

# Package ‘ProtGenerics’

July 17, 2025

**Title** Generic infrastructure for Bioconductor mass spectrometry packages

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

**Version** 1.40.0

**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** 7.3.2

**git\_url** <https://git.bioconductor.org/packages/ProtGenerics>

**git\_branch** RELEASE\_3\_21

**git\_last\_commit** 048a08d

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.21

**Date/Publication** 2025-07-16

## Contents

backendInitialize . . . . .	2
extractByIndex . . . . .	3
filterFeatures . . . . .	3
filterSpectra . . . . .	4
Param . . . . .	4
peaksData . . . . .	5



processingQueue . . . . .	5
ProcessingStep . . . . .	6
ProtGenerics . . . . .	7
<b>Index</b>	<b>10</b>

---

backendInitialize	<i>General backend methods</i>
-------------------	--------------------------------

---

## 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 <i>backend</i> object.
...	Optional parameters.
backend	A <i>backend</i> object.



---

extractByIndex	<i>Extracting elements by index</i>
----------------	-------------------------------------

---

**Description**

The `extractByIndex()` method allows to subset an object (or extract elements from it) by providing their integer indices.

**Usage**

```
extractByIndex(object, i)
```

**Arguments**

object	The object to subset/from which to extract elements.
i	integer with the indices.

---

filterFeatures	<i>Filter features</i>
----------------	------------------------

---

**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, ...)
```

**Arguments**

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



---

filterSpectra	<i>Filter Spectra</i>
---------------	-----------------------

---

### 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	<i>Generic parameter class</i>
-------	--------------------------------

---

### 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.

---

processingQueue*Processing Queue*

---

**Description**

These methods are related to the *\*processing queue\** implemented in the [Spectra](<https://github.com/RforMassSpectrometry>) and [Chromatograms](<https://github.com/RforMassSpectrometry/Chromatograms>) packages.

- `'addProcessing()'` adds a processing step to the processing queue.
- `'applyProcessing()'` execute the processing queue replacing the original data in `'object'` with the processed one.
- `'processingChunkSize()'` and `'processingChunkSize()<-'` are supposed to get and set the number of elements (e.g. spectra) for which the data is loaded into memory and processed at a time.
- `'processingChunkFactor()'`: defines a `'factor'` that can be used to split `'object'` into chunks defined by the length of `'object'` and its `'processingChunkSize()'`.



**Usage**

```

processingChunkSize(object, ...)

processingChunkSize(object, ...) <- value

addProcessing(object, ...)

applyProcessing(object, ...)

processingChunkFactor(object, ...)

```

**Arguments**

object	The object with the processing queue.
...	Additional parameters to be defined.
value	The replacement value.

---

ProcessingStep	<i>Processing step</i>
----------------	------------------------

---

**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.
object	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 an 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

```
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`, 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**

```
## 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, 7

accessions (ProtGenerics), 7

acquisitionNum (ProtGenerics), 7

addProcessing (processingQueue), 5

adjacencyMatrix (ProtGenerics), 7

aggregateFeatures (ProtGenerics), 7

alignRt (ProtGenerics), 7

analyser (ProtGenerics), 7

analyserDetails (ProtGenerics), 7

analyzer (ProtGenerics), 7

analyzerDetails (ProtGenerics), 7

applyProcessing (processingQueue), 5

as.list, Param-method (Param), 4

backendBpparam (backendInitialize), 2

backendInitialize, 2

backendMerge (backendInitialize), 2

backendParallelFactor  
(backendInitialize), 2

bin (ProtGenerics), 7

calculateFragments (ProtGenerics), 7

centroided (ProtGenerics), 7

centroided<- (ProtGenerics), 7

characterOrFunction-class  
(ProcessingStep), 6

chromatogram (ProtGenerics), 7

chromatograms (ProtGenerics), 7

collisionEnergy (ProtGenerics), 7

collisionEnergy<- (ProtGenerics), 7

combineFeatures (ProtGenerics), 7

compareChromatograms (ProtGenerics), 7

compareSpectra (ProtGenerics), 7

compounds (ProtGenerics), 7

database (ProtGenerics), 7

dataOrigin (ProtGenerics), 7

dataOrigin<- (ProtGenerics), 7

dataStorage (ProtGenerics), 7

dataStorage<- (ProtGenerics), 7

detectorType (ProtGenerics), 7

estimatePrecursorIntensity  
(ProtGenerics), 7

executeProcessingStep (ProcessingStep),  
6

expemail (ProtGenerics), 7

exptitle (ProtGenerics), 7

extractByIndex, 3

filterAcquisitionNum (ProtGenerics), 7

filterDataOrigin (ProtGenerics), 7

filterDataStorage (ProtGenerics), 7

filterEmptySpectra (ProtGenerics), 7

filterFeatures, 3

filterIntensity (ProtGenerics), 7

filterIsolationWindow (ProtGenerics), 7

filterMsLevel (ProtGenerics), 7

filterMz (ProtGenerics), 7

filterMzRange (ProtGenerics), 7

filterMzValues (ProtGenerics), 7

filterNA (ProtGenerics), 7

filterPolarity (ProtGenerics), 7

filterPrecursorCharge (ProtGenerics), 7

filterPrecursorMz (ProtGenerics), 7

filterPrecursorMzRange (ProtGenerics), 7

filterPrecursorMzValues (ProtGenerics),  
7

filterPrecursorScan (ProtGenerics), 7

filterProductMz (ProtGenerics), 7

filterProductMzRange (ProtGenerics), 7

filterProductMzValues (ProtGenerics), 7

filterRanges (ProtGenerics), 7

filterRt (ProtGenerics), 7

filterSpectra, 4

filterValues (ProtGenerics), 7

impute (ProtGenerics), 7



instrumentCustomisations  
     (ProtGenerics), 7  
 instrumentManufacturer (ProtGenerics), 7  
 instrumentModel (ProtGenerics), 7  
 intensity (ProtGenerics), 7  
 intensity<- (ProtGenerics), 7  
 ionCount (ProtGenerics), 7  
 ions (ProtGenerics), 7  
 ionSource (ProtGenerics), 7  
 ionSourceDetails (ProtGenerics), 7  
 isCentroided (ProtGenerics), 7  
 isolationWindowLowerMz (ProtGenerics), 7  
 isolationWindowLowerMz<-  
     (ProtGenerics), 7  
 isolationWindowTargetMz (ProtGenerics),  
     7  
 isolationWindowTargetMz<-  
     (ProtGenerics), 7  
 isolationWindowUpperMz (ProtGenerics), 7  
 isolationWindowUpperMz<-  
     (ProtGenerics), 7  
 isReadOnly (backendInitialize), 2  
  
 mass (ProtGenerics), 7  
 modifications (ProtGenerics), 7  
 msInfo (ProtGenerics), 7  
 msLevel (ProtGenerics), 7  
 msLevel<- (ProtGenerics), 7  
 mz (ProtGenerics), 7  
 mz<- (ProtGenerics), 7  
  
 Param, 4  
 Param-class (Param), 4  
 peaks (ProtGenerics), 7  
 peaks<- (ProtGenerics), 7  
 peaksData, 5  
 peaksData<- (peaksData), 5  
 peaksVariables (peaksData), 5  
 peptides (ProtGenerics), 7  
 polarity (ProtGenerics), 7  
 polarity<- (ProtGenerics), 7  
 precAcquisitionNum (ProtGenerics), 7  
 precScanNum (ProtGenerics), 7  
 precursorCharge (ProtGenerics), 7  
 precursorCharge<- (ProtGenerics), 7  
 precursorIntensity (ProtGenerics), 7  
 precursorIntensity<- (ProtGenerics), 7  
 precursorMz (ProtGenerics), 7  
 precursorMz<- (ProtGenerics), 7  
  
 processingChunkFactor  
     (processingQueue), 5  
 processingChunkSize (processingQueue), 5  
 processingChunkSize<-  
     (processingQueue), 5  
 processingData (ProtGenerics), 7  
 processingData<- (ProtGenerics), 7  
 processingQueue, 5  
 ProcessingStep, 6  
 ProcessingStep-class (ProcessingStep), 6  
 productMz (ProtGenerics), 7  
 productMz<- (ProtGenerics), 7  
 proteins (ProtGenerics), 7  
 ProtGenerics, 7  
 ProtGenerics-package (ProtGenerics), 7  
 psms (ProtGenerics), 7  
  
 quantify (ProtGenerics), 7  
  
 rtime (ProtGenerics), 7  
 rtime<- (ProtGenerics), 7  
  
 scanIndex (ProtGenerics), 7  
 scans (ProtGenerics), 7  
 selectMethod, 9  
 setBackend (backendInitialize), 2  
 setGeneric, 9  
 setMethod, 9  
 show, Param-method (Param), 4  
 show, ProcessingStep-method  
     (ProcessingStep), 6  
 showMethods, 9  
 smooth (ProtGenerics), 7  
 smoothed (ProtGenerics), 7  
 smoothed<- (ProtGenerics), 7  
 spectra (ProtGenerics), 7  
 spectra<- (ProtGenerics), 7  
 spectraData (ProtGenerics), 7  
 spectraData<- (ProtGenerics), 7  
 spectraNames (ProtGenerics), 7  
 spectraNames<- (ProtGenerics), 7  
 spectrapply (ProtGenerics), 7  
 spectraVariables (ProtGenerics), 7  
 supportsSetBackend (backendInitialize),  
     2  
  
 tic (ProtGenerics), 7  
 tolerance (ProtGenerics), 7  
  
 uniqueMsLevels (ProtGenerics), 7



`writeMSData(ProtGenerics)`, [7](#)