Package 'SpectraQL'

November 10, 2026

Title MassQL support for Spectra

Version 1.7.1

Description The Mass Spec Query Language (MassQL) is a domain-specific language enabling to express a query and retrieve mass spectrometry (MS) data in a more natural and understandable way for MS users. It is inspired by SQL and is by design programming language agnostic. The SpectraQL package adds support for the MassQL query language to R, in particular to MS data represented by Spectra objects. Users can thus apply MassQL expressions to analyze and retrieve specific data from Spectra objects.

Depends R (>= 4.4.0), ProtGenerics (>= 1.25.1)

Imports Spectra (>= 1.5.6), MsCoreUtils, methods

Suggests testthat, MsDataHub, roxygen2, rmarkdown, knitr, S4Vectors, BiocStyle, mzR

License Artistic-2.0

LazyData false

VignetteBuilder knitr

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

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

biocViews Infrastructure, Proteomics, MassSpectrometry, Metabolomics

Encoding UTF-8

Roxygen list(markdown=TRUE)

RoxygenNote 7.3.3

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

git_branch devel

git_last_commit cbc3f30

git_last_commit_date 2026-04-30

Repository Bioconductor 3.24

Date/Publication 2026-11-09

Author Johannes Rainer [aut, cre] (ORCID: <https://orcid.org/0000-0002-6977-7147>), Andrea Vicini [aut], Sebastian Gibb [ctb] (ORCID: <https://orcid.org/0000-0001-7406-4443>)

Maintainer Johannes Rainer <Johannes.Rainer@eurac.edu>

Contents

query 2

Index **5**

`query` *Query a Spectra object using MassQL*

Description

The `query` function allows to query and subset/filter a `Spectra` object using a Mass Spec Query Language MassQL expression.

A MassQL query is expressed in the form `"QUERY <type of data> WHERE <condition> AND <condition> FILTER <filter> AND <filter>"`, multiple *conditions* and *filters* can be combined with logical *and* operations. In the MassQL definition, *conditions* subsets the data to specific spectra while *filter* restricts the data within a spectrum. Note that at present MassQL *filters* are not supported. Also note that MassQL queries are interpreted case insensitive in `SpectraQL`.

See also the package vignette for more details.

Usage

```
## S4 method for signature 'Spectra'
query(x, query = character(), ...)
```

Arguments

`x`	The `Spectra` object to query.
`query`	`character(1)` with the MassQL query.
`...`	currently ignored.

Value

Depending on the `<type of data>` part of the MassQL query.

Type of data

The `"<type of data>"` allows to define which data should be extracted from the selected spectra. MassQL defines *type of data* being `"MS1DATA"` or `"MS2DATA"` to retrieve data from MS1 or MS2 scans. By default peak data will be returned, but in addition, MASSQL defines additional functions that can be applied to modify the data or select different data to be returned. In addition *SpectraQL* defines the special type of data `"*"` which will return the results as a `Spectra` object. *SpectraQL* supports:

- `"*"`: select all data and return the data subset as a `Spectra::Spectra()` object.
- `"MS1DATA"`: return the `Spectra::peaksData()` from all selected **MS1** spectra, i.e. a `list` with two column matrices with the peaks' m/z and intensity values.
- `"MS2DATA"`: return the `Spectra::peaksData()` from all selected **MS2** spectra, i.e. a `list` with two column matrices with the peaks' m/z and intensity values.
- `"scaninfo(MS1DATA)"`, `"scaninfo(MS2DATA)"`: return the `Spectra::spectraData()` of all selected spectra.
- `"scansum(MS1DATA)"`, `"scansum(MS2DATA)"`: sum of the peak intensities of the selected spectra (TIC, or XIC if combined with `"FILTER"`).

Conditions

Conditions define to which spectra the data set should be subsetted. A *condition* will subset a `Spectra` object to selected spectra, but will not (unlike *Filters*, see further below) filter peaks from a spectrum. Several conditions can be combined with `"and"` (case insensitive). The syntax for a condition is `"<condition> = <value>"`, e.g. `"MS2PROD = 144.1"`. Such conditions can be further refined by additional expressions that allow for example to define acceptable tolerances for m/z differences. `SpectraQL` supports (case insensitive):

- `"RTMIN"`: minimum retention time (in **seconds**).
- `"RTMAX"`: maximum retention time (in **seconds**).
- `"SCANMIN"`: the minimum scan number (acquisition number).
- `"SCANMAX"`: the maximum scan number (acquisition number).
- `"CHARGE"`: the charge for MS2 spectra.
- `"POLARITY"`: the polarity of the spectra (can be `"positive"`, `"negative"`, `"pos"` or `"neg"`, case insensitive).
- `"MS2PROD"` or `"MS2MZ"`: allows to select MS2 spectra that contain a peak with particular m/z value(s). See below for examples.
- `"MS2PREC"`: allows to select MS2 spectra with the defined precursor m/z value(s). See below for examples.
- `"MS1MZ"`: allows to select MS1 spectra containing peak(s) with the defined m/z value(s).
- `"MS2NL"`: allows to look for a neutral loss from precursor in MS2 spectra.

All conditions involving m/z values allow to specify a mass accuracy using the optional fields `"TOLERANCEMZ"` and `"TOLERANCEPPM"` that define the absolute and m/z-relative acceptable difference in m/z values. One or both fields can be attached to a *condition* such as `"MS2PREC=100:TOLERANCEMZ=0.1:TOLERAN` to select for example all MS2 spectra with a precursor m/z equal to 100 accepting a difference of 0.1 and 20 ppm. Note that in contrast to MassQL, the default tolarance and ppm is 0 for all calls.

Filters

Filters subset the data within spectra, i.e. select which peaks within spectra should be retrieved. *SpectraQL* supports the following filters:

- `"MS1MZ"`: filters MS1 spectra keeping only peaks with matching m/z values (tolerance can be specified with `"TOLERANCEMZ"` and `"TOLERANCEPPM"` as for conditions).
- `"MS2MZ"`: filters MS2 spectra keeping only peaks with matching m/z values (tolerance can be specified with `"TOLERANCEMZ"` and `"TOLERANCEPPM"` as for conditions).

Author(s)

Andrea Vicini, Johannes Rainer

Examples

```
## Read a data file with MS1 and MS2 spectra
library(MsDataHub)
library(Spectra)
fls <- MsDataHub::PestMix1_DDA.mzML()
sps_dda <- Spectra(fls)

## Subset to spectra measured between 300 and 400 seconds
```

```
query(sps_dda, "QUERY * WHERE RTMIN = 300 AND RTMAX = 400")

## To extract peaks data from MS1 or MS2 spectra use "MS1DATA" or "MS2DATA"
## instead of *. Note also that queries are case-insensitive.
pks <- query(sps_dda, "query ms1data where rtmin = 300 and rtmax = 400")
pks
head(pks[[1L]])

## To select (MS2) spectra with a certain precursor m/z the MS2PREC condition
## can be used. Below we extract all spectra with a precursor m/z of 99.9
## accepting also a difference of 10ppm
query(sps_dda, "QUERY * WHERE MS2PREC = 99.967:TOLERANCEPPM=10")

## It is also possible to specify multiple precursor m/z values:
query(sps_dda, "QUERY * WHERE MS2PREC = (99.967 OR 428.88):TOLERANCEPPM=10")

## To select all MS1 spectra that contain a peak with a certain m/z we can
## use the MS1MZ condition. Below we combine this with an absolute tolerance
## using TOLERANCEMZ.
query(sps_dda, "QUERY * WHERE MS1MZ = 100:TOLERANCEMZ=1")

## Using MS2DATA in combination with MS1MZ will not return any spectra.
query(sps_dda, "QUERY MS2DATA WHERE MS1MZ = 100:TOLERANCEMZ=1")

## In contrast, do select MS2 spectra containing a peak with a certain m/z
## we have to use the condition MS2PROD
query(sps_dda, "QUERY * WHERE MS2PROD = 100:TOLERANCEMZ=1")

## MS2MZ can be used as alternative to MS2PROD
query(sps_dda, "QUERY * WHERE MS2MZ = 100:TOLERANCEMZ=1")

## Select MS2 spectra containing a peak with neutral loss from
## precursor of 100 allowing a m/z relative ppm tolerance of 5)
res <- query(sps_dda, "QUERY MS2DATA WHERE MS2NL=100:TOLERANCEPPM=5")

## Combine two different conditions: selection of spectra with positive
## polarity and retention time greater than 200
res <- query(sps_dda, "QUERY * WHERE RTMIN = 200 AND POLARITY = Positive")
```

Index

query, 2
query,Spectra-method *(*query*)*, 2

Spectra::peaksData(), *2*
Spectra::Spectra(), *2*
Spectra::spectraData(), *2*