir** Character scalar specifying directory where downloaded file is stored. (Default: tempdir()) • **timeout** Integer scalar specifying timeout in seconds for loading a file. (Default: 5*60)
file	character vector specifying the files that are being fetched.

Details

The HoloFood database primarily comprises targeted metabolomic data, omitting non-targeted metabolomic information. Nonetheless, it features URLs linking to studies within the MetaboLights database. This functionality enables users to access non-targeted metabolomic data. The getMetaboLights function returns a structured list encompassing processed data in data.frame format for study metadata, assay metadata, and assay.

The metadata includes the file names of spectra data. Those files can be loaded with getMetaboLightsFile. Alternatively, once you've identified the study and files to fetch, you can refer to this [vignette](https://rformassspectrometry.g data-from-metabolights) for instructions on loading the data directly into an MsExperiment object, specifically designed for metabolomics spectra data.

Value

list, SummarizedExperiment or TreeSummarizedExperiment

See Also

getResult getData

Examples

```
# This example is not run, because the server fails to respond sometimes.
if( FALSE ){
    res <- getMetaboLights("MTBLS4381")
    file_paths <- getMetaboLightsFile(
        study.id = "MTBLS4381",
        file = res[["assay_meta"]][["Raw Spectral Data File"]]
        )
    # Get data as SummarizedExperiment
```

```
    se <- getMetaboLights("MTBLS3540", output = "SE")
}
```

`getResult` *Get data on samples from HoloFood database*

Description

Get data on samples from HoloFood database

Usage

```
getResult(accession, ...)
```

Arguments

`accession`	`Character vector` specifying the accession IDs of type samples.
`...`	optional arguments: • **use.cache** `Logical scalar` specifying whether to use cache. Note that when `get.metabolomic = TRUE` is specified, the file from the MetaboLights is stored in the local system to the location specified by `cache.dir` despite of the value of `use.cache`. (Default: `FALSE`) • **cache.dir** `Character scalar` specifying cache directory. (Default: `tempdir()`) • **clear.cache** `Logical scalar` specifying whether to use.cache (Default: `FALSE`) • **assay.type** `Character scalar` specifying the name of assay in resulting `TreeSummarizedExperiment` object. (Default: `"counts"`) • **get.metabolomic** `Logical scalar` specifying whether to retrieve processed metabolomic data from MetaboLights database. For retrieving spectra data, refer to `getMetaboLights` documentation. (Default: `FALSE`)

Details

With `getResult`, you can fetch data on samples from the HoloFood database. Compared to `getData`, this function is more convenient for fetching the samples data because it converts the data to `MultiAssayExperiment` where different omics are stored as `TreeSummarizedExperiment` objects which are optimized for downstream analytics. Columns of returned `MultiAssayExperiment` are individual animals. These columns are linked with individual samples that are stored in `TreeSummarizedExperiment` objects.

The HoloFood database lacks non-targeted metabolomic data but they can be fetched from MetaboLights resource. Certain datasets include processed features. Those datasets can be retrieved with the function `getResult` which integrates metabolomic data with other datasets from HoloFood.

Furthermore, while the HoloFoodR database does not include metagenomic assembly data, users can access such data from the MGnify database. The MGnifyR package provides a convenient interface for accessing this database. By employing `MGnifyR::getResult()`, users can obtain data formatted as a `MultiAssayExperiment` object, containing multiple `TreeSummarizedExperiment` objects. Consequently, data from both HoloFood and MGnify databases are inherently compatible for subsequent downstream analysis.

Value

MultiAssayExperiment

See Also

getData TreeSummarizedExperiment MultiAssayExperiment MGnifyR:getResult

Examples

```
# Find samples on certain animal
samples <- doQuery("samples", animal_accession = "SAMEA112904746")

# Get the data
mae <- getResult(samples[["accession"]])
mae
```

HoloFoodR HoloFoodR *package*

Description

HoloFoodR implements an interface to the EBI HoloFood database. See the vignette for a general introduction to this package, [about HoloFood](https://www.holofood.eu/) for general HoloFood information, and [API documentation](https://docs.holofooddata.org/api.html) for details on the JSONAPI implementation.

Author(s)

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

Authors:

- Artur Sannikov <arsann@utu.fi> (ORCID)
- Leo Lahti <leo.lahti@iki.fi> (ORCID)

See Also

TreeSummarizedExperiment MultiAssayExperiment

Index