Package 'ProtGenerics'

April 15, 2020

Title S4 generic functions for Bioconductor proteomics infrastructure

Description S4 generic functions needed by Bioconductor proteomics packages.

Version 1.18.0

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

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

biocViews Infrastructure, Proteomics, MassSpectrometry

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

Depends methods

License Artistic-2.0

NeedsCompilation no

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

git_branch RELEASE_3_10

git_last_commit ff9705b

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

R topics documented:

ProtGenerics 1

Index **4**

`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, ...)
```

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 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 <lg390@cam.ac.uk>

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

∗Topic **methods**
ProtGenerics, 1

accessions *(*ProtGenerics*)*, 1
acquisitionNum *(*ProtGenerics*)*, 1
analyser *(*ProtGenerics*)*, 1
analyserDetails *(*ProtGenerics*)*, 1
analyzer *(*ProtGenerics*)*, 1
analyzerDetails *(*ProtGenerics*)*, 1

centroided *(*ProtGenerics*)*, 1
centroided<- *(*ProtGenerics*)*, 1
chromatogram *(*ProtGenerics*)*, 1
chromatograms *(*ProtGenerics*)*, 1
collisionEnergy *(*ProtGenerics*)*, 1
collisionEnergy<- *(*ProtGenerics*)*, 1
combineFeatures *(*ProtGenerics*)*, 1

database *(*ProtGenerics*)*, 1
dataOrigin *(*ProtGenerics*)*, 1
dataOrigin<- *(*ProtGenerics*)*, 1
dataStorage *(*ProtGenerics*)*, 1
dataStorage<- *(*ProtGenerics*)*, 1
detectorType *(*ProtGenerics*)*, 1

expemail *(*ProtGenerics*)*, 1
exptitle *(*ProtGenerics*)*, 1

filterAcquisitionNum *(*ProtGenerics*)*, 1
filterDataOrigin *(*ProtGenerics*)*, 1
filterDataStorage *(*ProtGenerics*)*, 1
filterEmptySpectra *(*ProtGenerics*)*, 1
filterIsolationWindow *(*ProtGenerics*)*, 1
filterMsLevel *(*ProtGenerics*)*, 1
filterMz *(*ProtGenerics*)*, 1
filterPolarity *(*ProtGenerics*)*, 1
filterPrecursorMz *(*ProtGenerics*)*, 1
filterPrecursorScan *(*ProtGenerics*)*, 1
filterProductMz *(*ProtGenerics*)*, 1
filterRt *(*ProtGenerics*)*, 1

instrumentCustomisations *(*ProtGenerics*)*, 1
instrumentManufacturer *(*ProtGenerics*)*, 1
instrumentModel *(*ProtGenerics*)*, 1
intensity *(*ProtGenerics*)*, 1
intensity<- *(*ProtGenerics*)*, 1
ionCount *(*ProtGenerics*)*, 1
ions *(*ProtGenerics*)*, 1
ionSource *(*ProtGenerics*)*, 1
ionSourceDetails *(*ProtGenerics*)*, 1
isCentroided *(*ProtGenerics*)*, 1
isolationWindowLowerMz *(*ProtGenerics*)*, 1
isolationWindowLowerMz<- *(*ProtGenerics*)*, 1
isolationWindowTargetMz *(*ProtGenerics*)*, 1
isolationWindowTargetMz<- *(*ProtGenerics*)*, 1
isolationWindowUpperMz *(*ProtGenerics*)*, 1
isolationWindowUpperMz<- *(*ProtGenerics*)*, 1

mass *(*ProtGenerics*)*, 1
modifications *(*ProtGenerics*)*, 1
msInfo *(*ProtGenerics*)*, 1
msLevel *(*ProtGenerics*)*, 1
msLevel<- *(*ProtGenerics*)*, 1
mz *(*ProtGenerics*)*, 1
mz<- *(*ProtGenerics*)*, 1

peaks *(*ProtGenerics*)*, 1
peaks<- *(*ProtGenerics*)*, 1
peptides *(*ProtGenerics*)*, 1
polarity *(*ProtGenerics*)*, 1
polarity<- *(*ProtGenerics*)*, 1
precAcquisitionNum *(*ProtGenerics*)*, 1
precScanNum *(*ProtGenerics*)*, 1
precursorCharge *(*ProtGenerics*)*, 1
precursorCharge<- *(*ProtGenerics*)*, 1
precursorIntensity *(*ProtGenerics*)*, 1
precursorIntensity<- *(*ProtGenerics*)*, 1
precursorMz *(*ProtGenerics*)*, 1
precursorMz<- *(*ProtGenerics*)*, 1
processingData *(*ProtGenerics*)*, 1
processingData<- *(*ProtGenerics*)*, 1
productMz *(*ProtGenerics*)*, 1
productMz<- *(*ProtGenerics*)*, 1
proteins *(*ProtGenerics*)*, 1

ProtGenerics, 1
ProtGenerics-package (ProtGenerics), 1
psms (ProtGenerics), 1

rtime (ProtGenerics), 1
rtime<- (ProtGenerics), 1

scanIndex (ProtGenerics), 1
scans (ProtGenerics), 1
selectMethod, *3*
setGeneric, *3*
setMethod, *3*
showMethods, *3*
smooth (ProtGenerics), 1
smoothed (ProtGenerics), 1
smoothed<- (ProtGenerics), 1
spectra (ProtGenerics), 1
spectra<- (ProtGenerics), 1
spectraData (ProtGenerics), 1
spectraData<- (ProtGenerics), 1
spectraNames (ProtGenerics), 1
spectraNames<- (ProtGenerics), 1
spectraVariables (ProtGenerics), 1

tic (ProtGenerics), 1
tolerance (ProtGenerics), 1

writeMSData (ProtGenerics), 1