Analysis of GC-MS metabolomics data with metaMS

Ron Wehrens

May 3, 2016

1 Introduction

Many packages are available for the analysis of data from GC-MS and LC-MS experiments – typically, hardware vendors provide software that is optimized for the instrument and allow for a direct interaction of the lab scientist with the data. For high-throughput applications, however, more automatic pipelines are needed. Open-source alternatives such as **xcms** [1] not only are able to handle data from several different types of spectrometers, but can also be integrated easily in web interfaces [2], allowing large numbers of files to be processed simultaneously.

Because of the generality of packages like **xcms**, several other packages have been written to tweak the functionality of **xcms** for optimal performance in a particular context. Package **metaMS** does so for the field of untargeted metabolomics; this vignette focuses on the analysis of GC-MS data. In comparison with the usual **xcms** pipeline several changes have been implemented, the most important of which are:

- the package collects all user-changeable settings in one list, with elements for the individual stages of the data processing pipeline. This improves consistency and maintainability;
- rather than a feature-based analysis, as is the case with **xcms**, **metaMS** performs a pseudospectrum-based analysis, where the basic entity is a collection of (mz, I) pairs at specific retention times. This avoids the alignment step, which can be extremely difficult for GC-MS data;
- support has been added for the creation of in-house databases of standards; these in-house databases can then be used for annotation purposes;
- comparison with databases of standard spectra is made on the bases of the pseudospectra, using a tried and tested similarity function that is fast enough to also search databases of hundreds of thousands of compounds.

One of the goals of setting up **metaMS** was to set up a simple system with few user-settable parameters, capable of handling the vast majority of untargeted metabolomics experiments. Users not proficient in R can be accomodated by setting up web interfaces, e.g. using RWui [3] – in our institute, the only pieces of information the users have to provide in such a web interface are the location of the data, and the protocol used (e.g. GC triple-quad). This will automatically link the appropriate database for annotation and use the optimized settings. Results are returned both in the form of spreadsheets and R objects.

2 Example data

Because experimental data are quite big in general, this part has been split off in a separate package called **metaMSdata**; since the data are unlikely to see many updates, the user of **metaMS** can download future versions without having to download the example data again. Package **metaMSdata** provides a small number of example data files that illustrate the functionality of **metaMS**. For the GC-MS part, they consist of four injections of mixtures of chemical standards [4]. One of these injections will be used to create a database for three of the compounds present: Linalool, Methyl salicylate and Ethyl hexanoate.

Once the database of standards has been created, it can be used for annotation purposes. Intermediate results are typically orders of magnitude smaller than the raw data files, and **metaMS** itself contains a number of these to illustrate the concepts and to speed up the examples in the manual pages.

3 GC-MS data processing in metaMS

In untargeted metabolomics, the application of MS/MS and related techniques is usually restricted to follow-up experiments, where identification of potentially interesting features is the goal. In initial experiments most often simple LC-MS and GC-MS experiments are done, where annotation is based on reliable data of chemical standards, often measured in very similar conditions. Package **metaMS** supports building these databases, using exactly the same data processing pipeline as is used for real samples. The settings are gathered in an object of class `metaMSsettings`:

```
> library(metaMS)
> data(FEMsettings)
> TSQXLS.GC

Object of class 'metaMSsettings'
Instrument: TSQXLS.QQQ.GC
Chromatography: GC

> metaSetting(TSQXLS.GC, "PeakPicking")

$method
[1] "matchedFilter"

$step
[1] 0.5

$steps
[1] 2

$mzdiff
[1] 0.5

$fwhm
[1] 5

$snthresh
[1] 2

$max
[1] 500
```

The settings used for this particular set of samples are fine-tuned for a Thermo Scientific TSQXLS triple-quad GC. The `PeakPicking` field in the settings contains all elements that are passed to the `xcmsSet` function from **xcms**.

3.1 Analysis of samples

The standard workflow of **metaMS** for GC-MS data is the following:

1. peak picking;
2. definition of pseudospectra;

3. identification and elimination of artefacts;
4. annotation by comparison to a database of standards;
5. definition of unknowns;
6. output.

This has been implemeted in function `runGC`, which takes a vector of file names, corresponding to the samples, and a settings list as mandatory arguments. In addition, some extra arguments can be provided. In particular, a database of standards, as discussed in the previous section, can be provided for annotation purposes. The call therefore can be as simple as:

```
> library(metaMSdata)
> data(threeStdsDB)        ## provides DB
> cdfdir <- system.file("extdata", package = "metaMSdata")
> cdffiles <- list.files(cdfdir, pattern = "_GC_",
+                        full.names = TRUE, ignore.case = TRUE)
> result <- runGC(files = cdffiles, settings = TSQXLS.GC, DB = DB,
+                 nSlaves = 2)
```

Alternatively, if the peak picking by **xcms** is already done, one can provide the `xcmsSet` object:

```
> result <- runGC(xset = GCset, settings = TSQXLS.GC, DB = DB)
```

In both cases, the result is a list containing a set of patterns corresponding with the compounds that have been found, either annotated or unknown, the relative intensities of these patterns in the individual annotations, and possibly the `xcmsSet` object for further inspection. In practice, the `runGC` function is all that users need to use. However, to give more details, each of the steps in the workflow will be discussed briefly below.

All results and intermediate results from this vignette are available in data object `GCresults` – this is used here to demonstrate the structure of the data objects without having to create them on the fly, which simply takes too much time:

```
> data("GCresults")
```

3.1.1 Peak picking

The peak picking is performed by the usual **xcms** functions. A wrapper function, `peakDetection`, has been written in **metaMS** to allow the individual parameters to be passed to the function as a settings list. The result is that the whole of the **xcms** functionality is available, simply by changing the values of some settings, or by adding fields. In the `runGC` function, this step is performed by

```
> GCset <- peakDetection(cdffiles,
+                        settings = metaSetting(TSQXLS.GC, "PeakPicking"),
+                        convert2list = TRUE, nSlaves = 2)
```

Since this part can take quite some time, it is operated in parallel wherever possible (using **Rmpi** or **snow**, and shows some feedback to the user. The last argument of the function determines whether the results are to be presented for all samples together (the format of **xcms**), or should be split into a list with one entry for each individual file. The latter case is useful here, in the analysis of GC data, but also when setting up a database of standards.

3.1.2 Definition of pseudospectra

Rather than individual peaks, the basic data structure in the GC-MS part of **metaMS** is a pseudospectrum, i.e. a set of m/z values showing a chromatographic peak at the same retention time. This choice is motivated by several considerations. First of all, in GC the amount of overlap is much less than in LC: peaks are much narrower. This means that even a one- or two-second difference

in retention time can be enough to separate the corresponding mass spectra. Secondly, fragmentation patterns for many compounds are available in extensive libraries like the NIST library (see `http://www.nist.gov/srd/nist1a.cfm`). In addition, the spectra are somewhat easier to interpret since adducts, such as found in LC, are not present. The main advantage of pseudospectra, however, is that their use allows the results to be interpreted directly as relative concentrations of chemical compounds: a fingerprint in terms of chemical composition is obtained, rather than a fingerprint in terms of hard-to-interpret features.

The pseudospectra are obtained by simply clustering on retention time, using the `runCAMERA` wrapper function, which for GC data calls `groupFWHM`:

```
> allSamples <- lapply(GCset, runCAMERA, chrom = "GC",
+                      settings = metaSetting(TSQXLS.GC, "CAMERA"))
```

Again, all the usual parameters for the `groupFWHM` function can be included in the `CAMERA` slot of the settings object. The most important parameter is `perfwhm`, which determines the maximal retention time difference of features in one pseudospectrum.

The final step is to convert the CAMERA objects into easily handled lists, which are basically the R equivalent of the often-used `msp` format from the AMDIS software [5]. In `runGC`, this step is implemented as:

```
> allSamples.msp <- lapply(allSamples, to.msp, file = NULL,
+                          settings = metaSetting(TSQXLS.GC, "DBconstruction"))

> sapply(allSamples.msp, length)

STDmix_GC_01 STDmix_GC_02 STDmix_GC_03 STDmix_GC_04
         241          237          232          235

> allSamples.msp[[1]][[26]]

       mz    maxo       rt
 [1,]  38 2145159 38.18892
 [2,] 101 1338453 38.19257
 [3,] 144  352465 38.19257
 [4,] 170  341177 38.19620
 [5,] 188  182418 38.19257
 [6,] 205  139864 38.19257
 [7,] 219  278031 38.18525
 [8,] 223  105532 38.18892
 [9,] 226  161979 38.18525
[10,] 233   93438 38.18525
[11,] 245   82342 38.18162
[12,] 259   81633 38.18162
[13,] 260   67324 38.19620
[14,] 262  145054 38.19620
[15,] 266   77631 38.18162
[16,] 268   56122 38.18162
[17,] 337   44739 38.18892
[18,] 339   30957 38.18525
```

Object `allsamples.msp` is a nested list, with one entry for each sample, and each sample represented by a number of fields. In this case, more than 300 pseudospectra are found in each sample (even though the samples are mixtures of only fourteen chemical standards). The pseudospectra are three-column matrices, containing m/z, intensity and retention time information, respectively. One can plot the individual spectra for visual inspection using the function `plotPseudoSpectrum` – an example is shown in Figure 1.

```
> plotPseudoSpectrum(allSamples.msp[[1]][[26]])
```

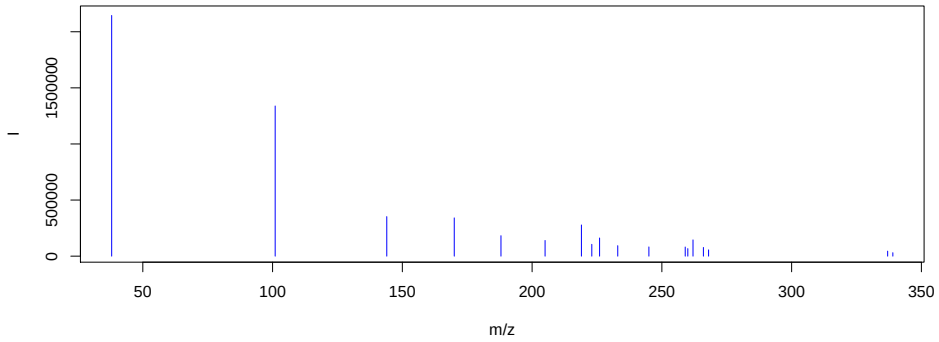

Figure 1: A pseudospectrum from one of the samples.

3.1.3 Annotation

Once we have identified our pseudospectra, we can start the annotation process. This is done by comparing every pseudospectrum to a database of spectra. As a similarity measure, we use the weighted dot product as it is fast, simple, and gives good results [6]. The first step in the comparison is based on retention, since a comparison of either retention time or retention index is much faster than a spectral comparison. The corresponding function is `matchSamples2DB`. Since the weighted dot product uses scaled mass spectra, the scaling of the database is done once, and then used in all comparisons:

```
> DB.treated <- treat.DB(DB)
> allSam.matches <-
+     matchSamples2DB(allSamples.msp, DB = DB.treated,
+                     settings = metaSetting(TSQXLS.GC, "match2DB"),
+                     quick = FALSE)
> allSam.matches

$annotations
$annotations$STDmix_GC_01
  pattern annotation alternatives
1      21          2
2      54          3
3     104          1

$annotations$STDmix_GC_02
  pattern annotation alternatives
1      21          2
2      43          3
3     119          1

$annotations$STDmix_GC_03
  pattern annotation alternatives
1      12          2
2      15          3
3      47          1
```

```
> matchExpSpec(allSamples.msp[[1]][[4]], DB.treated,
+              DB.treated = TRUE, plotIt = TRUE)

[1] 0.009241914 0.000000000 0.000000000
```

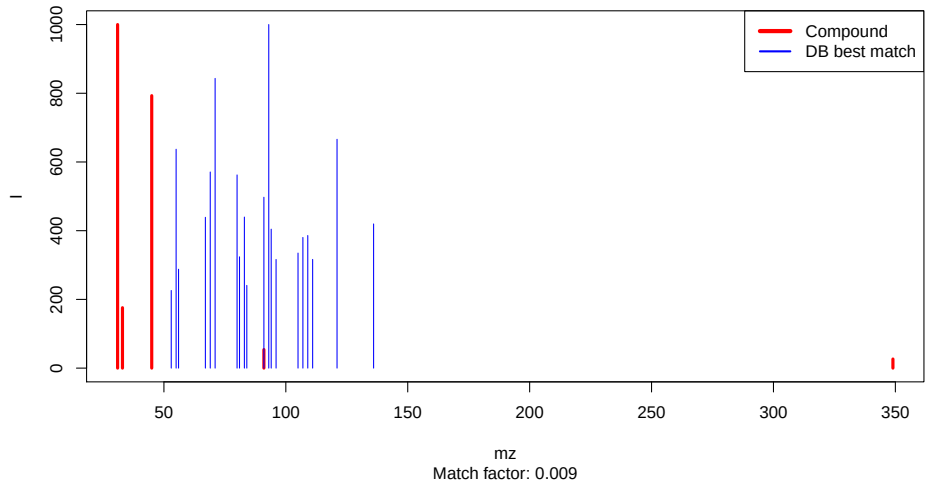

Figure 2: Best match between an experimental pattern (in red) and a database entry (in blue).

```
$annotations$STDmix_GC_04
  pattern annotation alternatives
1      10          2
2      17          3
3      53          1
```

This returns a table where all patterns that have a match with a DB entry are shown in the first column, and the DB entry itself in the second column. If for a particular experimental pattern more than one match is found, the alternatives (with a lower match factor) are shown in the last column. In this case, all three patterns in the database match with exactly one pattern in each of the experimental samples.

To see the match for a particular pattern, one can use the function `matchExpSpec`, returning match factors (numbers between 0 and 1, where the latter means a perfect match) for all entries in the database. If the `plotIt` argument is `TRUE`, the best match is shown – see Figure 2.

Samples may contain compounds that are not of any interest, such as plasticizers, internal standards, column material etcetera. These can be filtered out before doing an annotation: **metaMS** allows certain categories of database entries (defined in slot `matchIrrelevants` of the settings object) to be removed before further annotation. If the spectra of these compounds are very specific (and they often are), the retention criterion may be bypassed by setting the maximal retention time difference to very high values, which leads to the removal of such spectra wherever they occur in the chromatogram.

3.1.4 Unknowns

The most important aspect of untargeted metabolomics is the definition of unknowns, patterns that occur repeatedly in several samples, but for which no annotation has been found. In **metaMS** these unknowns are found by comparing all patterns within a certain retention time (or retention index) difference on their spectral characteristics. The same match function is used, but the threshold may be different from the threshold used to match with the database of standards. Likewise, the maximum retention time (index) difference may be different, too. In defining unknowns we have so far used settings that are more

strict than when comparing to a database: since all samples are typically measured in one single run, expected retention time differences are rather small. In addition, one would expect reproducible spectra for a single compound. A true unknown, or at least an interesting one, is also present in a significant fraction of the samples. All these parameters are gathered in the `betweenSamples` element of the settings object.

Since the matching is done using scaled patterns, we need to created a scaled version of the experimental pseudospectra first:

```
> allSamples.msp.scaled <- lapply(allSamples.msp, treat.DB,
+                                 isMSP = FALSE)
> allSam.matches <-
+     matchSamples2Samples(allSamples.msp.scaled,
+                          allSamples.msp,
+                          annotations = allSam.matches$annotations,
+                          settings = metaSetting(TSQXLS.GC, "betweenSamples"))
> names(allSam.matches)

[1] "annotations" "unknowns"
```

For large numbers of samples, this process can take quite some time (it scales quadratically), especially if the allowed difference in retention time is large. The result now is a list of two elements: the first is the annotation table that we also saw after the comparison with the database, and the second is a list of pseudospectra corresponding to unknowns. In the annotation table, negative indices correspond to the pseudospectra in this list.

```
> allSam.matches$annotations[[1]]

   pattern annotation alternatives
1       21          2
2       54          3
3      104          1
4        1         -1
5        5         -2
6       14         -3
7       19         -4
8       67         -5
9       68         -6
10      82         -7
11      84         -8
12     128         -9
```

In the example above we see that pattern 10 in the first sample corresponds to the first unknown.

3.1.5 Output

At this stage, all elements are complete: we have the list of pseudospectra with an annotation, either as a chemical standard from the database, or an unknown occurring in a sizeable fraction of the injections. The only things left to do is to calculate relative intensities for the pseudospectra, and to put the results in an easy-to-use table. This table consists of two parts. The first part is the information on the "features", which here are the pseudospectra:

```
> features.df <- getFeatureInfo(stdDB = DB, allMatches = allSam.matches,
+                               sampleList = allSamples.msp)
> features.df[, c(1:3, ncol(features.df) - 2:0)]
```

```
                Name    CAS RTman std.rt  rt.sd     rt
1           Linalool  78706 17.86 17.860 0.0012 17.859
2  Methyl salicylate 119368 22.44 22.439 0.0016 22.439
3    Ethyl hexanoate 123660 10.77 10.770 0.0007 10.770
4          Unknown 1     NA    NA     NA 0.0092  3.792
5          Unknown 2     NA    NA     NA 0.0014  4.915
6          Unknown 3     NA    NA     NA 0.0021 30.559
7          Unknown 4     NA    NA     NA 0.0017 19.669
8          Unknown 5     NA    NA     NA 0.0012 19.038
9          Unknown 6     NA    NA     NA 0.0016 13.013
10         Unknown 7     NA    NA     NA 0.0025  9.754
11         Unknown 8     NA    NA     NA 0.0018 22.087
12         Unknown 9     NA    NA     NA 0.0018  9.915
13        Unknown 10     NA    NA     NA 0.0023 19.039
14        Unknown 11     NA    NA     NA 0.0084  3.786
15        Unknown 12     NA    NA     NA 0.0224  3.444
16        Unknown 13     NA    NA     NA 0.0021 19.667
17        Unknown 14     NA    NA     NA 0.0000  9.753
18        Unknown 15     NA    NA     NA 0.0045 20.194
```

The first three lines are the standards, and the next two are the two unknowns that are identified by the pipeline. The second part of the table contains the intensities of these features in the individual injections.

In manual interpretation of this kind of data, the intensities of one or two "highly specific" features are often used to achieve relative quantitation. In an automatic pipeline, this is a risky strategy: not only can the intensity of a peak vary quite dramatically (relative standard deviations of up to 30% are assumed acceptable in GC-MS, e.g. when SPME is applied), but these errors are all the more pronounced in high-intensity peaks (hence the common use of a relative standard deviation). In addition, one is ignoring the information in the other peaks of the pseudospectrum. In **metaMS**, pseudospectrum intensity is expressed as a multiple of the corresponding reference pattern (either a database pattern or an unknown), where the intensity ratio is determined using robust regression to avoid one deviating feature to influence the results too much [4]. First, we define an object containing all relevant pseudospectra, and next the intensities are generated:

```
> PseudoSpectra <- constructExpPseudoSpectra(allMatches = allSam.matches,
+                                            standardsDB = DB)
> ann.df <- getAnnotationMat(exp.msp = allSamples.msp, pspectra = PseudoSpectra,
+                            allMatches = allSam.matches)
> ann.df
```

```
      STDmix_GC_01 STDmix_GC_02 STDmix_GC_03 STDmix_GC_04
 [1,]    0.5393561    0.5106029     1.000000    0.9740733
 [2,]    0.5139159    0.5143993     1.000000    1.0048997
 [3,]    0.5506663    0.4716969     1.000000    0.9449879
 [4,]    1.0000000    0.9779254     0.000000    0.0000000
 [5,]    1.0000000    1.0678439     1.012443    1.0116312
 [6,]    1.0000000    0.8567742     1.779322    1.9436918
 [7,]    1.0000000    1.0727171     0.000000    0.0000000
 [8,]    1.0000000    0.0000000     0.000000    1.9793002
 [9,]    1.0000000    0.0000000     1.886385    0.0000000
[10,]    1.0000000    0.9750801     0.000000    0.0000000
[11,]    1.0000000    0.0000000     1.745452    0.0000000
[12,]    0.5967444    0.5416304     1.002923    1.0000000
[13,]    0.0000000    1.0000000     1.786243    0.0000000
```

```
[14,]    0.0000000    0.0000000     1.000000    0.9995801
[15,]    0.0000000    0.0000000     1.000000    1.0017294
[16,]    0.0000000    0.0000000     1.000000    1.0221790
[17,]    0.0000000    0.0000000     1.000000    1.0708144
[18,]    0.0000000    0.0000000     1.000000    1.1444677
```

Since relative intensities are hard to interpret for mass spectrometrists, these are converted into numbers corresponding to peak heights or peak areas. This is done by multiplication by the highest intensity in the reference spectrum:

```
> ann.df2 <- sweep(ann.df, 1, sapply(PseudoSpectra,
+                                    function(x) max(x$pspectrum[, 2])),
+                  FUN = "*")
> ann.df2
```

```
      STDmix_GC_01 STDmix_GC_02 STDmix_GC_03 STDmix_GC_04
 [1,]     727018.1     688260.6      1347937    1312989.4
 [2,]    2252467.1    2254586.1      4382949    4404424.0
 [3,]    1354776.8    1160492.4      2460250    2324906.6
 [4,]  715872128.0  700069546.2            0          0.0
 [5,]  361640992.0  386176144.4    366140725  365847309.5
 [6,]    3386176.0    2901188.3      6025097    6581682.4
 [7,]    2269447.0    2434474.5            0          0.0
 [8,]    1184045.0          0.0            0    2343580.6
 [9,]    1176973.0          0.0      2220224          0.0
[10,]     897047.0     874692.7            0          0.0
[11,]     855648.0          0.0      1493492          0.0
[12,]     642690.2     583332.7      1080142    1076994.0
[13,]          0.0    1105883.0      1975375          0.0
[14,]          0.0          0.0    711693760  711394911.2
[15,]          0.0          0.0    698653312  699861583.3
[16,]          0.0          0.0      4265898    4360511.1
[17,]          0.0          0.0      1657900    1775303.2
[18,]          0.0          0.0       244832     280202.3
```

The three standards (the first three lines) have been identified in all four samples; the two unknowns in three and two cases, respectively. The final result is obtained by simply concatenating the two tables column-wise.

3.2 Building a database of standards

From the previous discussion it should be clear how important an in-house database of standards, measured on the same system as the samples, really is. The point is that the retention behaviour of chemical compounds can differ substantially across systems – an in-house database is the best way to avoid many false negative and false positive hits.

Fortunately, the pipeline as discussed so far can be used easily to also process injections of standards. These may be injected as mixtures, provided that their retention times are not completely overlapping. Additional information is obtained from a user-provided `csv` file containing information like CAS number, retention time, and molecular weight. Such a file is processed with the function `readStdInfo`. For the three chemical standards in the **metaMSdata** package, this leads to the following information:

```
> library(metaMSdata)
> stddir <- system.file("extdata", package = "metaMSdata")
> input.file <- list.files(stddir, pattern = "csv", full.names = TRUE)
```

```
> threeStdsInfo <- readStdInfo(input.file, stddir, sep = ";", dec = ",")
> threeStdsInfo[,"stdFile"] <- file.path(stddir, "STDmix_GC_03.CDF")
> threeStdsInfo[,c(1:4, 8)]
```

```
     CAS              Name RTman ChemspiderID   monoMW
1  78706          Linalool 17.86     13849981 154.1358
2 119368 Methyl salicylate 22.44     13848808 152.0473
3 123660   Ethyl hexanoate 10.77        29005 144.1150
```

The system gives a warning that some `CDF` files in the data directory are not used, which can be useful feedback – in this case, it is so by design. The result is a `data.frame` object containing all information about the chemical standards: where to look for them, but also it provides identifiers such as a CAS number that can be used to compare the data to other databases. This is an essential step, since many patterns will be identified in the raw data, even when the samples consist of clean mixtures of chemical standards: for automatic DB construction, a validation step of some kind has to be part of the pipeline. The main benefit of setting up one's own database is the exact retention time information, something that will be essential in the annotation phase lateron. Continuing the example for the three standards, the in-house database can simply be created by issuing

```
> data(threeStdsNIST)                          ## provides smallDB
> DB <- createSTDdbGC(threeStdsInfo, TSQXLS.GC, extDB = smallDB,
+                     nSlaves = 2)
```

The system returns some feedback about the process, which for large numbers of files can take some time. If parallel processing is supported (`Rmpi` or `snow`), this is shown, too. If no validation by comparing to an external database of spectra is possible, **metaMS** allows to add manually validated spectra to the data base using the `manualDB` argument to the `createSTDdbGC` function.

```
> names(DB[[1]])
```

```
 [1] "CAS"          "Name"         "RTman"        "ChemspiderID" "SMILES"
 [6] "InChI"        "csLinks"      "monoMW"       "stdFile"      "bestDBmatch"
[11] "pspectrum"    "Class"        "date"         "std.rt"       "std.rt.sd"
```

The database contains all information from the input file, as well as information on the match with the external database, and the validated spectrum.

4 Alternative annotation strategies

Instead of the strategy outlined above, one could also use the building blocks provided by **metaMS** in different ways. Instead of first comparing all pseudospectra in the injections with the database patterns, one could also first identify all unknowns. Then the pseudospectra corresponding to the unknowns could be compared with the database to see if any of them matches known standards. This strategy would get rid of "single hits", i.e. chemical compounds that are identified in only one or very few injections, and would make full use of the fact that all injections have been done in a single run, so would lead to very similar retention time behaviour and very similar spectra. One could hypothesize that over the course of several months or even years, a database of spectra of standards could show bigger and bigger deviations with actual measurements, and this alternative strategy could possibly lead to more easily interpretable, or at least easier-to-correct results.

There are also downsides to this strategy, as can be imagined: first of all, it is slower, especially with large numbers of samples. Since the comparison with a database scales linearly with the number of injections, and the defintion of unknowns quadratically, this can be quite a big issue. In addition, the definition of what is an unknown not only depends on spectral and chromatographic similarities, but also on the fraction of injections in which this pattern is found: an additional parameter that can make quite

a big difference in the outcome. Finally, it is easy to remove annotations from the "standard" strategy that occur in only very few samples: one can even assess whether alternative annotations are present that are more in line with what is found in the other samples.

Whichever strategy is most useful depends on the data and the scientific questions at hand. Given the building blocks provided by **xcms** and **CAMERA**, and on a more abstract level by **metaMS**, it is now quite easy to implement different strategies for different purposes.

References

[1] C. A. Smith, E. J. Want, G. O'Maille, R. Abagyan, and G. Siuzdak. **XCMS**: Processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Anal. Chem.*, 78:779–787, 2006.

[2] R. Tautenhahn, G.J. Patti, D. Rinehart, and G. Siuzdak. XCMS Online: A web-based platform to process untargeted metabolomic data. *Anal. Chem.*, 84(11):5035–5039, 2012.

[3] R. Newton and L. Wernisch. RWui: a web application to create user friendly web interfaces for R scripts. *R News*, 7:32–35, 2007.

[4] R. Wehrens, G. Weingart, and F. Mattivi. An open-source pipeline for GC-MS-based untargeted metabolomics. *J. Chrom. B*, 966:109–116, 2014.

[5] S.E. Stein. An integrated method for spectrum extraction and compound identification from gas chromatography/mass spectrometry data. *J. Am. Soc. Mass Spectrom.*, 10:770–781, 1999.

[6] S.E. Stein and D.R. Scott. Optimization and testing of mass spectral library search algorithms for compound identification. *J. Am. Soc. Mass Spectrom.*, 5:859–866, 1994.