

The TargetSearch Package

***Alvaro Cuadros-Inostroza¹, Jan Lisec¹, Henning Redestig¹,
and Matthew A Hannah¹***

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

September 13, 2021

Abstract

This document describes how to use [TargetSearch](#) to preprocess GC-MS data.

Package

TargetSearch 1.49.3

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

Contents

1	Supplied files	3
1.1	NetCDF files	3
1.2	Sample Information	3
1.3	Retention index markers	4
2	Pre-processing	5
2.1	Conversion of CDF files to TargetSearch format	5
2.2	Baseline correction	6
2.3	Peak detection	6
2.4	RI correction	7
3	Library Search	8
3.1	Reference Library File	8
3.2	Library Search Algorithm	9
4	Metabolite Profile	11
5	Peaks and Spectra Visualisation	12
6	TargetSearch GUI	13
7	Untargeted search	13
8	Session info	15

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 *TargetSearch-Data*.

1.1 NetCDF files

TargetSearch can currently read only NetCDF files. Many GC-MS software vendors are able to convert their proprietary raw chromatograms to NetCDF. There are however two issues you have to keep in mind:

- Make sure that you export the CDF files as nominal mass, not centroid nor profile mode, as *TargetSearch* cannot handle this type of data. Nevertheless, if such an option is not available, there files can be converted to nominal mass automatically.
- It is recommended to baseline correct your chromatograms before exporting to NetCDF to increase processing speed. If not, *TargetSearch* implements two methods for baseline correction that you can use at the cost of performance.

Though only NetCDF is supported, it is still possible to use other formats, such as mzXML, mzData or mzML using the package *mzR*. This is not yet implemented but it will in the next release.

1.2 Sample Information

After exporting the CDF files into a convenient location, prepare a tab-delimited text file describing your samples. It must 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. Additionally, if a column named "SAMPLE_NAME" is found, this column will be used as sample names instead of the CDF files (useful if your CDF files are not human readable friendly).

Other columns such as 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.

The TargetSearch Package

```
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)
```

1.3 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 threshold) 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 (a subset of) the samples. By default, the option `single` is `TRUE`, which means only a (randomly chosen) sample will be used for the figure (the reason is we usually do not need to see all the samples,

The TargetSearch Package

and it is computationally expensive to display too many samples at the same time). To display multiple samples, set the said option to `FALSE`. In this case, it is recommended to use *subsetting* to select specific sample to review (in particular in large datasets).

The example below demonstrates the multiple display option restricted to the first three samples. Some basic plot styling is also possible by passing arguments to `matplot()`, such as, `col` and `lty`. The generated plot is shown in figure 1. Note that the gray area indicates the search window of each marker.

```
checkRimLim(samples[1:3], rimLimits, single=FALSE, col=2:4, lty=1)
```

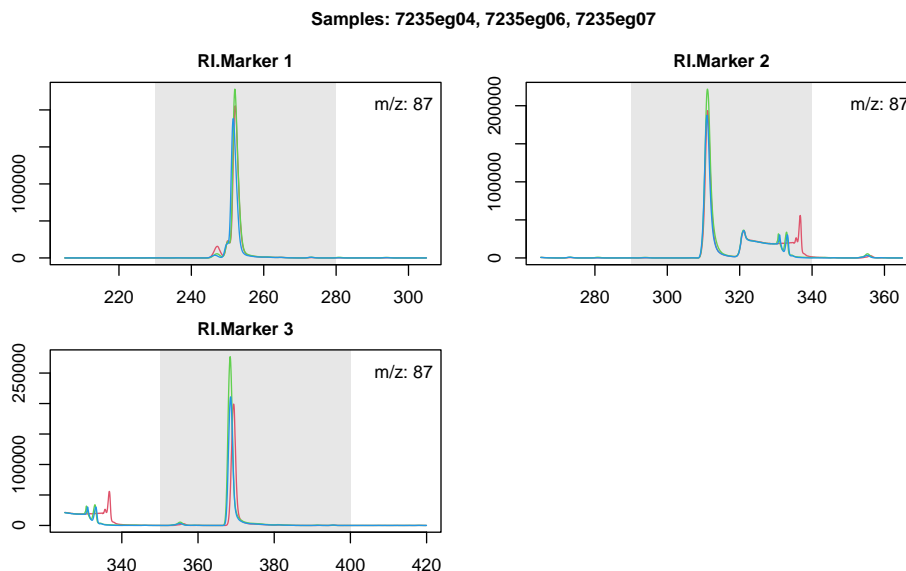


Figure 1: Example of the RI markers m/z traces in samples

2 Pre-processing

2.1 Conversion of CDF files to TargetSearch format

Starting from *TargetSearch* 1.42, a CDF4 custom format was introduced. This format has many advantages over the typical CDF3 that is exported by commercial software, mainly compression and fast data access, which can speed up the internal *TargetSearch* computations (specially in plotting functions). Another advantage is that you can store baseline corrected data in this file, which is not possible to do in CDF file. And finally, the conversion is required if the CDF files are not in nominal mass format. The main disadvantage is that this format is unique to *TargetSearch* and it is unlikely to be used elsewhere, and the extra storage space required for these files.

To transform the files, simply run the function `ncdf4Convert()`. In the example below, the files are converted to CDF4 format and saved to the working directory. If the parameter `path` is missing, then the files will be placed in the same directory as the original CDF files.

```
samples <- ncdf4Convert(samples, path=".")
```

If you need to perform baseline correction on the CDF files, please check the following section instead.

The TargetSearch Package

Note: Although this conversion is optional, it is highly recommended. It is still possible to use the classic workflow at the cost of computation speed.

2.2 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 two baseline correction methods. The first one is implemented in `baselineCorrection()` and is based on the work of Chang et al. [1]. This method is referred as “classic” in the man-pages as it has been present since the beginning. The second method called “quantiles”, introduced in *TargetSearch* 1.42.0, estimates the baseline by computing a given quantile in a sliding window (the quantile is 50% by default). This method is implemented in `baselineCorrectionQuant()`.

A wrapper function called `baseline()` is also provided, which has the advantage that it can take a CDF file name as input (the other functions take matrices). *Note: These functions are meant to be run internally and only exposed for advanced users.*

To apply baseline correction to the chromatograms, we simply call the function `ncdf4Convert()`, similarly as the section above, but with extra parameters that are required so the baseline correction procedure takes place. An example of such a call is shown below.

```
samples <- ncdf4Convert(samples, path=".", baseline=TRUE, bsline_method="quantiles", ...)
```

Here, the parameter `baseline` is set to `TRUE` (by default is `FALSE`) and we choose the method “quantiles”. The ... contain the parameters passed to `baselineCorrectionQuant()`. Similarly, if the chosen method is “classic”, then the extra parameters will be passed to `baselineCorrection()`. Please refer to the manpages for details on these methods.

2.3 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()`. Parameters to this function are sample and retention time limits objects, the intensity threshold (`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.

Note: Formerly, we would use this function for baseline correction, but as of TargetSearch 1.42.0, this has to be done as explained in the section above.

```
RImatrix <- RIcorrect(samples, rimLimits, IntThreshold = 100,  
                      pp.method = "ppc", 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*

The TargetSearch Package

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 $2 \times \text{Window} + 1$.*

2.4 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-2021-09-13.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)
```

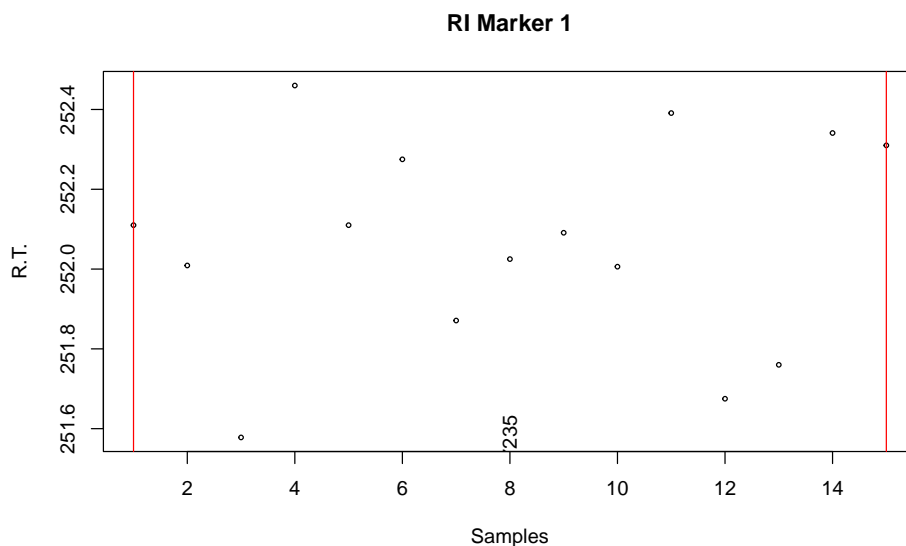


Figure 2: Retention Index Marker 1

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 possibilities. 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.

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 possible to examine 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.

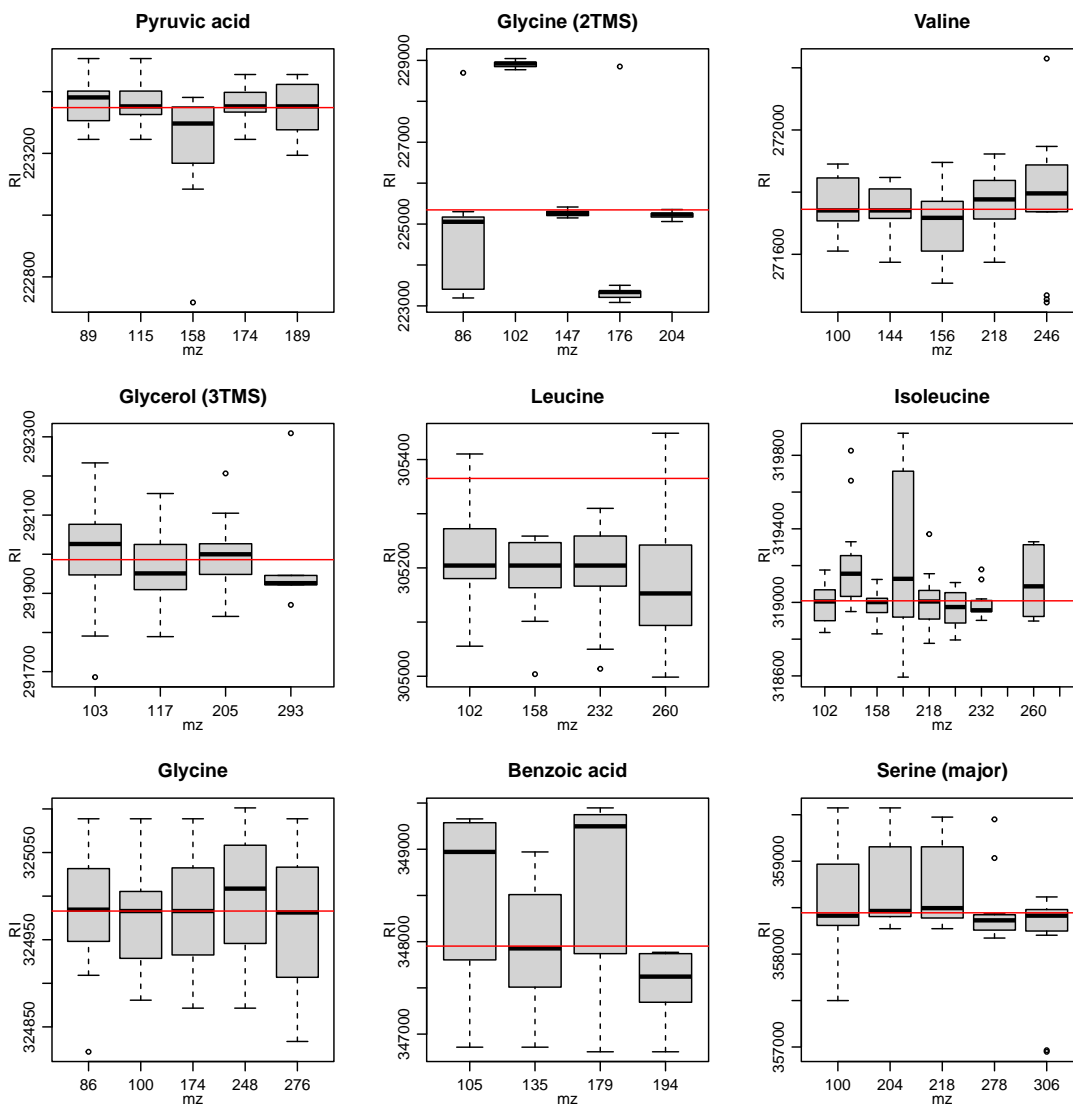


Figure 3: RI deviation of first 9 metabolites in the library

The TargetSearch Package

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
## 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	NA	301	855	170	865	977
## 176	239	547	234	232	729	149	729	758
## 116	271	1305	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	NA	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	NA	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	115	NA
## 101	666	153	516	130	NA	118	NA
## 91	624	213	525	NA	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 than `r_thresh`.

```
MetabProfile <- Profile(samples, lib, peakData, r_thresh = 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_thresh`) only the one with more correlated masses is retained.

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

The function returns a *msProfile* object. The `info` slot is similar as described above, but extra columns with a "Cor_" prefix (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")
```

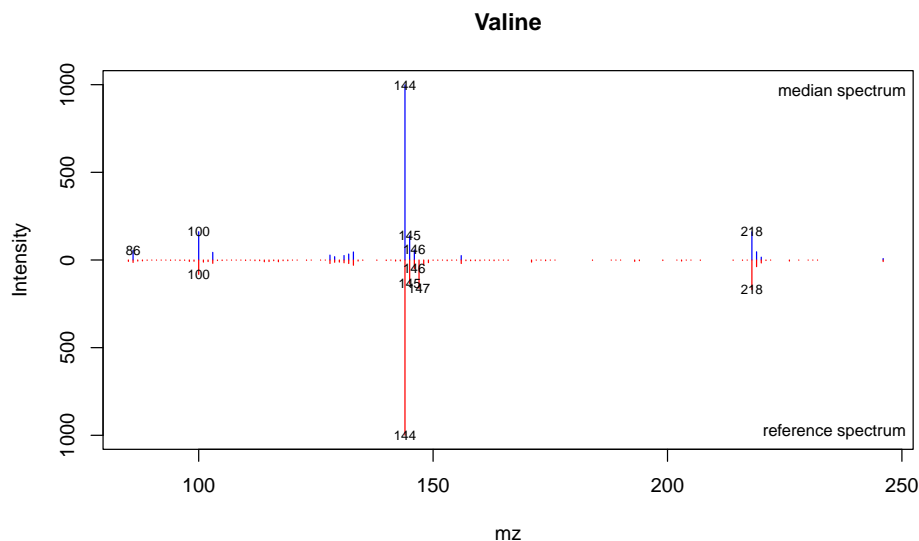


Figure 4: Spectra comparison of “Valine”

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)
```

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

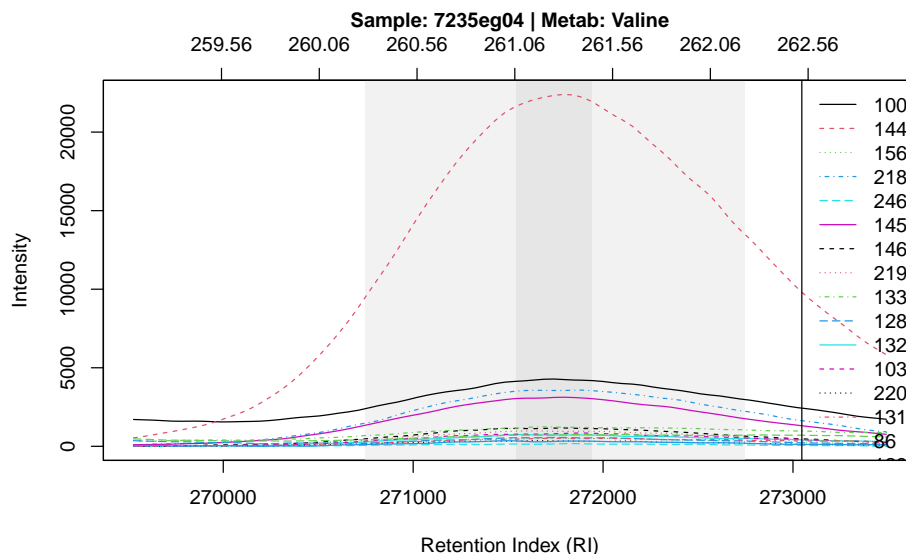


Figure 5: Chromatographic peak of Valine

6 TargetSearch GUI

Warning: The graphical user interface (GUI) has been removed from TargetSearch. Please use the regular functions.

Nevertheless, the old GUI is still available in my github repository. Please follow the instructions if you want to install it.

<https://github.com/acinostroza/TargetSearchGUI>

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.

The TargetSearch Package

```
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] 11

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):463–48, 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](https://doi.org/10.1002/9783527669882.ch18).

8 Session info

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

- R version 4.1.1 (2021-08-10), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_GB, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Running under: Ubuntu 20.04.3 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: TargetSearch 1.49.3, TargetSearchData 1.31.0
- Loaded via a namespace (and not attached): BiocManager 1.30.16, BiocStyle 2.21.3, assertthat 0.2.1, compiler 4.1.1, digest 0.6.27, evaluate 0.14, fastmap 1.1.0, highr 0.9, htmltools 0.5.2, knitr 1.34, magrittr 2.0.1, ncd4 1.17, rlang 0.4.11, rmarkdown 2.10, stringi 1.7.4, stringr 1.4.0, tools 4.1.1, xfun 0.25, yaml 2.2.1