`flagme`: Fragment-level analysis of GCMS-based metabolomics data

Mark Robinson
mrobinson@wehi.edu.au

April 14, 2011

1 Introduction

This document gives a brief introduction to the `flagme` package, which is designed to process, visualize and statistically analyze sets of GCMS samples. The ideas discussed here were originally designed with GCMS-based metabolomics in mind, but indeed some of the methods and visualizations could be useful for LC-MSMS datasets. The *fragment-level analysis* though, takes advantage of the rich fragmentation patterns observed from electron impact ionization.

There are many aspects of data processing for GCMS data. Generally, algorithms are run separately on each sample to detect features, or *peaks* (e.g. AMDIS). Due to retention time shifts from run-to-run, an alignment algorithm is employed to allow the matching of the same feature across multiple samples. Alternatively, if known standards are introduced to the samples, retention *indices* can be computed for each peak and used for alignment. After peaks are matched across all samples, further processing steps are employed to create a matrix of abundances, leading into detecting differences in abundance.

Many of these data processing steps are prone to errors and they often tend to be black boxes. But, with effective exploratory data analysis, many of the pitfalls can be avoided and any problems can be fixed before proceeding to the downstream statistical analysis. The package provides various visualizations to ensure the methods applied are not black boxes.

The `flagme` package gives a complete suite of methods to go through all common stages of data processing. In addition, R is especially well suited to the downstream data analysis tasks since it is very rich in analysis tools and has excellent visualization capabilities. In addition, it is freely available (`www.r-project.org`), extensible and there is a growing community of users and developers. For routine analyses, graphical user interfaces could be designed.

2 Reading and visualizing GCMS data

To run these examples, you must have the `gcspikelite` package installed. This data package contains several GCMS samples from a spike-in experiment we designed to interrogate data processing methods. So, first, we load the packages:

To load the data and corresponding peak detection results, we simply create vectors of the filenames and create a `peakDataset` object. Note that we can speed up the import time by setting the retention time range to a subset of the elution, as below:

```
> gcmsPath <- paste(.find.package("gcspikelite"), "data", sep = "/")
> data(targets)
> cdfFiles <- paste(gcmsPath, targets$FileName, sep = "/")
> eluFiles <- gsub("CDF", "ELU", cdfFiles)
> pd <- peaksDataset(cdfFiles, mz = seq(50, 550), rtrange = c(7.5,
+     8.5))
```

```
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_468.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_474.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_475.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_485.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_493.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_496.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_470.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_471.CDF
 Reading  /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_479.CDF
```

```
> pd <- addAMDISPeaks(pd, eluFiles)
```

```
Reading retention time range: 7.500133 8.499917
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_468.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_474.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_475.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_485.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_493.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_496.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_470.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_471.ELU ... Done.
Reading /loc/home/biocbuild/bbs-2.8-bioc/R/library/gcspikelite/data/0709_479.ELU ... Done.
```

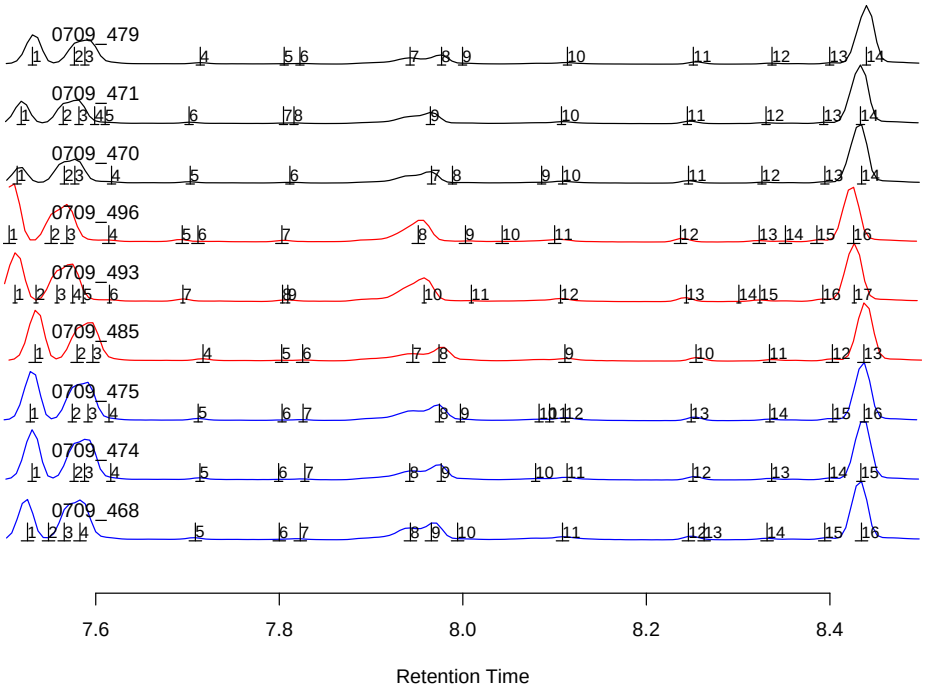
```
> pd
```

```
An object of class "peaksDataset"
9 samples: 0709_468 0709_474 0709_475 0709_485 0709_493 0709_496 0709_470 0709_471 0709_479
501 m/z bins - range: ( 50 550 )
scans: 175 175 175 175 175 174 175 175 175
peaks: 16 15 16 13 17 16 14 14 14
```

Here, we have added peaks from AMDIS, a well known and mature algorithm for deconvolution of GC-MS data. For GC-TOF-MS data, we have implemented a parser for the `ChromaTOF` output (see the analogous `addChromaTOF-Peaks` function). Support for XMCS or MzMine may be added in the future. Ask the author if another detection result format is desired as the parsers are generally easy to design.

Regardless of platform and peak detection algorithm, a useful visualization of a set of samples is the set of total ion currents (TIC), or extracted ion currents (XICs). To view TICs, you can call:

```
> plot(pd, rtrange = c(7.5, 8.5), plotPeaks = TRUE, plotPeakLabels = TRUE,
+     max.near = 8, how.near = 0.5, col = rep(c("blue", "red",
+         "black"), each = 3))
```



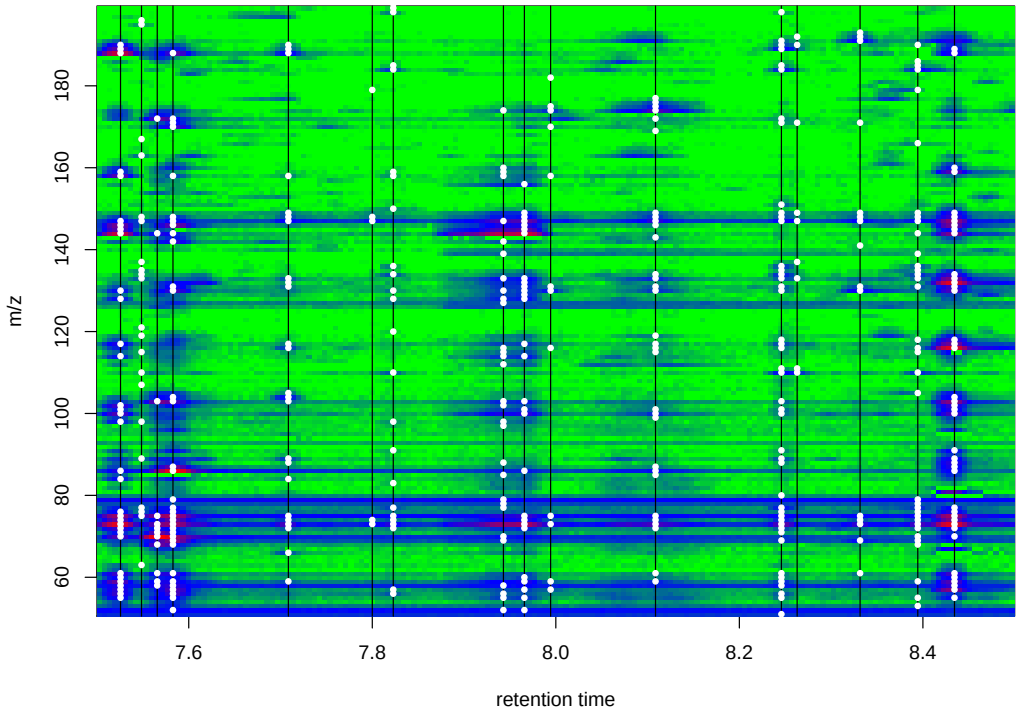
Note here the little *hashes* represent the detected peaks and are labeled with an integer index. One of the main challenges is to match these peak detections across several samples, given that the appear at slightly different times in different runs.

For XICs, you need to give the indices (of `pd@mz`, the grid of mass-to-charge values) that you want to plot through the `mzind` argument. This could be a single ion (e.g. `mzind=24`) or could be a range of indices if multiple ions are of interest (e.g. `mzind=c(24,25,98,99)`).

There are several other features within the `plot` command on `peaksDataset` objects that can be useful. See `?plot` (and select the flagme version) for full details.

Another useful visualization, at least for individual samples, is a 2D heatmap of intensity. Such plots can be enlightening, especially when peak detection results are overlaid. For example (with detected fragment peaks from AMDIS shown in white):

```
> r <- 1
> plotImage(pd, run = r, rtrange = c(7.5, 8.5), main = "")
> v <- which(pd@peaksdata[[r]] > 0, arr.ind = TRUE)
> abline(v = pd@peaksrt[[r]])
> points(pd@peaksrt[[r]][v[, 2]], pd@mz[v[, 1]], pch = 19, cex = 0.6,
+     col = "white")
```



3 Pairwise alignment with dynamic programming algorithm

One of the first challenges of GCMS data is the matching of detected peaks (i.e. metabolites) across several samples. Although gas chromatography is quite robust, there can be some drift in the elution time of the same analyte from run to run. We have devised a strategy, based on dynamic programming, that takes into account both the similarity in spectrum (at the apex of the called peak) and the similarity in retention time, without requiring the identity of each peak; this matching uses the data alone. If each sample gives a 'peak list' of the detected peaks (such as that from AMDIS that we have attached to our `peaksDataset` object), the challenge is to introduce gaps into these lists such that they are best aligned. From this a matrix of retention times or a matrix of peak abundances can be extracted for further statistical analysis, visualization and interpretation. For this matching, we created a procedure analogous to a multiple *sequence* alignment.

To highlight the dynamic programming-based alignment strategy, we first illustrate a pairwise alignment of two peak lists. This example also illustrates the selection of parameters necessary for the alignment. From the data read in previously, let us consider the alignment of two samples, denoted `0709_468` and `0709_474`. First, a similarity matrix for two samples is calculated. This is calculated based on a scoring function and takes into account the similarity in retention time and in the similarity of the apex spectra, according to:

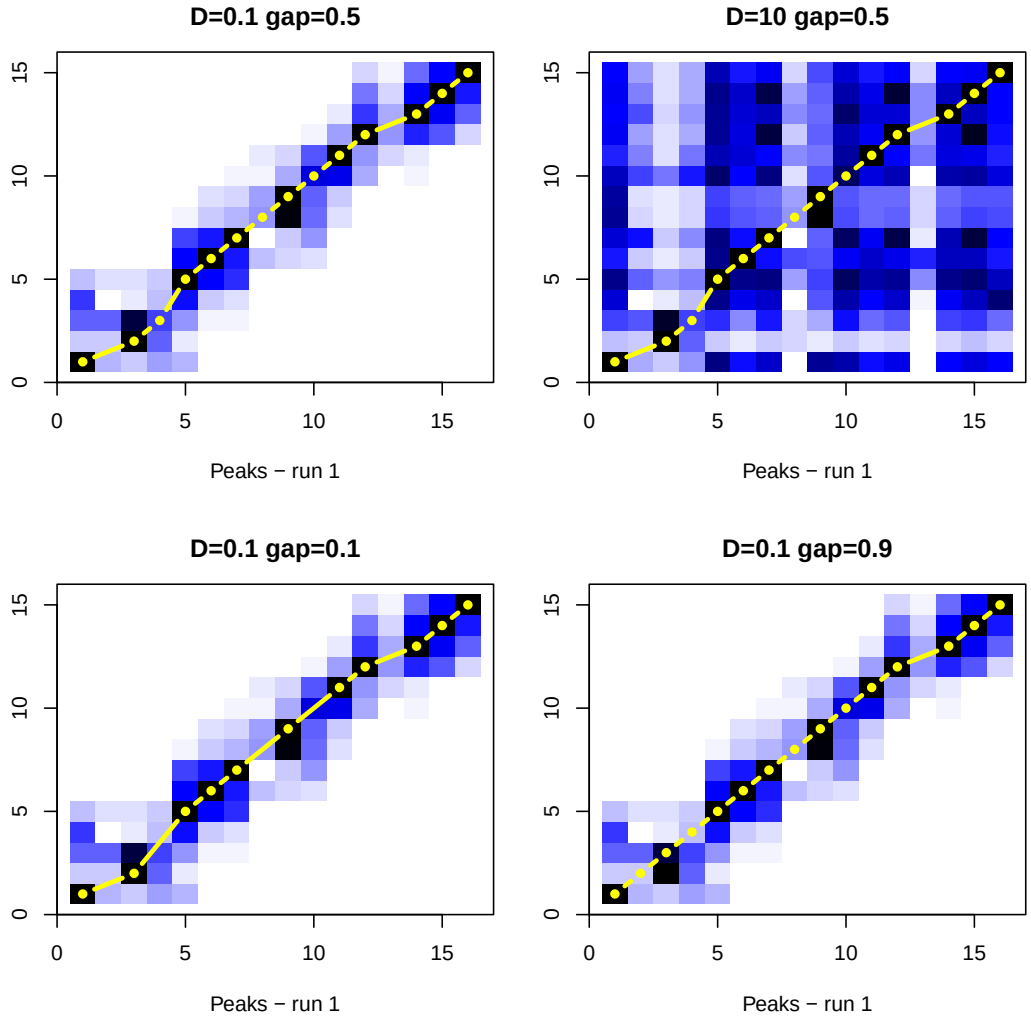
$$S_{ij}(D) = \frac{\sum_{k=1}^{K} x_{ik} y_{jk}}{\sqrt{\sum_{k=1}^{K} x_{ik}^2 \cdot \sum_{k=1}^{K} y_{jk}^2}} \cdot \exp\left(-\frac{1}{2}\frac{(t_i - t_j)^2}{D^2}\right)$$

where i is the index of the peak in the first sample and j is the index of the peak in the second sample, $\mathbf{x}_i$ and $\mathbf{y}_j$ are the spectra vectors and t_i and t_j are their respective retention times. As you can see, there are two components to the similarity: spectra similarity (left term) and similarity in retention time (right term). Of course, other metrics

for spectra similarity are feasible. Ask the author if you want to see other metrices implemented. We have some non-optimized code for a few alternative metrics.

The peak alignment algorithm, much like sequence alignments, requires a `gap` parameter to be set, here a number between 0 and 1. A high gap penalty discourages gaps when matching the two lists of peaks and a low gap penalty allows gaps at a very low *cost*. We find that a gap penalty in the middle range (0.4-0.6) works well for GCMS peak matching. Another parameter, `D`, modulates the impact of the difference in retention time penalty. A large value for `D` essentially eliminates the effect. Generally, we set this parameter to be a bit larger than the average width of a peak, allowing a little flexibility for retention time shifts between samples. Keep in mind the `D` parameter should be set on the scale (i.e. seconds or minutes) of the `peaksrt` slot of the `peaksDataset` object. The next example shows the effect of the `gap` and `D` penalty on the matching of a small ranges of peaks.

```
> Ds <- c(0.1, 10, 0.1, 0.1)
> gaps <- c(0.5, 0.5, 0.1, 0.9)
> par(mfrow = c(2, 2), mai = c(0.8466, 0.4806, 0.4806, 0.1486))
> for (i in 1:4) {
+     pa <- peaksAlignment(pd@peaksdata[[1]], pd@peaksdata[[2]],
+         pd@peaksrt[[1]], pd@peaksrt[[2]], D = Ds[i], gap = gaps[i])
+     plot(pa, xlim = c(0, 17), ylim = c(0, 16), matchCol = "yellow",
+         main = paste("D=", Ds[i], " gap=", gaps[i], sep = ""))
+ }

[peaksAlignment] Comparing 16 peaks to 15 peaks -- gap= 0.5 D= 0.1
[peaksAlignment]  14 matched.  Similarity= 0.1411328
[peaksAlignment] Comparing 16 peaks to 15 peaks -- gap= 0.5 D= 10
[peaksAlignment]  14 matched.  Similarity= 0.1222656
[peaksAlignment] Comparing 16 peaks to 15 peaks -- gap= 0.1 D= 0.1
[peaksAlignment]  11 matched.  Similarity= 0.01060904
[peaksAlignment] Comparing 16 peaks to 15 peaks -- gap= 0.9 D= 0.1
[peaksAlignment]  15 matched.  Similarity= 0.2153892
```



You might ask: is the flagme package useful without peak detection results? Possibly. There have been some developments in alignment (generally on LC-MS proteomics experiments) without peak/feature detection, such as Prince et al. 2006, where a very similar dynamic programming is used for a pairwise alignment. We have experimented with alignments without using the peaks, but do not have any convincing results. It does introduce a new set of challenges in terms of highlighting differentially abundant metabolites. However, in the `peaksAlignment` routine above (and those mentioned below), you can set `usePeaks=FALSE` in order to do *scan*-based alignments instead of *peak*-based alignments. In addition, the `flagme` package may be useful simply for its bare-bones dynamic programming algorithm.

3.1 Normalizing retention time score to drift estimates

In what is mentioned above for pairwise alignments, we are penalizing for differences in retention times that are non-zero. But, as you can see from the TICs, some differences in retention time are consistent. For example, all of the peaks from sample `0709_485` elute at later times than peaks from sample `0709_496`. We should be able to estimate this drift and normalize the time penalty to that estimate, instead of penalizing to zero. That is, we should replace $t_i - t_j$ with $t_i - t_j - \hat{d}_{ij}$ where $\hat{d}_{ij}$ is the expected drift between peak i of the first sample and peak j of the second sample.

More details coming soon.

3.2 Imputing location of undetected peaks

One goal of the alignment leading into downstream data analyses is the generation of a table of abundances for each metabolite across all samples. As you can see from the TICs above, there are some low intensity peaks that fall below detection in some but not all samples. Our view is that instead of inserting arbitrary low constants (such as 0 or half the detection limit) or imputing the intensities post-hoc or having missing data in the data matrices, it is best to return to the area of the where the peak should be and give some kind of abundance. The alignments themselves are rich in information with respect to the location of undetected peaks. We feel this is a more conservative and statistically valid approach than introducing arbitrary values.

More details coming soon.

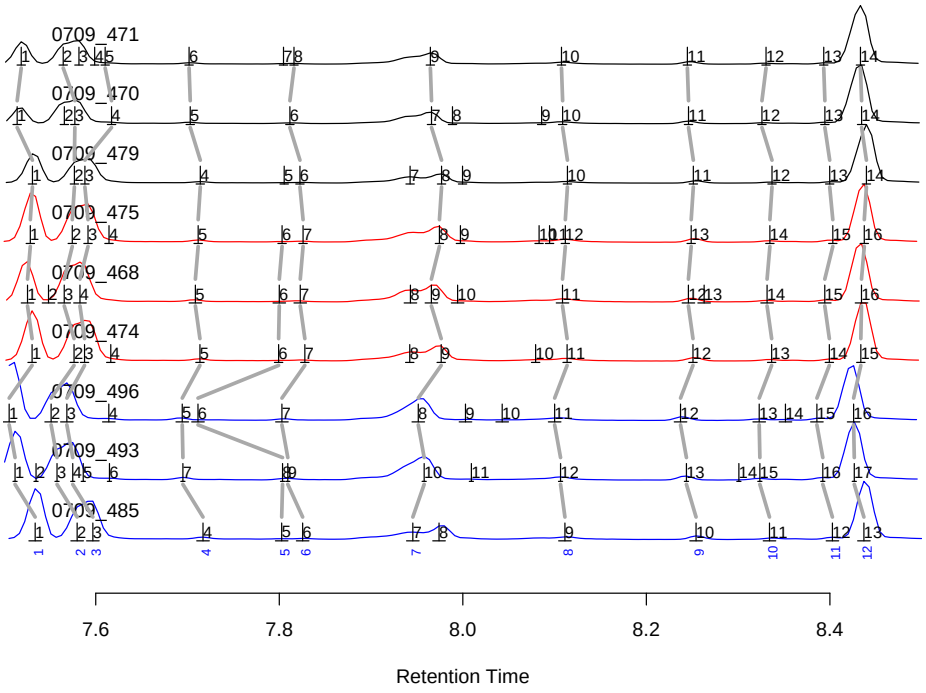
4 Multiple alignment of several experimental groups

Next, we discuss the multiple alignment of peaks across many samples. With replicates, we typically do an alignment within replicates, then combine these together into a summarized form. This cuts down on the computational cost. For example, consider 2 sets of samples, each with 5 replicates. Aligning first within replicates requires 10+10+1 total alignments whereas an all-pairwise alignment requires 45 pairwise alignments. In addition, this allows some flexibility in setting different gap and distance penalty parameters for the *within* alignment and *between* alignment. An example follows.

```
> print(targets)

      FileName Group
1 0709_468.CDF   mmA
2 0709_474.CDF   mmA
3 0709_475.CDF   mmA
4 0709_485.CDF   mmC
5 0709_493.CDF   mmC
6 0709_496.CDF   mmC
7 0709_470.CDF   mmD
8 0709_471.CDF   mmD
9 0709_479.CDF   mmD

> ma <- multipleAlignment(pd, group = targets$Group, wn.gap = 0.5,
+     wn.D = 0.05, bw.gap = 0.6, bw.D = 0.05, usePeaks = TRUE,
+     filterMin = 2, df = 50, verbose = FALSE)
> ma

An object of class "multipleAlignment"
3 groups: 3 3 3 samples, respectively.
12 merged peaks
```

If you set `verbose=TRUE`, many nitty-gritty details of the alignment procedure are given. Next, we can take the alignment results and overlay it onto the TICs, allowing a visual inspection.

```
> plot(pd, rtrange = c(7.5, 8.5), runs = ma@betweenAlignment@runs,
+     mind = ma@betweenAlignment@ind, plotPeaks = TRUE, plotPeakLabels = TRUE,
+     max.near = 8, how.near = 0.5, col = rep(c("blue", "red",
+         "black"), each = 3))
```



4.1 Gathering results

The alignment results can be extracted from the `multipleAlignment` object as:

```
> ma@betweenAlignment@runs
```

```
[1] 4 5 6 2 1 3 9 7 8
```

```
> ma@betweenAlignment@ind
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    1    1    1    1    1    1    1    1    1
 [2,]    2    3    2    2    3    2    2    3    2
 [3,]    3    4    3    3    4    3    3    4    5
 [4,]    4    7    5    5    5    5    4    5    6
 [5,]    5    8    6    6    6    6   NA   NA   NA
 [6,]    6    9    7    7    7    7    6    6    8
 [7,]    7   10    8    9    9    8    8    7    9
 [8,]    9   12   11   11   11   12   10   10   10
 [9,]   10   13   12   12   12   13   11   11   11
[10,]   11   15   13   13   14   14   12   12   12
[11,]   12   16   15   14   15   15   13   13   13
[12,]   13   17   16   15   16   16   14   14   14
```

This table would suggest that matched peak `8` (see numbers below the TICs in the figure above) corresponds to detected peaks `9, 12, 11` in runs `4, 5, 6` and so on, same as shown in the above plot.

In addition, you can gather a list of all the merged peaks with the `gatherInfo` function, giving elements for the retention times, the detected fragment ions and their intensities. The example below also shows the how to construct a table of retention times of the matched peaks (No attempt is made here to adjust retention times onto a common scale. Instead, the peaks are matched to each other on their original scale). For example:

```
> outList <- gatherInfo(pd, ma)
> outList[[8]]

$rt
   mmC.4    mmC.5    mmC.6    mmA.2    mmA.1    mmA.3    mmD.9    mmD.7
8.111250 8.106500 8.100067 8.113633 8.108633 8.111700 8.114000 8.108767
   mmD.8
8.107517

$mz
 [1]  59  73  74  75  86 100 130 131 133 147 148 149 174 175 176 217

$data
      mmC.4 mmC.5 mmC.6 mmA.2 mmA.1 mmA.3 mmD.9 mmD.7 mmD.8
 [1,]  9695 11732  8343  8055  9672 11005  8492 10511 10279
 [2,] 59448 76824 59128 60344 73472 72456 52960 64856 72640
 [3,]  5763  7979  6059  5474  7225  6718  5583  6344  6927
 [4,] 26488 30992 27696 25840 25560 27632 25464 28072 27472
 [5,] 15562 19968 12768 15983 17360 18144 13951 17176 16840
 [6,]  6212  9168  5787  6973  7661  9378  6121  8683  8293
 [7,]  6332  7733  5404  5956  6285  7722  5786  7015  7299
 [8,]  3243  4340  2942  2947  4120  3978  2910  4024  3591
 [9,]  7725 10539  6590  7648  8690  9048  7934  9404  9656
[10,] 37552 52016 44608 42008 43432 47072 42024 43920 47144
[11,]  6518  7554  7426  6689  6705  7355  6527  6457  7068
[12,]  4075  5973  4262  3589  4598  4636  4068  4513  4331
[13,] 75280 98752 51304 73520 78928 92696 68552 84504 83752
[14,] 13887 18088 10155 13086 13590 16592 13046 16062 16186
[15,]  5510  8212  3517  5894  6156  8166  5496  7460  6939
[16,]  6108  9586  9836  9327  8646  9482  9148  8702  9571

> rtmat <- matrix(unlist(lapply(outList, .subset, "rt"), use.names = FALSE),
+     nr = length(outList), byrow = TRUE)
> colnames(rtmat) <- names(outList[[1]]$rt)
> rownames(rtmat) <- 1:nrow(rtmat)
> round(rtmat, 3)

   mmC.4 mmC.5 mmC.6 mmA.2 mmA.1 mmA.3 mmD.9 mmD.7 mmD.8
1  7.534 7.512 7.506 7.531 7.526 7.529 7.531 7.514 7.519
2  7.580 7.558 7.551 7.576 7.566 7.574 7.577 7.577 7.565
3  7.597 7.575 7.569 7.588 7.583 7.592 7.588 7.617 7.610
4  7.717 7.695 7.694 7.714 7.709 7.712 7.714 7.703 7.702
5  7.803 7.804 7.711 7.799 7.800 7.803    NA    NA    NA
6  7.825 7.809 7.803 7.828 7.823 7.826 7.823 7.812 7.816
7  7.946 7.958 7.951 7.976 7.966 7.975 7.977 7.966 7.965
8  8.111 8.107 8.100 8.114 8.109 8.112 8.114 8.109 8.108
9  8.254 8.244 8.237 8.251 8.246 8.249 8.251 8.246 8.245
```

```
10 8.334 8.324 8.323 8.337 8.332 8.335 8.337 8.326 8.330
11 8.403 8.392 8.386 8.399 8.394 8.403 8.400 8.395 8.393
12 8.437 8.427 8.426 8.434 8.434 8.437 8.440 8.435 8.433
```

5 Future improvements and extension

There are many procedures that we have implemented in our investigation of GCMS data, but have not made part of the package just yet. Some of the most useful procedures will be released, such as:

1. Parsers for other peak detection algorithms (e.g. XCMS, MzMine) and parsers for other alignment procedures (e.g. SpectConnect) and perhaps retention indices procedures.
2. More convenient access to the alignment information and abundance table.
3. Statistical analysis of differential metabolite abundance.
4. Fragment-level analysis, an alternative method to summarize abundance across all detected fragments of a metabolite peak.

6 References

See the following for further details:

1. Robinson MD. *Methods for the analysis of gas chromatography - mass spectrometry data.* **Ph.D. Thesis**. October 2008. Department of Medical Biology (Walter and Eliza Hall Institute of Medical Research), University of Melbourne.
2. Robinson MD, De Souza DP, Keen WW, Saunders EC, McConville MJ, Speed TP, Likić VA. (2007) *A dynamic programming approach for the alignment of signal peaks in multiple gas chromatography-mass spectrometry experiments.* **BMC Bioinformatics**. 8:419.
3. Prince JT, Marcotte EM (2006) *Chromatographic alignment of ESI-LC-MS proteomics data sets by ordered bijective interpolated warping.* **Anal Chem**. 78(17):6140-52.

7 This vignette was built with/at ...

```
> sessionInfo()

R version 2.13.0 (2011-04-13)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [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] stats     graphics  grDevices utils     datasets  methods   base
```

```
other attached packages:
[1] flagme_1.8.0      gcspikelite_1.0.4

loaded via a namespace (and not attached):
[1] MASS_7.3-12  SparseM_0.88 gdata_2.8.1  gplots_2.8.0 gtools_2.6.2
[6] tools_2.13.0 xcms_1.26.0

> date()

[1] "Thu Apr 14 00:10:20 2011"
```