Package ‘ProtGenerics’

May 4, 2024

Title  Generic infrastructure for Bioconductor mass spectrometry packages

Description  S4 generic functions and classes needed by Bioconductor proteomics packages.

Version  1.36.0

Author  Laurent Gatto <laurent.gatto@uclouvain.be>,
        Johannes Rainer <johannes.rainer@eurac.edu>

Maintainer  Laurent Gatto <laurent.gatto@uclouvain.be>

biocViews  Infrastructure, Proteomics, MassSpectrometry

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

Depends  methods

Suggests  testthat

License  Artistic-2.0

NeedsCompilation  no

RoxygenNote  7.3.1

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

git_branch  RELEASE_3_19

git_last_commit  f1a8d20

git_last_commit_date  2024-04-30

Repository  Bioconductor 3.19

Date/Publication  2024-05-03

Contents

backendInitialize ........................................ 2
filterFeatures ........................................ 2
filterSpectra .......................................... 3
Param .................................................. 3
peaksData ............................................. 4
ProcessingStep ....................................... 4
ProtGenerics .......................................... 5
### Description

These methods are used for implementations of backends e.g. for Spectra or Chromatograms object to initialize the backend, merge backends or extract specific information from them. See the respective help pages (e.g. in the Spectra or Chromatograms packages) for information on the actual implementations of these methods.

### Usage

```
backendInitialize(object, ...)  
isReadOnly(object)  
setBackend(object, backend, ...)  
backendMerge(object, ...)  
backendBpparam(object, ...)  
backendParallelFactor(object, ...)  
supportsSetBackend(object, ...)  
```

### Arguments

- **object**
  - The backend object.
- **...**
  - Optional parameters.
- **backend**
  - A backend object.

### Description

Implementations of this generic filter function are supposed to filter features in object based on a filter criteria defined by parameter filter.

### Usage

```
filterFeatures(object, filter, ...)  
```
filterSpectra

Arguments

object  The object to filter.
filter  The filtering criteria on which object should be filtered.
...

Optional parameters.

Description

Implementations of this generic filter function are supposed to filter spectra (e.g. within a Spectra object) based on filter criteria defined with parameter filter.

Usage

filterSpectra(object, filter, ...)

Arguments

object  The object to filter.
filter  The filtering criteria based on which object should be filtered.
...

Optional parameters.

Param

Generic parameter class

Description

The ‘Param’ class is a virtual class which can be used as *base* class from which *parameter* classes can inherit.

The methods implemented for the ‘Param’ class are:
- ‘as.list’: coerces the ‘Param’ class to a ‘list’ with list elements representing the object’s slot values, names the slot names. *Hidden* slots (i.e. those with a name starting with '.') are not returned. In addition, a ‘Param’ class can be coerced to a ‘list’ using ‘as(object,"list")’.
- ‘show’: prints the content of the ‘Param’ object (i.e. the individual slots and their value).

Usage

## S4 method for signature 'Param'
as.list(x, ...)

## S4 method for signature 'Param'
show(object)
Arguments

x             ‘Param’ object.
...           ignored.
object        ‘Param’ object.

Author(s)

Johannes Rainer

peaksData

Get or set MS peak data

Description

These methods get or set mass spectrometry (MS) peaks data, which can be m/z, intensity or retention time values. See the respective help pages (e.g. in the Spectra or Chromatograms packages) for information on the actual implementations of these methods.

Usage

peaksData(object, ...)

peaksData(object) <- value

peaksVariables(object, ...)

Arguments

object  A data object.
...     Optional parameters.
value   Replacement for peaks data.

ProcessingStep

Processing step

Description

Class containing the function and arguments to be applied in a lazy-execution framework.

Objects of this class are created using the ProcessingStep() function. The processing step is executed with the executeProcessingStep() function.
Usage

ProcessingStep(FUN = character(), ARGS = list())

## S4 method for signature 'ProcessingStep'
show(object)

executeProcessingStep(object, ...)

Arguments

FUN  function or character representing a function name.
ARGS list of arguments to be passed along to FUN.
oobject  ProcessingStep object.
...  optional additional arguments to be passed along.

Details

This object contains all relevant information of a data analysis processing step, i.e. the function and all of its arguments to be applied to the data. This object is mainly used to record possible processing steps of a Spectra or OnDiskMSnExp object (from the Spectra and MSnbase packages, respectively).

Value

The ProcessingStep function returns and object of type ProcessingStep.

Author(s)

Johannes Rainer

Examples

## Create a simple processing step object
ps <- ProcessingStep(sum)

executeProcessingStep(ps, 1:10)

ProtGenerics  S4 generic functions for Bioconductor proteomics infrastructure

Description

These generic functions provide basic interfaces to operations on and data access to proteomics and mass spectrometry infrastructure in the Bioconductor project.

For the details, please consult the respective methods’ manual pages.
Usage

```r
psms(object, ...)
peaks(object, ...)
modifications(object, ...)
database(object, ...)
rtime(object, ...)
tic(object, ...)
spectra(object, ...)
intensity(object, ...)
mz(object, ...)
peptides(object, ...)
proteins(object, ...)
accessions(object, ...)
scans(object, ...)
mass(object, ...)
ions(object, ...)
chromatograms(object, ...)
chromatogram(object, ...)
isCentroided(object, ...)
writeMSData(object, file, ...)
## and many more
```

Arguments

- `object`: Object of class for which methods are defined.
- `file`: for `writeMSData`
  - the name of the file to which the data should be exported.
- `...`: Further arguments, possibly used by downstream methods.

Details

**When should one define a generics?:**

Generics are appropriate for functions that have *generic* names, i.e. names that occur in multiple circumstances, (with different input classes, most often defined in different packages) or, when (multiple) dispatching is better handled by the generics mechanism rather than the developer. The dispatching mechanism will then automatically call the appropriate method and save the user from explicitly calling `package::method` or the developer to handle the multiple input types cases. When no such conflict exists or is unlikely to happen (for example when the name of the function is specific to a package or domain, or for class slots accessors and replacement methods), the usage of a generic is arguably questionable, and in most of these cases, simple, straightforward functions would be perfectly valid.

**When to define a generic in ProtGenerics?:**

ProtGenerics is not meant to be the central package for generics, nor should it stop developers from defining the generics they need. It is a means to centralise generics that are defined in different packages (for example `mzR::psms` and `mzID::psms`, or `IRanges::score` and `mzR::score`, etc.).
now BioGenerics::score) or generics that live in a rather big package (say mzR) on which one wouldn’t want to depend just for the sake of that generics’ definition.

The goal of ProtGenerics is to step in when namespace conflicts arise so as to facilitate inter-operability of packages. In case such conflict appears due to multiple generics, we would (1) add these same definitions in ProtGenerics, (2) remove the definitions from the packages they stem from, which then (3) only need to import ProtGenerics. This would be very minor/straightforward changes for the developers and would resolve issues when they arise.

More generics can be added on request by opening an issue or sending a pull request on: https://github.com/lgatto/ProtGenerics

Author(s)

Laurent Gatto

See Also

• The BiocGenerics package for S4 generic functions needed by many Bioconductor packages.
• showMethods for displaying a summary of the methods defined for a given generic function.
• selectMethod for getting the definition of a specific method.
• setGeneric and setMethod for defining generics and methods.

Examples

```r
## List all the symbols defined in this package:
ls("package:ProtGenerics")

## Not run:
## What methods exists for 'peaks'
showMethods("peaks")

## To look at one method in particular
getMethod("peaks", "mzRpwiz")

## End(Not run)
```
Index

* methods
  ProtGenerics, 5
  dataStorage<- (ProtGenerics), 5
database (ProtGenerics), 5
dataOrigin (ProtGenerics), 5
dataOrigin<- (ProtGenerics), 5
dataStorage (ProtGenerics), 5

  as.list,Param-method (Param), 3
  backendBpparam (backendInitialize), 2
  backendInitialize, 2
  backendMerge (backendInitialize), 2
  backendParallelFactor
    (backendInitialize), 2
  bin (ProtGenerics), 5
  calculateFragments (ProtGenerics), 5
centroided (ProtGenerics), 5
centroided<- (ProtGenerics), 5
collaboratorFunction-class
  (ProcessingStep), 4
collaboratorOrFunction-class
  (ProcessingStep), 4
  chromatogram (ProtGenerics), 5
  chromatograms (ProtGenerics), 5
collisionEnergy (ProtGenerics), 5
collisionEnergy<- (ProtGenerics), 5
  combineFeatures (ProtGenerics), 5
  compareChromatograms (ProtGenerics), 5
  compareSpectra (ProtGenerics), 5
  compounds (ProtGenerics), 5

  impute (ProtGenerics), 5
  instrumentCustomisations
    (ProtGenerics), 5
  instrumentManufacturer (ProtGenerics), 5
  instrumentModel (ProtGenerics), 5

  expemail (ProtGenerics), 5
  exptitle (ProtGenerics), 5
  filterAcquisitionNum (ProtGenerics), 5
  filterDataOrigin (ProtGenerics), 5
  filterDataStorage (ProtGenerics), 5
  filterEmptySpectra (ProtGenerics), 5
  filterFeatures, 2
  filterIntensity (ProtGenerics), 5
  filterIsolationWindow (ProtGenerics), 5
  filterMsLevel (ProtGenerics), 5
  filterMz (ProtGenerics), 5
  filterMzRange (ProtGenerics), 5
  filterMzValues (ProtGenerics), 5
  filterNA (ProtGenerics), 5
  filterPolarity (ProtGenerics), 5
  filterPrecursorCharge (ProtGenerics), 5
  filterPrecursorMz (ProtGenerics), 5
  filterPrecursorMzRange (ProtGenerics), 5
  filterPrecursorMzValues (ProtGenerics), 5
  filterPrecursorScan (ProtGenerics), 5
  filterProductMz (ProtGenerics), 5
  filterProductMzRange (ProtGenerics), 5
  filterProductMzValues (ProtGenerics), 5
  filterRt (ProtGenerics), 5
  filterSpectra, 3

  impute (ProtGenerics), 5
  instrumentCustomisations
    (ProtGenerics), 5
  instrumentManufacturer (ProtGenerics), 5
  instrumentModel (ProtGenerics), 5

  executeProcessingStep (ProcessingStep), 4
  expemail (ProtGenerics), 5
  exptitle (ProtGenerics), 5
dataStorage<- (ProtGenerics), 5
detectorType (ProtGenerics), 5

  addProcessing (ProtGenerics), 5
  adjacencyMatrix (ProtGenerics), 5
  aggregateFeatures (ProtGenerics), 5
  alignRt (ProtGenerics), 5
  analyser (ProtGenerics), 5
  analyserDetails (ProtGenerics), 5
  analyzer (ProtGenerics), 5
  analyzerDetails (ProtGenerics), 5
intensity (ProtGenerics), 5
intensity<- (ProtGenerics), 5
ionCount (ProtGenerics), 5
ions (ProtGenerics), 5
ionSource (ProtGenerics), 5
ionSourceDetails (ProtGenerics), 5
isCentroided (ProtGenerics), 5
isolationWindowLowerMz (ProtGenerics), 5
isolationWindowLowerMz<- (ProtGenerics), 5
isolationWindowTargetMz (ProtGenerics), 5
isolationWindowTargetMz<- (ProtGenerics), 5
isolationWindowUpperMz (ProtGenerics), 5
isolationWindowUpperMz<- (ProtGenerics), 5
isReadOnly (backendInitialize), 2
mass (ProtGenerics), 5
modifications (ProtGenerics), 5
msInfo (ProtGenerics), 5
msLevel (ProtGenerics), 5
msLevel<- (ProtGenerics), 5
mz (ProtGenerics), 5
mz<- (ProtGenerics), 5
Param, 3
Param-class (Param), 3
peaks (ProtGenerics), 5
peaks<- (ProtGenerics), 5
peaksData, 4
peaksData<- (peaksData), 4
peaksVariables (peaksData), 4
peptides (ProtGenerics), 5
polarity (ProtGenerics), 5
polarity<- (ProtGenerics), 5
precAcquisitionNum (ProtGenerics), 5
precScanNum (ProtGenerics), 5
precursorCharge (ProtGenerics), 5
precursorCharge<- (ProtGenerics), 5
precursorIntensity (ProtGenerics), 5
precursorIntensity<- (ProtGenerics), 5
precursorMz (ProtGenerics), 5
precursorMz<- (ProtGenerics), 5
processingData (ProtGenerics), 5
processingData<- (ProtGenerics), 5
ProcessingStep, 4
ProcessingStep-class (ProcessingStep), 4
productMz (ProtGenerics), 5
productMz<- (ProtGenerics), 5
proteins (ProtGenerics), 5
ProtGenerics, 5
ProtGenerics-package (ProtGenerics), 5
psms (ProtGenerics), 5
quantify (ProtGenerics), 5
rtime (ProtGenerics), 5
rtime<- (ProtGenerics), 5
scanIndex (ProtGenerics), 5
scans (ProtGenerics), 5
selectMethod, 7
setBackend (backendInitialize), 2
setGeneric, 7
setMethod, 7
show,Param-method (Param), 3
show,ProcessingStep-method (ProcessingStep), 4
showMethods, 7
smooth (ProtGenerics), 5
smoothed (ProtGenerics), 5
smoothed<- (ProtGenerics), 5
spectra (ProtGenerics), 5
spectra<- (ProtGenerics), 5
spectraData (ProtGenerics), 5
spectraData<- (ProtGenerics), 5
spectraNames (ProtGenerics), 5
spectraNames<- (ProtGenerics), 5
spectrapply (ProtGenerics), 5
spectraVariables (ProtGenerics), 5
supportsSetBackend (backendInitialize), 2
tic (ProtGenerics), 5
tolerance (ProtGenerics), 5
uniqueMsLevels (ProtGenerics), 5
writeMSData (ProtGenerics), 5