LC/MS Preprocessing and Analysis with xcms

Colin A. Smith

January 24, 2008

Introduction

This document describes how to use xcms to preprocess LC/MS data for relative quantitation and statistical analysis. It gives examples of how visualization can be used throughout the process and to display final results. An overview of the preprocessing/analysis methodology, along with the function names associated with each step, is shown in Figure 1

1 Raw Data File Preparation

The xcms package reads full-scan LC/MS data from AIA/ANDI format NetCDF, mzXML, and mzData files. All data to be analyzed by xcms must first be converted to one of those file formats. Software packages for many instruments are able to export to NetCDF. For information about how to export to NetCDF, please consult the documentation that came with your instrument’s software. The online help of most packages frequently use the terms “CDF” or “AIA” when referring to NetCDF format. In addition to NetCDF, mzXML exporters for a number of instruments are also available.

After exporting all files to NetCDF/mzXML/mzData format, they should be put in a location that will remain the same throughout the analysis. That is because xcms records the location of the raw data files and refers back to them a number of times during preprocessing and analysis.

In most cases, LC/MS files that were acquired under different conditions should not be compared. For instance, positive and negative ionization mode files will have no ions in common and should thus be preprocessed separately. Similarly, data files acquired with different elution gradients should not be processed together.

Another important consideration is the directory structure in which the files are organized. xcms uses sample class information during preprocessing to help decide which groups of peaks are significant. If organized into subdirectories, samples will automatically be assigned to separate classes based on their location. Samples may be separated

http://sashimi.sourceforge.net/software_glossolalia.html
Figure 1: Flow chart showing a high-level overview of the preprocessing/analysis methodology employed by \textit{xcms}. Function/method names corresponding to each step are also given.
into class based on tissue type, mutation, gender, disease, or time. For example, if you are analyzing the longitudinal effect of a drug in two patient groups, you may wish to put the groups into two directories “GroupA” and “GroupB”. Within each of those directories, you could further separate the samples by the time they were taken, such as “Day1”, “Day2”, etc. In xcms, they will be automatically assigned class names “GroupA/Day1”, “GroupA/Day2”, etc.

For the purposes of demonstration, we will use a subset of the data from Saghatelian et al. (2004) examining the metabolic consequences of knocking out the fatty acid amide hydrolase (FAAH) gene in mice. The raw data files are contained in the cdf directory of the faahKO data package. There are samples from the spinal cords of 6 knockout mice and 6 wild-type mice placed in two subdirectories. Each file contains centroided data acquired in positive ion mode from 200-600 m/z and 2500-4500 seconds. To access the NetCDF files, we first locate the cdf directory in the faahKO package.

```r
> cdfpath <- system.file("cdf", package = "faahKO")
> list.files(cdfpath, recursive = TRUE)

```

### 2 Filtration and Peak Identification

The class of objects used for preprocessing analyte data from multiple LC/MS files is xcmsSet. It stores peak lists and provides methods for grouping and aligning those peaks. To create an xcmsSet object from a set of NetCDF files, use the xcmsSet() constructor function. It handles batch peak picking and generation of the xcmsSet object. There are a number of ways you can specify the files it should read. By default, it will recursively search through the current directory for NetCDF/mzXML/mzData files. Alternatively, you can manually specify the files you are interested in, as shown below.

During peak identification, xcms uses a separate line for each sample to report on the status of processing. It outputs out pairs of numbers separated by a colon. The first number is the m/z it is currently processing. The second number is the number of peaks that have been identified so far. It is important to note that the number may be significantly larger than the final number of peaks as a vicinity elimination postprocessing step removes duplicate peaks corresponding to the same ion.

```r
> library(xcms)
> cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
> xset <- xcmsSet(cdffiles)

```
A "xcmsSet" object with 12 samples

Time range: 2506.1-4147.7 seconds (41.8-69.1 minutes)
Mass range: 200.1-599.3338 m/z
Peaks: 4721 (about 393 per sample)
Peak Groups: 0
Sample classes: KO, WT

Profile settings: method = bin
step = 0.1

Memory usage: 0.709 MB

The default arguments for xcmsSet should work acceptably in most cases. However, there are a number of parameters that may need to be optimized for a particular instrument or group of samples. The full set of parameters can be seen by viewing the documentation for the xcmsSet function and findPeaks method.

The method findPeaks can make use of different algorithms for peak detection. Currently findPeaks.matchedFilter and findPeaks.centWave are available, findPeaks.matchedFilter is used by default. First, several of the most important parameters of findPeaks.matchedFilter will be discussed.

findPeaks.matchedFilter
One parameter to consider is the Gaussian model peak width used for matched filtration, an integral part of the peak detection algorithm. For a discussion of how model peak width affects the signal to noise ratio, see Danielsson et al. (2002). It can be specified as either the standard deviation (sigma) or full width at half maximum (fwhm). By default, a FWHM of 30 seconds is used. Depending on the type of chromatography, the correct model peak width can be quite different. One means of determining the peak width is to fit the Gaussian function to one or more peaks in representative samples.
produced with your experimental protocol. Functionality for doing so is provided in the
`plotChrom` method with the `fitgauss` argument set to TRUE.

Several parameters depend on the resolution your mass spectrometer. Prior to
matched filtration, the peak detection algorithm creates extracted ion base peak chro-
matograms (EIBPC) on a fixed step size defined by the `step` argument (default 0.1 m/z).
To take into account uncertainties in scan to scan mass accuracy, the peak identification
algorithm combines a given number of EIBPCs prior to filtration and peak detection,
as defined by the `steps` argument. The default value, 2, combines EIBPCs 1-2, 2-3, 3-4,
etc. If the peak width is significantly greater than the step size, you may wish to turn
off combination using a value of 1. If your scan to scan accuracy is worse, you may wish
to increase the number of scans combined. For example, a value of 3 would combine
EIBPCs 1-3, 2-4, 3-5, etc.

Another factor to consider is the algorithm by which EIBPCs are produced. One way
of thinking about that process is as a transformation of the data from being separate
lists of mass/intensity pairs (one list for each scan) to a matrix with rows representing
equally spaced masses and a column for each scan. Data transformed into such a matrix
is usually referred to as being in profile mode. To do so, each scan of unequally spaced
masses must be mapped onto a column of the final matrix. The algorithm used to do so
is selected using the `profmethod` argument and can be either “bin”, “binlin”, “binlinbase”,
or “intlin”.

The simplest algorithm, “bin”, simply bins the intensity into the matrix cell closest
to it in mass. If more than one intensity value is assigned to the same cell, then the
greatest intensity is used. All other matrix cells are left at zero. It is the default and
is especially useful for centroided data. “binlin” does the same thing except that it uses
linear interpolation to fill in cells that otherwise would have been left at zero. It works
well for sparsely populated continuum data.

Some mass spectrometry software allows the user to set an intensity threshold be-
low which no mass/intensity values are recorded in continuum mode. When the mass
spectral signal falls below that threshold, simple linear interpolation will not provide a
good approximation of the original signal, instead creating artificially high background.
To address that, the “binlinbase” method uses linear interpolation between data points
within 0.15 m/z of each other, and otherwise inserts a basal intensity value set to half of
the minimum intensity. Those specific parameters can be changed using the `profparam`
argument. See documentation for the function `profBinLinBase` for more details.

The last method, “intlin”, uses integration and linear interpolation between mass/intensity
pairs to determine the equally spaced intensity values. This has the advantage of being
numerically stable regardless of the mass step size. However, it is more useful for visu-
alization than peak identification and is generally not recommended as such.

`findPeaks.centWave`

The method `findPeaks.centWave` follows a different approach. This algorithm is most
suitable for high resolution centroid LC/TOF-MS data. If your machine software or con-
verter does not support the export of peak centroids, tools like the `TOPP-PeakPicker`
can be used for preprocessing. Due to the fact that peak centroids are used, a binning step is not necessary.

In the first phase of the algorithm areas of high peak density (characterised by having at least \textit{minEntries} peaks within \textit{dev} ppm in consecutive scans) in the LC/MS map are located. In the second phase these \textit{regions of interest} (ROI) are further analysed. Gaussian curves are fitted to the chromatographic peaks, and areas larger than \(2 \times \text{minPeakWidth}\) scans are checked whether they contain more than one peak (these may also overlap) using wavelet analysis.

This algorithm is much slower than \textit{findPeaks.matchedFilter} due to the extensive gaussian fits. It can be sped up by narrowing the ROI’s found in the first phase. This can be done by restricting the parameter \textit{dev} (default value is 140 ppm for a 10 ppm QTOF), depending on your machine’s mass accuracy.

The method is capable of detecting close-by-peaks and also overlapping peaks. Some efforts are made to detect the exact peak boundaries to get precise peak integrals. The peak attributes \textit{sn} (Signal/Noise Ratio) and \textit{egauss} (root-mean-square-error of the gaussian fit) can be used to assess the peak quality.

3 Matching Peaks Across Samples

After peak identification, peaks representing the same analyte across samples must be placed into groups. That is accomplished with the \textit{group} method, which returns a new \textit{xcmsSet} object with the additional group information. The grouping process is non-destructive and does not affect the other data stored in the \textit{xcmsSet} object. Therefore, we can safely replace the \textit{xset} object with the grouped version. The grouping algorithm processes the peak lists in order of increasing mass and will regularly output the mass it is currently working on.

```r
> xset <- group(xset)
```

262 325 387 450 512 575

There are several grouping parameters to consider optimizing for your chromatography and mass spectrometer. Please consult the \textit{group} documentation for more details. To see what the algorithm is doing while running, use the \textit{sleep} argument to specify a time (in seconds) to pause and plot each iteration. That can be quite useful for visualizing parameter effects.

4 Retention Time Correction

After matching peaks into groups, \textit{xcms} can use those groups to identify and correct correlated drifts in retention time from run to run. The aligned peaks can then be
used for a second pass of peak grouping which will be more accurate than the first. The whole process can be repeated in an iterative fashion, although we will only demonstrate a single pass of retention time alignment here.

Not all peak groups will be helpful for identifying retention time drifts. Some groups may be missing peaks from a large fraction of samples and thus provide an incomplete picture of the drift at that time point. Still others may contain multiple peaks from the same sample, which is a sign of improper grouping. `xcms` ignores those groups by only considering “well-behaved” peak groups which are missing at most one sample and have at most one extra peak. (Those values can be changed with the `missing` and `extra` arguments.)

For each of those well-behaved groups, the algorithm calculates a median retention time and, for every sample, a deviation from that median. Within a sample, the observed deviation generally changes over time in a nonlinear fashion. Those changes are approximated using a local polynomial regression technique implemented in the `loess` function. By default, the curve fitting is done using least-squares on all data points. However, it is possible to enable outlier detection and removal by setting the `family` argument to "symmetric", as shown here.

Retention time correction is performed by the `retcor` method, which returns an `xcms-Set` object with corrected retention times. Because it changes the retention times of all peaks, it is important to store the new object under a new variable name. That will allow you to backtrack and repeat retention time correction if necessary.

```r
> xset2 <- retcor(xset, family = "symmetric", plottype = "mdevden")
```

Retention Time Correction Groups: 133

The above command uses the `plottype` argument to produce a plot, shown in Figure 2, which is useful for supervising the algorithm. It includes the data points used for loess regression and the resulting deviation profiles. It additionally shows the distribution of peak groups across retention time.

After retention time correction, the initial peak grouping becomes invalid and is discarded. Therefore, the resulting object needs to be regrouped. Here, we decrease the inclusiveness of the grouping using the `bw` argument (default 30 seconds).

```r
> xset2 <- group(xset2, bw = 10)
```

5 Filling in Missing Peak Data

After the second pass of peak grouping, there will still be peak groups which are missing peaks from some of the samples. That can occur because peaks were missed during peak
Figure 2: Retention time deviation profiles used for aligning the samples. The data points used for generating each profile are also shown. All times are in seconds. A negative number indicates a sample was eluting before most of the others, and vice versa. Samples that were acquired on the same day are colored similarly and have correlated deviation profiles, as expected. Below, kernel density estimation is used to show the distribution of all peaks and those peaks used as standards for retention time correction. Examples of two peaks before and after alignment are shown in Figure [4].
identification or because an analyte was not present in a sample. In any case, those missing data points can be filled in by rereading the raw data files and integrating them in the regions of the missing peaks. That is performed using the `fillPeaks` method, which returns a `xcmsSet` object with the filled in peak data. While running, it outputs the name of the sample it is currently processing.

```r
> xset3 <- fillPeaks(xset2)
ko15 ko16 ko18 ko19 ko21 ko22 wt15 wt16 wt18 wt19 wt21 wt22
> xset3
An "xcmsSet" object with 12 samples

Time range: 2501.2-4148.8 seconds (41.7-69.1 minutes)
Mass range: 200.1-599.3338 m/z
Peaks: 6053 (about 504 per sample)
Peak Groups: 402
Sample classes: KO, WT

Profile settings: method = bin
  step = 0.1

Memory usage: 0.907 MB
```

## 6 Analyzing and Visualizing Results

A report showing the most statistically significant differences in analyte intensities can be generated with the `diffreport` method. It will automatically generate extracted ion chromatograms for a given number of them, in this case 10. Several of those chromatograms are shown in Figure 3.

```r
> reporttab <- diffreport(xset3, "WT", "KO", "example", 10, metlin = 0.15)
ko15 ko16 ko18 ko19 ko21 ko22 wt15 wt16 wt18 wt19 wt21 wt22
> reporttab[1:4, ]

<table>
<thead>
<tr>
<th>name</th>
<th>fold</th>
<th>tstat</th>
<th>pvalue</th>
<th>mzmed</th>
<th>mzmin</th>
<th>mzmax</th>
<th>rtmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>M300T3390</td>
<td>5.693594</td>
<td>14.44368</td>
<td>5.026336e-08</td>
<td>300.1898</td>
<td>300.1706</td>
<td>300.2000</td>
<td>3390.313</td>
</tr>
<tr>
<td>M301T3390</td>
<td>5.877538</td>
<td>15.57384</td>
<td>6.696196e-08</td>
<td>301.1879</td>
<td>301.1659</td>
<td>301.1949</td>
<td>3389.665</td>
</tr>
<tr>
<td>M298T3187</td>
<td>3.885650</td>
<td>11.95238</td>
<td>3.275387e-07</td>
<td>298.1508</td>
<td>298.1054</td>
<td>298.1592</td>
<td>3186.795</td>
</tr>
<tr>
<td>M491T3397</td>
<td>12.492266</td>
<td>15.45027</td>
<td>1.911815e-06</td>
<td>491.2000</td>
<td>491.1877</td>
<td>491.2063</td>
<td>3397.203</td>
</tr>
</tbody>
</table>
```
Figure 3: Auto-generated extracted ion chromatograms for the top four differentially regulated ions. Darkened lines indicate where the peaks were integrated for quantitation. The top two plots show the primary and secondary isotopic peaks of an N-acyl ethanolamine (NAE) with a 16 carbon acyl chain. The lower left plot shows the primary isotopic peak of an NAE with a 16 carbon, monounsaturated acyl chain. The lower right plot shows another potential FAAH substrate of unknown identity. Its peaks are not aligned because it is showing a different elution profile than the majority of the other metabolites. Compare it with peaks in the top two plots, which are also eluting at the same time but are correctly aligned.
rtmin  rtmax  npeaks  KO  WT
1  3386.756  3396.321  12  6  6
2  3386.756  3392.108  7  6  1
3  3184.127  3191.432  4  4  0
4  3367.067  3424.663  6  6  0

metlin
1 http://metlin.scripps.edu/metabo_list.php?mass_min=299.04&mass_max=299.34
2 http://metlin.scripps.edu/metabo_list.php?mass_min=300.04&mass_max=300.34
3 http://metlin.scripps.edu/metabo_list.php?mass_min=297&mass_max=297.3
4 http://metlin.scripps.edu/metabo_list.php?mass_min=490.05&mass_max=490.35

ko15  ko16  ko18  ko19
1  4534353.62273683  4980914.48421051  5290739.13866664  4564262.89684209
2  962353.429578945  1047934.14136842  1109303.04472222  946943.392842103
3  180780.817277777  203995.603575946  191015.910842105  190626.849523810
4  432037.001363632  332159.07255  386966.75145  334951.452952381

ko21  ko22  wt15  wt16
1  4733236.07999997  3931592.586  349660.88536842  491793.181333331
2  984787.204999993  806171.4729  86240.5964036599  120207.511987543
3  156878.609095697  220288.6218  16221.5440234364  43676.6299979770
4  294816.2356500  373577.607619048  43244.7547637125  13067.8118739993

wt18  wt19  wt21  wt22
1  645526.704947367  634108.848947367  1438254.44559999  1364627.84400000
2  143067.184686717  137215.082260249  218466.085919548  291392.971409092
3  53533.5670506814  76505.5153194862  54696.7080216332  49676.2302397415
4  47130.036980923  49628.0055557253  12069.305324329  7327.44884758512

If the metlin argument is set to a numeric value, the report will include links to the Metlin Metabolite Database ([http://metlin.scripps.edu/](http://metlin.scripps.edu/)) showing potential metabolite identities. A positive value indicates the data was acquired in positive ion mode and the neutral mass is calculated assuming all ions are M+H. A negative value does the opposite. The value itself indicates the uncertainty in mass accuracy. For instance, the first and third metabolites in the report produce the following URLs:

- [http://metlin.scripps.edu/metabo_list.php?mass_min=299.04&mass_max=299.34](http://metlin.scripps.edu/metabo_list.php?mass_min=299.04&mass_max=299.34)
- [http://metlin.scripps.edu/metabo_list.php?mass_min=297&mass_max=297.3](http://metlin.scripps.edu/metabo_list.php?mass_min=297&mass_max=297.3)

## 7 Selecting and Visualizing Peaks

It is also possible to generate extracted ion chromatograms for arbitrary peak groups selected using various criteria. Here we generate EICs for two analytes eluting at different times. They are shown using both unaligned and aligned retention times. The resulting plots are shown in Figure 4.
> gt <- groups(xset3)
> colnames(gt)

[1] "mzmed" "mzmin" "mzmax" "rtmed" "rtmin" "rtmax" "npeaks" "KO"
[9] "WT"

> groupidx1 <- which(gt[, "rtmed"] > 2600 & gt[, "rtmed"] < 2700 &
  + gt[, "npeaks"] == 12)[1]
> groupidx2 <- which(gt[, "rtmed"] > 3600 & gt[, "rtmed"] < 3700 &
  + gt[, "npeaks"] == 12)[1]
> eiccor <- getEIC(xset3, groupidx = c(groupidx1, groupidx2))

ko15 ko16 ko18 ko19 ko21 ko22 wt15 wt16 wt18 wt19 wt21 wt22

> eicraw <- getEIC(xset3, groupidx = c(groupidx1, groupidx2), rt = "raw")

ko15 ko16 ko18 ko21 ko22 wt15 wt16 wt18 wt19 wt21 wt22

> plot(eicraw, xset3, groupidx = 1)
> plot(eicraw, xset3, groupidx = 2)
> plot(eiccor, xset3, groupidx = 1)
> plot(eiccor, xset3, groupidx = 2)

References


Figure 4: Unaligned (top) and aligned (bottom) extracted ion chromatograms from two analytes eluting at 2624 and 3678 seconds. Darkened lines indicate where the peaks were integrated for quantitation. A plot illustrating the retention time correction is shown in Figure 2.