MetaBox Guide

Raphael Aggio[1,2]; Arno Mayor[2]; Sophie Reade[2]; Chris S.J. Probert[2] and Katya Ruggiero[1]

January 22, 2015

- 1 - Department of statistics and School of Biological Sciences - The University of Auckland - Private Bag 92019 - Auckland 1142 - New Zealand.
- 2 - Gastroenterology Unit - Cellular and Molecular Physiology - Institute of Translational Medicine - University of Liverpool - Nuffield Building, Crown Street, Liverpool L693BX, UK.

1 Introduction

MetaBox is an R package developed to process biological samples analyzed by Gas chromatography - Mass Spectrometry (GC-MS). MetaBox identifies and quantifies metabolites described in a mass spectral library, and generates reports in formats that facilitate further steps of data analysis. This document aims to describe few principles behind MetaBox and how to use it in metabolomics studies. The analysis of GC-MS-generated metabolomics datasets using MetaBox can be divided in 3 steps: library building, data processing and report filtering. Bellow you can find a detailed description of each step. We recommend reading the article "Aggio et. al (2014) Identifying and quantifying metabolites by scoring peaks of GC-MS data. BMC Bioinformatics", which contains more details about the algorithm behind MetaBox.

1.1 Metabolomics

Metabolomics is a new omics related technology that aims to study the impact of environmental and/or genetic perturbations in the metabolome - the complete set of small molecules ($\leq$ 1000 Da), or metabolites, present in a biological samples. Metabolites are intermediates of biochemical reactions and their concentrations depend mostly on the levels and properties of enzymes. Therefore, the levels of metabolites result from a complex function involving many regulatory processes occurring inside of the cells (e.g. regulation of transcription, translation and protein-protein interaction). In the hierarchical organization of omics technologies, metabolomics is located at the bottom of the list. Consequently, it concentrates the changes along the flow of information from gene expression and represents the most downstream effect of genetic or environmental perturbations. The profile of metabolites is then a result of the interaction of

the cell's genome with its environment and, ultimately, the level of metabolites represents the phenotype of a biological system. When combined to metabolic networks, the level of metabolites is considered key information to understand the regulation of cell's metabolism, which makes metabolomics one of the most relevant omics-technology applied to systems biology studies.

The identification and measurement of metabolites per se is not something developed by metabolomics. Back in 1776, metabolites present in the urine started to be identified by Matthew Dobson. Since then, identification and measurement of metabolites has been largely applied in medicine and today it is part of the daily routine of hospitals and clinics for diagnostics. The innovation brought by metabolomics was the approach of analyzing several metabolites at the same time and comparing their levels under different experimental conditions. This approach allows detecting changes in localized regions of the cell's metabolism (i.e. gene and metabolic pathways) in response to environmental or genetic perturbations.

The ultimate goal of metabolomics is to quantify all of the metabolites in a cellular system in a given state and at a given point in time. However, the high number of metabolites and their enormous diversity make it virtually impossible. Sample preparation processes such as metabolite extraction - obtaining metabolites present inside of the cells - and chemical derivatization - turning metabolites non-volatiles into volatile ones - are quite specific to a determined class of metabolites (e.g. organic acids). Therefore, the different methods applied during sample preparation determine the main class of metabolites reported by metabolomics experiments.

1.2 Gas Chromatography - Mass Spectrometry

GC-MS is a hyphenated analytical technique that combines gas chromatography (GC) and mass spectrometry (MS) to analyze metabolites present in a mixture of chemicals. The GC separates metabolites in a mixture, while the MS fragments and detects each of these metabolites. When a mixture of metabolites is injected into the GC-MS, it firstly enters the GC, which is composed of a capillary column (stationary phase) inside a computer-controlled oven. The capillary column is made of chemicals able to selectively attract metabolites in a mixture while the oven vaporizes metabolites injected into the capillary column. Entering the GC, vaporized metabolites are then carried through the capillary column by a mobile phase, which is generally an inert gas such as helium. While travelling through the capillary column, metabolites are separated according to their interaction with the stationary phase, where metabolites with weaker interactions travel faster.

Eluting from the GC, metabolites then enter the ion source. One of the main functions of the ion source is the ionization of metabolites. For that, the most common ionization techniques are called electron impact ionization (EI) and electrospray ionization (ESI). When using EI, metabolites enter the ion source and are bombarded with a stream of electrons that break them into ion mass fragments (IMFs). These fragments

eventually reach the recorder, a computer that registers the mass to charge ratio (m/z) and the current or intensity of each fragment. When using EI, a metabolite is expected to always fragment in the same manner and the intensity of its IMFs tends to always show the same proportion between them. For example, a compound X generates the IMFs 102 m/z, 59 m/z, 178 m/z, and 132 m/z; and the intensities of 59 m/z, 178 m/z and 132 m/z in relation to the IMF 102 m/z are nearly always 0.673, 0.342 and 0.073, respectively. The fragmentation pattern of each compound is stored in a mass spectrum, a list of mass to charge ratios with their respective intensities (Figure 1).

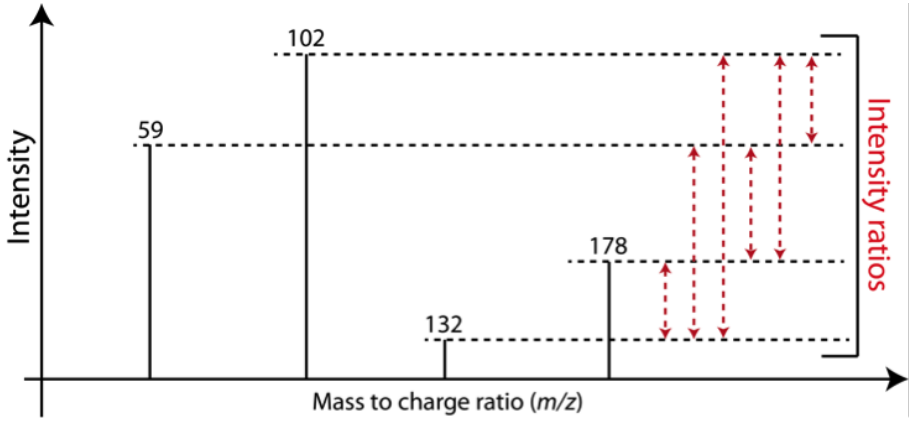

Figure 1: Mass spectrum. Each metabolite analyzed by gas chromatography - mass spectrometry is represented by a mass spectrum, which describes the fragmentation pattern of each metabolite and the intensity ratio between mass fragments.

In addition, a metabolite analyzed by GC-MS takes a specific time to travel through the gas-chromatography column, which is called retention time (RT). Although environmental conditions (e.g. humidity, the age of the capillary column, etc.) and other factors may shift the RT of each compound, a specific metabolite is expected to show approximately the same RT when analyzed by GC-MS. Therefore, each compound can be identified by its fragmentation pattern and its approximate RT. As described above, when a compound elutes or is released from the capillary column, it is bombarded by electrons and broken into fragments that eventually reach the recorder. However, not every molecule of this specific compound elutes from the capillary column at the same time; molecules start to be released at low and increasing quantities until it reaches its maximum concentration followed by a trail with the last few molecules. Consequently, if one plots a chromatogram - the intensity detected for a fragment or a group of fragments plotted in relation to time, each metabolite will be ideally represented by a single peak (Figure 2).

If the different metabolites in a mixture are well separated when travelling through the GC, a single peak of the chromatogram is likely to contain all fragments of a single compound. However, if the GC separation is not efficient, different metabolites coelute - elute from the capillary column simultaneously - and fragments from these metabo-

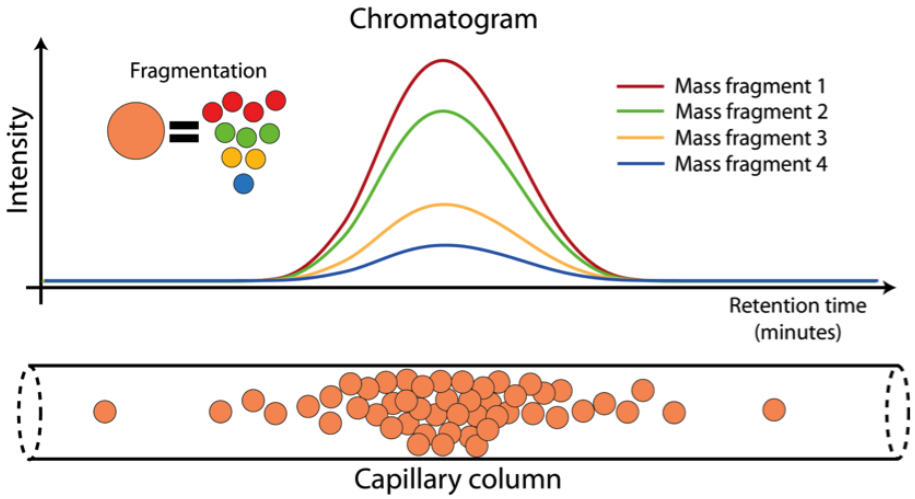

Figure 2: The capillary column and the gas chromatography - mass spectrometry (GC-MS) chromatogram. Molecules travel through the capillary column of the gas chromatography and start eluting at low and increasing concentration until they reach the maximum concentration followed by a trail with the last few molecules. These molecules are then fragmented and finally reach the recorder, which ultimately generates the chromatogram. The peak represented in the chromatogram is a result of the number of molecules from the same metabolite elute from the capillary column per time.

lites will reach the recorder about the same time. Consequently, a single peak of the chromatogram will contain fragments originated from different metabolites, which considerably difficult their identification. In these cases, a process called deconvolution is applied, where fragments of each metabolite are analyzed separately in order to distinguish multiple metabolites represented in the same peak.

2 Requirements

MetaBox requires 4 additional R packages. These packages are xcms, svDialogs, pander and MassSpecWavelet. You can install these packages directly from R using:

```
install.packages(c("xcms", "svDialogs", "pander", "MassSpecWavelet"))
```

or

```
source("http://bioconductor.org/biocLite.R") biocLite(c("xcms", "svDialogs", "pander", "MassSpecWavelet"))
```

Alternatively, you can download them from the CRAN website (http://cran.r-project.org/) or from the Bioconductor website (www.bioconductor.org).

Table 1: This table shows an example of the mass spectral library required by MetaBox, which contains each standard compound's name (Compound), its expected RT (E_{RT}) in minutes, the m/z ratio of its four (generally) most IMFs (M_1, M_2, M_3 and M_4) and the relative intensities (R_2, R_3 and R_4) to that of M_1.

Compound	ERT	M1	M2	M3	M4	R2	R3	R4
ethanol	6.644	31	45	46	29	0.777	0.343	0.249
acetone	7.373	43	58	42	39	0.262	0.076	0.044
isopropyl alcohol	7.582	45	43	41	39	0.177	0.103	0.077
acetonitril	7.905	41	40	39	38	0.546	0.223	0.137
ethyl acetate	10.593	43	45	70	61	0.137	0.116	0.105
1-butanol	13.381	56	41	43	31	0.72	0.543	0.346
2-pentanone	13.959	43	86	41	71	0.249	0.127	0.109
pyridine	16.426	79	52	51	50	0.564	0.275	0.205
zylene1	20.395	91	106	77	51	0.327	0.08	0.077
zylene2	20.697	91	106	105	77	0.533	0.223	0.115
zylene3	21.803	91	106	105	77	0.488	0.189	0.109
benzaldehyde	25.712	106	105	77	51	0.99	0.935	0.404
indole	38.634	117	90	89	63	0.414	0.313	0.103

3 Library building

The compound identification and quantification performed by MetaBox is based on a mass spectral library, which we will call here as ionLib. The ionLib contains, for each compound: its name, its expected retention time, the m/z value of its four most abundant fragments and the intensity ratios between these fragments in relation to the most abundant fragment. See Table 1 for an example of ionLib.

The ionLib can be manually built as a comma separated values (.csv) format file or it can be built using the Automated Deconvolution and Identification System - AMDIS. Any Excel-like software or even text editors can be used to manually build the ionLib, while the function `buildLib` converts an existing AMDIS library to the format required by MetaBox. If you already have your own mass spectral library built using AMDIS, the

function `buildLib` allows you to use MetaBox with AMDIS-built existing mass spectral libraries.

3.1 Manual library building

MetaBox has no function to automatically identify analytes in the absence of an ionLib. Therefore, manually building an ionLib using only MetaBox can be considerably time-consuming if compared to building an ionlib through AMDIS and then applying the `buildLib` for converting it to the MetaBox format. Thus, if you have access to AMDIS (for the identification of anlytes one must also have access to the NIST mass spectral database (http://chemdata.nist.gov/mass-spc/amdis/)), we strongly encourage you to use them instead of manually building. If you have no access to the NIST database, you may have to use other databases to identify the analytes or peaks found in the analyzed samples. The future versions of MetaBox may include functions to facilitate library building with no need for AMDIS and NIST. Although MetaBox has no function to automatically identify analytes in the absence of an ionLib, it does contain a function, `plotIons`, to obtain the mass spectrum and RT of desired peaks. The function `plotIons` can be used to build an ionLib using provisional names for peaks or analytes. Then, in a further step, the peaks considered important for obtaining the biological interpretation (e.g. peaks present at significantly different levels across experimental conditions) can be searched against a mass spectral database. The function `plotIons` collects the intensity values of the ion mass fragments present at a specific RT window, plot them in a new window and returns the mass spectrum associated with a specific time point in the chromatogram.

The function ***`plotIons`*** *can be applied following the steps bellow:*

- 1 - Load the MetaBox library;

```
library(MetaBox)
```

- 2 - Before applying the function `plotIons` in the R console, it is important that you already know the values desired for each of its arguments. Thus, we recommend you to define the values that best suit your analysis. The possible values for the arguments of the function `plotIons` are listed bellow:
  - data = this argument may receive the path to the GC-MS file in CDF format to be analyzed (e.g. /Users/You/Project_1/Cond1/File1.CDF). Alternatively, this argument may be left empty, which will result in a pop up dialog box allowing the user to select the desired CDF file to be analyzed.
  - ions = this argument must receive the specific mass-to-charge ratio values of the fragments you wish to be highlighted in the plot, if any. For building a

library, you may not know yet the mass values of the most abundant fragments. In this case, you can leave the argument ions empty, set the argument plotAbundantIons to TRUE and use the argument numberOfIons to define the number of most abundant fragments to be shown in the plot.

- RT = this argument must receive two numerical values, one with the initial RT value to be analyzed and another with the final RT value. For example, if you are looking for peaks in the chromatogram region between 10 and 11 minutes, the RT argument will receive the value c(10,11). If you need a shorter time window, define RT = c(10.5, 11), for example.
- yscale = this argument must receive two numerical values to define the minimun and maximun values to be shown in the Y axis. For example, if yscale = c(0, 200), the Y axis will be showing fragments with intensity between 0 and 200. This argument can be very useful to visualyse peaks closer to the noise level. The default value of yscale is to set the Y axis from 0 to the maximun TIC value in the selected RT window.
- color = this argument must receive the name of the color to be used for coloring the ion mass fragments that will be highlighted in the plot.
- plotGraph = when this argument is set to TRUE, the default value, a plot with the chromatogram is plot. If plotGraph = FALSE, no chromatogram is plot. This option is useful if you are only interested in the values of the ion mass fragments during the RT window defined through the argument RT.
- plotAbundantIons = if this argument is TRUE, the most abundant ion mass fragments during the RT window defined will be highlighted.
- numberOfIons = this argument must receive the number of ion mass fragments to be highlighted in the plot, in addition to the ions defined through the argument ions.
- save = if save = TRUE, a csv file containing the intensities of the fragments present in the RT window defined through the argument RT.
- folder = this argument defines the path to the folder where the csv file generated by `plotIons` will be stored.
- output = this argument defines the name of the csv file that will be generated if save = TRUE.

- 3 - Once you know the values of the arguments of the function `plotIons` you can apply this function in the R console. For this, follow the code bellow:

```
fragmentValues <- plotIons(ions = c(define here the ions you want to highlight or leave it empty), RT = c(define here the time window))
```

- 4 - If save = TRUE, a new window will pop up allowing you to choose the folder where the results will be saved. Select the folder and click on choose;

- 5 - A new window will open allowing you to select the CDF file to be analyzed. Select the file and click on choose;

- 6 - A new window will pop up showing you the chromatogram and the desired number of fragments (See Figure 3);

- 7 - Here comes the most interesting part. When you click on the top of a peak or any other region of the chromatogram, a new window will pop up with the mass spectrum associated with the RT where you clicked (See Figure 4). If you click in a new region of the chromatogram or a different peak, the mass spectrum shown in the other window will be updated with the mass spectrum associated with the new region. When you decided that the last clicked region is the best mass spectrum to represent a peak, you can click on any region bellow the X axis and the results will be generated. As default, the function `plotIons` generates a dataframe containing the intensities of the fragments present during the RT window defined and another dataframe containing the mass spectrum at the clicked region. The dataframe containing the mass spectrum shows the mass-to-charge ratios of the fragments as row.names and the first column contains their relative values and their rations. The fragment showing the relative value = 1 is the most abundant fragment. MetaBox provides you with the mass spectrum associated with a peak, however, we can not provide you with the identification or potential identifications for this peak, as MetaBox does not include any mass spectral database at the moment. Therefore, a mass spectral database must be used in order to obtain a definitive identification of the mass spectrum and its associated peak.

- 8 - Use *?plotIons* in the R console to see more details about this function.

IMPORTANT: The ionLib must be built using the chromatograms or CDF files as they are going to be further analyzed by MetaBox. For example, if any noise removal processes is applied to a chromatogram at any stage before compound identification and quantification is performed by MetaBox, these processes must be applied to this chromatogram before building the ionLib. This advice applies to libraries built manually or using AMDIS and NIST. Actually, this advice applies to any software applied to the identification and quantification of metabolites analyzed by GC-MS data. It is required because the information contained in the CDF files must reflect the mass spectra present in the mass spectral library. If a library is built before noise removal, the proportion between fragments in the ionLib will not match the proportion of these fragments in the sample or CDF file.

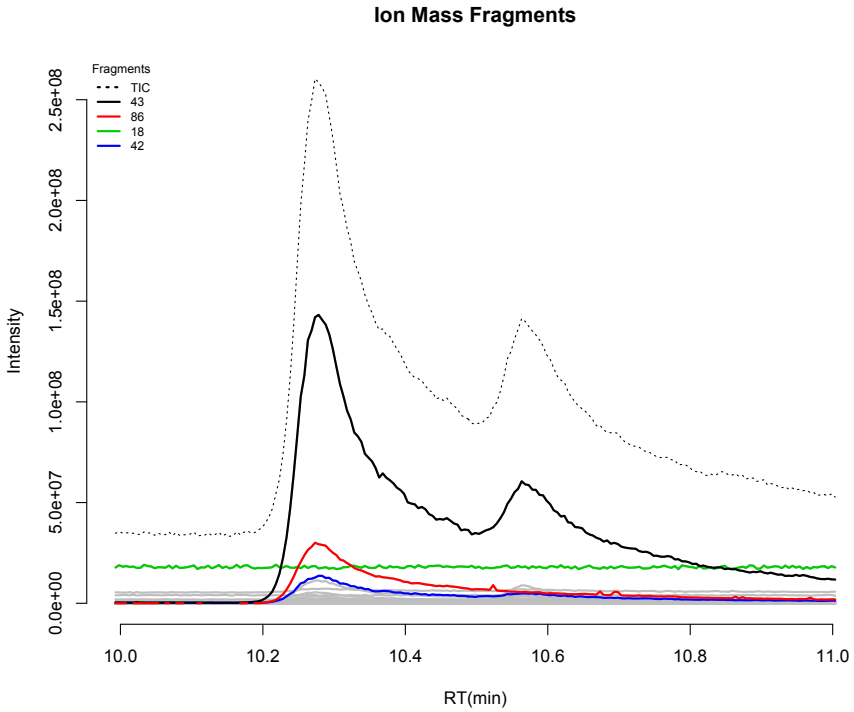

Figure 3: An example of the plot produced by the function `plotIons`.

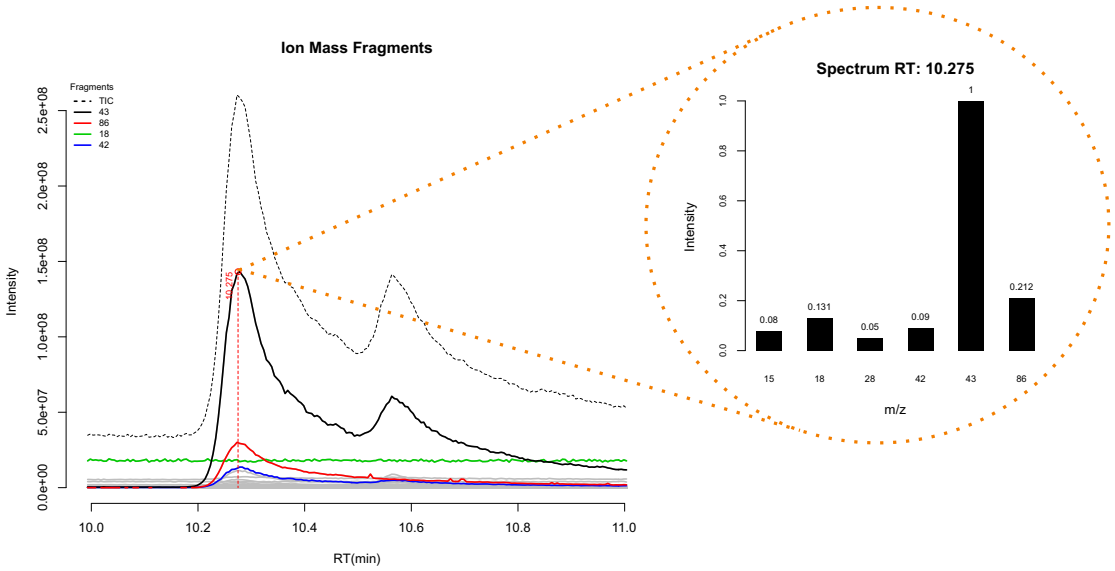

Figure 4: An example of the plot produced by the function `plotIons`.

3.2 Converting an AMDIS library

The user manual that accompanies the AMDIS installation files contains all the required information to build a library, therefore, we will not cover this topic in this document.

An AMDIS-generated library is composed of two files: a .msl file and a .CID file. For converting an AMDIS library to the format required by MetaBox, the function `buildLib` can be applied using the following code in the R console:

- 1 - Load the MetaBox library, if not yet loaded;

```
library(MetaBox)
```

- 2 - Apply the function `buildLib` to convert the AMDIS library and store it in a R object called `MyNewLibrary` (feel free to modify the name of the R object):

```
MyNewLibrary <- buildLib()
```

IMPORTANT: Storing the converted library in an R object is not a requirement. It will be saved to a .csv file if the argument save = TRUE, which is the default behavior of `buildLib`.

- 3 - A new window called "Select the .msl file of the AMDIS library in use" will open.
- 4 - Select the .msl file from the desired AMDIS library and click on choose.
- 5 - A new window called "Select the folder where the output file will be saved" will open.
- 6 - Select the folder to save the new MetaBox library and click on Choose.
- 7 - A new file called ion_lib.csv will be saved in the folder indicated above. In addition, the converted library will be stored in the variable MyNewLibrary.
- 8 - Use *?buildLib* in the R console to see more details about this function.

4 Identifying compounds in a metabolomics project: `matchSpectra`

The function `matchSpectra` is the main function of MetaBox. `matchSpectra` compares the spectrum of each metabolite present in the ionLib against specific RT windows of each GC-MS sample in analysis, and generates two reports containing the identified compounds and their respective abundance in the different GC-MS samples analyzed. For this, the function `matchSpectra` uses a system of points to predict which specific retention time (RT) of the GC-MS samples is most likely to be associated to each mass spectrum present in the ionLib. For each metabolite in the ionLib, the function `matchSpectra` gives points to RTs in the GC-MS sample based on five criteria: the number of reference ion mass fragments showing a peak at the same RT (total of 4 points); the number of reference ions showing a positive intensity at this RT (total of 4 points); the number of reference ions showing the expected intensity in relation to the most abundant reference ion [IMPORTANT: a similarity threshold can be defined by the user to indicate how similar the intensity of expected and observed ion mass fragments must be to be considered a positive identification] (total of 3 points); the number of reference ions showing a positive correlation [IMPORTANT: a correlation threshold can be specified by the user in order to define the minimum value of the Pearson's correlation coefficient to define a positive correlation] (total of 3 points); and the number of reference ions having their highest intensity at a specific RT (total of 3 points). The maximum score of a RT is then 18 score points. If two different RTs show the maximum score for a specific metabolite in the ionLib, the most similar RT value in relation to the expected RT (the RT defined in the ionLib for this specific compound) is then selected as the best RT to represent this metabolite. The ion mass fragment defined in the ionLib as the most abundant fragment, or M1, is used for quantification. The intensity of M1 is obtained as the abundance or intensity of a metabolite. See "Aggio et. al (2014) Identifying and quantifying metabolites by scoring peaks of GC-MS data. BMC Bioinformatics, doi:10.1186/s12859-014-0374-2." to a description of the steps performed by `matchSpectra`.

4.1 Preparing the data to use the function `matchSpectra`

The function `matchSpectra` was developed to automatically identify the experimental conditions associated to each GC-MS sample under analysis. For this, the GC-MS samples you are willing to analyze must be converted to CDF format and organized in folders named according to their respective experimental conditions. The ideal way for organization the GC-MS samples before applying `matchSpectra` is: (1) create a folder and give it a name that represents the project involved with the samples to be analyzed. For example, you can name it as Project_yeast. This folder will be the main folder of your analysis. It is where all your samples related to the Project_yeast will be located; (2) save your ionLib inside of the main folder, in this case Project_yeast;

(3) now, inside of this main folder, create one additional folder for each experimental condition. For example, one folder named Experimental_condition_1 and another folder named Experimental_condition_2; (4) then, allocate each GC-MS file you are willing to analyze into the folder named after its respective experimental condition. See bellow a description of how GC-MS samples must be organized:

- Project_yeast
  - Experimental_condition_1
    - Sample1.CDF
    - Sample2.CDF
    - Sample3.CDF
  - Experimental_condition_2
    - Sample4.CDF
    - Sample5.CDF
    - Sample6.CDF
  - Experimental_condition_3
    - Sample7.CDF
    - Sample8.CDF
    - Sample9.CDF

After the GC-MS samples are organized as above, you can proceed to the compound identification using the function `matchSpectra`. All the results produced by `matchSpectra` will be stored in the main folder, or Project_yeast in the example above.

IMPORTANT: GC-MS samples must be converted to CDF format. MetaBox does not recognize any other format of GC-MS files. Most GC-MS equipment include a software able to convert results to CDF format.

IMPORTANT: Organizing the GC-MS files as described above is not a requirement. Alternatively, you can store all the CDF files in a single folder and apply the function `matchSpectra`. In this case, a dialog box will pop up allowing you to select the number of experimental conditions under analysis and the CDF files belonging to each experimental condition.

4.2 Applying the function `matchSpectra`

If the GC-MS samples were organized as suggested in the subsection above, you can apply the function `matchSpectra` following the steps bellow:

- 1 - Load the MetaBox library, if not yet loaded;

```
library(MetaBox)
```

- 2 - Define the values of the arguments of the function `matchSpectra`. You can see all the arguments of `matchSpectra` using the command *?matchSpectra* in the R console. Here we will describe the arguments we consider most important for applying `matchSpectra`, however, we strongly recommend visualizing the help file obtained with *?matchSpectra*:
  - dataFolder = this argument must receive the path to the main folder. For example, dataFolder = "/Users/You/Project_yeast". If dataFolder is set as default, a dialog box will allow you to select the main folder or the folder containing all the CDF files.
  - ionLib = this argument must receive the path to a CSV file containing the ionLib or an R object containing the ionLib. If ionLib is set as default, a dialog box will pop up allowing you to select the CSV or the MSL file containing the ionLib.
  - searchWindow = this argument must receive a numeric value which will be used to calculate the time window used to identify each metabolite in the ionLib. For example, if searchWindow = 0.2 and the expected RT of a metabolite X in the ionLib is 10 minutes, the metabolite X will be searched by `matchSpectra` in the RT window going from 9.8 (10 - 0.2) to 10.2 (10 + 0.2). The more reproducible or repeatable the GC-MS platform in use is, the smaller the searchWindow value will be.
  - matchFactor = this argument must receive a numerical value going from 0 to 0.99. It indicates proportion of similarity between expected ratios, values in the ionLib, and observed ratios of fragments, values in the CDF sample. For example, if matchFactor = 0.7, observed and expected spectra must be at least 70% similar to be considered as a potential identification. The spectra similarity is calculated for each fragment in the spectrum. For example, if a spectrum contains the fragments 54, 74, 102 and 134, the intensity ratio of fragments 74 and 54, or 74/54 must be at least 70% similar to the value R2 present in the ionLib to generate a positive score. The same applies to the ratios R3 and R4, or 102/54 and 134/54.
  - correlation = this argument must receive a numerical value going from 0 to 0.99. One of the stages in the `matchSpectra` algorithm is the calculation of the correlation between ion mass fragments. The argument correlation defines the minimum correlation coefficient to generate a positive score.
  - peakFindMethod = this argument must receive a numerical value of 1, 2 or 3. Each numerical value represents one specific algorithm for detecting peaks

in a chromatogram. You may try different methods in order to find which works the best with your samples.

- scoreCut = this argument must receive a numerical value from 0 to 18. During the analysis performed by the function `matchSpectra`, it gives scores to RT potentially representing metabolites present in the ionLib. Then, two reports are produced: (1) a report containing all the scored RTs and their respective potential identities; and (2) another report containing only the compounds that were most likely present in the analyzed samples. These compounds are the compounds associated to RTs that received scores higher or equal to the value defined in scoreCut. For example, if scoreCut = 13, only compounds associated with RTs that receive 13 or more score points will be reported.

IMPORTANT: Several graphs will be produced for each sample analyzed by `matchSpectra`. The argument saveGraphs may be used to disable this feature.

- 3 - Once the best values for each argument has been defined, you can apply the function `matchSpectra` in the R console:

```
project1 <- matchSpectra(edit here with your values for each argument)
```

- 4 - If dataFolder = default value, a new window called "Select the folder containing the GC-MS data in CDF format (NetCDF or AIA)" will open;

- 5 - Select the Main_folder (the folder containing the subfolders for each experimental condition or Project_yeast in the example above) and click on Choose;

- 6 - A new window called "Select a CSV file containing the ionLib or the .msl file of the AMDIS library in use." will open;

- 7 - Select an ionLib file and click on Choose;

- 8 - The GC-MS files will be analyzed using the defined arguments of `matchSpectra` and, according to their values, PDF files containing three different types of graphs will be saved in the main folder. In addition, two reports and one log file containing the results will be generated and stored in the main folder.

4.3 Analyzing the results produced by `matchSpectra`

The function `matchSpectra` generates two reports, one log file and one graph for each compound in the ionLib, depending on its arguments values. One of the reports generated by `matchSpectra` is called MatchReportTotal. It is stored as a CSV file in the main folder (the Project_yeast of our example) and contains for each metabolite identified in each sample analyzed: the RT where the metabolite was identified, the difference between expected RT (the RT defined in the ionLib) and observed RT (the RT where the compound was actually identified) associated with this metabolite, its score calculated by `matchSpectra` and the intensity of the main reference ion mass fragment (M1 in the ionLib) (See Table 2).

Table 2: MatchReportTotal.csv. `matchSpectra` generates a report called MatchReportTotal that is stored in a CSV file. This report contains the names of identified metabolites and for each sample: the Retention Time (RT) where each metabolite was identified, the difference between expected (the RT defined in the ionLib used) and real RT (the RT where the metabolite was identified), the score calculated by MetaBox and the intensity of the ion used as main reference ion (M1 in the ionLib used).

Name	RT_Sample1	DiffRT_Sample1	Score_Sample1	Sample1
Replicates	100uL	100uL	100uL	100uL
ethanol	6.64612	0.002	14	23259136
acetone	7.36968	0.003	16	247545856
isopropyl alcohol	7.57977	0.002	16	82120704
acetonitril	7.9007	0.004	15	86769664
ethyl acetate	10.59667	0.004	16	340213760
1-butanol	13.38012	0.001	18	169279488
2-pentanone	13.95783	0.001	17	369115136
pyridine	16.43788	0.012	16	766640128
zylene1	20.39425	0.001	15	29983744
zylene2	20.6977	0.001	18	86278144
zylene3	21.81225	0.009	17	44974080
benzaldehyde	25.71023	0.002	15	534659072
indole	38.63548	0.001	18	157777920

The second report produced by `matchSpectra` is called MatchReportCutOff. It is also stored as a CSV file in the main folder. However, the MatchReportCutOff contains only those metabolites showing a calculated score equal or higher than the value defined

in the scoreCut argument applied. (See Table 3). It is organized with the names of the identified metabolites in the first column and their abundances in each analyzed sample in the subsequent columns.

Table 3: MatchReporCutOff.csv. `matchSpectra` generates a report called MatchReportCutOff that is stored in a CSV file. This report is a shortlist of the MatchReportTotal.csv. For every metabolite showing a score equal or above the cutOff score applied with `matchSpectra`, MatchReportCutOff contains metabolite's names in the first column and the intensity of the main reference ion (M1 in the ionLib) in each analyzed sample in the following columns.

Name	Sample1	Sample2	Sample3	Sample6	Sample7	Sample8
Replicates	100uL	100uL	100uL	50uL	50uL	50uL
ethanol	23259136	24012800	24326144	11761664	13939712	15432704
acetone	247545856	285147136	271532032	140296192	183844864	211861504
isopropyl alcohol	82120704	75304960	82141184	40357888	53235712	48267264
acetonitril	86769664	96366592	94244864	47382528	60628992	65150976
ethyl acetate	340213760	437469184	427343872	201703424	242745344	316309504
1-butanol	169279488	176668672	181108736	34881536	51818496	76873728
2-pentanone	369115136	412483584	419954688	195510272	231981056	314114048
pyridine	766640128	858587136	843120640	369508352	415711232	482574336
zylene1	29983744	53858304	41267200	25606144	27378688	35684352
zylene2	86278144	141459456	118910976	75735040	79917056	100077568
zylene3	44974080	82784256	66351104	44204032	43917312	62398464
benzaldehyde	534659072	603521024	589234176	142163968	163987456	207470592
indole	157777920	165347328	181731328	70967296	80273408	86499328

The log file generated by `matchSpectra` contains the values of the parameters used when analyzing samples. It stores the path to the mainFolder analyzed, the path to the ionLib used, the values of the searchWindow, matchFactor, correlation, scoreCut and peakFindMethod applied among others when analyzing samples. In addition, `matchSpectra` generates a line graph for each metabolite identified with a score higher or equal to the cutOff value applied. This line graph presents a frame or slice of the sample's chromatogram where each metabolite was identified. It contains the RT plotted in X and the intensities of the TIC and the 4 reference ions associated with each compound (M1, M2, M3 and M4 in the ionLib) plotted in Y (Figure 5).

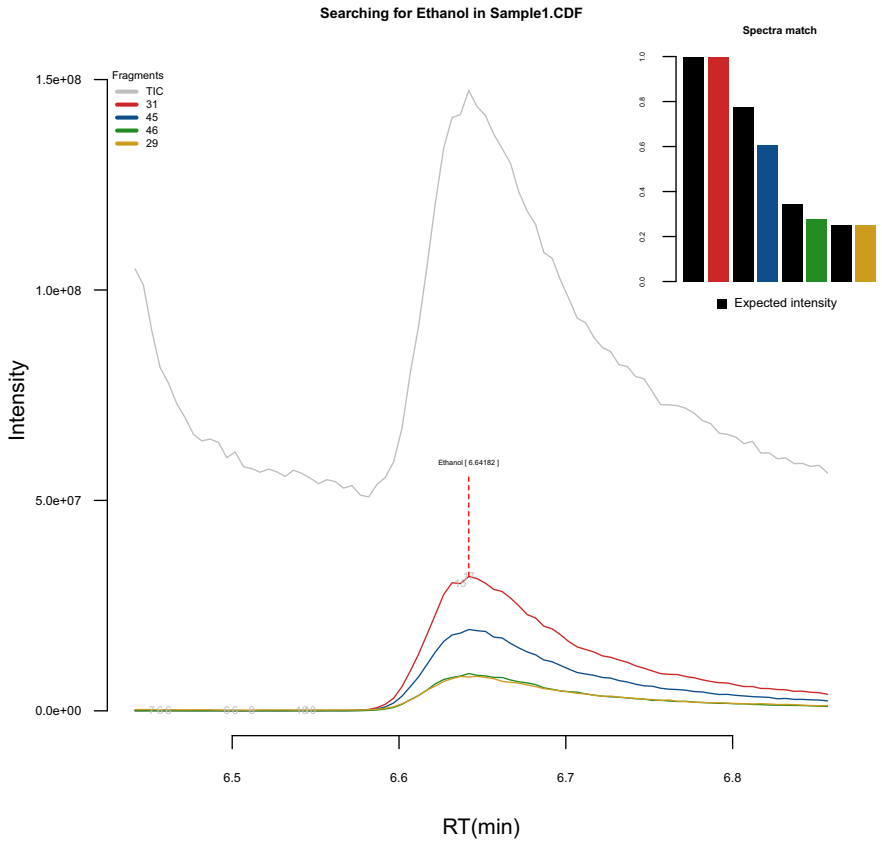

Figure 5: Sample_compound.pdf. For each compound identified, `matchSpectra` generates a line graph showing the exactly point where the metabolite was identified and the intensities of the TIC and the 4 ion mass fragments (M1, M2, M3 and M4 in the ionLib) associated to each compound.

5 Filtering the MatchReportTotal using the function `filterTotalReport`

The analysis performed by the function `matchSpectra` may be time-consuming depending on the number of samples to be analyzed and the number of metabolites in ionLib. Commonly, users must verify if the scoreCut value applied fits with the dataset analyzed by applying different cutOff values and comparing their results. However, re-analyzing the whole dataset may take too long. Thus, we created the function `filterTotalReport`, which allows users to obtain results using different values of scoreCut without having to re-analyze the whole dataset. For this, the function `filterTotalReport` filters an existing MatchReportTotal.csv file and generates a new report containing the names of identified metabolites in the first column and the intensity of their respec-

tive main ion mass fragments (M1 in ionLib) in each analyzed sample in the subsequent columns. Basically, `filterTotalReport` allows users to re-generate MatchReportCutOff files using different values of scoreCut. The CSV file produced by `filterTotalReport` has the same format as MatchReportCutOff.csv (Table 3). For applying the function `filterTotalReport`, follow the steps bellow:

- 1 - Load the MetaBox library, if not yet loaded:

```
library(MetaBox)
```

- 2 - Apply the function `filterTotalReport` selecting the desired scoreCut value:

```
newReport <- filterTotalReport()
```

- 3 - A new window called "Select the CSV file containing the input data" will open;
- 4 - Select the existing MatchReportTotal.csv file generated by the function `matchSpectra` and click on Choose.
- 5 - A new window called "Select the cut off score" will open.
- 6 - Select the new scoreCut value to be used and click on OK.
- 7 - A new window called "Select the folder where the output file will be saved" will open.
- 8 - Select the folder where the results will be saved and click on Choose.
- 9 - A new CSV file names "filteredReport.csv" will be saved in the chosen folder.

When applying `filterTotalReport` as described above, the default parameters will be used. Alternatively, the user may modify any of the 5 arguments used by `filterTotalReport`:

- inputData = a character value defining the path to the CSV file MatchReportTotal.csv or to a data frame containing the report MatchReportTotal.csv.
- scoreCut = a numeric value used to define the value used as cutOff or scoreCut. Compounds that received a lower score than the value defined in scoreCut will not be selected to the results (default = open a dialog box allowing the user to select the score to be used).

- save = if TRUE, a CSV file will be generated in the directory defined in folder (default = TRUE).
- folder = a character value defining the path to the directory where the CSV file containing the results will be saved (default = open a dialog box allowing the user to point and click on the selected directory).
- output = a character value defining the name of the report that will be generated (default = "filteredReport").

```
> print(sessionInfo(), locale = FALSE)

R version 3.1.1 (2014-07-10)
Platform: x86_64-apple-darwin13.1.0 (64-bit)

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] MetaBox_1.0.0          MassSpecWavelet_1.30.0 waveslim_1.7.3
 [4] pander_0.3.8           svDialogs_0.9-55       svGUI_0.9-55
 [7] xcms_1.40.0            Biobase_2.24.0         BiocGenerics_0.10.0
[10] mzR_1.10.7             Rcpp_0.11.2

loaded via a namespace (and not attached):
[1] codetools_0.2-8 digest_0.6.4    tools_3.1.1
```