MSnbase2 - disk access is the limit

Laurent Gatto and Johannes Rainer

December 7, 2016

Contents

1 Introduction

Find this script at

`https://github.com/lgatto/EuroBioc2016-Basel-MSnbase2`

Most (if not all) of what I will be showing today was implemented by Johannes Rainer and Laurent Gatto during summer 2016 and was released in October 2016 in MSnbase version 2.0.

2 The `MSnExp` class

An `MSnExp` is a mean to store raw data (i.e. spectra) and metadata from 1 or more mass spectrometry (MS) acquisitions. It looks like this:

```
suppressPackageStartupMessages(library("MSnbase"))
getClass("MSnExp")
```

(See `?MSnExp` and the MSnbase development vignette for details.)

The original **in-memory** implementation imported all spectra as `Spectrum1` (or `Spectrum2`) objects in the `assayData` environment, with the obvious effect of consuming quite a bit of memory.

Example: On a server with 128 Gb RAM, loading all MS2 spectra from 14 LC-MSMS acquistions (4.9 Gb on disk) took 90 minutes and resulted in a 3.3 Gb raw data object. That's a reasonable data set from 4 years ago.

3 Keeping data on disk

The `mzR` package uses C/C++ code for fast and random on-disk access of the raw XML-based data files (`mzML`, `mzXML`, …). This is what enables a new **on-disk** version of the `MSnExp` class, called `OnDiskMSnExp` (which extends `MSnExp`).

```
getClass("OnDiskMSnExp")
```

(See `?OnDiskMSnExp` and the MSnbase development vignette for details.)

4 Demo and benchmarking

```
setMSnbaseVerbose(FALSE)
basename(f <- msdata::proteomics(full.names = TRUE)[1])
cat("On disk\n")
on <- readMSData2(f[1]) ## on disk
cat("In memory\n")
im <- readMSData(f[1])  ## in memory
```

```
on
```

```
im
```

Both have the same API (so far at least).
See the MSnbase benchmarking vignette

1. reading data
2. object size
3. accessing spectra
4. subsetting expriments
5. (quantitation)

5 A bit more about the implementation

5.1 Feature data

1. What we have

   ```
   fvarLabels(im)
   ```

   ```
   fvarLabels(on)
   ```

2. And how we use it

   These are used is the filtering functions whenever possible.

   ```
   length(on)
   table(msLevel(on))
   length(filterMsLevel(on, 3L))
   ```

5.2 Processing queue

Whenever possible, access/processing to/of data is delayed (*lazy* processing).

```
library("magrittr")
on2 <- on %>%
    filterMsLevel(3L) %>%
    filterMz(c(126, 132))
on2
```

```
setMSnbaseVerbose(TRUE)
plot(on[["X009.1"]], full = TRUE, centroided = FALSE)
plot(on2[["X009.1"]], full = TRUE, centroided = FALSE)
```

5.3 Parallelisation

Systematic use of `BiocParallel`.

5.4 C-level code

On-the-fly construction of spectra is done using C-level constuctors. This is much faster, mostly due to bypassing method dispatching and the validity check.

5.5 Validity

The default `validObject` doesn't verify the validity on the spectra (as there aren't any to check). Hence, we have a `validateOnDiskMSnExp` function that instantiates all spectra and checks their validity (in addition to calling `validObject`).

```
validObject(on)
validateOnDiskMSnExp(on)
```

5.6 Serialisation

- Yes for in-memory, no for on-disk (can't guarantee that the raw files will stay).
- But on-disk can be coerced to in-memory with `as(on2, "MSnExp")`.

6 More new features:

- Can store any combination of MS levels (in-memory `MSnExp` can only cope with a single level at a time.)

```
table(msLevel(on))
```

- Heavy disk access lead to unconvering (and fixing) a few bugs in `mzR`!
- Consistent filtering functions, convenient with piping.

```
grep("^filter", ls("package:MSnbase"), value = TRUE)
```

7 Conclusion

- `MSnExp` were focused with providing convenient access to raw and meta-data. `OnDiskMSnExp` focus on speed and efficiency.
- Currently both co-exists, with identical (similar) APIs
- This will lead to more common infrastructure/collaboration between proteomics and metabolomics (`xcms3` will be using `OnDiskMSnExp` objects).

8 Acknowledgements

- MSnbase contributors, in particular Sebastian Gibb
- Funding: Biotechnology and Biological Sciences Research Council

Thank you for your attention!