The TargetSearch Package

Alvaro Cuadros-Inostroza, Jan Lisec,
Henning Redestig and Matthew A Hannah
Max Planck Institute for Molecular Plant Physiology
Potsdam, Germany
http://www.mpimp-golm.mpg.de/

October 28, 2009

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

1 Supplied Files

This section describes the files that have to be prepared before running *TargetSearch*. They are the sample file, the reference library file and the retention marker definition. For example purposes, 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. It is also recommended to baseline correct your chromatograms before exporting to NetCDF. Please refer to your software documentation for details.

After exporting the NetCDF files, please place them in a convenient location. Then prepare a text file to describe your samples. It must be tab-delimited and have at least two columns: "CDF_FILE" and "MEASUREMENT_DAY". 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.

To import the sample list into R, use the function `ImportSamples()`. You also need to specify the directory where the NetCDF files are (*CDFpath*) and a directory where the transformed cdf files, the so called RI files, will be saved (*RIpath*).

```
> library(TargetSearchData)
> library(TargetSearch)
```

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, "samples.txt"

```
> 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 = ".")
```

You could alternatively create a `tsSample` object by using the sample class methods.

```
> cdffiles <- dir(cdf.path, pattern = "cdf$")
> rifiles <- paste("RI_", sub("cdf", "txt", cdffiles), sep = "")
> days <- substring(cdffiles, 1, 4)
> smp_names <- sub("\\.cdf", "", cdffiles)
> smp_data <- data.frame(CDF_FILE = cdffiles, GROUP = gl(5, 3))
> samples <- new("tsSample", Names = smp_names, CDFfiles = cdffiles,
+     CDFpath = cdf.path, RIpath = ".", days = days, RIfiles = rifiles,
+     data = smp_data)
```

1.2 Retention Time Markers

The RI markers time definition, time window and m/z values has to be provided in a tab-delimited text file. The first two columns indicate the lower and upper window limits where the retention marker will be searched. The third column is the RI of that particular marker. An example is shown in table 2.

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

Table 2: Retention time definition file example, "rimLimits.txt"

Use the function `ImportFameSettings()` to import the limits and to set the mz/z markers. Alternatively, the markers could be specified by an additional column in the RI markers file.

```
> 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.

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.

2 Baseline correction, peak identification and RI correction

Initially, *TargetSearch* identifies the local apex intensities in all chromatograms, finds the retention time of the RI markers and converts the retention time to RI using linear interpolation (Van den Dool and Kratz, 1963). This is done by the function `RIcorrect()`. It takes as parameter the sample and retention time limits objects, the mass range to extract (*MassRange = c(85,500)*), the intensity threshhold (*IntThreshold = 10*), 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 the retention time markers and creates a tab-delimited file (RI file) per every chromatogram in the selected directory. These files contein the extracted peak list of the respective NetCDF file.

```
> RImatrix <- RIcorrect(samples, rimLimits, massRange = c(85, 500),
+     IntThreshold = 10, pp.method = "smoothing", Window = 7)
```

There are two peak picking methods available. "smoothing" implements the algorithm used by Tagfinder Luedemann et al. (2008). The "ppc" algorithm is a port from the package *ppc* function `ppc.peaks`. It looks for the local maxima within a given time window. The *Window* parameter sets the window size of the chosen peak picking method.

In case that the chromatograms were not baseline corrected by the platform specific GC-MS software, it is possible to do so with the function `RIcorrect()` as well. There are to extra arguments you have to include: *baseline* is a logical value that tells `RIcorrect()` whether to baseline correct the chromatograms or not. *baseline.opts* is a list of options passed to `baselineCorrection()` function. This algorithm is based on the work of Chang et al. (2007) and a description is found in `baselineCorrection()` documentation.

After that, it is possible to check for outliers in the samples by using the function `FAMEoutliers()`. It creates a PDF report of the retention time markers informing of possible outliers that the user can remove or not. Alternatively, retention index markers can be checked manually by the function `plotFAME`. Figure 1 shows the retention times of the marker 1.

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

```
Outliers Report:
===============

No outliers were found.
```

Here *threshold* sets the number of standard deviation a sample will be considered outlier. In this example, no outliers were detected.

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

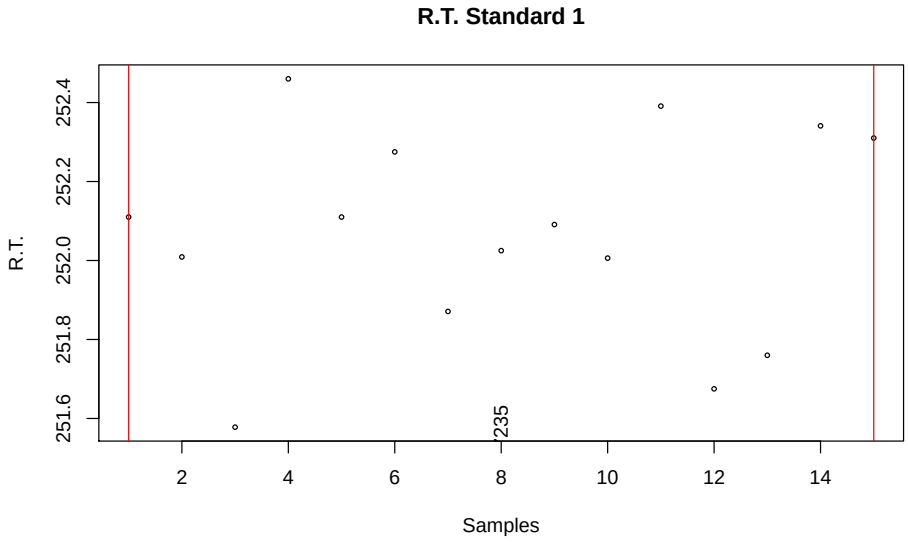

Figure 1: Retention Index Marker 1.

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://csbdb.mpimp-golm.mpg.de/csbdb/gmd/gmd.html` at *The Golm Metabolome Database* (Kopka et al., 2005).

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, "library.txt"

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 2. 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

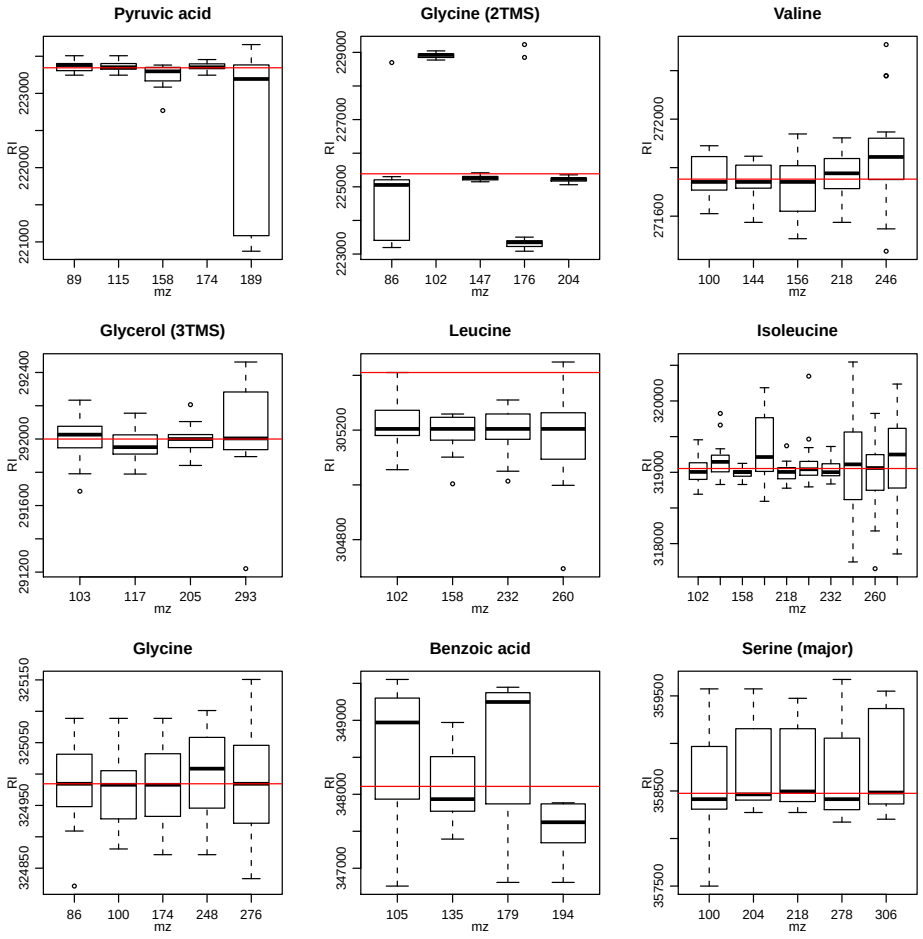

Figure 2: RI deviation of first 9 metabolites in the library.

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 matrices can be accessed by using the *Intensity* and *retIndex* methods.

```
> met.RI <- retIndex(peakData)
> met.Intensity <- Intensity(peakData)
```

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 The total number of masses that were found in the samples.

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 3).

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 4).

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 5.

This is a descript of all the GUI options.

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

```
[1] 3
```

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

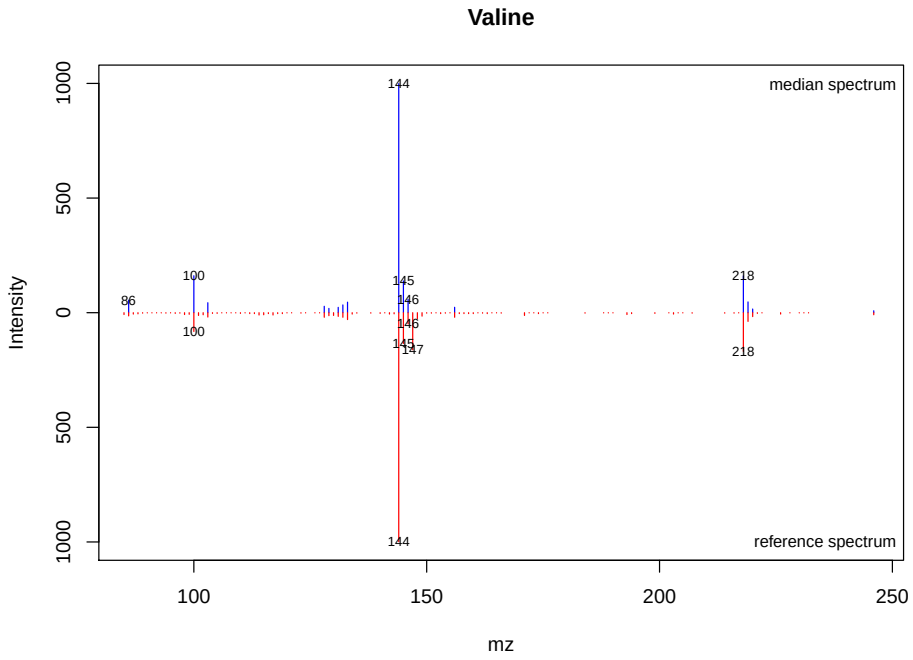

Figure 3: Spectra comparison of “Valine”

```
> top.masses <- topMass(lib)[[3]]
> sample.id <- 1
> cdf.file <- file.path(cdf.path, cdffiles[sample.id])
> rawpeaks <- peakCDFextraction(cdf.file, massRange = c(85, 500))
> plotPeak(rawpeaks, time.range = libRI(lib)[3] + c(-2000, 2000),
+     masses = top.masses, useRI = TRUE, rimTime = RImatrix[, sample.id],
+     standard = rimStandard(rimLimits), massRange = c(85, 500),
+     main = "Valine")
```

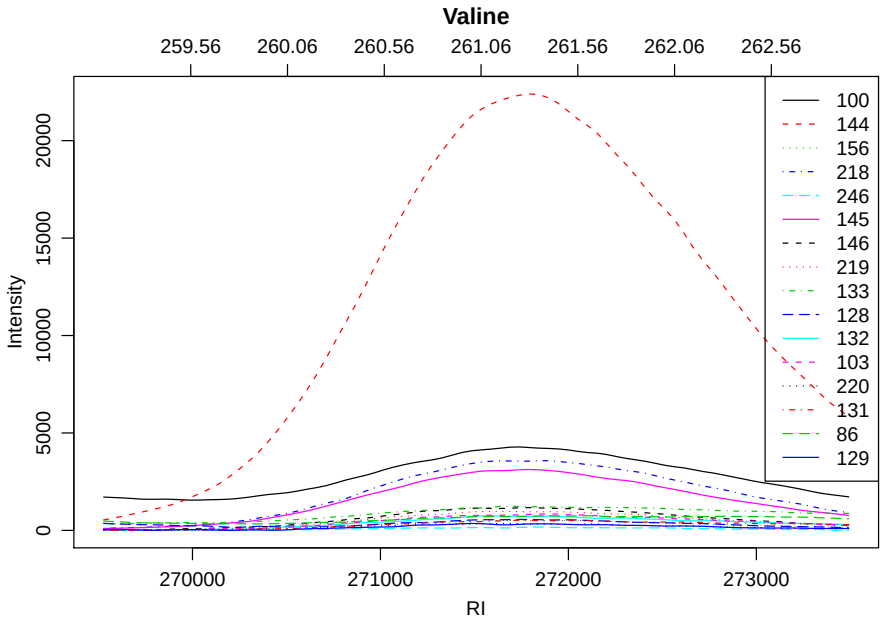

Figure 4: Chromatographic peak of Valine.

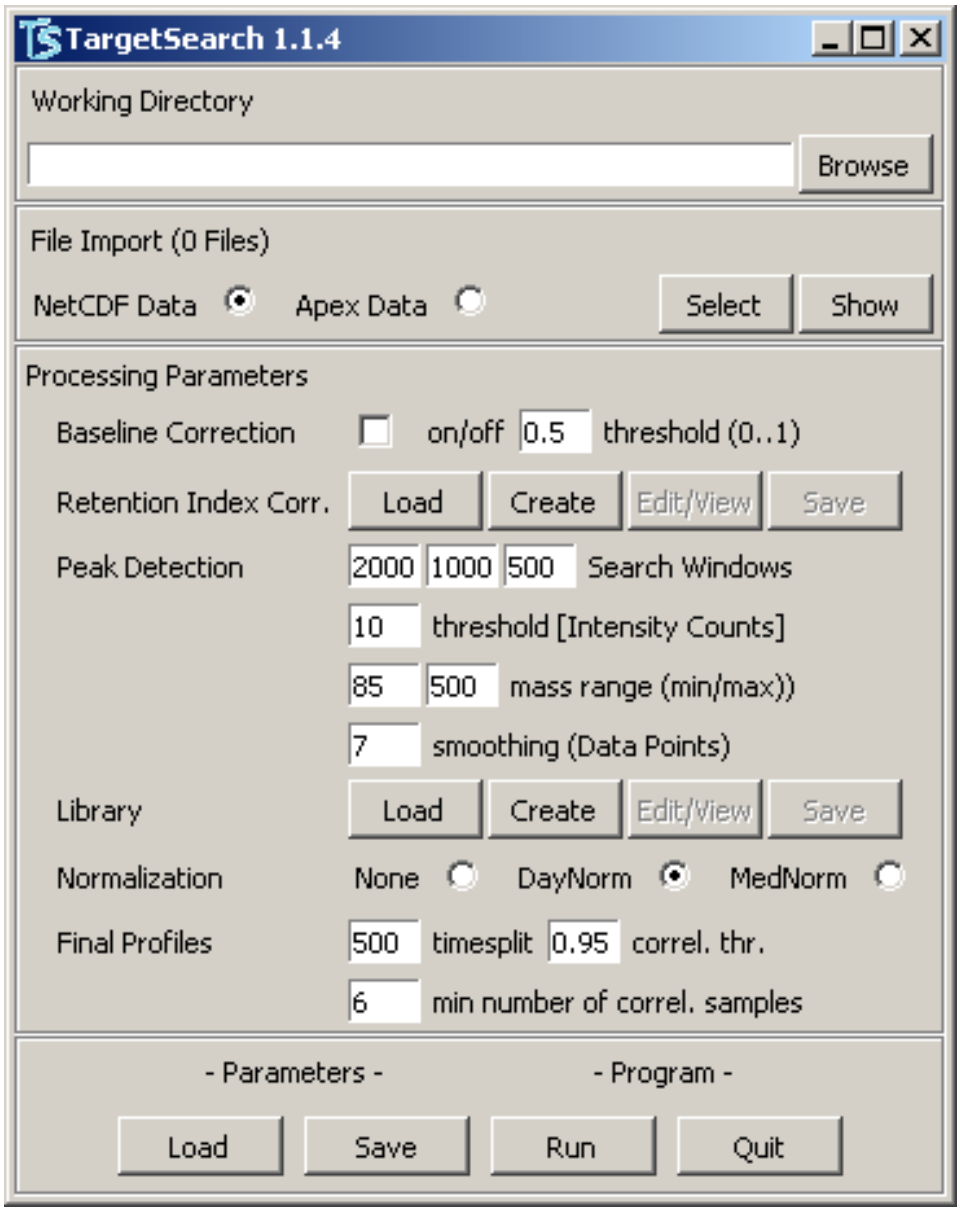

Figure 5: The *TargetSearch* GUI.

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.

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.

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.

Peak Detection : *Search Windows* refers to the allowed RI deviation of your metabolites which are narrowed in 3 consecutive searches. *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.

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`.

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.

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(2e+05, 3e+05, 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(
+     ";"))))
> 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

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.

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.

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.

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.