

RMassBank for XCMS

Erik Müller

May 3, 2016

Contents

1 Introduction

As the RMassBank-workflow is described in the other manual, this document mainly explains how to utilize the XCMS-, MassBank-, and peaklist-readMethods for Step 1 of the workflow.

2 Input files

2.1 LC/MS data

RMassBank handles high-resolution LC/MS spectra in mzML or mzdata format in centroid¹ or in profile mode. Data in the examples was acquired using a QTOF instrument.

In the standard workflow, the file names are used to identify a compound: file names must be in the format `xxxxxxx_1234_xxx.mzXML`, where the `xxx` parts denote anything and the `1234` part denotes the compound ID in the compound list (see below). Advanced and alternative uses can be implemented; consult the implementation of `msmsRead`, `msms_workflow` and `findMsMSHRperX.direct` for more information.

3 Additional Workflow-Methods

The data used in the following example is available as a package *RMassBankData*, so both libraries have to be installed to run this vignette.

```
> library(RMassBank)
> library(RMassBankData)
```

¹The term "centroid" here refers to any kind of data which are not in profile mode, i.e. don't have continuous m/z data. It does not refer to the (mathematical) centroid peak, i.e. the area-weighted mass peak.

3.1 Options

In the first part of the workflow, spectra are extracted from the files and processed. In the following example, we will process the Glucolesquerellin spectra from the provided files.

For the workflow to work correctly, we use the default settings, and modify then to match the data acquisition method. The settings have to contain the same parameters as the mzR-method would for the workflow.

```
> RmbDefaultSettings()
> rmbo <- getOption("RMassBank")
> rmbo$spectraList <- list(
+   list(mode="CID", ces="10eV", ce="10eV", res=12000),
+   list(mode="CID", ces="20eV", ce="20eV", res=12000)
+ )
> rmbo$multiplicityFilter <- 1
> rmbo$annotations$instrument <- "Bruker micrOTOFq"
> rmbo$annotations$instrument_type <- "LC-ESI-QTOF"
> rmbo$recalibrator$MS1 <- "recalibrate.identity"
> rmbo$recalibrator$MS2 <- "recalibrate.identity"
> options("RMassBank" = rmbo)
>
>
```

3.2 XCMS-workflow

First, a workspace for the msmsWorkflow must be created:

```
> msmsList <- newMsmsWorkspace()
```

The full paths of the files must be loaded into the container in the array files:

```
> msmsList@files <- list.files(system.file("spectra.Glucolesquerellin",
+                                           package = "RMassBankData"),
+                               "Glucolesquerellin.*mzData", full.names=TRUE)
```

Note the position of the compound IDs in the filenames. Historically, the "pos" at the end was used to denote the polarity; it is obsolete now, but the ID must be terminated with an underscore. If you have multiple files for one compound, you have to give them the same ID, but thanks to the polarity at the end being obsolete, you can just enumerate them.

Additionally, the compound list must be loaded using `loadList`:

```
> loadList(system.file("list/PlantDataset.csv", package="RMassBankData"))
```

Basically, the changes to the workflow using XCMS can be described as follows:

The MS2-Spectra (and optionally the MS1-spectrum) are extracted and peakpicked using XCMS. You can pass different parameters for the `findPeaks` function of XCMS using the `findPeaksArgs`-argument to detect actual peaks. Then, CAMERA processes the peak lists and creates pseudospectra (or compound spectra). The obtained pseudospectra are stored in the array `specs`.

Please note that "findPeaksArgs" has to be a list with the list elements named after the arguments that the method you want to use contains, as `findPeaks` is called by `do.call`. For example, if you want to use `centWave` with a peakwidth from 5 to 12 and 25 ppm, `findPeaksArgs` would look like this:

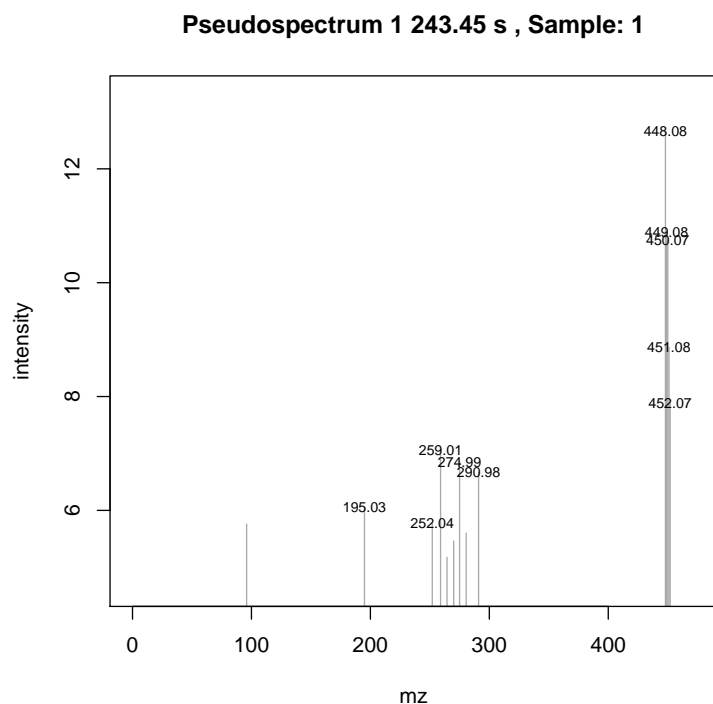
```
> Args <- list(method="centWave",  
+             peakwidth=c(5,12),  
+             prefilter=c(0,0),  
+             ppm=25, snthr=2)
```

If you want to utilize XCMS for Step 1 of the workflow, you have to set the `readMethod`-parameter to "xcms" and - if you don't want to use standard values for `findPeaks` - pass on `findPeaksArgs` to the workflow.

```
> msmsList <- msmsRead(msmsList, files= msmsList@files,  
+                     readMethod = "xcms", mode = "mH", Args = Args, plots = TRUE)
```

MS2 spectra without precursorScan references, using estimation
MS2 spectra without precursorScan references, using estimation

```
> msmsList <- msmsWorkflow(msmsList, steps=2:8,  
+                           mode="mH", readMethod="xcms")
```



You can of course run the rest of the workflow as usual, by - like here - setting steps to 1:8

3.3 Export the records

To export the records from the XCMS workflow, follow the same procedure as the standard RMassBank workflow, i.e.:

```

> mb <- newMbWorkspace(msmsList)
> mb <- resetInfolists(mb)
> mb <- loadInfolist(mb, system.file("infolists/PlantDataset.csv",
+                                   package = "RMassBankData"))
> ## Step
> mb <- mbWorkflow(mb, steps=1:8)

```

3.4 peaklist-workflow

The peaklist-workflow works akin to the normal mzR-workflow with the only difference being, that the supplied data has to be in .csv format and contain 2 columns: "mz" and "int". You can look at an example file in the RMassBankData-package in spectra.Glucolesquerellin. Please note that the naming of the csv has to be similar to the mzdata-files, with the only difference being the filename extension. The readMethod name for this is "peaklist"

```

> msmsPeaklist <- newMsmsWorkspace()
> msmsPeaklist@files <- list.files(system.file("spectra.Glucolesquerellin",
+                                             package = "RMassBankData"),
+                                 "Glucolesquerellin.*csv", full.names=TRUE)
> msmsPeaklist <- msmsWorkflow(msmsPeaklist, steps=1:8,
+                              mode="mH", readMethod="peaklist")

```

The records can then be generated and exported with mbWorkflow.

4 Session information

```
> sessionInfo()
```

```

R version 3.3.0 RC (2016-04-26 r70550)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

```

```
locale:
```

[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:

[1] parallel stats graphics grDevices utils datasets methods
[8] base

other attached packages:

[1] xcms_1.48.0 Biobase_2.32.0 ProtGenerics_1.4.0
[4] BiocGenerics_0.18.0 mzR_2.6.0 gplots_3.0.1
[7] RMassBankData_1.9.0 RMassBank_2.0.0 Rcpp_0.12.4.5

loaded via a namespace (and not attached):

[1] lattice_0.20-33 png_0.1-7 gtools_3.5.0
[4] digest_0.6.9 foreach_1.4.3 chron_2.3-47
[7] plyr_1.8.3 acepack_1.3-3.3 mzID_1.10.0
[10] stats4_3.3.0 ggplot2_2.1.0 BiocInstaller_1.22.0
[13] zlibbioc_1.18.0 data.table_1.9.6 gdata_2.17.0
[16] S4Vectors_0.10.0 rpart_4.1-10 Matrix_1.2-6
[19] preprocessCore_1.34.0 splines_3.3.0 BiocParallel_1.6.0
[22] stringr_1.0.0 foreign_0.8-66 igraph_1.0.1
[25] RCurl_1.95-4.8 munsell_0.4.3 fingerprint_3.5.2
[28] pcaMethods_1.64.0 nnet_7.3-12 gridExtra_2.2.1
[31] Hmisc_3.17-4 IRanges_2.6.0 codetools_0.2-14
[34] XML_3.98-1.4 bitops_1.0-6 grid_3.3.0
[37] RBGL_1.48.0 CAMERA_1.28.0 gtable_0.2.0
[40] affy_1.50.0 magrittr_1.5 rcdk_3.3.2
[43] scales_0.4.0 graph_1.50.0 KernSmooth_2.23-15
[46] stringi_1.0-1 impute_1.46.0 reshape2_1.4.1
[49] affyio_1.42.0 doParallel_1.0.10 limma_3.28.0
[52] latticeExtra_0.6-28 Formula_1.2-1 rjson_0.2.15
[55] RColorBrewer_1.1-2 iterators_1.0.8 tools_3.3.0
[58] MSnbase_1.20.0 rcdklibs_1.5.8.4 survival_2.39-2
[61] yaml_2.1.13 colorspace_1.2-6 cluster_2.0.4
[64] vsn_3.40.0 caTools_1.17.1 MALDIquant_1.14
[67] rJava_0.9-8