Package: MsBackendMassIVE (via r-universe)

November 25, 2026

Title Retrieve Mass Spectrometry Data from MassIVE

Version 1.0.0

Description MassIVE is one of the main public repositories for storage of metabolomics experiments. The MsBackendMassIVE package provides functionality to retrieve and represent mass spectrometry (MS) data from MassIVE. Data files are downloaded and cached locally avoiding repetitive downloads. MS data from metabolomics experiments can thus be directly and seamlessly integrated into R-based analysis workflows with the Spectra and MsBackendMassIVE package.

Depends R (>= 4.2.0), Spectra (>= 1.15.12)

Imports httr2, xml2, ProtGenerics, BiocFileCache, S4Vectors, methods, MsCoreUtils (>= 1.23.9), progress, jsonlite, rvest

Suggests testthat, rmarkdown, mzR, knitr, BiocStyle

License Artistic-2.0

Encoding UTF-8

VignetteBuilder knitr

BugReports https://github.com/RforMassSpectrometry/MsBackendMassIVE/issues

URL https://github.com/RforMassSpectrometry/MsBackendMassIVE

biocViews Infrastructure, MassSpectrometry, Metabolomics, DataImport, Proteomics

Roxygen list(markdown=TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libssl-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:30 UTC

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

RemoteRef RELEASE_3_24

RemoteSha ea6013a0479897a0f1481bc538d90baf55488c6d

Contents

GNPS2-utils 2
MassIVE-utils 3
MsBackendMassIVE 6

Index **10**

`GNPS2-utils` *Query the GNPS2 datasetchache resource*

Description

The GNPS2 *datasetcache* collects and provides general information on data sets/experiments with their related MS data files for various repositories including MassIVE and MetaboLights. The resource is updated on a regular basis. *MsBackendMassIVE* provides utility functions to retrieve information from this resource directly in R:

- `gnps2_query()`: query the datasetcache for metadata of data sets with the provided (MassIVE) dataset ID(s). Returns a `data.frame` with one row per file entry from the *filename* table.
- `gnps2_usi_download_link()`: retrieve the download link for a specific USI. Returns a `character(1)` with the link.

Usage

```
gnps2_query(id = character(), usi_pattern = "*", filepath_pattern = "*")

gnps2_usi_download_link(usi = character())
```

Arguments

`id`	for `gnps2_query()`: `character` with the ID(s) of the MassIVE data set(s).
`usi_pattern`	for `gnps2_query()`: `character(1)` defining a pattern to filter the *USI*, such as `usi_pattern = ".mzML"` to retrieve the USI of all files of the data set (i.e., files with extension `".mzML"`). This parameter is passed to the `grepl()` function.
`filepath_pattern`	for `gnps2_query()`: `character(1)` defining a pattern to filter the `filepath`, such as `filepath_pattern = "metadata"` to retrieve the `filepath` of all files of the data set (i.e., files with metadata info). This parameter is passed to the `grepl()` function.
`usi`	for `gnps2_usi_download_link()`: `character(1)` with the USI of a file in GNPS2 DB.

Details

The `gnps2_query()` function queries the GNPS2 Datasette API at `https://datasetcache.gnps2.org/datasette/databa` by executing a SQL query on the *filename* table filtered by dataset IDs. It returns all matching file metadata records. This metadata is used by downstream functions to determine the FTP paths and to download files. The `gnps2_usi_download_link()` makes a GET request to the GNPS2 dashboard to get the download link of a specific USI.

Value

- For `gnps2_query()`: a `data.frame` with the all information in the GNPS2 datasetcache database for the data set IDs provided.
- For `gnps2_usi_download_link()`: a `character(1)` with the downlaod link of the USI.

Note

The Datasette API enforces a maximum limit of 50,000 rows per query. Longer results will thus be truncated.

Author(s)

Gabriele Tomè

Examples

```
## Get the GNPS2 table to the data set MSV000080547
gnps2_query("MSV000080547")

## Get link for an USI
gnps2_usi_download_link("mzspec:MTBLS39:FILES/AM063A.cdf")
```

`MassIVE-utils` *Utility functions for the MassIVE repository*

Description

MassIVE (Mass Spectrometry Interactive Virtual Environment) is a community resource developed by the NIH-funded Center for Computational Mass Spectrometry to promote the global, free exchange of mass spectrometry data. MassIVE supports deposition of both proteomics and metabolomics experiments, and is a full member of the ProteomeXchange consortium, allowing datasets to be assigned ProteomeXchange accessions to satisfy publication requirements. Submitted data can include raw mass spectrometry files, identification results, and quantification data. The repository also provides online workflows for reanalysis of public datasets and tools for comparison of identification results across datasets.

Each experiment in MassIVE is identified with its unique identifier, starting with *MSV* followed by a number. The data (raw MS files, metadata, and result files) of a dataset are available for public download and online browsing once the dataset has been made public by its submitter.

The functions listed here allow to query and retrieve information of a data set/experiment from MassIVE.

- `massive_ftp_path()`: returns the FTP path for a provided MassIVE ID. If the MassIVE ID does not exist the function throws an error. With `mustWork = TRUE` (the default) the function throws an error either because the data set does not exist in GNPS2 DB (No mzML/CDF/mzXML files available) or no internet connection is available. The function returns a `character(1)` with the FTP path to the data set folder.
- `massive_cached_data_files()`: lists locally cached data files from MassIVE. Since this function evaluates only local content it does not require an internet connection. With the default parameters all available data files are listed. The parameters can be used to restrict the lookup.
- `massive_list_files()`: returns the available files (and directories) for the specified MassIVE data set (i.e., the FTP directory content of the data set). The function returns a `character` vector with the relative file names to the absolute FTP path (`massive_ftp_path()`) of the data set. Parameter `pattern` allows to filter the file names and define which file names should be returned.
- `massive_sync_data_files()`: synchronize data files of a specified MassIVE data set eventually downloading and locally caching them. Parameter `fileName` allows to specify names of selected data files to sync.
- `massive_download_file()`: download files from the MassIVE repository for a specified MassIVE dataset. Use `pattern` to filter files by name using a regular expression (downloads all files by default). Use `fileName` to specify one or more exact file names to download. Use `path` to set the destination directory for downloaded files.
- `massive_param_file()`: download and parse the `params.xml` files of the data set. The function return a `data.frame` or a `list` of `data.frame` with 2 columns (Parameter Name, Value). Use `fileName` to parse additional `xml` files in the data.set.
- `massive_number_files()`: return the number of data files in a specified MassIVE data set. Use `pattern` to filter files by name using a regular expression, default: `pattern = "mzML$|CDF$|cdf$|mzXML$"`.
- `massive_delete_cache()`: removes all local content for the MassIVE data set with ID `massiveId`. This will delete eventually present locally cached data files for the specified data set. This does not change any other data eventually present in the local `BiocFileCache`.

Usage

```
massive_ftp_path(x = character(), mustWork = TRUE)

massive_list_files(x = character(), pattern = NULL)

massive_download_file(
  massiveId = character(),
  pattern = "*",
  fileName = character(),
  path = "./",
  overwrite = FALSE
)

massive_param_file(massiveId = character(), fileName = "params.xml")
```

```
massive_number_files(
  massiveId = character(),
  pattern = "mzML$|CDF$|cdf$|mzXML$"
)

massive_sync_data_files(
  massiveId = character(),
  pattern = "mzML$|CDF$|cdf$|mzXML$",
  fileName = character()
)

massive_cached_data_files(
  massiveId = character(),
  pattern = "*",
  fileName = character()
)

massive_delete_cache(massiveId = character())
```

Arguments

x	`character(1)` with the ID of the MassIVE data set (usually starting with a *MSV* followed by a number).
mustWork	for `massive_ftp_path()`: `logical(1)` whether the validity of the path should be verified or not. By default (with `mustWork = TRUE`) the function throws an error if either the data set does not exist or if the folder can not be accessed (e.g. if no internet connection is available).
pattern	for `massive_list_files()`, `massive_sync_data_files()`, `massive_cached_data_files()`, `massive_donwload_file()`, and `massive_number_files()`: `character(1)` defining a pattern to filter the file names, such as `pattern = "mzML$"` to retrieve the file names of all files of the data set (i.e., files with extension `"mzML"`). This parameter is passed to the `grepl()` function.
massiveId	`character(1)` with the ID of a single MassIVE data set/experiment.
fileName	for `massive_sync_data_files()`, `massive_cached_data_files()` and `massive_download_file()`: optional `character` defining the names of specific data files of a data set that should be downloaded and cached.
path	for `massive_download_file()`: optional `character` defining the directory where download the files.
overwrite	for `massive_download_file()`: `logical(1)` whether existing files should be overwritten. Defaults to `FALSE`, in which case files that already exist in `path` are skipped.

Details

Data retrieval follows three main steps. First, the package queries the GNPS2 DB to list all files for the provided `massiveId`, filtering them by `filePattern` to retain only formats supported by `MsBackendMzR` (mzML, CDF, mzXML). Second, the FTP link is retrieved from MassIVE. If

the requested files are in the ccms_peak folder, the FTP link is updated by changing the volume from the project-specific one to volume z01, which contains the ccms_peak folder for all projects. Each file is then downloaded from the MassIVE FTP server and cached locally. Files already present in the cache are not re-downloaded. Third, the cached local paths are passed to Spectra::MsBackendMzR() to read and index the spectral data. Two additional per-spectrum variables are populated: "massive_id" and "data_file". When offline = TRUE, the remote query is skipped and only previously cached content is used.

Value

- For massive_ftp_path(): character(1) with the ftp path to the specified data set on the MassIVE ftp server.
- For massive_list_files(): character with the names of the files in the data set's base ftp directory.
- For massive_sync_data_files() and massive_cached_data_files(): a data.frame with the MassIVE ID, the name(s) and remote and local file names of the synchronized data files sorted by filename ("rpath").
- For massive_number_files(): integer(1) with the number of data files in the data set.

Author(s)

Johannes Rainer, Philippine Louail, Gabriele Tomè

Examples

```
## Get the FTP path to the data set MSV000080547
massive_ftp_path("MSV000080547")

## Retrieve available files (and directories) for the data set MSV000080547
massive_list_files("MSV000080547")

## Retrieve the available .mzML files.
mzMLfiles <- massive_list_files("MSV000080547", pattern = "mzML$")
mzMLfiles

## Download parameter file for the data set MSV000080547
massive_download_file("MSV000080547", pattern = "params.xml",
                      path = tempdir())
```

MsBackendMassIVE *MsBackend representing MS data from MassIVE*

Description

MsBackendMassIVE retrieves and represents mass spectrometry (MS) data from proteomics and metabolomics experiments stored in the MassIVE (Mass Spectrometry Interactive Virtual Environment) repository, a community resource developed by the NIH-funded Center for Computational Mass Spectrometry at UC San Diego. The backend directly extends the Spectra::MsBackendMzR backend from the *Spectra* package and hence supports MS data in mzML, netCDF and mzXML format. Data in other formats can not be loaded with MsBackendMassIVE. Upon initialization with the backendInitialize() method, the MsBackendMassIVE backend downloads and caches the MS data files of a dataset locally, avoiding repeated download of the data. The local data cache is managed by Bioconductor's *BiocFileCache* package. See the help and vignettes from that package for details on cached data resources. Additional utility functions for management of cached files are also provided by *MsBackendMassIVE*. See help for massive_cached_data_files() for more information.

Usage

```
MsBackendMassIVE()

## S4 method for signature 'MsBackendMassIVE'
backendInitialize(
  object,
  massiveId = character(),
  filePattern = "mzML$|CDF$|cdf$|mzXML$",
  offline = FALSE,
  ...
)

## S4 method for signature 'MsBackendMassIVE'
backendRequiredSpectraVariables(object, ...)

massive_sync(x, offline = FALSE)
```

Arguments

object	an instance of MsBackendMassIVE.
massiveId	character(1) with the ID of a single MassIVE data set/experiment.
filePattern	character with the pattern defining the supported (or requested) file types. Defaults to filePattern = "mzML$\|CDF$|cdf$\|mzXML$" hence restricting to mzML, CDF and mzXML files which are supported by *Spectra*'s MsBackendMzR backend.
offline	logical(1) whether only locally cached content should be evaluated/loaded.
...	additional parameters; currently ignored.
x	an instance of MsBackendMassIVE.

Details

File names for data files are by default extracted from the column "filepath" of the GNPS2 database.

The backend uses the BiocFileCache package for caching of the data files. These are stored in the default local *BiocFileCache* cache along with additional metadata that includes the MassIVE ID. Note that at present only MS data files in *mzML*, *CDF* and *mzXML* format are supported.

The `MsBackendMassIVE` backend defines and provides additional spectra variables `"massive_id"` and `"data_file"` that list the MassIVE ID, and the original data file name on the MassIVE ftp server for each individual spectrum. The `"data_file"` can be used for the mapping between the experiment's samples and the individual data files, respective their spectra.

The `MsBackendMassIVE` backend is considered *read-only* and does thus not support changing *m/z* and intensity values directly.

Value

- For `MsBackendMassIVE()`: an instance of `MsBackendMassIVE`.
- For `backendInitialize()`: an instance of `MsBackendMassIVE` with the MS data of the specified MassIVE data set.
- For `backendRequiredSpectraVariables()`: `character` with spectra variables that are needed for the backend to provide the MS data.
- For `massive_sync()`: the input `MsBackendMassIVE` with the paths to the locally cached data files being eventually updated.

Initialization and loading of data

New instances of the class can be created with the `MsBackendMassIVE()` function. Data is loaded and initialized using the `backendInitialize()` function which can be configured with parameters `massiveId` and `filePattern`. `massiveId` must be the ID of a **single** (existing) MassIVE dataset (e.g. `"MSV000079514"`). Optional parameter `filePattern` defines the pattern used to filter the file names of the MS data files. It defaults to data files with file endings of supported MS data formats. `backendInitialize()` requires an active internet connection as the function first compares the remote file content to the locally cached files and eventually synchronizes changes/updates. This can be skipped with `offline = TRUE` in which case only locally cached content is queried.

The `backendRequiredSpectraVariables()` function returns the names of the spectra variables required for the backend to provide the MS data.

The `massive_sync()` function can be used to *synchronize* the local data cache and ensure that all data files are locally available. The function will check the local cache and eventually download missing data files from the MassIVE repository.

Note

To account for high server load and eventually failing or rejected downloads from the MassIVE FTP server (`ftp://massive-ftp.ucsd.edu/`), the download functions repeatedly retry to download a file. An error is thrown if the download fails for 5 consecutive attempts. Between each attempt, the function waits for an increasing time period (5 seconds between the first and second and 10 seconds between the 2nd and 3rd attempt). This time period can also be configured with the `"massive.sleep_mult"` option, which defines the *sleep time multiplicator* (defaults to 5).

Author(s)

Gabriele Tomè, Philippine Louail, Johannes Rainer

Examples

```
library(MsBackendMassIVE)

## List files of a MassIVE data set
massive_list_files("MSV000080547")

## Initialize a MsBackendMassIVE representing all MS data files of
## the data set with the ID "MSV000080547". This will download and cache all
## files and subsequently load and represent them in R.

be <- backendInitialize(MsBackendMassIVE(), "MSV000080547",
                        filePattern = "11.mzML$")
be

## The `massive_sync()` function can be used to ensure that all data files
## are available locally. This function will eventually download missing data
## files or update their paths.
be <- massive_sync(be)
```

Index

backendInitialize,MsBackendMassIVE-method
(MsBackendMassIVE), 6
backendRequiredSpectraVariables,MsBackendMassIVE-method
(MsBackendMassIVE), 6

GNPS2-utils, 2
gnps2_query (GNPS2-utils), 2
gnps2_usi_download_link (GNPS2-utils), 2
grepl(), 2, 5

MassIVE-utils, 3
massive_cached_data_files
(MassIVE-utils), 3
massive_cached_data_files(), 7
massive_delete_cache (MassIVE-utils), 3
massive_download_file (MassIVE-utils), 3
massive_ftp_path (MassIVE-utils), 3
massive_list_files (MassIVE-utils), 3
massive_number_files (MassIVE-utils), 3
massive_param_file (MassIVE-utils), 3
massive_sync (MsBackendMassIVE), 6
massive_sync_data_files
(MassIVE-utils), 3
MsBackendMassIVE, 6
MsBackendMassIVE-class
(MsBackendMassIVE), 6

Spectra::MsBackendMzR, 7
Spectra::MsBackendMzR(), 6