MSnbase, efficient R-based processing and visualisation of raw mass spectrometry data

Laurent Gatto,[*,†] Sebastian Gibb,[‡] and Johannes Rainer[¶]

†*Computational Biology Unit, de Duve Institute, Université catholique de Louvain, Brussels, Belgium*

‡*Department of Anaesthesiology and Intensive Care of the University Medicine Greifswald, Germany*

¶*Institute for Biomedicine, Eurac Research, Affiliated Institute of the University of Lübeck, Bolzano, Italy*

E-mail: laurent.gatto@uclouvain.be

Abstract

We present version 2 of the `MSnbase` R/Bioconductor package. `MSnbase` provides infrastructure for the manipulation, processing and visualisation of mass spectrometry data. We focus on the new *on-disk* infrastructure, that allows the handling of large raw mass spectrometry experiment on commodity hardware and illustrate how the package is used for data processing, method development and visualisation.

Introduction

Mass spectrometry is a powerful technology to assays chemical and biological samples. It is used routinely with well characterised protocol, as well a development platform, to improve on existing methods and devise new ones. The complexity and diversity of mass spectrometry

yields data that is itself complex and often times of considerable size, that requires non trivial processing before producing interpretable results. This is particularly relevant, and can constitute a significant challenge for method developers: in addition to the development of sample processing and mass spectrometry methods, there is a need to process, analyse, interpret and assess these new data to demonstrate the improvement in their technical and analytical workflows.

There exists a very diverse catalogue of software tools to explore, process and interpret mass spectrometry data. These range from low level software libraries that are aimed at programmers to develop new applications, to user-oriented applications. Example of software libraries include Java-based jmzML[1] or C/C++-based ProteoWizard,[2] that are used as application programming interfaces. ProteomeDiscoverer (Thermo Scientific), MaxQuant[3] and PeptideShaker[4] are among the most widely user-centric application. They generally provide a limited and fixed set of functionality featuring often advanced graphical user interfaces.

In this software note, we present version 2 of the `MSnbase`[5] software package, available from the Bioconductor[6] project. `MSnbase`, like other Python-based software packages such as pyOpenMS,[7] spectrum_utils[8] or Pyteomics,[9] offers a platform that lies between low level libraries and end-use software. `MSnbase` provides a flexible R[10] command line environment for metabolomics and proteomics mass spectrometry-based application. It allows a detailed step-by-step processing, analysis and exploration of the data and development of novel computational mass spectrometry methods.

Software functionality

In `MSnbase`, mass spectrometry experiments are handled as `MSnExp` objects. While the implementation is more complex, it is useful to schematise a raw data experiment as being composed of raw data, i.e. a collection of individual spectra, as well as spectra-level metadata. Each spectrum is composed of m/z values and associated intensities. The metadata

are represented by a table with variables along the columns and each row associated to a spectrum. Among the metadata available for each spectrum, there are MS level, acquisition number, retention time, precursor m/z and intensity (for MS level 2 and above), and many more. `MSnbase` provides a rich interface to manipulate such objects. The code chunk below illustrates such an object as displayed in the R console and an enumeration of the metadata fields.

```
> show(ms)
MSn experiment data ("OnDiskMSnExp")
Object size in memory: 0.54 Mb
- - - Spectra data - - -
 MS level(s): 1 2 3
 Number of spectra: 994
 MSn retention times: 45:27 - 47:6 minutes
- - - Processing information - - -
Data loaded [Sun Apr 26 15:40:58 2020]
 MSnbase version: 2.13.6
- - - Meta data  - - -
phenoData
  rowNames: MS3TMT11.mzML
  varLabels: sampleNames
  varMetadata: labelDescription
Loaded from:
  MS3TMT11.mzML
protocolData: none
featureData
  featureNames: F1.S001 F1.S002 ... F1.S994 (994 total)
  fvarLabels: fileIdx spIdx ... spectrum (35 total)
```

```
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
> fvarLabels(ms)
 [1] "fileIdx"                   "spIdx"
 [3] "smoothed"                  "seqNum"
 [5] "acquisitionNum"            "msLevel"
 [7] "polarity"                  "originalPeaksCount"
 [9] "totIonCurrent"             "retentionTime"
[11] "basePeakMZ"                "basePeakIntensity"
[13] "collisionEnergy"           "ionisationEnergy"
[15] "lowMZ"                     "highMZ"
[17] "precursorScanNum"          "precursorMZ"
[19] "precursorCharge"           "precursorIntensity"
[21] "mergedScan"                "mergedResultScanNum"
[23] "mergedResultStartScanNum"  "mergedResultEndScanNum"
[25] "injectionTime"             "filterString"
[27] "spectrumId"                "centroided"
[29] "ionMobilityDriftTime"      "isolationWindowTargetMZ"
[31] "isolationWindowLowerOffset" "isolationWindowUpperOffset"
[33] "scanWindowLowerLimit"      "scanWindowUpperLimit"
[35] "spectrum"
```

In the following sections, we will describe demonstrate how `MSnbase` effichiently manages large raw mass spectrometry data and how it is used for data processing and visualisation. We will also illustrate how it makes use of the forward-pipe operator (`%>%`) defined in the `magrittr` package. This operator has proved useful to develop non-trivial analyses by combining individual functions into easily readable pipelines.

On-disk backend

The main feature in version 2 of the `MSnbase` package was the addition of different backends for raw data storage, namely *in-memory* and *on-disk*. The following code chunk demonstrates how to create two `MSnExp` objects, tailored to manage mass spectrometry experiments, storing data in-memory or on-disk.

```
library("MSnbase")
raw_mem <- readMSData("file.mzML", mode = "inMemory")
raw_dsk <- readMSData("file.mzML", mode = "onDisk")
```

The former is the legacy storage mode, implemented in the first version of the package, that loads all the raw data and the metadata in memory. This solution doesn't scale for modern large dataset, and was complemented by the on-disk backend, that only loads metadata into memory and accesses the spectra in the original files when needed. There are two direct benefits using the on-disk backend, namely faster reading and reduced memory footpring. Figure 1 shows 5-fold faster reading times (a) and over a 10-fold reduction in memory usage (b).

The on-disk backend also offers efficient data manipulation by way of *lazy processing*. Operations on the raw data are stored in a processing queue and only effectively applied when raw data is accessed on disk. As an example, the following short analysis pipeline, that can equally be applied to on in-memory or on-disk data retains MS2 spectra acquired between 1000 and 3000 seconds, extract the m/z range corresponding to the TMT 6-plex range and focuses on the MS2 spectra with a precursor intensity greater than 11×10^6 (the median precursor intensity).

```
ms <- ms %>%
    filterRt(c(1000, 3000)) %>%
    filterMz(120, 135)
```

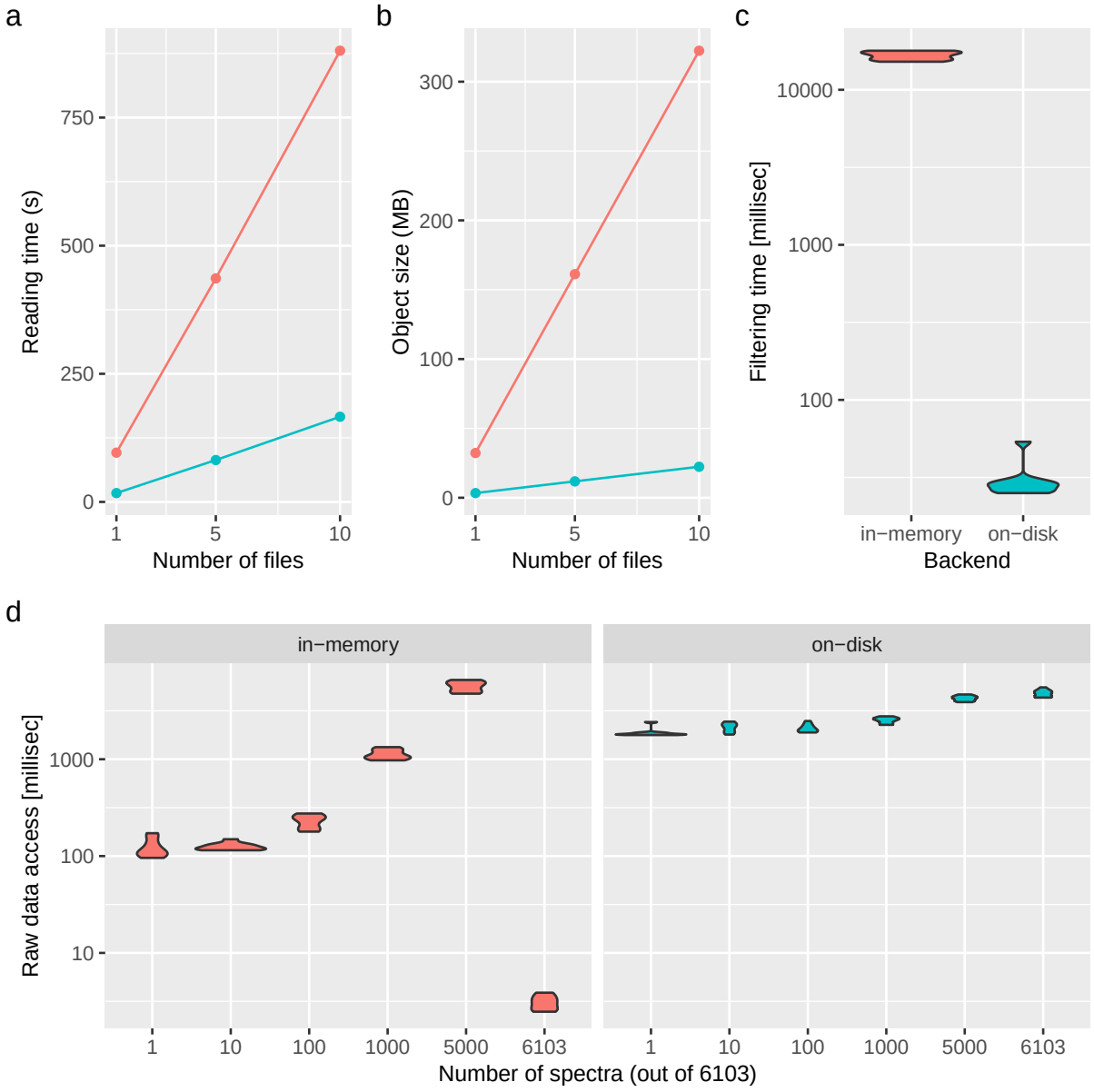

Figure 1: (a) Reading time (in seconds) and (b) data size in memory (in MB) to read/store 1, 5 and 10 files containing 1431 MS1 (on-disk only) and 6103 MS2 (on-disk and in-memory) spectra. (c) Filtering benchmark assessed over 10 interations on in-memory and on-disk data containing 6103 MS2 spectra. (d) Access time to spectra for the in-memory (left) and on-disk (right) backends for 1, 10, 100 1000, 5000 and all 6103 spectra. On-disk backend: blue. In-memory backend: red.

```
ms[precursorIntensity(ms) > 11e6, ]
```

As shown on Figure 1 (c), this lazy mechanism si significantly faster than its application on in-memory data. The advantageous reading and execution times and memory footprint of the on-disk backend are possible by avoiding unnecessary access to the raw data. Once access to the spectra m/z and intensity values become mandatory (for example for plotting), then the in-memory backend becomes more efficient, as illustrated on Figure 1 (d). This gain is maximal when the whole dataset is the be accessed (i.e. all spectra are already in memory) and negligeable when large fractions of the data need to be subset.

Prototyping

The `MSnExp` data structure and its interface constitute a efficient prototyping environment for computational method development. We illustrate this by demonstrating how to implement the BoxCar[11] acquisition method. In a nutshell, BoxCar acquisition aims at improving the detection of intact precursor ions by distributing the charge capacity over multiple narrow m/z segments and thus limiting the proportion of highly abundant precursors in each segment. A full scan is reconstructed by combining the respective adjacent segments of the BoxCar acquisitions. The `MSnbaseBoxCar` package[12] is a small package that demonstrates this. The simple method is composed of three steps is described below and illustrated with code from `MSnbaseBoxCar` in the following code chunk.

1. Identify and filter the groups of spectra that represent adjacent BoxCar acquisitions (Figure 2 (b)).. This can be done using the 'filterString' metadata variable that identifies BoxCar spectra by their adjacent m/z segments with the `bc_groups()` function and filtering relevant spectra with the `filterBoxCar()`.

2. Remove any signal outside the BoxCar segnments using the `bc_zero_out_box()` function from `MSnbaseBoxCar` (Figures 2 (c) and (d)).

3. Using the `combineSpectra` function from the `MSnbase`, combine the cleaned BoxCar spectra into a new, full spectrum (Figure 2 (e)).

```
bc <- readMSData("boxcar.mzML", mode = "onDisk") %>%
      bc_groups() %>%       ## identify BoxCar groups (bc_groups)
      filterBoxCar() %>%    ## keep only BoxCar spectra
      bc_zero_out_box() %>% ## remove signal outside of BoxCar segments
      combineSpectra(fcol = "bc_groups", ## reconstruct full spectrum
                     method = boxcarCompbine)
```

All the functions for the processing of BoxCar spectra and segments in `MSnbaseBoxCar` were developed using existing functionality implemented in `MSnbase`, illustrating the flexibility and adaptability of the `MSnbase` package for computational mass spectrometry method development.

Visualisation

The R environment is well known for the quality of its visualiation capacity. This also holds true for mass spectrometry.[13–16] Here, we conclude the overview of version 2 of the `MSnbase` package by highlighting the flexibility of the software to visualise and assess the efficiency of raw data processing. Figure 3 compares the raw MS profile data for the serine and the same data after smoothing, centroiding and m/z refinement. Detailed execution and description of these operations can be found in the *MSnbase: centroiding of profile-mode MS data* `MSnbase` vignette.

```
serine_mz <- 106.049871
serine_proc <- ms %>%
    smooth(method = "SavitzkyGolay", halfWindowSize = 4L) %>%
    pickPeaks(refineMz = "descendPeak") %>%
```

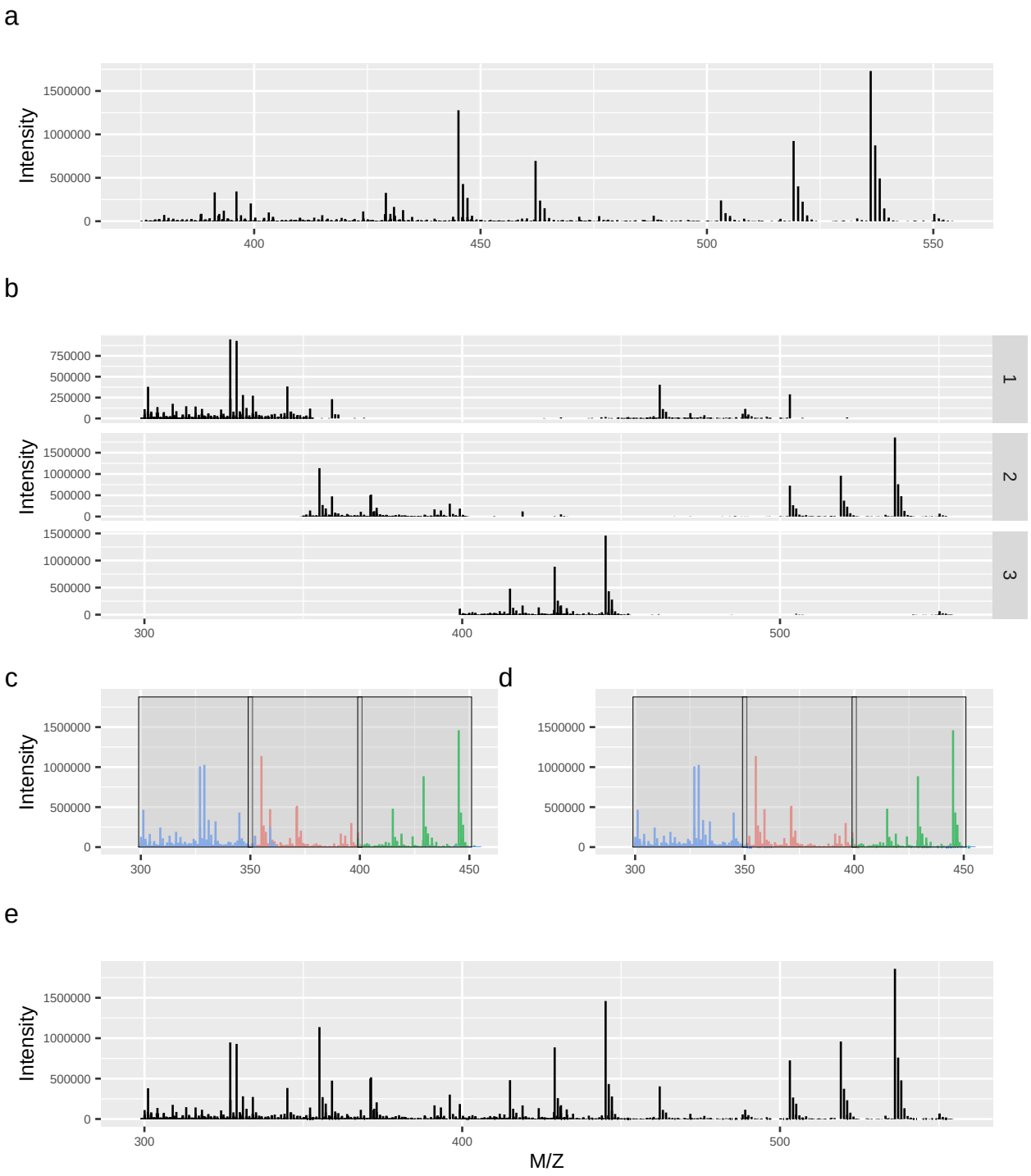

Figure 2: BoxCar processing with `MSnbase`. (a) Standard full scan with (b) three corresponding BoxCar scans showing the adjacent segments. Figure (c) shows the overlapping intact BoxCar segments and (d) the same segments after cleaning, i.e. where peaks outside of the segments were removed. The reconstructed full scan is shown on panel (e).

```
    filterRt(c(serine_mz - 0.01, serine_mz + 0.01)) %>%
    filterMz(c(175, 187))
```

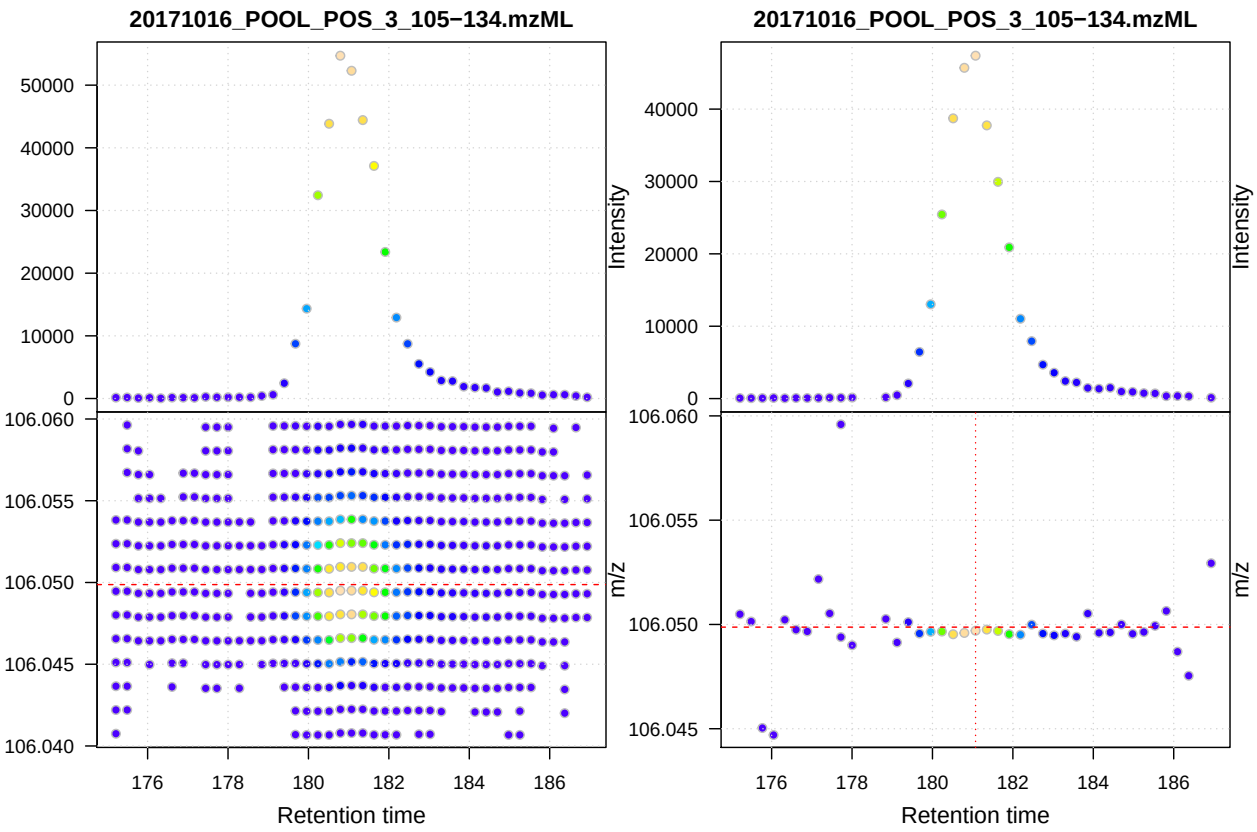

Figure 3: Visualisation of data smoothing and m/z refinement using `MSnbase`. (a) Raw MS profile data for serine. Upper panel shows the base peak chromatogram (BPC), lower panel the individual signals in the retention time – m/z space. The horizontal dashed red line indicates the theoretical m/z of the [M+H]+ adduct of serine. (b) Smoothed and centroided data for serine with m/z refinement. The horizontal red dashed line indicates the theoretical m/z for the [M+H]+ ion and the vertical red dotted line the position of the maximum signal.

Discussion

We have presented here some some important functionality of the version 2 of the `MSnbase` package. The new on-disk infrastructure enables large scale data analyses[17] using `MSnbase` or other packages that rely on it such as `xcms`. We have also illustrated how `MSnbase` can

be used for standard data analysis and visualisation, and how it can be used for method development and prototyping.

The first public commit to the `MSnbase` GitHub repository was in October 2010. Since then, the package benefited from 12 contributors[18] that added various features, some particularly significant ones such as the on-disk backend described above. According to the package's Bioconductor page, there are 36 Bioconductor packages that depend, import or suggest it. Among these are `pRoloc`[19] to analyse mass spectrometry-based spatial proteomics data, `msmsTests`,[20] `DEP`,[21] `DAPAR` and `ProStaR`[22] for the statistical analysis for quantitative proteomics data, `RMassBank`[23] to process metabolomics tandem MS files and build MassBank records, `MSstatsQC`[24] for longitudinal system suitability monitoring and quality control of targeted proteomic experiments and the widely used `xcms`[25] packge for the processing and analysis of metabolomics data. `MSnbase` is also used in non-R/Bioconductor software, such as for example IsoProt,[26] that provides a reproducible workflow for iTRAQ/TMT experiments. `MSnbase` currently ranks 101 out of 1823 packages based on the monthly downloads from unique IP addresses, tallying over 1000 downloads from unique IP addresses each months.

As is custom with Bioconductor packages, `MSnbase` comes with ample documentation. Every user-accessible function is documented in a dedicated manual page. In addition, the package offers 5 vignettes, including one aimed at developers. Questions from users and developers are answered on the Bioconductor support forum as well as on the package GitHub page. The package provides several sample and benchmarking datasets, and relies on other dedicated *experiment packages* such as `msdata`[27] for raw data or `pRolocdata`[19] for quantiative data. `MSnbase` is available on Windows, Mac OS and Linux under the open source Artistic 2.0 license and easily installable using standard installation procedures.

The version of `MSnbase` used in this manuscript is 2.13.9. The main features presented here were available since version 2.0. The code to reproduce the analyses and figures in this article is available at `https://github.com/lgatto/2020-msnbase-v2/`.

Acknowledgement

The authors thank the various contributors and users who have provided constructive input and feedback that have helped, over the years, the improvement of the package. The authors declare no conflict of interest.

References

(1) Côté Richard G, M. L., Reisinger Florian jmzML, an open-source Java API for mzML, the PSI standard for MS data. *Proteomics* **2010**, *10*, 1332–1335.

(2) Chambers, M. C. et al. A cross-platform toolkit for mass spectrometry and proteomics. *Nat Biotechnol* **2012**, *30*, 918–20.

(3) Cox, J.; Mann, M. MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. *Nat Biotechnol* **2008**, *26*, 1367–1372.

(4) Vaudel, M.; Burkhart, J. M.; Zahedi, R. P.; Oveland, E.; Berven, F. S.; Sickmann, A.; Martens, L.; Barsnes, H. PeptideShaker enables reanalysis of MS-derived proteomics data sets. *Nat Biotechnol* **2015**, *33*, 22–24.

(5) Gatto, L.; Lilley, K. S. `MSnbase` - an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation. *Bioinformatics* **2012**, *28*, 288–9.

(6) Huber, W. et al. Orchestrating high-throughput genomic analysis with Bioconductor. *Nat Methods* **2015**, *12*, 115–21.

(7) Rost, H. L.; Schmitt, U.; Aebersold, R.; Lars, M. pyOpenMS: a Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics* **2014**, *14*, 74–77.

(8) Bittremieux, W. spectrum_utils: A Python package for mass spectrometry data processing and visualization. *Analytical Chemistry* **2020**, *92*, 659–661.

(9) Goloborodko, A. A.; Levitsky, L. I.; Ivanov, M. V.; Gorshkov, M. V. Pyteomics - a Python Framework for Exploratory Data Analysis and Rapid Software Prototyping in Proteomics. *J. Am. Soc. Mass Spectrom.* **2013**, *24*, 301–304.

(10) R Core Team, R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing: Vienna, Austria, 2020.

(11) Meier, F.; Geyer, P.; Virreira Winter, S.; Juergen, C.; Matthias, M. BoxCar acquisition method enables single-shot proteomics at a depth of 10,000 proteins in 100 minutes. *Nature Methods* **2018**, *15*, 440–448.

(12) Gatto, L. `MSnbaseBoxCar`: BoxCar Data Processing with MSnbase. 2020; R package version 0.2.0.

(13) Gatto, L.; Breckels, L. M.; Naake, T.; Gibb, S. Visualization of proteomics data using R and Bioconductor. *PROTEOMICS* **2015**, *15*, 1375–1389.

(14) Panse, C.; Grossmann, J. `protViz`: Visualizing and Analyzing Mass Spectrometry Related Data in Proteomics. 2020; R package version 0.6.

(15) Sauteraud, R.; Jiang, M.; Gottardo, R. `Pviz`: Peptide Annotation and Data Visualization using Gviz. 2019; R package version 1.21.0.

(16) Bemis, K. D.; Harry, A.; Eberlin, L. S.; Ferreira, C.; van de Ven, S. M.; Mallick, P.; Stolowitz, M.; Vitek, O. `Cardinal`: an R package for statistical analysis of mass spectrometry-based imaging experiments. *Bioinformatics* **2015**,

(17) Nothias, L. F. et al. Feature-based Molecular Networking in the GNPS Analysis Environment. *bioRxiv* **2019**,

(18) Gatto, L. `MSnbase` contributors 2010 - 2020. 2020; `https://lgatto.github.io/msnbase-contribs-2/`.

(19) Gatto, L.; Breckels, L. M.; Wieczorek, S.; Burger, T.; Lilley, K. S. Mass-spectrometry-based spatial proteomics data analysis using `pRoloc` and `pRolocdata`. *Bioinformatics* **2014**, *30*, 1322–4.

(20) Gregori, J.; Sanchez, A.; Villanueva, J. msmsTests: LC-MS/MS Differential Expression Tests. 2019; R package version 1.25.0.

(21) Zhang, X.; Smits, A. H.; van Tilburg, G. B.; Ovaa, H.; Huber, W.; Vermeulen, M. Proteome-wide identification of ubiquitin interactions using UbIA-MS. *Nature Protocols* **2018**, *13*, 530–550.

(22) Wieczorek, S.; Combes, F.; Lazar, C.; Giai Gianetto, Q.; Gatto, L.; Dorffer, A.; Hesse, A. M.; Couté, Y.; Ferro, M.; Bruley, C.; Burger, T. `DAPAR` & `ProStaR`: software to perform statistical analyses in quantitative discovery proteomics. *Bioinformatics* **2017**, *33*, 135–136.

(23) Stravs, M. A.; Schymanski, E. L.; Singer, H. P.; Hollender, J. Automatic Recalibration and Processing of Tandem Mass Spectra using Formula Annotation. *Journal of Mass Spectrometry* **2013**, *48*, 89–99.

(24) Dogu, E.; Mohammad-Taheri, S.; Abbatiello, S. E.; Bereman, M. S.; MacLean, B.; Schilling, B.; Vitek, O. `MSstatsQC`: Longitudinal System Suitability Monitoring and Quality Control for Targeted Proteomic Experiments. *Mol Cell Proteomics* **2017**, *16*, 1335–1347.

(25) Smith, C. A.; Want, E. J.; O'Maille, G.; Abagyan, R.; Siuzdak, G. XCMS: processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Anal Chem* **2006**, *78*, 779–87.

(26) Griss, J.; Vinterhalter, G.; Schwämmle, V. IsoProt: A Complete and Reproducible Workflow To Analyze iTRAQ/TMT Experiments. *J Proteome Res* **2019**, *18*, 1751–1759.

(27) Neumann, S.; Gatto, L.; Rainer, J. `msdata`: Various Mass Spectrometry raw data example files. 2019; R package version 0.27.0.