MSnbase development

Laurent Gatto
lg390@cam.ac.uk
Cambridge Center for Proteomics
Kathryn S. Lilley Group
University of Cambridge

February 16, 2012

Abstract

This vignette describes the classes implemented in MSnbase package. It is intended as a starting point for developers or users who would like to learn more or further develop/extend `pSet`.

Keywords: Mass Spectrometry (MS), proteomics, infrastructure.

Contents

Foreword

MSnbase is in an early development (see section 4 for details about packages and version used in this vignette). Although main data structures have been thought out and are meant to be compatible with other existing mature infrastructure in the Bioconductor project, changes may occur in the future. Current functionality will evolve and new one will be added. Although at an early stage, this package is released in the hope that it may foster new developments in proteomics data analysis within R by providing a common infrastructure. Several package developers working with mass spectrometry and proteomics data met at the Bioconductor Developer Meeting Europe[1] held in Heidelberg in November 2010, and agreed to combine efforts. This library is one attempt to do so.

You are welcome to contact me for questions, bugs, typos or suggestions about MSnbase. If you wish to reach a broader audience for general questions about proteomics analysis using R, you may want to use the Bioconductor mailing list[2].

1 Introduction

This document is not a replacement for the individual manual pages, that document the slots of the MSnbase classes. It is a centralised high-level description of the package design.

MSnbase aims at being compatible with the Biobase infrastructure Gentleman et al. (2004). Many meta data structures that are used in `eSet` and associated classes are also used here. As such, knowledge of the *Biobase development and the new eSet* vignette[3] would be beneficial.

The initial goal is to use the MSnbase infrastructure for labelled quantitation using reporter ions (iTRAQ (Ross et al., 2004) and TMT (Thompson et al., 2003)). Spectral counting should be trivial to apply with current features, as long as identification data is at hand. Currently, no effort is invested to streamline label-free quantitative proteomics, although some effort has been done to keep the infrastructure flexible enough to accommodate more designs.

2 MSnbase classes

All classes have a `.__classVersion__` slot, of class `Versioned` from the `Biobase` package. This slot documents the class version for any instance to be used for debugging and object update purposes. Any change in a class implementation should trigger a version change.

[1] http://bioconductor.org/help/course-materials/2010/HeidelbergNovember2010/

[2] https://stat.ethz.ch/mailman/listinfo/bioconductor

[3] The vignette can directly be accessed with `vignette("BiobaseDevelopment",package="Biobase")` once Biobase is loaded.

2.1 pSet: a virtual class for raw mass spectrometry data and meta data

This virtual class is the main container for mass spectrometry data, i.e spectra, and meta data. It is based on the `eSet` implementation for genomic data. The main difference with `eSet` is that the `assayData` slot is an environment containing any number of `Spectrum` instances (see section 2.6).

One new slot is introduced, namely `processingData`, that contains one `MSnProcess` instance (see section 2.4). and the `experimentData` slot is now expected to contain `MIAPE` data (see section 2.5). The `annotation` slot has not been implemented, as no prior feature annotation is known in shotgun proteomics.

```
> getClass("pSet")
```

```
Virtual Class "pSet" [package "MSnbase"]

Slots:

Name:            assayData           phenoData         featureData
Class:         environment NAnnotatedDataFrame  AnnotatedDataFrame

Name:       experimentData        protocolData      processingData
Class:               MIAxE  AnnotatedDataFrame          MSnProcess

Name:              .cache   .__classVersion__
Class:         environment            Versions

Extends: "Versioned"

Known Subclasses: "MSnExp"
```

Future work Currently, few setters have been implemented.

2.2 MSnExp: a class for MS experiments

`MSnExp` extends `pSet` to store MS experiments. It does not add any new slots to `pSet`. Accessors and setters are all inherited from `pSet` and new ones should be implemented for `pSet`. Methods that manipulate actual data in experiments are implemented for `MSnExp` objects.

```
> getClass("MSnExp")
```

```
Class "MSnExp" [package "MSnbase"]
```

```
Slots:

Name:            assayData           phenoData         featureData
Class:         environment NAnnotatedDataFrame  AnnotatedDataFrame

Name:       experimentData        protocolData      processingData
Class:               MIAxE  AnnotatedDataFrame          MSnProcess

Name:               .cache   .__classVersion__
Class:         environment            Versions

Extends:
Class "pSet", directly
Class "Versioned", by class "pSet", distance 2
```

2.3 MSnSet: a class for quantitative proteomics data

This class stores quantitation data and meta data after running `quantify` on an `MSnExp` object. The quantitative data is in form of a $n \times m$ matrix, where m is the number of features/spectra originally in the `MSnExp` used as parameter in `quantify` and m is the number of reporter ions (see section 2.7).

This prompted to keep a similar implementation as the `ExpressionSet` class, while adding the proteomics-specific annotation slot introduced in the `pSet` class, namely `processingData` for objects of class `MSnProcess` (see section 2.4).

The `MSnSet` class extends the virtual `eSet` class to provide compatibility for `ExpressionSet`-like behaviour. The experiment meta-data in `experimentData` is also of class `MIAPE` (see section 2.5). The `annotation` slot, inherited from `eSet` is not used.

```
> getClass("MSnSet")


Class "MSnSet" [package "MSnbase"]

Slots:

Name:       experimentData     processingData               qual
Class:              MIAPE         MSnProcess         data.frame

Name:           assayData          phenoData        featureData
Class:          AssayData AnnotatedDataFrame AnnotatedDataFrame

Name:          annotation       protocolData  .__classVersion__
Class:          character AnnotatedDataFrame           Versions

Extends:
```

```
Class "eSet", directly
Class "VersionedBiobase", by class "eSet", distance 2
Class "Versioned", by class "eSet", distance 3
```

2.4 MSnProcess: a class for logging processing meta data

This class aims at recording specific manipulations applied to `MSnExp` or `MSnSet` instances. The `processing` slot is a `character` vector that describes major processing. Most other slots are of class `logical` that indicate whether the data has been centroided, smoothed, ... although many of the functionality is not implemented yet. Any new processing that is implemented should be documented and logged here.

It also documents the raw data file from which the data originates (`files` slot) and the MSnbase version that was in use when the `MSnProcess` instance, and hence the `MSnExp`/`MSnSet` objects, were originally created.

```
> getClass("MSnProcess")
```

```
Class "MSnProcess" [package "MSnbase"]

Slots:

Name:              files         processing             merged            cleaned
Class:         character          character            logical            logical

Name:       removedPeaks           smoothed            trimmed         normalised
Class:         character            logical            numeric            logical

Name:     MSnbaseVersion .__classVersion__
Class:         character           Versions

Extends: "Versioned"
```

2.5 MIAPE: Minimum Information About a Proteomics Experiment

The Minimum Information About a Proteomics Experiment (Taylor et al., 2007, 2008) `MIAPE` class describes the experiment, including contact details, information about the mass spectrometer and control and analysis software.

Raw data is currently imported from `mzXML` files (Pedrioli et al., 2004) < using the `xcms:::rampRawData` and `xcms:::rampRawDataMSn` functions from the xcms package (Smith et al., 2006). These functions do not give access to the meta data. New importer functions are under development (see for instance mzR[4]) that will hopefully give programmatic access to meta data stored in the

[4] https://github.com/sneumann/mzR/blob/master/DESCRIPTION

data file to populate the MIAPE object.

```
> getClass("MIAPE")
```

```
Class "MIAPE" [package "MSnbase"]

Slots:

Name:                    title                      url
Class:               character                character

Name:                 abstract                pubMedIds
Class:               character                character

Name:                  samples            preprocessing
Class:                    list                     list

Name:                    other                dateStamp
Class:                    list                character

Name:                     name                      lab
Class:               character                character

Name:                  contact           instrumentModel
Class:               character                character

Name:    instrumentManufacturer instrumentCustomisations
Class:               character                character

Name:              softwareName          softwareVersion
Class:               character                character

Name:         switchingCriteria           isolationWidth
Class:               character                  numeric

Name:             parameterFile                ionSource
Class:               character                character

Name:          ionSourceDetails                 analyser
Class:               character                character

Name:           analyserDetails              collisionGas
Class:               character                character

Name:         collisionPressure          collisionEnergy
```

```
Class:                  numeric                character

Name:              detectorType      detectorSensitivity
Class:                character                character

Name:         .__classVersion__
Class:                 Versions

Extends:
Class "MIAxE", directly
Class "Versioned", by class "MIAxE", distance 2
```

2.6 Spectrum *et al.*: classes for MS spectra

`Spectrum` is a virtual class that defines common attributes to all types of spectra. MS1 and MS2 specific attributes are defined in the `Spectrum1` and `Spectrum2` classes, that directly extend `Spectrum`.

The choices of attributes has been dictated by the `xcms:::rampRawData` and `xcms:::rampRawDataMSn` functions and what data from the `mzXML` file they gave access to. It is expected that some hopefully minor changes might come up here when migrating to other data import packages, that allow random access to `mzXML` data and support `mzML` (Martens et al., 2010).

```
> getClass("Spectrum")


Virtual Class "Spectrum" [package "MSnbase"]

Slots:

Name:            msLevel        peaksCount                rt    acquisitionNum
Class:           integer           integer           numeric           integer

Name:          scanIndex                mz         intensity          fromFile
Class:           integer           numeric           numeric           integer

Name:         centroided .__classVersion__
Class:           logical          Versions

Extends: "Versioned"

Known Subclasses: "Spectrum2", "Spectrum1"


> getClass("Spectrum1")
```

```
Class "Spectrum1" [package "MSnbase"]

Slots:

Name:           polarity           msLevel        peaksCount                rt
Class:           integer           integer           integer           numeric

Name:     acquisitionNum         scanIndex                mz         intensity
Class:           integer           integer           numeric           numeric

Name:           fromFile        centroided .__classVersion__
Class:           integer           logical          Versions

Extends:
Class "Spectrum", directly
Class "Versioned", by class "Spectrum", distance 2


> getClass("Spectrum2")


Class "Spectrum2" [package "MSnbase"]

Slots:

Name:              merged        precScanNum        precursorMz
Class:            numeric            integer            numeric

Name:  precursorIntensity    precursorCharge    collisionEnergy
Class:            numeric            integer            numeric

Name:             msLevel         peaksCount                rt
Class:            integer            integer           numeric

Name:      acquisitionNum          scanIndex                mz
Class:            integer            integer           numeric

Name:           intensity           fromFile         centroided
Class:            numeric            integer            logical

Name:   .__classVersion__
Class:           Versions

Extends:
Class "Spectrum", directly
Class "Versioned", by class "Spectrum", distance 2
```

2.7 ReporterIons: a class for isobaric tags

The iTRAQ and TMT (or any other peak of interest) are implemented `ReporterIons` instances, that essentially defines an expected MZ position for the peak and a width around this value as well a names for the reporters.

```
> getClass("ReporterIons")
```

```
Class "ReporterIons" [package "MSnbase"]

Slots:

Name:               name     reporterNames       description                mz
Class:         character         character         character           numeric

Name:                col             width .__classVersion__
Class:         character           numeric          Versions

Extends: "Versioned"
```

2.8 NAnnotatedDataFrame: multiplexed AnnotatedDataFrames

The simple expansion of the `AnnotatedDataFrame` classes adds the `multiplex` and `multiLabel` slots to document the number and names of multiplexed samples.

```
> getClass("NAnnotatedDataFrame")
```

```
Class "NAnnotatedDataFrame" [package "MSnbase"]

Slots:

Name:          multiplex       multiLabels       varMetadata              data
Class:           numeric         character        data.frame        data.frame

Name:          dimLabels .__classVersion__
Class:         character          Versions

Extends:
Class "AnnotatedDataFrame", directly
Class "Versioned", by class "AnnotatedDataFrame", distance 2
```

3 Miscellaneous

Unit tests MSnbase implements unit tests with the testthat package.

Processing methods Methods that process raw data, i.e. spectra should be implemented for `Spectrum` objects first and then `eapply`'ed (or similar) to the `assayData` slot of an `MSnExp` instance in the specific method.

Speed and memory requirements Raw mass spectrometry file are generally several hundreds of MB large and most of this is used for binary raw spectrum data. As such, data containers can easily grow very large and thus require large amounts of RAM. This requirement may be tackled in the future by avoiding to load the raw data into memory and using random access to the content of `mzXML`/`mzML` data files on demand. When focusing on reporter ion quantitation, a direct solution for this is to trim the spectra using the `trimMz` method to select the area of interest and thus substantially reduce the size of the `Spectrum` objects. This is illustrated in section **??** on page **??** of the `MSnbase-demo` vignette.

The independent handling of spectra is ideally suited for parallel processing. This will be added soon.

4 Session information

- R version 2.14.1 (2011-12-22), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, tools, utils
- Other packages: Biobase 2.14.0, MSnbase 1.2.3, Rcpp 0.9.9, cacheSweave 0.6, codetools 0.2-8, filehash 2.2, formatR 0.3-4, getopt 1.17, ggplot2 0.9.0, highlight 0.3.1, mzR 1.0.2, optparse 0.9.5, parser 0.0-14, pgfSweave 1.2.1, stashR 0.3-4, tikzDevice 0.6.2
- Loaded via a namespace (and not attached): BiocInstaller 1.2.1, IRanges 1.12.6, MASS 7.3-17, RColorBrewer 1.0-5, affy 1.32.1, affyio 1.22.0, colorspace 1.1-1, dichromat 1.2-4, digest 0.5.1, grid 2.14.1, lattice 0.20-0, limma 3.10.2, memoise 0.1, munsell 0.3, plyr 1.7.1, preprocessCore 1.16.0, proto 0.3-9.2, reshape 0.8.4, reshape2 1.2.1, scales 0.1.0, stringr 0.6, vsn 3.22.0, xcms 1.30.3, zlibbioc 1.0.0

References

Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus,

Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5(10):–80, 2004. doi: 10.1186/gb-2004-5-10-r80. URL http://dx.doi.org/10.1186/gb-2004-5-10-r80.

Lennart Martens, Matthew Chambers, Marc Sturm, Darren Kes sner, Fredrik Levander, Jim Shofstahl, Wilfred H Tang, Andreas Ro mpp, Steffen Neumann, Angel D Pizarro, Lu isa Montecchi-Palazzi, Natalie Tasman, Mike Coleman, Florian Reisinger, Pune et Souda, Henning Hermjakob, Pierre-Alain Binz, and Eric W Deutsch. mzml - a community standard for mass spectrometry data. *Molecular & Cellular Proteomics : MCP*, 2010. doi: 10.1074/mcp.R110.000133.

Patrick G A Pedrioli, Jimmy K Eng, Robert Hubley, Mathijs Vogelzang, Eric W Deutsch, Brian Raught, Brian Pratt, Erik Nilsson, Ruth H Angeletti, Rolf Apweiler, Kei Cheung, Catherine E Costello, Henning Hermjakob, Sequin Huang, Randall K Julian, Eugene Kapp, Mark E McComb, Stephen G Oliver, Gilbert Omenn, Norman W Paton, Richard Simpson, Richard Smith, Chris F Taylor, Weimin Zhu, and Ruedi Aebersold. A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.*, 22(11):1459–66, 2004. doi: 10.1038/nbt1031.

Philip L. Ross, Yulin N. Huang, Jason N. Marchese, Brian Williamson, Kenneth Parker, Stephen Hattan, Nikita Khainovski, Sasi Pillai, Subhakar Dey, Scott Daniels, Subhasish Purkayastha, Peter Juhasz, Stephen Martin, Michael Bartlet-Jones, Feng He, Allan Jacobson, and Darryl J. Pappin. Multiplexed protein quantitation in saccharomyces cerevisiae using amine-reactive isobaric tagging reagents. *Mol Cell Proteomics*, 3(12):1154–1169, Dec 2004. doi: 10.1074/mcp.M400129-MCP200. URL http://dx.doi.org/10.1074/mcp.M400129-MCP200.

C.A. Smith, E.J. Want, G. O'Maille, R. Abagyan, and G. Siuzdak. Xcms: Processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching and identification. *Analytical Chemistry*, 78:779–787, 2006.

Chris F. Taylor, Norman W. Paton, Kathryn S. Lilley, Pierre-Alain Binz, Randall K. Julian, Andrew R. Jones, Weimin Zhu, Rolf Apweiler, Ruedi Aebersold, Eric W. Deutsch, Michael J. Dunn, Albert J. R. Heck, Alexander Leitner, Marcus Macht, Matthias Mann, Lennart Martens, Thomas A. Neubert, Scott D. Patterson, Peipei Ping, Sean L. Seymour, Puneet Souda, Akira Tsugita, Joel Vandekerckhove, Thomas M. Vondriska, Julian P. Whitelegge, Marc R. Wilkins, Ioannnis Xenarios, John R. Yates, and Henning Hermjakob. The minimum information about a proteomics experiment (miape). *Nat Biotechnol*, 25(8):887–893, Aug 2007. doi: 10.1038/nbt1329. URL http://dx.doi.org/10.1038/nbt1329.

Chris F Taylor, Pierre-Alain Binz, Ruedi Aebersold, Michel Affolter, Robert Barkovich, Eric W Deutsch, David M Horn, Andreas HÃijhmer, Martin Kussmann, Kathryn Lilley, Marcus Macht, Matthias Mann, Dieter MÃijller, Thomas A Neubert, Janice Nickson, Scott D Patterson, Roberto Raso, Kathryn Resing, Sean L Seymour, Akira Tsugita, Ioannis Xenarios, Rong Zeng, and Randall K Julian. Guidelines for reporting the use of mass spectrometry in proteomics. *Nat. Biotechnol.*, 26(8):860–1, 2008. doi: 10.1038/nbt0808-860.

Andrew Thompson, Jürgen Schäfer, Karsten Kuhn, Stefan Kienle, Josef Schwarz, Günter Schmidt, Thomas Neumann, R Johnstone, A Karim A Mohammed, and Christian Hamon. Tandem mass tags: a novel quantification strategy for comparative analysis of complex protein mixtures by MS/MS. *Anal. Chem.*, 75(8):1895–904, 2003.