The TargetSearch Package

***Alvaro Cuadros-Inostroza*[1], *Jan Lisec*[1], *Henning Redestig*[1], *and Matthew A Hannah*[1]**

[1]Max Planck Institute for Molecular Plant Physiology, Potsdam, Germany

February 27, 2019

Abstract

This document describes how to use *TargetSearch* to preprocess GC-MS data.

Package

TargetSearch 1.38.2

Report issues on https://github.com/acinostroza/TargetSearch/issues

Contents

1 Supplied files

This section describes the files that have to be prepared before running *TargetSearch*. These comprise a sample definition file, a reference library file and a retention marker definition file. Throughout this manual, we will use the example files provided by the package *TargetSearchData*.

1.1 NetCDF files and sample File

TargetSearch can currently read only NetCDF files. Many GC-MS software packages are able to convert raw chromatograms to NetCDF. Although some baseline correction functionality is included in *TargetSearch*, it is recommended to baseline correct your chromatograms before exporting to NetCDF to increase processing speed. Please refer to your vendor's software documentation for further details.

Export the NetCDF files into a convenient location. Then prepare a tab-delimited text file describing your samples. It must be contain (at least) the two columns entitled: "CDF_FILE" and "MEASUREMENT_DAY", or, column names that match "cdf" and "day", respectively. If more than one column matches the pattern, then the first (leftmost) is taken. Other columns such as sample name, sample group, treatment, etc. may be additionally included to aid sample sub-setting and downstream analyses. An example is shown in table 1.

CDF_FILE	MEASUREMENT_DAY	TIME_POINT
7235eg08.cdf	7235	1
7235eg11.cdf	7235	1
7235eg26.cdf	7235	1
7235eg04.cdf	7235	3
7235eg30.cdf	7235	3
7235eg32.cdf	7235	3
...		

Table 1: Sample file example

To import the sample list into R, use the function `ImportSamples()` specifying the options `CDFpath` (directory containing the NetCDF files) and `RIpath` (directory where the transformed cdf files, containing the retention index corrected apex data, also called RI-files, will be saved). This function will warn you if CDF samples are not found in the chosen path.

```
library(TargetSearchData)
library(TargetSearch)
cdf.path <- system.file("gc-ms-data", package = "TargetSearchData")
sample.file <- file.path(cdf.path, "samples.txt")
samples <- ImportSamples(sample.file, CDFpath = cdf.path, RIpath = ".")
```

Alternatively, you could create a `tsSample` object by using the sample class methods.

```
cdffiles <- dir(cdf.path, pattern="cdf$")
# take the measurement day info from the cdf file names.
days <- sub("^([[:digit:]]+).*$","\\1",cdffiles)
# sample names
```

```
smp_names <- sub("\\.cdf", "", cdffiles)
# add some sample info
smp_data <- data.frame(CDF_FILE =cdffiles, GROUP = gl(5,3))
# create the sample object
samples <- new("tsSample", Names = smp_names, CDFfiles = cdffiles,
               CDFpath = cdf.path, RIpath = ".", days = days,
               data = smp_data)
```

1.2 Retention index markers

To align different chromatograms using RI markers, a tab-delimited definition file for these markers has to be provided. At a minimum it should contain three columns specifying the search window (lower and upper threshod) and the fixed standard value for each marker (table 2). Further, a characteristic ion mass (m/z) has to be provided either in an additional column or as an argument `mass` to function `ImportFameSettings()`.

```
rim.file  <- file.path(cdf.path, "rimLimits.txt")
rimLimits <- ImportFameSettings(rim.file, mass = 87)
```

This will import the limits in '`rimLimits.txt`' file and set the marker mass to 87 for all markers.

LowerLimit	UpperLimit	RIstandard
230	280	262320
290	340	323120
350	400	381020

Table 2: Retention time definition file example

If you do not use RI markers, you can skip this part by setting the parameter `rimLimits` to `NULL` in `RIcorrect()` function. Please note that in this case, no retention time correction will be performed.

To make sure that you have correctly set the time ranges for the RI markers, the function `checkRimLim()` can be used. This function plots the m/z of all the markers in one, randomly chosen, sample. Repeat the function to see other samples. In the figure 1 a example is shown. The gray area indicates the search window of each marker.

```
checkRimLim(samples, rimLimits)
```

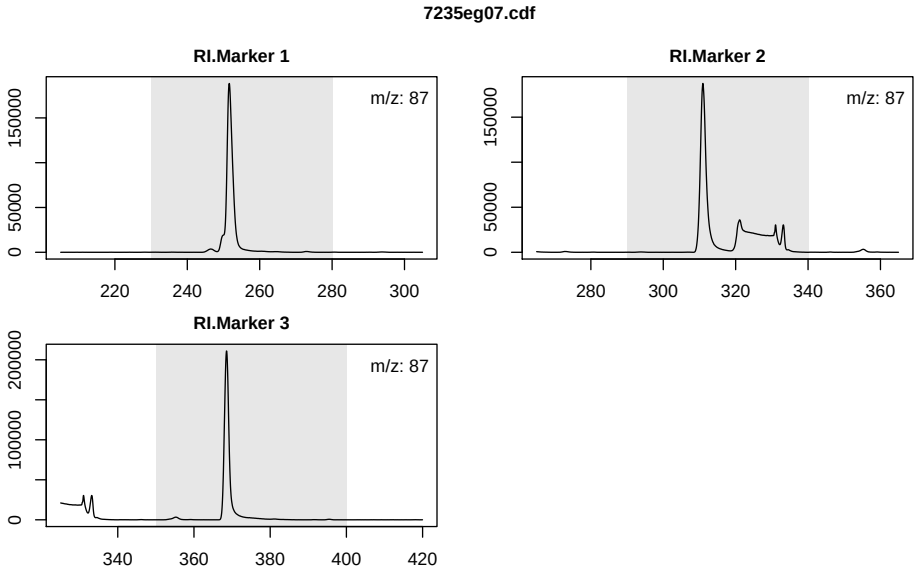

Figure 1: Example of the RI markers m/z traces in one sample

2 Pre-processing

2.1 Baseline correction

TargetSearch expects baseline corrected NetCDF files, which is usually performed by your GC-MS vendor's software. If that is the case, you may continue reading the following subsection.

However, if your chromatograms are not baseline corrected, *TargetSearch* provides a method implemented in the function `baselineCorrection()`. This algorithm is based on the work of Chang et al. [1] and a description is found in the `baselineCorrection()` documentation.

There is no need to call this function directly, as it is automatically called by `RIcorrect()`, which will be explained in the next subsection.

2.2 Peak detection

Initially, *TargetSearch* identifies the local apex intensities in all chromatograms, finds the retention time (RT) of the RI markers and converts RT to RI using linear interpolation [2]. This is done by the function `RIcorrect()`. Parameter to this function are sample and retention time limits objects, the intensity threshhold (`IntThreshold = 100`), the peak picking method (`pp.method`) and a `Window` parameter that will be used by said method. The function will return a matrix with the retention times of all RI markers and create a file with all extracted peaks of the respective NetCDF file, which will we call *RI file*. This file can be a tab-delimited file (RI file) (old format) or a custom binary file (default format). The difference between them is that the binary file allows much faster parsing compared to the other format.

If you chromatograms are not baseline corrected, it is at this point where you can pass the parameters `baseline=TRUE` to `RIcorrect()`. Additional baseline correction parameters can be passed via `baseline.opts`.

```
RImatrix <- RIcorrect(samples, rimLimits, IntThreshold = 100,
                      pp.method = "smoothing", Window = 15)
```

There are three peak picking methods available: *smoothing* implements the algorithm used by Tagfinder [3], *gaussian* uses basically the same algorithm, but the difference is the smoother is gaussian based, and the *ppc* algorithm, which is an adaptation of the function `ppc.peaks` from the package *ppc*. *Comment: This package seems to be deprecated. It is not in CRAN anymore*. This method simply searches for local maxima within a given time window. The `Window` parameter sets the window size of the chosen peak picking method in terms of number of scan points. *Note: The number of points actually used is* $2 \times Window + 1$.

2.3 RI correction

After running `RIcorrect()`, your chromatograms have been already retention time corrected. However, it is necessary to check that the RI markers detection was actually correct. To do so, *TargetSearch* examines the generated RI matrix `RImatrix` with the function `FAMEoutliers()`. It creates a PDF report of the RI markers including information about possible outliers that the user can remove or not. Alternatively, RI markers can be checked manually using the function `plotFAME` (Figure 2).

```
outliers <- FAMEoutliers(samples, RImatrix, threshold = 3)
```

```
##
##  Outliers Report:
##  ================
##
##
## No outliers were found.
```

```
## FAMEs were saved in TargetSearch-2019-02-27.FAME-report.pdf
```

Here, `threshold` sets the number of standard deviations a value has to be away from the mean before it will be considered an outlier. In this example, however, no outliers were detected.

```
plotFAME(samples, RImatrix, 1)
```

Note that *TargetSearch* assumes that the RI markers are injected together with the biological samples. A different approach is to inject them separately. If this is the case, please look at the vignette *RI correct extra* and its accompanying file '`RetentionIndexCorrection.R`' for a workaround.

In case of outliers, there is no single method to fix them as this really depends on the particular measurement conditions. However, there are few posibilities. i) If the time deviation from the mean is small, for example, less than one second, as a rule of thumb the outliers can usually be ignored. ii) if the outliers are due to chromatogram shift, then it is also fine to ignore them, because that is the point of the markers. iii) On the contrary, if the shift is because of a problem with the marker itself (eg, overloaded peak, low intense peak) and ***not*** because of a chromatogram shift, then do not ignore the outliers. In this case, you can manually set the retention time of the outlier to the mean/median, or choose another peak (which is stable across samples) as a new marker, or even just remove the RI marker from the list.

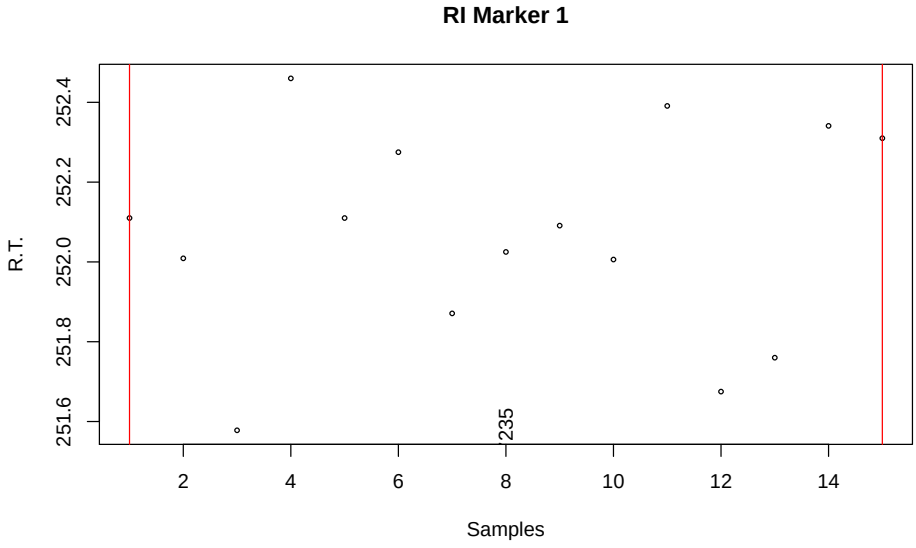

Figure 2: Retention Index Marker 1

3 Library Search

3.1 Reference Library File

The *reference library* file contains the information of the metabolites or mass spectral tags (MSTs) that will be searched for in the chromatograms. A public spectra database could be found here http://gmd.mpimp-golm.mpg.de/ at *The Golm Metabolome Database* [4, 5, 6].

Required information is the metabolite name ("Name"), expected retention time index ("RI"), selective masses ("SEL_MASS"), most abundant masses ("TOP_MASS"), spectrum ("SPECTRUM") and RI deviations ("Win_1", "Win_2", "Win_3"). See example in table 3. The columns "Name" and "RI" are mandatory and you have at least to include one of the columns "SEL_MASS", "TOP_MASS" or "SPECTRUM" in the file (see below). The RI deviation columns are optional.

Name	RI	Win_1	SEL_MASS	SPECTRUM
Pyruvic acid	222767	4000	89;115;158;174;189	85:7 86:14 87:7 88:5 8...
Glycine (2TMS)	228554	4000	86;102;147;176;204	86:26 87:19 88:8 89:4...
Valine	271500	2000	100;144;156;218;246	85:8 86:14 87:6 88:5 8...
Glycerol (3TMS)	292183	2000	103;117;205;293	85:14 86:2 87:16 88:13...
Leucine	306800	1500	102;158;232;260	158:999 159:148 160:45...
Isoleucine	319900	1500	102;103;158;163;218	90:11 91:2 92:1 93:1 9...
Glycine	325000	2000	86;100;174;248;276	85:6 86:245 87:24 88:12...

Table 3: Reference Library example

In this file, masses and intensities must be positive integers. RIs and RI deviations can be any positive real number. The selective and most abundant masses list must be delimited by semicolon (;). The spectrum is described by a list of mass and intensity pair. Every mass-intensity pair is separated by colon (:) and different pairs are separated by spaces.

The function `ImportLibrary()` imports the reference file.

```
lib.file <- file.path(cdf.path, "library.txt")
lib      <- ImportLibrary(lib.file, RI_dev = c(2000,1000,200),
            TopMasses = 15, ExcludeMasses = c(147, 148, 149))
```

Here we set the RI window deviations to 2000, 1000 and 200 RI units. Since "Win_1" column is already in the file, the first value (2000) is ignored. Also, the 15th most abundant masses are taken but excluding the masses 147, 148 and 149 (common confounding masses)

3.2 Library Search Algorithm

The library search is performed in three steps. First, for every metabolite, selective masses are searched in a given time window around the expected RI. This is done by the function `medianRILib()`. This function calculates the median RI of the selective masses and return new library object with the updated RI. The time deviation is given either in the library file (column "Win_1") or when the library is imported (see `ImportLibrary()`).

```
lib <- medianRILib(samples, lib)
```

It is also posible to examinate visually the RI deviation of the metabolites by setting the parameter `makeReport=TRUE`, which creates a pdf report like the one shown in figure 3. This may help to set or update the expected RI deviation.

In the second step, the function `sampleRI()` searches the selective masses again, but using the updated RI and the RI deviation defined in the library object ("Win_2"). After that, the intensities of the selected masses are normalised to the median of the day, and then used to extract other masses with correlated apex profiles. The masses for which the Pearson correlation coefficient is above `r_thres` are taken as metabolite markers and their RIs are averaged on a per sample basis. This average RI represents the exact position where the metabolite elutes in the respective sample, which is returned in a matrix form.

```
cor_RI <- sampleRI(samples, lib, r_thres = 0.95,
                   method = "dayNorm")
```

The third step will look up for all the masses (selective and most abundant masses) in all the samples. This is done by the function `peakFind()`. It returns a *tsMSdata* object with the intensities and RI of every mass (rows) and every sample (columns) that were search for.

```
peakData <- peakFind(samples, lib, cor_RI)
```

The intensity and RI slots can be accessed by using the `Intensity` and `retIndex` methods. Each slot is a list, where every component of the list is a matrix, which is related to a metabolite in the library. There should be as many components as metabolites in the library. In every matrix, columns are samples and rows are masses.

```
met.RI        <- retIndex(peakData)
met.Intensity <- Intensity(peakData)
# show the intensity values of the first metabolite.
met.Intensity[[1]]
```

```
##     7235eg04 7235eg06 7235eg07 7235eg08 7235eg09 7235eg11 7235eg12 7235eg15
## 89      3185     7974     3211     3544     9475     2172     9373     9873
## 115     1811     4279     1652     1924     5138     1081     5112     5460
```

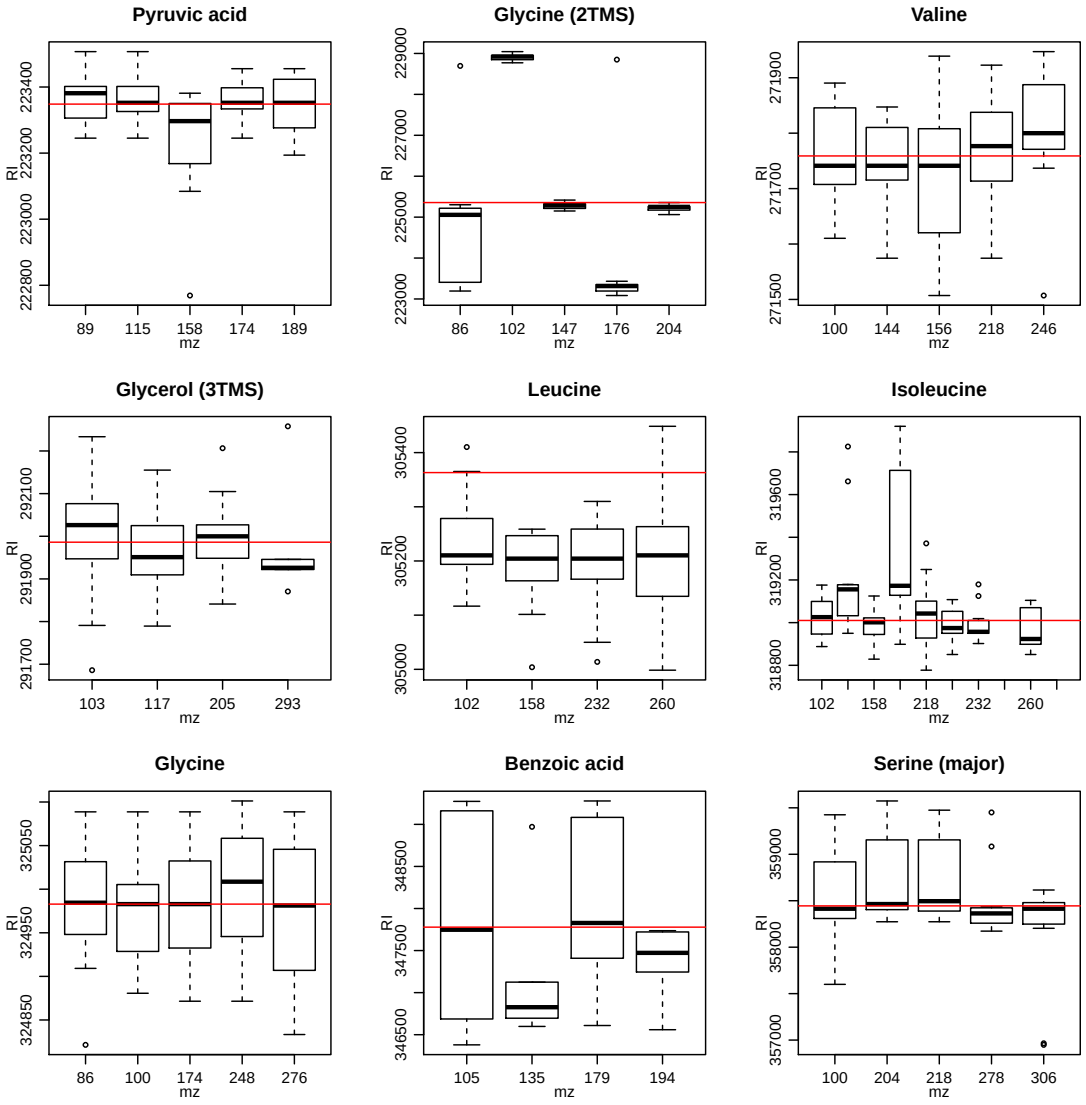

Figure 3: RI deviation of first 9 metabolites in the library

```
## 158      377    1162     400     458    1139     288    1122    1214
## 174     5840   14031    5690    6414   16970    3928   16852   17885
## 189       NA     200      NA     101     250      NA     237     248
## 99      1131    3034    1191    1260    3658     753    3621    3793
## 100     1021    2672    1037    1046    3074     630    3136    3269
## 175      762    1828     731     795    2307     467    2265    2443
## 117      359     957     342     380    1208     225    1214    1259
## 90        NA     729     282     301     855     170     865     977
## 176      239     547     234     232     729      NA     729     758
## 116      271    1294      NA      NA    1012     155    1185    1262
## 114      153     338     133     150     455      NA     413     439
## 101      130      NA     158     182     469      NA     507     528
## 91       178     411     172     168     503     103     490     592
```

```
##     7235eg20 7235eg21 7235eg22 7235eg25 7235eg26 7235eg30 7235eg32
## 89     11701     3245    10407     2738     1577     2330      460
## 115     6384     1819     5662     1488      822     1311      253
## 158     1451      410     1229       NA       NA      325       NA
## 174    20963     5826    18655     5135     2971     4425      909
## 189      292       NA      293       NA       NA       NA       NA
## 99      4471     1193     3945     1027      580      867      183
## 100     3876     1058     3408       NA      446      696      170
## 175     2831      725     2501      661      354      528      122
## 117     1451      343     1350      303      138      265       NA
## 90      1111      252      961      241       NA      215       NA
## 176      926      237      780       NA      100      192       NA
## 116     1451       NA     1193      335      121      194       NA
## 114      520      135      483       NA       NA       NA       NA
## 101      666      153      516      130       NA      118       NA
## 91       624      213      525      103      121      134       NA
```

4 Metabolite Profile

The function `Profile` makes a profile of the MS data by averaging all the normalised mass intensities whose Pearson coefficient is greater that `r_thresh`.

```
MetabProfile <- Profile(samples, lib, peakData, r_thres = 0.95,
                        method = "dayNorm")
```

A *msProfile* object is returned. The averaged intensities and RI matrices that can be obtained by `Intensity` and `retIndex` methods. The profile information is represented by a *data.frame* in the `info` slot (accessible by `profileInfo` method). The columns are:

Name The metabolite/analyte name.

Lib_RI The expected RI (library RI).

Mass_count The number of correlating masses.

Non_consecutive_Mass_count Same as above, but not counting the consecutive masses.

Sample_Count_per_Mass The number of samples in which a correlating mass was found. The numbers are separated by semi-colon (;) and each number corresponds to one correlating masses (same order). If all the numbers are the same, only that unique number is shown.

Masses The correlating masses.

RI The average RI.

Score_all_masses The similarity score calculated using the average intensity of all the masses that were searched for, regardless of whether they are correlating masses.

Score_cor_masses Same as above, but only correlating masses are considered.

As metabolites with similar selective masses and RIs can be present in metabolite libraries, it is necessary to reduce redundancy. This is performed by the function `ProfileCleanUp` which selects peaks for which the RI gap is smaller than `timeSplit` and computes the Pearson correlation between them. When two metabolites within such a time-group are tightly correlated (given by `r_thres`) only the one with more correlated masses is retained.

```
finalProfile <- ProfileCleanUp(MetabProfile, timeSplit = 500,
                               r_thres = 0.95)
```

The function returns a *msProfile* object. The `info` slot is similar as described above, but extra columns with a "Cor_" preffix (e.g., "Cor_Name") are included. They provide information about metabolite redundancy.

5 Peaks and Spectra Visualisation

Finally, it may be of interest to check the chromatographic peak of selected metabolites and compare the median spectra of the metabolites, i.e., the median intensities of the selected masses across all the samples, with the reference spectra of the library. There are two functions to do so: `plotPeak` and `plotSpectra`.

For example, we can compare the median spectrum of "Valine" against its spectrum reference. Here we look for the library index of "Valine" and plot the spectra comparison in a "head-tail" plot (figure 4).

```
grep("Valine", libName(lib))

## [1] 3

plotSpectra(lib, peakData, libId = 3, type = "ht")
```

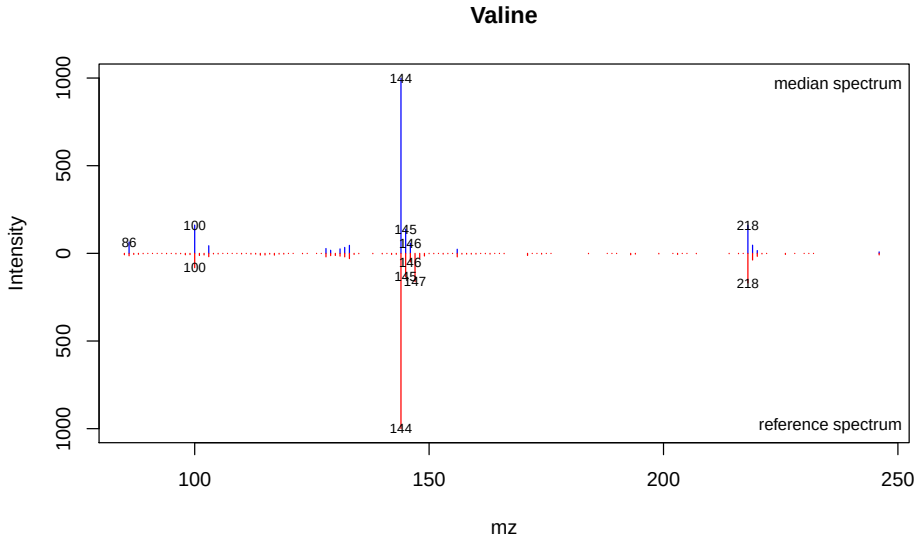

Figure 4: Spectra comparison of "Valine"

To look at the chromatographic peak of "Valine" in a given sample, we use the functions `peakCDFextraction` to extract the raw chromatogram and `plotPeak` to plot the peak (figure 5).

```
# we select the first sample
sample.id <- 1
cdf.file  <- file.path(cdf.path, cdffiles[sample.id])
rawpeaks  <- peakCDFextraction(cdf.file)
# which.met=3 (id of Valine)
plotPeak(samples, lib, MetabProfile, rawpeaks, which.smp=sample.id,
    which.met=3, corMass=FALSE)
```

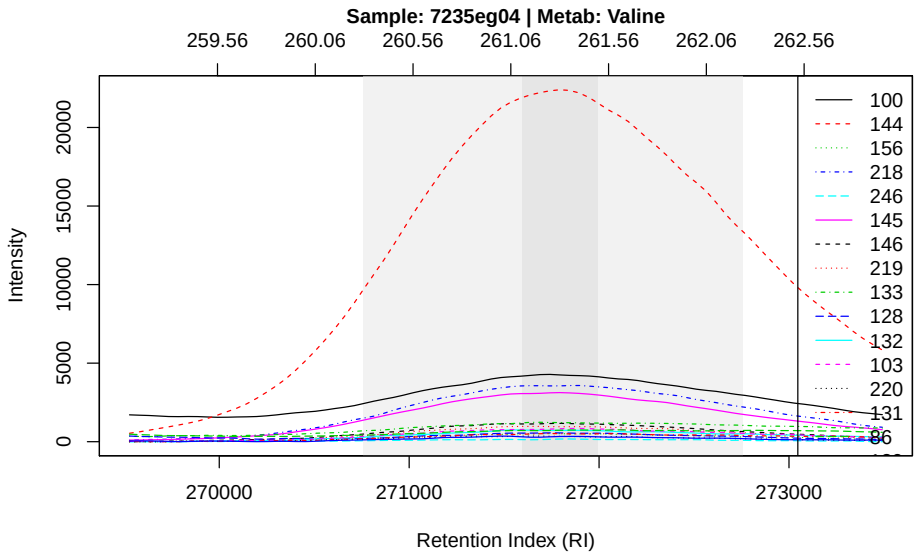

Figure 5: Chromatographic peak of Valine

Refer to the documentation of the functions `plotPeak` and `plotSpectra` for further options not covered here.

6 TargetSearch GUI

We provide a grahical user interface intended to facilitate the use of *TargetSearch* for users unfamiliar with R. Many parameters that would be set calling the individual *TargetSearch* functions as described in this document can be set here "in one go" before running the complete analysis. A screenshot of the GUI is shown in figure 6.

This is a descript of all the GUI options.

Working Directory : Use the *Browse*-button to select the folder on your hard drive containing all your GC-MS data files. The output of *TargetSearch* will be written to this folder too.

File Import : Clicking *NetCDF Data* or *Apex Data* radio buttons will open a file select dialog. Choose the files you would like to be processed. You may check your selection pressing the *Show*-button.

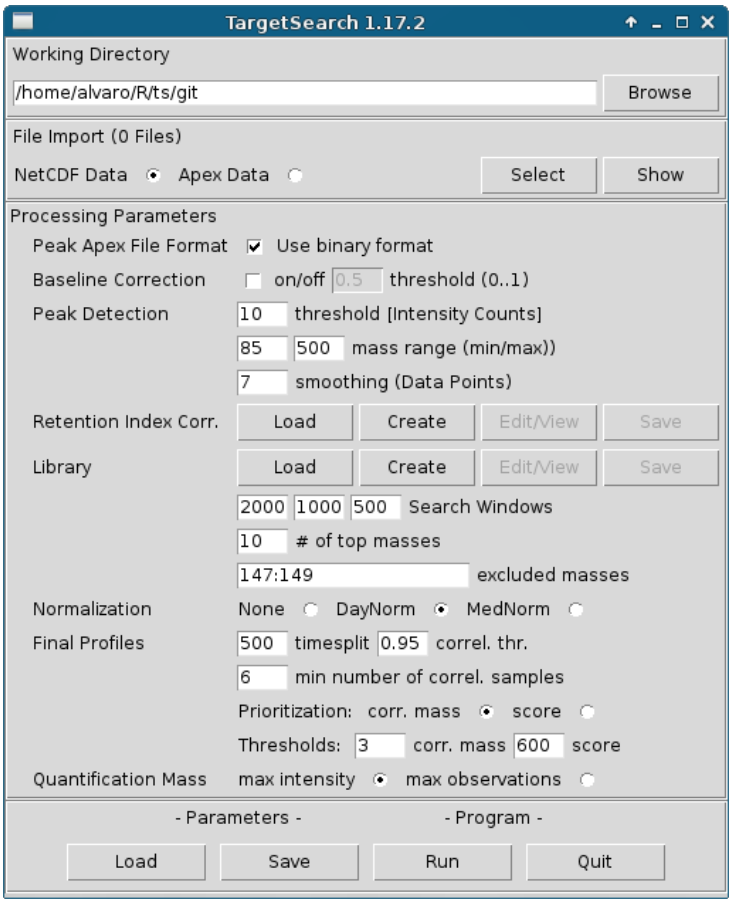

Figure 6: The *TargetSearch* GUI

Baseline Correction : Clicking *on/off* button will perform baseline correction before peak detection. If selected, the `threshold` parameter is a numeric value between 0 and 1. A value of one returns a baseline above the noise, 0.5 in the middle of the noise and 0 below the noise. See `baselineCorrection` documentation for further details.

Peak Detection : *Intensity Counts threshold* defines the minimum apex intensity incorporated in the analysis. A value of 1 would include all peaks. *Mass Range* allows to limit the mass values (m/z) to be included in the analysis. *Smoothing* averages raw data to eliminate some inherent noise leading to multiple peaks otherwise.

Retention Index Correction : Retention Index Correction is neccessary and applied only if you supply NetCDF Data (Apex Data contain already Retention Indices). You may *Load* or *Create* the search windows for your RI-Markers here.

Library : A Library (to detect metabolites) usable by *TargetSearch* contains at least information about the metabolite 'Name', its expected 'RI' and the selective masses in its spectrum 'SEL_MASS'. You may *Load* or *Create* one yourself using the respective buttons. A more detailed description of the file formats can be found in `ImportLibrary`. *Search Windows* refers to the allowed RI deviation of your metabolites which are narrowed in 3 consecutive searches. *No. of top masses* is the number of most abundant masses that will be selected from the spectra, besides the selected masses.

ExcludeMasses is only used when masses are obtained from the spectra. For example, "126,147:149,160" means that the masses 126, 147, 148, 149, and 160 will be excluded from the analysis (unless they are used as selected masses).

Normalization : This selects how the data will be normalized during the metabolite search. Options are `dayNorm`, a day based median normalization, `medianNorm`, normalization using the median of all the intensities of a given mass, and `none`, no normalization at all.

Final Profiles : Here you may set the parameters used by the functions `Profile` and `ProfileCleanUp`. *timesplit* sets an RI window that will be used to look for metabolites that could have been redundanly identified. *correl. thr.* is the correlation threshold and *min. number of correlation samples* is a threshold used to make sure that correlations are computed with at least said number of observations. *prioritization* is used to resolv ambiguities by taking the metabolite with the most correlating masses (*corr. mass*) or the highest similarity *score*. Metabolites with corr. mass and score higher than the thresholds will be suggested above the one that do not pass the thresholds, which are marked as "unidentified."

Quantification Mass : This section is used to generate an intensity matrix based on one quantification mass. The quantification mass can be specified in the library file (column "QUANT_MASS") or selected automatically by choosing *max intensity* or *max observations*. *max intensity* selects the mass with the highest intensity across the samples, while *max observation* takes the mass with less missing values.

Parameters : You may *Save* the current parameters as an `*.RData` file or *Load* previously saved parameters to compare the outcome of different settings or just repeat the analysis.

Program : *Run* starts to process all currently selected files using the current parameters and saving output to *Working Directory*. *Quit* closes the GUI.

7 Untargeted search

Although *TargetSearch* was designed to be targeted oriented, it is possible to perform untargeted searches. The basic idea is to create a library that contains evenly distributed "metabolites" in time and every "metabolite" uses the whole range of possible masses as selective masses. An example:

```
metRI    <- seq(200000, 300000, by = 5000)
metMZ    <- 85:250
metNames <- paste("Metab",format(metRI,digits=6), sep = "_")
```

Here we define a set of metabolites located every 5000 RI units from each other in the RI range 200000-300000, with selective masses in the range of 85-300, and assign them a name. After that, we create an `tsLib` object.

```
metLib   <- new("tsLib", Name = metNames, RI = metRI,
     selMass = rep(list(metMZ), length(metRI)), RIdev = c(3000, 1500, 500))
```

Now we can use this library object to perform a targetive search with this library on the *E. coli* samples as we did before.

```
metLib <- medianRILib(samples, metLib)
metCorRI <- sampleRI(samples, metLib)
metPeakData <- peakFind(samples, metLib, metCorRI)
metProfile <- Profile(samples, metLib, metPeakData)
metFinProf <- ProfileCleanUp(metProfile, timeSplit = 500)
```

The `metFinProf` object can be used to create a new library by taking only the metabolites that have, for example, more than 5 correlating masses and using the correlating masses as selective ones.

```
sum(  profileInfo(metFinProf)$Mass_count > 5)

## [1] 12

tmp   <-  profileInfo(metFinProf)$Mass_count > 5
metRI    <- profileInfo(metFinProf)$RI[tmp]
metNames <-  as.character( profileInfo(metFinProf)$Name[tmp] )
metMZ <- sapply(profileInfo(metFinProf)$Masses[tmp],
        function(x) as.numeric(unlist(strsplit(x,";"))) )
metLib   <- new("tsLib", Name = metNames, RI = metRI,
     selMass = metMZ, RIdev = c(1500, 750, 250))
```

After the new library object is created, the process can be repeated as shown above.

Finally, by using the function `writeMSP`, it is possible to export the spectrum of the unknown metabolites to MSP format used by NIST mass spectra search (http://www.nist.gov/srd/mslist.htm), so the unknown spectra can be search against known metabolite spectra databases.

References

[1] D. Chang, C. D. Banack, and S. L. Shah. Robust baseline correction algorithm for signal dense NMR spectra. *J. Magn. Reson.*, 187(2):288–292, 2007.

[2] H. Van den Dool and P. D. Kratz. A generalization of retention index system including linear temperature programmed gas-liquid partition chromatography. *J. Chromatography*, 11(4):463–&, 1963.

[3] A. Luedemann, K. Strassburg, A. Erban, and J. Kopka. Tagfinder for the quantitative analysis of gas chromatography - mass spectrometry (GC-MS)-based metabolite profiling experiments. *Bioinformatics*, 24(5):732–737, 2008.

[4] J. Kopka, N. Schauer, S. Krueger, C. Birkemeyer, B. Usadel, E. Bergmuller, P. Dormann, W. Weckwerth, Y. Gibon, M. Stitt, L. Willmitzer, A. R. Fernie, and D. Steinhauser. Gmd@csb.db: the Golm Metabolome database. *Bioinformatics*, 21(8):1635–1638, 2005.

[5] J. Hummel, J. Selbig, D. Walther, and J. Kopka. The golm metabolome database: a database for gc-ms based metabolite profiling. In J. Nielsen and M.C. Jewett, editors, *Metabolomics*, volume 18, pages 75–96. Springer-Verlag, Berlin, Heidelberg, New York, 2007. doi:10.1007/4735_2007_0229.

[6] J. Hummel, N. Strehmel, C. Bölling, S. Schmidt, D. Walther, and J. Kopka. Mass spectral search and analysis using the golm metabolome database. In *The Handbook of Plant Metabolomics*, pages 321–343. Wiley-VCH Verlag GmbH & Co. KGaA, 2013. doi:10.1002/9783527669882.ch18.

8 Session info

Output of `sessionInfo()` on the system on which this document was compiled.

- R version 3.5.2 (2018-12-20), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Running under: `Windows Server 2012 R2 x64 (build 9600)`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: TargetSearch 1.38.2, TargetSearchData 1.20.0, ncdf4 1.16
- Loaded via a namespace (and not attached): BiocManager 1.30.4, BiocStyle 2.10.0, Rcpp 1.0.0, compiler 3.5.2, digest 0.6.18, evaluate 0.13, highr 0.7, htmltools 0.3.6, knitr 1.21, magrittr 1.5, rmarkdown 1.11, stringi 1.3.1, stringr 1.4.0, tcltk 3.5.2, tools 3.5.2, xfun 0.5, yaml 2.2.0