A parser for raw and identification mass-spectrometry data

Bernd Fischer*
Steffen Neumann†
Laurent Gatto‡
Qiang Kou§

July 30, 2016

Contents

1 Introduction

The *mzR* package aims at providing a common interface to several mass spectrometry data formats, namely `mzData` [**?**], `mzXML` [**?**], `mzML` [**?**] for raw data, and `mzIdentML` [**?**], somewhat similar to the Bioconductor package affyio for affymetrix raw data. No processing is done in *mzR*, which is left to packages such as *XCMS* [**?**, **?**] or *MSnbase* [**?**].

Most importantly, access to the data should be fast and memory efficient. This is made possible by allowing on-disk random file access, i.e. retrieving specific data of interest without having to sequentially browser the full content nor loading the entire data into memory.

The actual work of reading and parsing the data files is handled by the included C/C++ libraries or "backends". The `mzRramp` RAMP parser, written at the Institute for Systems Biology (ISB) is a fast and lightweight parser in pure C. Later, it gained support for the `mzData` format. The C++ reference implementation for the `mzML` is the proteowizard library [**?**] (pwiz in short), which in turn makes use of the boost C++ (http://www.boost.org/) library. RAMP is able to access `mzML` files by calling pwiz methods. More recently, the proteowizard[1] [**?**] has been fully integrated using the `mzRpwiz` backend for raw data. The `mzRnetCDF` backend provides support to `CDF`-based formats. Finally, the `mzRident` backend is available to access identification data (`mzIdentML`) through pwiz.

warning: It is anticipated to switch to the `mzRpwiz` backend in Bioconductor 3.1. We advise users and developers to test it and report any issues on the github issue tracker https://github.com/sneumann/mzR/issues.

The *mzR* package is in essence a collection of wrappers to the C++ code, and benefits from the C++ interface provided through the Rcpp package [**?**].

2 Mass spectrometry raw data

All the mass spectrometry file formats are organized similarly, where a set of metadata nodes about the run is followed by a list of spectra with the actual masses and intensities. In addition, each of these spectra has its own set of metadata,

*bernd.fischer@embl.de
†sneumann@ipb-halle.de
‡lg390@cam.ac.uk
§qkou@umail.iu.edu
[1] http://proteowizard.sourceforge.net/

such as the retention time and acquisition parameters.

2.1 Spectral data access

Access to the spectral data is done via the `peaks` function. The return value is a list of two-column mass-to-charge and intensity matrices or a single matrix if one spectrum is queried.

2.2 Identification result access

The main access to identification result is done via `psms`, `score` and `modifications`. `psms` and `score` will return the detailed information on each psm and scores. `modifications` will return the details on each modification found in peptide.

2.3 Metadata access

Run metadata is available via several functions such as `instrumentInfo()` or `runInfo()`. The individual fields can be accessed via e.g. `detector()` etc.

Spectrum metadata is available via `header()`, which will return a list (for single scans) or a dataframe with information such as the `basePeakMZ`, `peaksCount`, ... or, for higher-order MS the `msLevel` and precursor information.

Identification metadata is available via `mzidInfo()`, which will return a list with information such as the `software`, `ModificationSearched`, `enzymes`, `SpectraSource` and other information for this identification result.

The availability of this metadata can not always be guaranteed, and depends on the MS software which converted the data.

3 Example

3.1 `mzXML/mzML/mzData` files

A short example sequence to read data from a mass spectrometer. First open the file.

```
library(mzR)
```

```
## Loading required package:  Rcpp
```

```
library(msdata)

mzxml <- system.file("threonine/threonine_i2_e35_pH_tree.mzXML",
                     package = "msdata")
aa <- openMSfile(mzxml) ## ramp, default backend
```

We can obtain different kind of header information.

```
runInfo(aa)
```

```
## $scanCount
## [1] 55
##
## $lowMz
## [1] 50.0036
##
## $highMz
## [1] 298.673
##
## $dStartTime
## [1] 0.3485
##
## $dEndTime
## [1] 390.027
##
## $msLevels
## [1] 1 2 3 4
```

```
instrumentInfo(aa)
```

```
## $manufacturer
## [1] "Thermo Scientific"
##
## $model
## [1] "LTQ Orbitrap"
##
## $ionisation
## [1] "ESI"
##
## $analyzer
## [1] "FTMS"
##
## $detector
## [1] "unknown"
```

```
header(aa,1)
```

```
## $seqNum
## [1] 1
##
## $acquisitionNum
## [1] 1
##
## $msLevel
## [1] 1
##
## $polarity
## [1] 1
##
## $peaksCount
## [1] 684
##
## $totIonCurrent
## [1] 341427000
##
```

```
## $retentionTime
## [1] 0.3485
##
## $basePeakMZ
## [1] 120.066
##
## $basePeakIntensity
## [1] 211860000
##
## $collisionEnergy
## [1] 0
##
## $ionisationEnergy
## [1] 0
##
## $lowMZ
## [1] 50.3254
##
## $highMZ
## [1] 298.673
##
## $precursorScanNum
## [1] 0
##
## $precursorMZ
## [1] 0
##
## $precursorCharge
## [1] 0
##
## $precursorIntensity
## [1] 0
##
## $mergedScan
## [1] 0
##
## $mergedResultScanNum
## [1] 0
##
## $mergedResultStartScanNum
## [1] 0
##
## $mergedResultEndScanNum
## [1] 0
```

Read a single spectrum from the file.

```
pl <- peaks(aa,10)
peaksCount(aa,10)
```

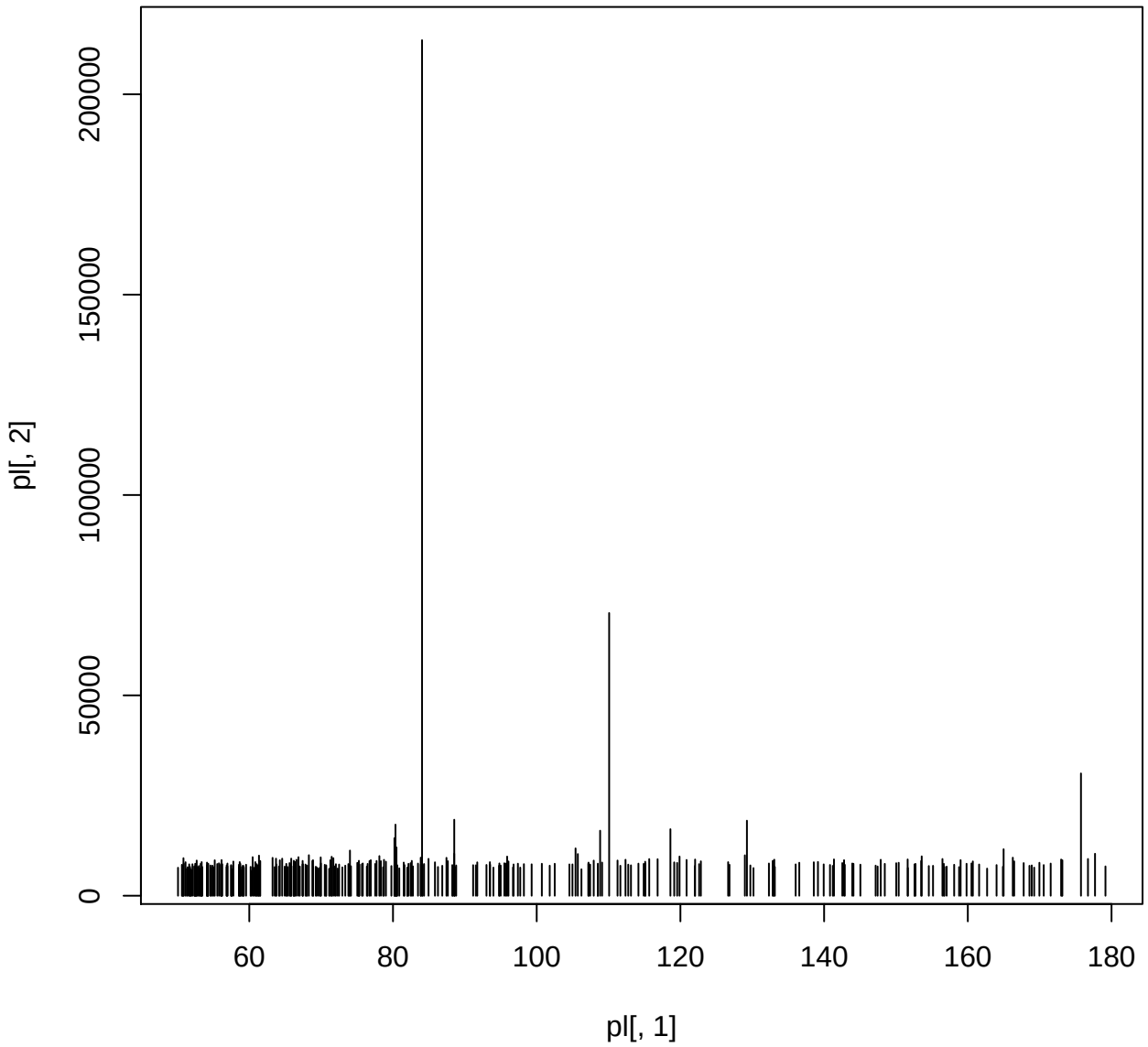
```
## [1] 317
```

```
head(pl)
```

```
##          [,1]     [,2]
## [1,] 50.08176 6984.858
```

```
## [2,] 50.62267 7719.419
## [3,] 50.70530 7185.290
## [4,] 50.73298 7509.140
## [5,] 50.83848 9366.624
## [6,] 50.88303 8012.808
```

```
plot(pl[,1], pl[,2], type="h", lwd=1)
```



One should always close the file when not needed any more if you are using RAMP backend. This will release the memory of cached content.

```
close(aa)
```

3.2 mzIdentML files

You can use openIDfile to read a mzIdentML file (version 1.1), which use the pwiz backend.

```
library(mzR)
library(msdata)

file <- system.file("mzid", "Tandem.mzid.gz", package="msdata")
x <- openIDfile(file)
```

mzidInfo function will return general information about this identification result.

```
mzidInfo(x)
```

```
## $FileProvider
## [1] "researcher"
##
## $CreationDate
## [1] "2012-07-25T14:03:16"
##
## $software
## [1] "xtandem x! tandem CYCLONE (2010.06.01.5) "
## [2] "ProteoWizard MzIdentML 3.0.6239 ProteoWizard"
##
## $ModificationSearched
## [1] "Oxidation"       "Carbamidomethyl"
##
## $FragmentTolerance
## [1] "0.8 dalton"
##
## $ParentTolerance
## [1] "1.5 dalton"
##
## $enzymes
## $enzymes$name
## [1] "Trypsin"
##
## $enzymes$nTermGain
## [1] "H"
##
## $enzymes$cTermGain
## [1] "OH"
##
## $enzymes$minDistance
## [1] "0"
##
## $enzymes$missedCleavages
## [1] "1"
##
##
## $SpectraSource
## [1] "D:/TestSpace/NeoTestMarch2011/55merge.mgf"
```

psms will return the detailed information on each peptide-spectrum-match, include spectrumID, chargeState, sequence. modNum and others.

```
p <- psms(x)
colnames(p)
```

```
##  [1] "spectrumID"               "chargeState"              "rank"
##  [4] "passThreshold"            "experimentalMassToCharge" "calculatedMassToCharge"
##  [7] "sequence"                 "modNum"                   "isDecoy"
## [10] "post"                     "pre"                      "start"
## [13] "end"                      "DatabaseAccess"           "DBseqLength"
## [16] "DatabaseSeq"              "DatabaseDescription"      "acquisitionNum"
```

The modifications information can be accessed using `modifications`, which will return the `spectrumID`, `sequence`, `name`, `mass` and `location`.

```
m <- modifications(x)
head(m)
```

```
##   spectrumID                 sequence            name    mass location
## 1   index=12  LCYIALDFDEEMKAAEDSSDIEK Carbamidomethyl 57.0215        2
## 2   index=12  LCYIALDFDEEMKAAEDSSDIEK       Oxidation 15.9949       12
## 3  index=285   KDLYGNVVLSGGTTMYEGIGER       Oxidation 15.9949       15
## 4   index=83   KDLYGNVVLSGGTTMYEGIGER       Oxidation 15.9949       15
## 5   index=21 VIDENFGLVEGLMTTVHAATGTQK       Oxidation 15.9949       13
## 6  index=198          GVGGAIVLVLYDEMK       Oxidation 15.9949       14
```

Since different software will use different scoring function, we provide a `score` to extract the scores for each psm. It will return a data.frame with different columns depending on software generating this file.

```
scr <- score(x)
colnames(scr)
```

```
## [1] "spectrumID"          "X.Tandem.expect"     "X.Tandem.hyperscore"
```

4 Future plans

Other file formats provided by HUPO, such as `mzQuantML` for quantitative data [**?**] are also possible in the future.

5 Session information

- R version 3.3.1 (2016-06-21), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Rcpp 0.12.6, msdata 0.10.0, mzR 2.6.3
- Loaded via a namespace (and not attached): Biobase 2.32.0, BiocGenerics 0.18.0, BiocStyle 2.0.2, ProtGenerics 1.4.0, codetools 0.2-14, evaluate 0.9, formatR 1.4, highr 0.6, knitr 1.13, magrittr 1.5, parallel 3.3.1, stringi 1.1.1, stringr 1.0.0, tools 3.3.1