RMassBank for XCMS

Erik Müller

April 27, 2020

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 centroid[1] 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 `xxxxxxxx_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)
```

[1]The 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:

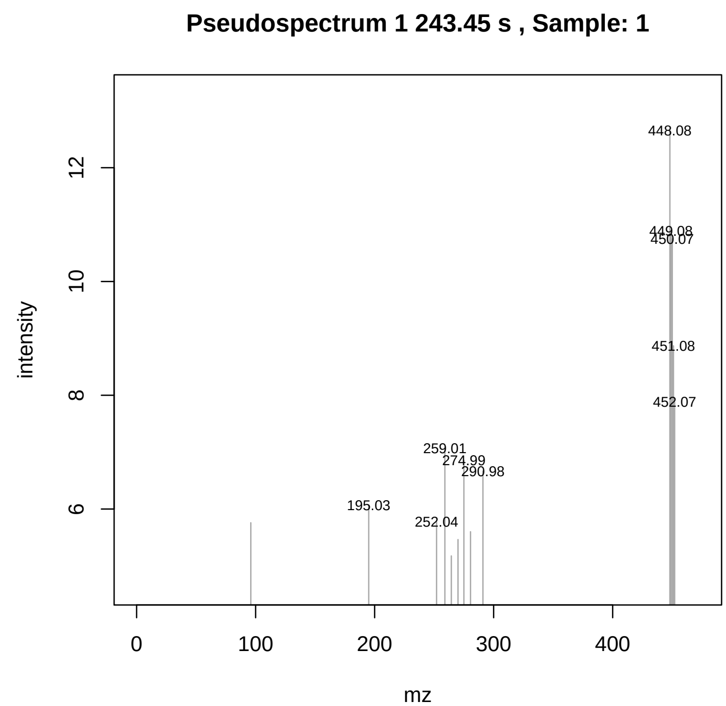
```
> Args <- list(method="centWave",
+                        peakwidth=c(5,12),
+                        prefilter=c(0,0),
+                        ppm=25, snthr=2)
```

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 4.0.0 (2020-04-24)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Mojave 10.14.6

Matrix products: default
```

```
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

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

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] xcms_3.10.0          MSnbase_2.14.0       ProtGenerics_1.20.0
 [4] S4Vectors_0.26.0     mzR_2.22.0           BiocParallel_1.22.0
 [7] Biobase_2.48.0       BiocGenerics_0.34.0  gplots_3.0.3
[10] RMassBankData_1.25.0 RMassBank_2.16.0     Rcpp_1.0.4.6

loaded via a namespace (and not attached):
  [1] bitops_1.0-6                matrixStats_0.56.0
  [3] doParallel_1.0.15           RColorBrewer_1.1-2
  [5] httr_1.4.1                  GenomeInfoDb_1.24.0
  [7] backports_1.1.6             tools_4.0.0
  [9] R6_2.4.1                    affyio_1.58.0
 [11] rpart_4.1-15                KernSmooth_2.23-17
 [13] enviPat_2.4                 Hmisc_4.4-0
 [15] colorspace_1.4-1            nnet_7.3-14
 [17] gridExtra_2.3               tidyselect_1.0.0
 [19] curl_4.3                    compiler_4.0.0
 [21] MassSpecWavelet_1.54.0      preprocessCore_1.50.0
 [23] graph_1.66.0                htmlTable_1.13.3
 [25] DelayedArray_0.14.0         checkmate_2.0.0
 [27] caTools_1.18.0              scales_1.1.0
 [29] DEoptimR_1.0-8              robustbase_0.93-6
 [31] affy_1.66.0                 RBGL_1.64.0
 [33] stringr_1.4.0               digest_0.6.25
 [35] foreign_0.8-79              XVector_0.28.0
 [37] htmltools_0.4.0             jpeg_0.1-8.1
 [39] base64enc_0.1-3             pkgconfig_2.0.3
 [41] itertools_0.1-3             limma_3.44.0
 [43] htmlwidgets_1.5.1           rlang_0.4.5
 [45] rstudioapi_0.11             impute_1.62.0
```

```
 [47] mzID_1.26.0                 gtools_3.8.2
 [49] acepack_1.4.1               dplyr_0.8.5
 [51] RCurl_1.98-1.2              magrittr_1.5
 [53] GenomeInfoDbData_1.2.3      Formula_1.2-3
 [55] MALDIquant_1.19.3           Matrix_1.2-18
 [57] munsell_0.5.0               lifecycle_0.2.0
 [59] vsn_3.56.0                  stringi_1.4.6
 [61] yaml_2.2.1                  MASS_7.3-51.6
 [63] SummarizedExperiment_1.18.0 zlibbioc_1.34.0
 [65] plyr_1.8.6                  grid_4.0.0
 [67] gdata_2.18.0                crayon_1.3.4
 [69] lattice_0.20-41             rcdklibs_2.3
 [71] splines_4.0.0               knitr_1.28
 [73] pillar_1.4.3                igraph_1.2.5
 [75] GenomicRanges_1.40.0        rjson_0.2.20
 [77] codetools_0.2-16            rcdk_3.5.0
 [79] XML_3.99-0.3                glue_1.4.0
 [81] latticeExtra_0.6-29         data.table_1.12.8
 [83] pcaMethods_1.80.0           BiocManager_1.30.10
 [85] CAMERA_1.44.0               png_0.1-7
 [87] vctrs_0.2.4                 foreach_1.5.0
 [89] gtable_0.3.0                RANN_2.6.1
 [91] purrr_0.3.4                 assertthat_0.2.1
 [93] ggplot2_3.3.0               xfun_0.13
 [95] ncdf4_1.17                  survival_3.1-12
 [97] tibble_3.0.1                rJava_0.9-12
 [99] iterators_1.0.12            fingerprint_3.5.7
[101] IRanges_2.22.0              cluster_2.1.0
[103] ellipsis_0.3.0
```