

# xcms-based preprocessing of a large LC-MS/MS data set; xcms version 3 with MSnbase infrastructure

Supporting Information for: *xcms in peak form: Now anchoring a complete metabolomics  
data preprocessing and analysis software ecosystem*

Philippine Louail<sup>1,2</sup>, Carl Brunius<sup>3</sup>, Mar Garcia-Aloy<sup>4</sup>, William Kumler<sup>5</sup>, Norman Storz<sup>6</sup>, Jan Stanstrup<sup>7</sup>,  
Hendrik Treutler<sup>6</sup>, Pablo Vangeenderhuysen<sup>8</sup>, Michael Witting<sup>9,10</sup>, Steffen Neumann<sup>6,11,\*</sup>, Johannes Rainer<sup>1</sup>

<sup>1</sup>Institute for Biomedicine, Eurac Research, 39100, Bolzano, Italy.

<sup>2</sup>Chair for Bioinformatics, Friedrich-Schiller-University Jena, 07743, Jena, Germany.

<sup>3</sup>Department of Life Sciences, Food and Nutrition Science, Chalmers University of Technology, SE-412 96 Göte-borg, Sweden.

<sup>4</sup>Metabolomics Unit, Research and Innovation Centre, Fondazione Edmund Mach, 38098, San Michele all'Adige (TN), Italy.

<sup>5</sup>School of Oceanography, University of Washington, Seattle, WA, 98195, USA.

<sup>6</sup>Leibniz Institute of Plant Biochemistry, MetaCom, Weinberg 3, 06120 Halle, Germany.

<sup>7</sup>Department of Nutrition, Exercise and Sports, University of Copenhagen, Rolighedsvej 26, 1958 Frederiksberg C, Denmark.

<sup>8</sup>Laboratory of Integrative Metabolomics (LIMET), Ghent University, 9820 Merelbeke, Belgium.

<sup>9</sup>Metabolomics and Proteomics Core, Helmholtz Zentrum München, Ingolstädter Landstraße 1, 85764 Neuherberg, Germany.

<sup>10</sup>Chair of Analytical Food Chemistry, TUM School of Life Sciences, Technical University of Munich, Maximus-von-Imhof-Forum 2, 85354 Freising-Weihenstephan, Germany.

<sup>11</sup>German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Puschstraße 4, 04103 Leipzig, Germany.

\* corresponding author: sneumann@ipb-halle.de

## Abstract

This document describes the preprocessing of a large untargeted metabolomics dataset consisting of about 1,000 samples using the *xcms* package (Louail, Brunius, et al. 2025) and export of its results for molecular networking with GNPS. For the present analysis, *xcms* version 3.7.2, part of Bioconductor release 3.10, was used. This version of *xcms* uses the mass spectrometry data infrastructure and data objects from the *MSnbase* package. To evaluate the performance, memory usage and timing of function calls are recorded during the analysis.

## Introduction

In this document we perform the preprocessing of a large untargeted metabolomics dataset using the *xcms* package (Louail, Brunius, et al. 2025) and record memory usage and timing of function calls during the analysis. The dataset consists of approximately **1,000 samples** (mzXML) files which can be downloaded from the *MassIVE* repository. The accession ID of the data set is *MSV000080030*. Preprocessing of this data set was also performed in (Nothias et al. 2020) to exemplify large-scale analysis with *xcms* in the framework of the GNPS environment. The analysis in this document uses the same *xcms* functions and classes from that analysis, i.e., relies on *MSnbase* for MS data handling.

The analysis in this document was defined based on Bioconductor version 3.10 with R version 3.7.3. The versions of the *xcms* and *MSnbase* packages are listed in the *Session and computation information* section. Running the code from this file on newer versions of R/Bioconductor, respectively versions of the *xcms* package will most likely result in errors. The present analysis was run using the *RELEASE\_3\_10* Bioconductor docker image (install with `docker pull bioconductor/bioconductor_docker:RELEASE_3_10`).

Note that the settings for the individual preprocessing steps were not optimized for the present data set. For information on how to derive settings specific for a data set see the Metabonaut resource (Louail, Graeve, et al. 2025).

## Data import

It is assumed that the mzXML files were downloaded from MassIVE and the *MSV000080030* folder was placed within the same directory this quarto document resides. We first load all required libraries and configure the parallel processing setup. For the present analysis we use 4 parallel processes. To track memory usage and duration of the various function calls we use the `peakRAM()` function from the same-named R package. Note however that this function fails to properly track memory usage in R if parallel processing is used.

```
library(MSnbase)
library(xcms)
library(peakRAM)
```

Setup parallel processing.

```
NCORES <- 4
register(MulticoreParam(NCORES), default = TRUE)
```

We next define all data files of the project, drop measurements of blanks and import the data as an `OnDiskMSnExp` object.

```
fls <- dir("MSV000080030", pattern = "mzXML$", full.names = TRUE,
          recursive = TRUE)
fls <- fls[!grep(fls, pattern = "/Blank", fixed = TRUE)]
pr <- peakRAM(
  od <- readMSData(fls, mode = "onDisk")
)
```

The size of the object in memory is shown below. Note that only spectra metadata is loaded into memory, while the mass peak data is retrieved on-the-fly from the original data files.

```
print(object.size(od), units = "GB")
```

```
## 1.4 Gb
```

## Chromatographic peak detection

Chromatographic peak detection is performed using the *CentWave* algorithm using the same settings as in (Nothias et al. 2020).

```
cwp <- CentWaveParam(snthresh = 3, noise = 800, peakwidth = c(2, 30), ppm = 20)
tmp <- peakRAM(
  xod <- findChromPeaks(od, param = cwp)
)
pr <- rbind(pr, tmp)
```

In total 4105820 chromatographic peaks were identified.

Peak refinement is not available with *xcms* version used in this analysis (3.8.2), thus we have to skip this step.

The size of the result object in memory is:

```
print(object.size(xod), units = "GB")
```

```
## 1.4 Gb
```

## Alignment

Prior retention time alignment we need to perform an initial correspondence analysis. Among the such defined features (i.e., chromatographic peaks grouped across samples) *anchor peaks* will be defined that are used to reduce retention time shifts and hence align the samples.

```
pdp <- PeakDensityParam(sampleGroups = rep(1, length(fileNames(xod))),
  bw = 2, binSize = 0.1, minFraction = 0.5)
tmp <- peakRAM(
  xod <- groupChromPeaks(xod, param = pdp)
)
```

```
## Processing 38561 mz slices ... OK
```

```
pr <- rbind(pr, tmp)
```

Next we perform the retention time alignment.

```
pgp <- PeakGroupsParam(minFraction = 0.7, extraPeaks = 100, span = 0.3)
tmp <- peakRAM(
  xod <- adjustRtime(xod, param = pgp)
)
```

```
## Performing retention time correction using 387 peak groups.
```

```
## Warning: Adjusted retention times had to be re-adjusted for some files to
## ensure them being in the same order than the raw retention times. A call to
## 'dropAdjustedRtime' might thus fail to restore retention times of
## chromatographic peaks to their original values. Eventually consider to increase
## the value of the 'span' parameter.
```

```
## Applying retention time adjustment to the identified chromatographic peaks ...
```

```
## OK
```

```
pr <- rbind(pr, tmp)
```

The size of the result object in memory is:

```
print(object.size(xod), units = "GB")
```

```
## 1.4 Gb
```

## Correspondence analysis

We next perform the correspondence analysis.

```
pdp <- PeakDensityParam(sampleGroups = rep(1, length(fileNames(xod))),
                        bw = 2, binSize = 0.015, minFraction = 0.03)
tmp <- peakRAM(
  xod <- groupChromPeaks(xod, param = pdp)
)
```

```
## Processing 257066 mz slices ... OK
```

```
pr <- rbind(pr, tmp)
```

After correspondence analysis we have 19941 features.

The size of the result object in memory is:

```
print(object.size(xod), units = "GB")
```

```
## 1.4 Gb
```

## Gap filling

As a last step in the preprocessing, we perform the gap-filling to reduce the number of missing values. The only gap-filling method available for the currently used *xcms* version 3.8.2 is the `FillChromPeaksParam()` method.

```
## Fill missing peak data
medWidth <- median(chromPeaks(xod)[, "rtmax"] -
                  chromPeaks(xod)[, "rtmin"])
tmp <- peakRAM(
  xod <- fillChromPeaks(
    xod, param = FillChromPeaksParam(fixedRt = medWidth))
)
```

```
## Defining peak areas for filling-in .... OK
```

```
## Start integrating peak areas from original files
```

```
pr <- rbind(pr, tmp)
```

The size of the result object in memory is:

```
print(object.size(xod), units = "GB")
```

```
## 1.4 Gb
```

## Summary of preprocessing performance

The time elapse for the individual preprocessing steps along with memory usage is shown in the table below. Note that on parallel processing setups the memory usage is not tracked properly by the `peakRAM()` function.

```
library(pander)
pr$Elapsed_Time_sec <- pr$Elapsed_Time_sec / 60
colnames(pr)[colnames(pr) == "Elapsed_Time_sec"] <- "Elapsed_Time_minutes"
pandoc.table(pr, style = "rmarkdown")
```

Table 1: Table continues below

Function_Call		
od<-readMSData(fl,mode="onDisk")		
xod<-findChromPeaks(od,param=cwp)		
xod<-groupChromPeaks(xod,param=pdp)		
xod<-adjustRtime(xod,param=pgp)		
xod<-groupChromPeaks(xod,param=pdp)		
xod<-fillChromPeaks(xod,param=FillChromPeaksParam(fixedRt=medWidth))		
Elapsed_Time_minutes	Total_RAM_Used_MiB	Peak_RAM_Used_MiB
26.65	1184	3821
77.29	486.2	2492
0.5264	7.5	2008
7.926	105.9	2024
1.796	25.7	1954
696.3	2093	10728

```
dr <- "xcms-preprocessing-msnbase"
dir.create(dr, showWarnings = FALSE)
save(xod, file = file.path(dr, "xod.RData"))
```

## Prepare and export data for GNPS

We next identify all MS2 spectra for all defined LC-MS features and export these, along with the feature abundances for use in the GNPS environment as described in (Nothias et al. 2020).

We first extract the feature values from the result object.

```
fall <- featureDefinitions(xod)[, c("mzmin", "mzmed", "mzmax",
                                   "rtmin", "rtmed", "rtmax")]
fall <- cbind(feature_id = rownames(fall),
              fall,
              featureValues(xod, value = "into"))
write.table(fall, file.path(dr, "xcms_all_features.txt"), sep = "\t",
            quote = FALSE, row.names = FALSE)
```

Next we extract all MS2 spectra associated with any of the features' detected chromatographic peaks. Note that we disable parallel processing for this step, because we would run out of memory on the computer setup used.

```
bpstop()
register(SerialParam())
source("https://raw.githubusercontent.com/jorainer/xcms-gnps-tools/master/customFunctions.R")
pr_ms2 <- peakRAM(
  ms2_all <- clean(featureSpectra(xod, return.type = "Spectra"), all = TRUE)
)
```

In total 2649099 MS2 spectra were identified and extracted for the 19941 LC-MS features.

We next select a single MS2 spectrum for each feature and export this reduced set also as an .mgf file.

```
tmp <- peakRAM(
  ## Select for each feature the MS2 spectrum with the largest TIC
  ms2_maxtic <- combineSpectra(ms2_all, fcol = "feature_id",
                              fun = maxTic)
)
```

```
## Warning: Parameter 'fun' is deprecated. Please use 'method' instead
```

```
pr_ms2 <- rbind(pr_ms2, tmp)
```

Next we export the data to a file which can then be submitted to GNPS feature-based molecular networking.

```
tmp <- peakRAM(
  writeMgfData(ms2_maxtic, file.path(dr, "ms2_maxtic.mgf"))
)
pr_ms2 <- rbind(pr_ms2, tmp)
```

## Summary of data export performance

The time elapse for the individual result extraction, formatting for GNPS and export steps along with memory usage is shown in the table below.

```
pr_ms2$Elapsed_Time_sec <- pr_ms2$Elapsed_Time_sec / 60
colnames(pr_ms2)[colnames(pr_ms2) == "Elapsed_Time_sec"] <-
  "Elapsed_Time_minutes"
pandoc.table(pr_ms2, style = "rmarkdown")
```

Table 3: Table continues below

Function_Call		
ms2_all<-clean(featureSpectra(xod,return.type="Spectra"),all=TRUE)		
ms2_maxtic<-combineSpectra(ms2_all,fcol="feature_id",fun=maxTic)		
writeMgfData(ms2_maxtic,file.path(dr,"ms2_maxtic.mgf"))		
Elapsed_Time_minutes	Total_RAM_Used_MiB	Peak_RAM_Used_MiB
45.6	21127	44085
0.4003	0.4	512.5
0.8925	0.1	1999

## Session and computation information

The present analysis was performed using the Bioconductor docker images for the developmental version 3.22. The analysis was run on 4 CPUs of a Framework 13' notebook with a 13th Gen Intel(R) Core(TM) i7-1370P CPU (20 cores) and 64 GB of main memory.

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 10 (buster)
```

```
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/x86_64-linux-gnu/libopenblas-r0.3.5.so
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
## [3] LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
## [5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=C
## [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
## [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4      parallel  stats      graphics  grDevices  utils      datasets
## [8] methods     base
##
## other attached packages:
## [1] pander_0.6.6      peakRAM_1.0.2      xcms_3.8.2
## [4] BiocParallel_1.20.1 MSnbase_2.12.0      ProtGenerics_1.18.0
## [7] S4Vectors_0.24.4  mzR_2.20.0         Rcpp_1.0.4.6
## [10] Biobase_2.46.0     BiocGenerics_0.32.0 BiocStyle_2.14.4
## [13] rmarkdown_2.30     BiocManager_1.30.10
##
## loaded via a namespace (and not attached):
## [1] lattice_0.20-41      assertthat_0.2.1      digest_0.6.25
## [4] foreach_1.5.2        R6_2.4.1              plyr_1.8.9
## [7] mzID_1.24.0          evaluate_1.0.5        ggplot2_3.3.0
## [10] pillar_1.4.3         zlibbioc_1.32.0       rlang_1.1.6
## [13] Matrix_1.2-18        preprocessCore_1.48.0 splines_3.6.3
## [16] munsell_0.5.0        compiler_3.6.3        xfun_0.53
## [19] pkgconfig_2.0.3      multtest_2.42.0       pcaMethods_1.78.0
## [22] htmltools_0.5.8.1    tibble_3.0.0          RANN_2.6.2
## [25] IRanges_2.20.2       codetools_0.2-16     XML_3.99-0.3
## [28] fansi_0.4.1          crayon_1.3.4          MASS_7.3-51.5
## [31] MassSpecWavelet_1.52.0 grid_3.6.3            gtable_0.3.0
## [34] lifecycle_0.2.0      affy_1.64.0           magrittr_1.5
## [37] scales_1.1.0         ncd4_1.24             cli_2.0.2
## [40] impute_1.60.0        affyio_1.56.0         doParallel_1.0.17
## [43] limma_3.42.2         robustbase_0.99-6     ellipsis_0.3.0
## [46] vctr_0.2.4           RColorBrewer_1.1-3    iterators_1.0.14
## [49] tools_3.6.3          glue_1.4.0            DEoptimR_1.1-4
## [52] fastmap_1.2.0        survival_3.1-12       yaml_2.2.1
## [55] colorspace_1.4-1     vsn_3.54.0            MALDIquant_1.19.3
## [58] knitr_1.50
```

## References

Louail, Philippine, Carl Brunius, Mar Garcia-Aloy, William Kumler, Norman Storz, Jan Stanstrup, Hendrik Treutler, et al. 2025. “Xcms at 20 and Still in Peak Form: Anchoring a Complete Metabolomics Data Preprocessing and Analysis Software Ecosystem.” ChemRxiv. <https://doi.org/10.26434/chemrxiv-2025-2n2kh>.

Louail, Philippine, Marilyn De Graeve, Anna Tagliaferri, Vinicius Verri Hernandez, Daniel Marques de Sá e Silva, and Johannes Rainer. 2025. “Rformassspectrometry/Metabonaut: V1.2.0.” Zenodo. <https://doi.org/10.5281/zenodo.15554287>.

Nothias, Louis-Félix, Daniel Petras, Robin Schmid, Kai Dührkop, Johannes Rainer, Abinesh Sarvepalli, Ivan Protsyuk, et al. 2020. “Feature-Based Molecular Networking in the GNPS Analysis Environment.” *Nature Methods* 17 (9): 905–8. <https://doi.org/10.1038/s41592-020-0933-6>.