Package: SpectraStash (via r-universe)

June 18, 2026

Title Serialize and restore Spectra objects in interoperable file formats

Version 0.97.4

Description The serialization mechanism of R allows to save and load R data objects in a binary format, that can however not be read by other programming languages or software. The SpectraStash package implements the serialization methods from the MsStash package for Spectra objects and its MsBackend classes. A variety of different file formats and types, including HDF5 and JSON-based formats defined by the Bioconductor *alabaster* package are supported. The file type can be defined and configured through a second argument `param` of the save/read methods.

Depends R (>= 4.6.0), MsStash

Imports methods, Spectra (>= 1.23.2), utils, S4Vectors, alabaster.base, jsonlite

Suggests BiocStyle, MsDataHub, testthat, mzR

License GPL-3

Encoding UTF-8

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

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

biocViews Infrastructure, MassSpectrometry, Metabolomics, DataImport, Proteomics

Roxygen list(markdown=TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev libssl-dev libnode-dev zlib1g-dev

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

Date/Publication 2026-06-18 08:58:46 UTC

RemoteUrl https://github.com/rformassspectrometry/SpectraStash

RemoteRef HEAD

RemoteSha 24a8d18cf3ab31981b321d9d413c367ae66d7b00

Contents

`MsBackendCachedStash` *Stash for* `MsBackendCached`

Description

The Spectra::MsBackendCached backend keeps a cache of spectra metadata (spectra variables) in an internal `data.frame` adding thus support for (temporarily) changing or adding spectra variables for purely read-only `MsBackend` classes. This backend is used e.g. by the MsBackendSql::MsBackendSql backend allowing to add or replace spectra variables without affecting the content of the underlying data base used by the `MsBackendSql`.

The stash of a `MsBackendCached` contains therefore the local spectra data cache (if present), the names of the available spectra variables and the total number of spectra. Supported stash formats are listed in the sections below.

Usage

```
## S4 method for signature 'MsBackendCached,PlainTextParam'
saveMsObject(object, param, ...)

## S4 method for signature 'MsBackendCached,PlainTextParam'
readMsObject(object, param, ...)

## S4 method for signature 'MsBackendCached'
saveObject(x, path, ...)

## S4 method for signature 'MsBackendCached,AlabasterParam'
saveMsObject(object, param, ...)

## S4 method for signature 'MsBackendCached,AlabasterParam'
readMsObject(object, param, ...)
```

Arguments

`object`	An `MsBackendCached` object.
`param`	Either a `PlainTextParam` or `AlabasterParam`.
`...`	Currently ignored.
`x`	An `MsBackendCached` object.
`path`	For `saveObject()`: `character(1)` with the path where the object should be stored into.

Details

Notes for stash-functionality for `MsBackend` objects extending `MsBackendCached`:

- `saveMsObject()` and `saveObject()` will fail if the stash directory already exist. Thus, stash functions of backend implementations extending `MsBackendCached` should **first** call the `MsBackendCached`'s `saveMsObject()` or `saveObject()` **before** exporting their respective content to the stash directory.

Text-file format, `PlainTextParam`

The data files written into the stash are:

- *ms_backend_data.txt*: tabulator separated text file with the content of the `@localData` slot.
- *ms_backend_nspectra.txt*: the number of spectra.
- *ms_backend_spectra_variables.txt*: the names of the object's spectra variables (tabulator separated).

alabaster-based format, `AlabasterParam`

The content from all slots of the `MsBackendCached` are stored using functionality from the *alabaster.base* package into separate sub-folders of the base stash directory. These are:

- *local_data*: for the `data.frame` with the locally cached spectra variables (slot `@localData`).
- *nspectra*: (`integer(1)`) with the number of spectra.
- *spectra_variables*: (`character`) with the names of the object's spectra variables.

Author(s)

Johannes Rainer

Examples

```
library(Spectra)

## Create an empty `MsBackendCached` object
be <- MsBackendCached()

## Stash the object in alabaster-format in a temporary directory
ap <- AlabasterParam(file.path(tempdir(), "cache-stash"))
saveMsObject(be, ap)

## The content of the stash folder:
dir(file.path(tempdir(), "cache-stash"))

## Restore the object
res <- readMsObject(MsBackendCached(), ap)
res
```

MsBackendHdf5PeaksStash

MsBackendHdf5Peaks Stash

Description

MsBackendHdf5Peaks classes can be stashed to (or read from) plain text file-based or *alabaster*-based formats using the saveMsObject() and readMsObject() functions combined with the PlainTextParam and AlabasterParam parameter objects, respectively. In both cases, data files are stored into the specified directory. By default, only the backend's spectra metadata is stored in that folder. Setting parameter consolidate = TRUE will also copy the HDF5-format peaks data files (containing the *m/z* and intensity values) of the backend into the folder generating a self-consistent stash.

Details on the stored files are provided in the sections below.

Usage

```
## S4 method for signature 'MsBackendHdf5Peaks,PlainTextParam'
saveMsObject(object, param, consolidate = FALSE)

## S4 method for signature 'MsBackendHdf5Peaks,PlainTextParam'
readMsObject(object, param, spectraPath = character())

## S4 method for signature 'MsBackendHdf5Peaks'
saveObject(x, path, consolidate = FALSE, ...)

## S4 method for signature 'MsBackendHdf5Peaks,AlabasterParam'
saveMsObject(object, param, consolidate = FALSE)

## S4 method for signature 'MsBackendHdf5Peaks,AlabasterParam'
readMsObject(object, param, spectraPath = character())
```

Arguments

object	An MsBackendHdf5Peaks object.
param	Either a PlainTextParam or AlabasterParam.
consolidate	logical(1) whether in addition to the spectra metadata also the peaks data file (in HDF5 format) should be stored in the stash folder. Default is consolidate = FALSE.
spectraPath	For readMsObject(): optional character(1) with the path to the peaks data in HDF5-format (in case they are on longer available in the folder referred to by the original stashed MsBackendHdf5Peaks object).
x	An MsBackendHdf5Peaks object.
path	For saveObject(): character(1) with the path where the object should be stored in.
...	Currently ignored.

Details

A `MsBackendHdf5Peaks` stash will by default only contain the spectra metadata (spectra variables) but no *m/z* and intensity values. The reference to the original HDF5-format peaks data files are stored as spectra variable *dataStorage*. If these files were moved or if the stash was copied to a different computer, the path to these original data files needs to be provided to the `readMsObject()` function with parameter `spectraPath`. Alternatively, with `consolidate = TRUE`, it is also possible to **copy** the peaks data files to the stash directory generating a self-contained data stash. Note however that in this case two copies of all data files exist (in the original location **and** the stash directory).

Value

`readMsObject()` returns an Spectra::MsBackendHdf5Peaks‘ object.

Text-file format, `PlainTextParam`

The `saveMsObject()` function with the `PlainTextParam` stores the spectra metadata (spectra variables) of an `MsBackendHdf5Peaks` to a plain tabulator delimited text file with the name *ms_backend_spectra_data.txt* in the directory specified with parameter `path` of the `PlainTextParam` object. Depending on parameter `consolidate` also the peaks data files (in HDF5 format) will be copied to the stash folder (with `consolidate = TRUE`).

abalbaster-based format, `AlabasterParam`

With `AlabasterParam`, the spectra metadata will be exported (or imported) through the *alabaster* framework. Similar to the `PlainTextParam`, `consolidate = TRUE` will also copy the HDF5-format peaks data files to the stash directory.

Author(s)

Johannes Rainer

Examples

```
library(Spectra)
library(SpectraStash)
## Create an example MsBackendHdf5Peaks backend from a single mzML file
library(MsDataHub)
tmp <- backendInitialize(MsBackendMzR(), X20171016_POOL_POS_1_105.134.mzML())
be_h5 <- backendInitialize(MsBackendHdf5Peaks(), data = spectraData(tmp),
    hdf5path = file.path(tempdir(), "h5_backend"))
be_h5

d <- file.path(tempdir(), "example_hdf5")
ptp <- PlainTextParam(path = d)

## Store the object into the stash, including the peaks data files.
saveMsObject(be_h5, ptp, consolidate = TRUE)

## List the content of the folder: ms_backend_spectra_data.txt file
```

```
## with the spectra metadata and an HDF5 file with the peaks data:
dir(d)

## Restore the stashed object
res <- readMsObject(MsBackendHdf5Peaks(), ptp)

## Store the object in an alabaster-format stash
d <- file.path(tempdir(), "example_hdf5_2")

ap <- AlabasterParam(d)
saveMsObject(be_h5, ap)

## Check the content of the stash; with the default (`consolidate = FALSE`)
## no HDF5 data file was moved.
dir(d)

## Restore the object again
res <- readMsObject(MsBackendHdf5Peaks(), ap)
res
```

MsBackendMzRStash *MsBackendMzR Stash*

Description

MsBackendMzR classes can be stashed to (or read from) plain text file-based or *alabaster*-based formats using the saveMsObject() and readMsObject() functions combined with the PlainTextParam and AlabasterParam parameter objects, respectively. Setting parameter consolidate = TRUE in the saveMsObject() or saveObject() function will copy also the original MS data files into the folder generating a self-consistent stash.

Additional properties of the stash formats are described in detail in the sections below.

Usage

```
## S4 method for signature 'MsBackendMzR,PlainTextParam'
saveMsObject(object, param, consolidate = FALSE)

## S4 method for signature 'MsBackendMzR,PlainTextParam'
readMsObject(object, param, spectraPath = character())

## S4 method for signature 'MsBackendMzR'
saveObject(x, path, consolidate = FALSE, ...)

## S4 method for signature 'MsBackendMzR,AlabasterParam'
saveMsObject(object, param, consolidate = FALSE)

## S4 method for signature 'MsBackendMzR,AlabasterParam'
readMsObject(object, param, spectraPath = character())
```

Arguments

object	An MsBackendMzR object.
param	Either a PlainTextParam or AlabasterParam.
consolidate	logical(1) whether in addition to the spectra metadata also the original MS data files should be stored in the stash folder. Default is consolidate = FALSE.
spectraPath	For readMsObject(): optional character(1) with the path to the MS data files (mzML, mzXML or CDF) in case they are on longer available in the folder referred to by the original stashed MsBackendMzR object.
x	An MsBackendMzR object.
path	For saveObject(): character(1) with the path where the object should be stored in.
...	Currently ignored.

Details

MsBackendMzR objects don't contain any peaks data (i.e., *m/z* and intensity values) but retrieve these from the original MS data files (in mzML, mzXML or CDF format). Unless consolidate = TRUE is used, a MsBackendMzR stash will therefore only contain the spectra metadata (i.e., the spectra variables) but no peaks data. The reference to the original MS data files is stored as spectra variable *dataStorage* and if the files are no longer available in the directory specified by *dataStorage* the restored object will not be valid, unless the new location is provided with parameter spectraPath.

Value

readMsObject() returns an Spectra::MsBackendMzR‘ object.

Text-file format, PlainTextParam

The saveMsObject() function with the PlainTextParam stores the spectra metadata (spectra variables) of an MsBackendMzR to a plain tabulator delimited text file with the name *ms_backend_spectra_data.txt* in the directory specified with parameter path of the PlainTextParam object. Importantly, the peaks data (the *m/z* and intensity values) are **not** exported with saveMsObject().

readMsObject() restores a previously stashed MsBackendMzR object from the directory specified with parameter path of the PlainTextParam.

The additional parameter spectraPath of readMsObject() allows to define the path to the MS data files containing the full MS data (i.e., the mzML, mzXML or CDF files referred to by the MsBackendMzR).

alabaster-based format, AlabasterParam

The saveMsObject() with an AlabasterParam parameter object stashes the provided MsBackendMzR object in an *alabster*-based format into the directory defined with argument param of the AlabasterParam. readMsObject() with AlabasterParam restores a previously stashed MsBackend object. Optional parameter spectraPath allows to specify the storage path of the MS data files referenced by the MsBackendMzR (in case they are no longer in the same directory when saving the object).

In addition, the *alabaster* methods saveObject() and readObject() can be used to save and read MsBackendMzR objects.

Author(s)

Philippine Louail, Johannes Rainer

Examples

```
library(SpectraStash)
library(Spectra)
library(MsDataHub)

## Create a MsBackendMzR from test data
be <- backendInitialize(MsBackendMzR(), PestMix1_DDA.mzML())
be

## Define a folder where to stash the object
pth <- file.path(tempdir(), "mzr_stash")

## Stash the object to this folder in a plain text-based format
saveMsObject(be, PlainTextParam(pth))

## Restore the stashed object
res <- readMsObject(MsBackendMzR(), PlainTextParam(pth))
res

## Clean-up
unlink(pth, recursive = TRUE)

## Store the data in *alabaster* format including also the original MS
## data files (`consolidate = TRUE`)
saveMsObject(be, AlabasterParam(pth), consolidate = TRUE)

## Get the directory content of the stash folder:
dir(pth)

## Restore the object
res <- readMsObject(MsBackendMzR(), AlabasterParam(pth))
res

## If the data is exported with `consolidate = FALSE` (the default), the
## new location of MS data files could be provided with parameter
## `spectraPath` of the `readMsObject()` function in case they are no
## longer in the path referenced by the stashed object.

## Clean-up
unlink(pth, recursive = TRUE)
```

SpectraStash *Spectra Stash*

Description

Spectra objects can be stashed to (or read from) plain text file-based or *alabaster*-based formats using the saveMsObject() or readMsObject() functions configured with the PlainTextParam or AlabasterParam parameter objects, respectively. In both cases, the data from the Spectra object is stored into the *stash* directory defined with the parameter object. Depending on the used MS backend (and parameters used), the stash can also contain the MS data files.

At present, Spectra objects using one of the following MS data backends are supported:

- MsBackendMzR: see MsBackendMzRStash for details and options.
- MsBackendHdf5Peaks: see MsBackendHdf5PeaksStash for details and options.

The data backend of any Spectra backend can eventually be changed to one of the above backends using the Spectra::setBackend() method to support saving the object into a Spectra stash. Support for additional backends respectively data representations might also be provided by separate R packages.

Details on the stash formats are provided in the respective sections below.

Usage

```
## S4 method for signature 'Spectra,PlainTextParam'
saveMsObject(object, param, ...)

## S4 method for signature 'Spectra,PlainTextParam'
readMsObject(object, param, ...)

## S4 method for signature 'Spectra'
saveObject(x, path, ...)

## S4 method for signature 'Spectra,AlabasterParam'
saveMsObject(object, param, ...)

## S4 method for signature 'Spectra,AlabasterParam'
readMsObject(object, param, ...)
```

Arguments

object	A Spectra object.
param	Either a PlainTextParam or AlabasterParam.
...	additional arguments passed to the saveMsObject or readMsObject method of the Spectra's MsBackend, such as for example consolidate or spectraPath. See the saveMsObject() and readMsObject() documentation of the used MsBackend class for information on supported arguments.
x	A Spectra object.
path	For saveObject(): character(1) with the path where the object should be stored in.

Value

`readMsObject()` returns a Spectra::Spectra object.

Text-file format, `PlainTextParam`

For this format, the data content of a `Spectra` object is stored into the files:

- *spectra_slots.txt*: plain text file containning the *processing queue* variables (separated by a `"|"`), the processing log messages (separated by a `"|"`), the processing chunk size and the `MsBackend` class used.
- *spectra_processing_queue.json*: the object's processing queue serialized in JSON format. It can be unserialized using `jsonlite::unserializeJSON()`.

For information on the MS backend's data see the respective documentation.

abalbaster-based format, `AlabasterParam`

With `AlabasterParam`, the `Spectra` object will be exported (or imported) through the *alabaster* framework as a set of JSON and/or HDF5 files. The content of each slot is stored to a separate file with the name matching the slot name (converted to *snake_case*). The object's `MsBackend` is stored into a sub-folder *backend* within the stash folder.

For information on the MS backend's data stash see the respective documentation.

Author(s)

Johannes Rainer, Philippine Louail

Examples

```
## Create a Spectra object from two example MS data files (from MsDataHub)
library(Spectra)
library(MsDataHub)
s <- Spectra(
    c(X20171016_POOL_POS_1_105.134.mzML(),
      X20171016_POOL_POS_3_105.134.mzML()))
s

## Filter the intensities of the Spectra removing peaks with an intensity
## below 100
s <- filterIntensity(s, intensity = 100)

## Define the format and location of the `Spectra` stash: use the
## *alabaster*-based format and store the stash in a folder named
## *spectra_stash* in a temporary directory
ap <- AlabasterParam(file.path(tempdir(), "spectra_stash"))

## Stash the `Spectra` object copying in addition the MS data files into the
## stash (`consolidate = TRUE`).
saveMsObject(s, ap, consolidate = TRUE)

## Show the content of the stash
```

```
dir(ap@path)

## Read the `Spectra` object from the stash:
res <- readMsObject(Spectra(), ap)
res

## It is also possible to read individual contents from the stash. The
## directory *backend* contains for example the stashed `MsBackend` of the
## `Spectra` object. To read only the `MsBackend`:
ap2 <- AlabasterParam(file.path(tempdir(), "spectra_stash", "backend"))
b <- readMsObject(MsBackendMzR(), ap2)
b

## Alternatively, that data can also be read directly with the `readObject()`
## method from the *alabaster.base* package:
library(alabaster.base)
b <- readObject(file.path(ap@path, "backend"))
b
```

Index