An R package to process LC/MS metabolomic data: MAIT (Metabolite Automatic Identification Toolkit)

Francesc Fernández-Albert
Polytechnic University of Catalonia
University of Barcelona

Rafael Llorach
University of Barcelona

Cristina Andrés-Lacueva
University of Barcelona

Alexandre Perera
Polytechnic University of Catalonia

Abstract

Processing metabolomic liquid chromatography and mass spectrometry (LC/MS) data files is time consuming. Currently available R tools allow for only a limited number of processing steps and online tools are hard to use in a programmable fashion. This paper introduces the metabolite automatic identification toolkit MAIT package, which allows users to perform end-to-end LC/MS metabolomic data analysis. The package is especially focused on improving the peak annotation stage and provides tools to validate the statistical results of the analysis. This validation stage consists of a repeated random sub-sampling cross-validation procedure evaluated through the classification ratio of the sample files. MAIT also includes functions that create a set of tables and plots, such as principal component analysis (PCA) score plots, cluster heat maps or boxplots. To identify which metabolites are related to statistically significant features, MAIT includes a metabolite database for a metabolite identification stage.

Keywords: Metabolomics, Peak Aggregation Measures, LC/MS.

1. Introduction

Liquid Chromatography and Mass Spectrometry (LC/MS) is an analytical technique widely used in metabolomics to detect molecules in biological samples (Theodoridis, Gika, Want, and Wilson 2012). It breaks the molecules down into pieces, some of which are detected as peaks in the mass spectrometer. Metabolic profiling of LC/MS samples basically consists of a peak detection and signal normalisation step, followed by multivariate statistical analysis such as Principal Components Analysis (PCA) and a wide range of statistical tests such as ANOVA, Welch’s test or Kruskal-Wallis test (Theodoridis et al. 2012; Tulipani, Llorach, Jáuregui, López-Uriarte, Garcia-Aloy, Bullo, Salas-Salvadó, and Andrés-Lacueva 2011).
As analysing metabolomic data is time consuming, a wide array of software tools are available, including commercial tools such as Analyst® software. There are programmatic R packages, such as XCMS (Smith, Want, O’Maille, Abagyan, and Siuzdak 2006; Tautenhahn, Böttcher, and Neumann 2008; Benton, Want, and Ebbels 2010) to detect peaks or CAMERA package (Kuhl, Tautenhahn, and Neumann 2011) and AStream (Alonso, Julia, Beltran, Vinaixa, Díaz, Ibañez, Correig, and Marsal 2011), which cover only peak annotation. Other modularly-designed proposals coded in JAVA such as MZmine or mzMatch are also available (Katajamaa, Miettinen, and Oresic 2006; Pluskal, Castillo, Villar-Briones, and Oresic 2010; Scheltema, Jankevics, Jansen, Swertz, and Breitling 2011). These tools are mainly focused on LC/MS data pre-processing and visualisation. Another category of free available tools consists of those having online access through a graphical user interface (GUI), such as XCMS Online (http://xcmsonline.scripps.edu) or MetaboAnalyst (Xia, Psychogios, Young, and Wishart 2009), both extensively used. These online tools are difficult to use in a programmable fashion. They are also designed and programmed to be used step by step with user intervention, making it difficult to set up metabolomic data analysis workflow.

We introduce a new R package called metabolite automatic identification toolkit (MAIT) for automatic LC/MS analysis. The goal of the MAIT package is to provide an array of tools for programmable metabolomic end-to-end analysis. It consequently has special functions to improve peak annotation through the processes called biotransformations. Specifically, MAIT is designed to look for statistically significant metabolites that separate the classes in the data. MAIT has the following dependencies in terms of R packages: pls, plots, e1071, caret, plsgenomics and agricolae.

2. Available Computational Tools

Table 1 contains a capability comparison between MAIT and some of the most widely used computational tools when processing LC/MS metabolomic data. Among the programable tools, there are R packages such as XCMS which is focused on preprocessing the data but it also is able to perform a simple statistical analysis of the data. MZmine or mzMatch are highly modularised tools based on JAVA and also centred on the data preprocessing and visualisation methods than in the statistical processing of the LC/MS data. On the other hand, the main approach of the online tool MetaboAnalyst is on the statistical analysis of the LC/MS data but it lacks of the programmable approach and off-line capabilities of the previous tools. In this context, MAIT aims at filling the gap of programmatic tools that allow for a full statistical study of LC/MS metabolomic data.
Table 1: Comparison of some of the main available computational tools for analysing LC/MS data.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Environment</th>
<th>Programmatic</th>
<th>Statistical Tests</th>
<th>Predictive power methods</th>
<th>Other Statistical Methods and functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>XCMS</td>
<td>R package</td>
<td>Yes</td>
<td>Parametric tests: ANOVA, Welch's t test</td>
<td>—</td>
<td>Metabolite Identification</td>
</tr>
<tr>
<td>mzMine</td>
<td>Java</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>Clustering Methods Heatmap plots Biotransformations/Neutral losses Annotation Projection Plots (PCA, PLS) Peak Annotation Metabolite Identification</td>
</tr>
<tr>
<td>mzMatch</td>
<td>Java</td>
<td>Yes</td>
<td>—</td>
<td>—</td>
<td>Peak Annotation Metabolite Identification Metabolite Quantification Supports PeakML format</td>
</tr>
<tr>
<td>metaboAnalyst</td>
<td>R-based Online Tool</td>
<td>No</td>
<td>Parametric tests: ANOVA, T-test, SAM, EBAM</td>
<td>PLSDA, SVM, SOM, Random Forest</td>
<td>Clustering Methods Heatmap plots Projection Plots (PCA, PLS) Pathway Analysis Metabolite Identification Support for Analysing External Peak Data</td>
</tr>
<tr>
<td>MAIT</td>
<td>R package</td>
<td>Yes</td>
<td>Parametric tests: ANOVA, Welch's T test, Student's T test Non-parametric tests: Kruskal-Wallis test Mann-Whitney test Supports user-defined tests</td>
<td>PLSDA, SVM, KNN</td>
<td>Heatmap plots Projection Plots (PCA, PLS) Peak Annotation Biotransformations/Neutral losses Annotation Metabolite Identification Support for Peak Aggregation Methods Support for Analysing External Peak Data</td>
</tr>
</tbody>
</table>
3. MAIT Modularity

Modularity is a highly desirable property of any software tool. A modular package is made of functions each one of which performs highly specific tasks. These functions are used by the package as building blocks to perform more complex procedures. Under the modular design of MAIT, the functions of the package operate with objects of a characteristic class named MAIT-class objects. The main functions of the MAIT workflow fill certain slots of the object and then return the updated MAIT-class object as an output. In this context, Table 2 shows the slots of the MAIT-class objects that are necessary to run each of the main MAIT functions and also the slots that are filled after the function run. From the same table, it is shown that just a few slots are necessary to run the functions. In particular, considering all the main functions, the required slots are four: @PhenoData@classes, @PhenoData@classNum, FeatureInfo@featureSigID and one of the following slots @RawData@data or @FeatureData@extendedTable. If a certain module is to be added, it is only necessary to fill the slots with the appropriate data. Function getScoresTable returns a peak scores table generated from a xsAnnotate object (see documentation of CAMERA package) if available (it comes from the peakAnnotation function) to extract the peak data. If there is no data in the slot, the table saved in @FeatureData@extendedTable is taken instead. New modules for peak detection and peak annotation stages (see Figure 1) to be implemented in the MAIT workflow, should create an overload of the function getScoresTable to extract the appropriate data (see help file of the function getScoresTable). Another option would be to use the MAITbuilder function instead (Sections 5.5 and 6.9).
Table 2: Slots of the MAIT-class object filled for each step. Optional slots are labelled with an asterisk.

<table>
<thead>
<tr>
<th>Step</th>
<th>Function</th>
<th>Input Slots</th>
<th>MAIT-class object Slots filled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Detection</td>
<td>(sampleProcessing())</td>
<td>Raw Sample files</td>
<td>@RawData@data: Object containing the raw peak data as an xcms-Set-object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@PhenoData@classes: Vector containing the class names.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@PhenoData@classNum: Vector containing the number of samples for each class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@PhenoData@resultsPath: Character containing the path where the results are going to be saved.</td>
</tr>
<tr>
<td>Peak Annotation</td>
<td>(peakAnnotation())</td>
<td>@RawData@data @PhenoData@classes* @PhenoData@classNum*</td>
<td>@RawData@data &lt;- Object containing the peak annotated data.</td>
</tr>
<tr>
<td>Statistical Analysis</td>
<td>(spectralSigFeatures())</td>
<td>@PhenoData@classes @PhenoData@classNum @FeatureData@extendedTable @RawData@data</td>
<td>@FeatureData@pvalues &lt;- Vector containing the p-values of the tests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@FeatureData@LSDResults &lt;- Vector containing the results of the Fisher tests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@FeatureData@pvaluesCorrection &lt;- Name of the post hoc method used (if any)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@FeatureData@scores &lt;- Full dataset having rows as variables and columns as samples.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@FeatureData@featureSigID &lt;- Numeric vector with the found significant variables for the dataset saved in the slot @FeatureData@scores</td>
</tr>
<tr>
<td>Biotransformations</td>
<td>(Biotransformations())</td>
<td>@FeatureData@featureSigID @FeatureData@extendedTable @RawData@data</td>
<td>@FeatureInfo@biotransformations: Matrix containing the biotransformations found.</td>
</tr>
<tr>
<td>Metabolite Identification</td>
<td>(identifyMetabolites())</td>
<td>@FeatureInfo@biotransformations* @FeatureData@extendedTable @RawData@data</td>
<td>@FeatureInfo@metaboliteTable: Dataframe containing the results of the metabolite identification and the statistical analysis.</td>
</tr>
<tr>
<td>Predictive Model</td>
<td>(Validation())</td>
<td>@PhenoData@classes @PhenoData@classNum @FeatureData@extendedTable @RawData@data</td>
<td>@Validation@classifRatioClasses: The Classification Ratio per class, classifier and iteration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>@Validation@ovClassifRatioTable: Table with the mean and standard error of the Classification Ratio per classifier</td>
</tr>
<tr>
<td>PLS Model/Plots</td>
<td>(plotPLS())</td>
<td>@PhenoData@classes @PhenoData@classNum @FeatureData@extendedTable @RawData@data</td>
<td>@FeatureData@plsModel: The PLS model of the statistically significant data is sassed in this slot.</td>
</tr>
<tr>
<td>PCA Model/Plots</td>
<td>(plotPLS())</td>
<td>@PhenoData@classes @PhenoData@classNum @FeatureData@extendedTable @RawData@data</td>
<td>@FeatureData@pcaModel: The PCA model of the statistically significant data is sassed in this slot.</td>
</tr>
</tbody>
</table>
4. Methodology

The main processing steps for metabolomic LC/MS data include the following stages: peak detection, peak annotation and statistical analysis. In the peak detection stage, the objective is to detect the peaks in the LC/MS sample files. The peak annotation stage identifies the metabolites in the metabolomic samples better by increasing the chemical and biological information in the data set. A statistical analysis step is essential to obtain significant sample features. All these 3 steps are covered in the MAIT workflow (see Figure 1).

4.1. Peak Detection

Peak detection in metabolomic LC/MS data sets is a complex issue for which several approaches have been developed. Two of the most well established techniques are matched filter (Danielsson, Bylund, and Markides 2002) and the centWave algorithm (Tautenhahn et al. 2008). MAIT can use both algorithms through the XCMS package.

4.2. Peak Annotation

The MAIT package uses 3 complementary steps in the peak annotation stage.

- The first annotation step uses a peak correlation distance approach and a retention time window to ascertain which peaks come from the same source metabolite, following the procedure defined in the CAMERA package (Kuhl et al. 2011). The peaks within each peak group are annotated following a reference adduct/fragment table and a mass tolerance window.

- The second step uses a mass tolerance window inside the peak groups detected in the first step to look for more specific mass losses called biotransformations. To do this, MAIT uses a predefined biotransformation table where the biotransformations we want to find are saved. A user-defined biotransformation table can be set as an input following the procedure defined in Section (6.6).

- The third annotation step is the metabolite identification stage, in which a predefined metabolite database is mined to search for the significant masses, also using a tolerance window. This database is the Human Metabolome Database (HMDB) (Wishart, Knox, Guo, Eisner, Young, Gautam, Hau, Psychogios, Dong, Bouatra, and et al. 2009), 2009/07 version.

4.3. Statistical Analysis

The objective of analysing metabolomic profiling data is to obtain the statistically significant features that contain the highest amount of class-related information. To gather these features, MAIT applies standard parametrical and non-parametrical statistical tests on every feature and selects the significant set of features by setting up a user-defined threshold P-value. Depending on the number of classes defined in the data, MAIT can use Student’s T-test, Welch’s T-tests and Mann-Whitney tests for two classes or ANOVA and Kruskal-Wallis tests for more than two classes. Furthermore, MAIT supports adding user-defined tests in a straightforward way (see section 6.4 for an example using the Fisher’s exact test).
Different multiple testing correction methods including false discovery rate and Bonferroni are implemented in **MAIT** through R function `p.adj`.

We propose a validation test to quantify how well the data classes are separated by the statistically significant features. The separation is validated through a repeated random subsampling cross-validation using partial least squares and discriminant analysis (PLS-DA), support vector machine (SVM) with a radial Kernel and K-nearest neighbours (KNN) (Hastie, Tibshirani, and Friedman 2003). Overall and class-related classification ratios are obtained to evaluate the class-related information of the significant features.

### 4.4. Support for Peak Aggregation Techniques

**MAIT** optionally supports peak aggregation techniques that might lead to better feature selection (Fernández-Albert, Llorach, Andrés-Lacueva, and Perera-Lluna 2014) through the commercial **pagR** package.

### 5. MAIT workflow

**MAIT** accepts LC/MS files in the open formats mzData and netCDF. Sample files should be placed in a folder having a set of subfolders, each of which is going to be a class in the data (see function `sampleProcessing()` in Section 6 for details).

The package centrepiece consists of the S4 **MAIT**-class objects. In terms of traceability, objects belonging to this class are designed to contain all the information related to the processing steps already run. The reason for this design is that using a single R object throughout the workflow improves the traceability of the analysis. The contents of a **MAIT**-class object are shown below. The slots of the **MAIT**-class objects are:

```r
Formal class 'MAIT' [package "MAIT"] with 5 slots
  ..@ FeatureInfo:Formal class 'MAIT.FeatureInfo' [package "MAIT"] with 3 slots
    .. ..@ biotransformations: logi [1, 1] NA
    .. ..@ peakAgMethod : chr ""
    .. ..@ metaboliteTable : 'data.frame': 0 obs. of 0 variables
    ..@ RawData :Formal class 'MAIT.RawData' [package "MAIT"] with 2 slots
      ..@ parameters:Formal class 'MAIT.Parameters' [package "MAIT"] with 10 slots
        .. ..@ sampleProcessing : list()
        .. ..@ peakAnnotation : list()
        .. ..@ peakAggregation : list()
        .. ..@ sigFeatures : list()
        .. ..@ biotransformations : list()
        .. ..@ identifyMetabolites: list()
        .. ..@ classification : list()
        .. ..@ plotPCA : list()
        .. ..@ plotPLS : list()
        .. ..@ plotHeatmap : list()
        ..@ data : list()
```
MAIT (Metabolite Automatic Identification Toolkit)

..@ Validation : Formal class 'MAIT.Validation' [package "MAIT"] with 3 slots
   .. .. ..@ ovClassifRatioTable: logi [1, 1] NA
   .. .. ..@ ovClassifRatioData : list()
   .. .. ..@ classifRatioClasses: logi [1, 1] NA
..@ PhenoData : Formal class 'MAIT.PhenoData' [package "MAIT"] with 3 slots
   .. .. ..@ classes : logi(0)
   .. .. ..@ classNum : logi(0)
   .. .. ..@ resultsPath: chr ""
..@ FeatureData: Formal class 'MAIT.FeatureData' [package "MAIT"] with 12 slots
   .. .. ..@ scores : logi [1, 1] NA
   .. .. ..@ featureID : logi(0)
   .. .. ..@ featureSigID : logi(0)
   .. .. ..@ LSDResults : logi [1, 1] NA
   .. .. ..@ models : list()
   .. .. ..@ pvalues : logi(0)
   .. .. ..@ pvaluesCorrection: chr ""
   .. .. ..@ pcaModel : list()
   .. .. ..@ plsModel : list()
   .. .. ..@ masses : num(0)
   .. .. ..@ rt : num(0)
   .. .. ..@ extendedTable : 'data.frame': 0 obs. of 0 variables

A MAIT-class object is built of 5 different S4 classes:

- **FeatureInfo-class**: The information regarding the peak annotation is saved in this class.
- **RawData-class**: This class contains the data imported from the metabolomic LC/MS (xcmsSet-class object or xsAnnotate-class object depending on the last function run)
- **Validation-class**: This contains the results of the cross-validation classification stage.
- **PhenoData-class**: All the class-related information and the results path is contained in this class.
- **FeatureData-class**: This class contains the information related to the features, its P-values and the mathematical models used.

Figure 1 shows the flowchart of the main functions of the MAIT package, their output files and their functionality. Table 3 shows the specific outputs of each function shown in Figure 1.

The MAIT package uses the wrapper function sampleProcessing() to call the required XCMS functions to perform the peak detection step. These functions include xcmsSet(), group(), retcor() and fillPeaks(). The peaks detected are saved as a xcmsSet-class object inside a MAIT-class object.
5.1. Peak Annotation

The default tables used to perform all the peak annotation steps are provided in MAIT as an R Data object called MAITtables.RData. When this file is loaded, the following objects can be found:

- **posAdducts**: The possible annotations for the first annotation step when the polarisation mode in the sample acquisition is set to positive.
- **negAdducts**: The possible annotations for the first annotation step when the polarisation mode in the sample acquisition is set to negative.
- **biotransformationsTable**: This table contains the specific biotransformations for the second annotation step.
- **Database**: The metabolite database table to perform the metabolite identification stage (third peak annotation step). This database is the Human Metabolome Database (HMDB)(Wishart et al. 2009), 2009/07 version.

The MAIT package uses a CAMERA package wrapper function called peakAnnotation() to perform the first step in the peak annotation stage. CAMERA groups the peaks using a retention time window followed by a correlation cut-off approach. An adduct table is required to launch this step. A user-defined adduct table or a MAIT default adduct table (posAdducts or negAdducts) can be selected. The user-defined table should be created following the CAMERA adduct table layout, which is:

```
name nmol charge massdiff oidscore quasi ips
1  [M+H]+  1  1  1.007276  1  1  1.0
2  [M+Na]+  1  1  22.989218  8  1  1.0
3  [M+K]+  1  1  38.963158 10  1  1.0
4  [M+NH4]+  1  1  18.033823 16  1  1.0
5  [M+2Na-H]+  1  1  44.971160 34  0  0.5
6  [M+2K-H]+  1  1  76.919040 60  0  0.5
```

The second peak annotation step is performed by the function called Biotransformations(). The function is codified to perform the procedure defined in Section 4.2. As is shown in Figure 1, function Biotransformations() should be launched after detecting the significant features using function spectralSigFeatures() (see Section 5.2). The first 10 entries of the Biotransformation Table are shown below. When 2 peaks in the same peak group have mass differences (within tolerance) equal to a value of the MASSDIFF column, they are related to each other by that biotransformation and are annotated accordingly.

```
NAME MASSDIFF
1  debenzylation -90.0468
2  tert-butyl dealkylation -56.0624
```
Likewise, to perform the third peak annotation step, function `identifyMetabolites()` mines the metabolite database file to find suitable metabolites for each peak. The function outputs a table (see Table 3) that contains all the possible matches for all the peaks. If no peak aggregation technique was applied through function `peakAggregation()` (see Section 5.3), the set of features to be identified are all the significant features found in the statistical tests (Section 5.2). A user-defined database can be used as an input object as well. To do so, the user file should have the following format:

<table>
<thead>
<tr>
<th>ENTRY</th>
<th>NAME</th>
<th>FORMULA</th>
<th>MASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HMDB00001 1-Methylhistidine</td>
<td>C7H11N3O2</td>
<td>169.085129</td>
</tr>
<tr>
<td>2</td>
<td>HMDB00002 1,3-Diaminopropane</td>
<td>C3H10N2</td>
<td>74.084396</td>
</tr>
<tr>
<td>3</td>
<td>HMDB00005 2-Ketobutyric acid</td>
<td>C4H6O3</td>
<td>102.031693</td>
</tr>
<tr>
<td>4</td>
<td>HMDB00008 2-Hydroxybutyric acid</td>
<td>C4H8O3</td>
<td>104.047340</td>
</tr>
<tr>
<td>5</td>
<td>HMDB00010 2-Methoxyestrone</td>
<td>C19H24O3</td>
<td>300.172546</td>
</tr>
<tr>
<td>6</td>
<td>HMDB00011 (R)-3-Hydroxybutyric acid</td>
<td>C4H8O3</td>
<td>104.047340</td>
</tr>
</tbody>
</table>

Each of these 3 annotation steps is implemented through a function. These 3 functions all have an input parameter where a user-defined table can be used instead of the MAIT default tables. In particular, in function `peakAnnotation()` there is the argument `adductTable`, in function `Biotransformations()`, the argument is called `bioTable` and the input argument for function `identifyMetabolites()` is called `database`.

5.2. Statistical Analysis

`spectralSigFeatures()` performs a univariate statistical test on each feature to gather the statistically significant variables that separate the classes in the data. The results of these statistical tests are saved in the MAIT-class object and are easily retrieved from it by applying function `sigPeaksTable()`. The validation procedure defined in Section 4.3 is launched
using function \texttt{Validation()}. The overall and class-related classification ratios are saved in boxplots and tables (see Table 3) in the folder called "Validation". The confusion matrices for each iteration and classifier are saved in the folder named "Confusion_Tables".

5.3. Support for Peak Aggregation Techniques

The peak aggregation techniques, optional in \texttt{MAIT} workflow, are applied through function \texttt{peakAggregation()}. This function allows the use of several different methods to obtain the peak aggregation measures. If the chosen method is \texttt{None}, no other packages are required and no peak aggregation technique is applied. Any other valid choice (\texttt{Mean, Single, PCA, NMF}) requires the additional commercial package \texttt{pagR} (see Figure 1).

5.4. Statistical Plots

The package also contains functions that create statistical plots to evaluate analysis results. These plots include 2D PCA score plots and an interactive 3D PCA score plot through function \texttt{plotPCA()}. The interactive 3D PCA score plot is generated by the package \texttt{rgl} (Adler and Murdoch 2012). Function \texttt{plotHeatmap()} produces an array of heat maps using different thresholds for the P-values and hierarchical clustering distances (Euclidean and Pearson’s; see Table 3), whereas Function \texttt{plotBoxplot()} makes it possible to create a boxplot for each significant feature found. As is shown in Figure 1, all 3 functions require the significant features to be found to run the functions correctly and create the plots.
Figure 1: Flowchart showing the main MAIT functions. Each box refers to a function and each circle points to the functionality of the function in the workflow. Solid arrows refer to possible data processing path. The left column plots contain the output of the functions.
Table 3: Table showing the output files generated by the main MAIT functions shown in Figure 1.

<table>
<thead>
<tr>
<th>MAIT Function</th>
<th>Output File Name</th>
<th>Output type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>peakAnnotation</td>
<td>Spectra.csv</td>
<td>Table</td>
<td>This table summarises the correspondence between peaks and spectra.</td>
</tr>
<tr>
<td>spectralSigFeatures</td>
<td>significantFeatures.csv</td>
<td>Table</td>
<td>In this table the results of the univariate tests performed for every feature and the information of the peak annotation are saved.</td>
</tr>
<tr>
<td>identifyMetabolites</td>
<td>metaboliteTable.csv</td>
<td>Table</td>
<td>This table summarises the results of all the previous functions in the workflow (see Figure 1), including peakAnnotation, spectralSigFeatures and Biotransformations. The possible metabolite identification matches are also included in the table.</td>
</tr>
<tr>
<td>plotPCA</td>
<td>Scoreplot_PC12.png</td>
<td>Plot</td>
<td>This file contains the PCA score plot for Principal Component 1 vs Principal Component 2.</td>
</tr>
<tr>
<td></td>
<td>Scoreplot_PC13.png</td>
<td>Plot</td>
<td>This file contains the PCA score plot for Principal Component 1 vs Principal Component 3.</td>
</tr>
<tr>
<td></td>
<td>Scoreplot_PC23.png</td>
<td>Plot</td>
<td>This file contains the PCA score plot for Principal Component 2 vs Principal Component 3.</td>
</tr>
<tr>
<td>plotPLS</td>
<td>Scoreplot_PLS_(Ncomps).png</td>
<td>Plots</td>
<td>Depending on the number of components found, this function generates one PLS Scoreplot (1 and 2 components) or three PLS Scoreplots (3 components)</td>
</tr>
<tr>
<td>plotHeatmap</td>
<td>X_Distance_Heatmap_pY.png</td>
<td>Plots</td>
<td>This set of files contain the heat maps after applying a hierarchical clustering using X distance (X=Euclidean or Correlation) and Y P-value (Y=0.05, 0.01, 0.001, 1e-4, 1e-5)</td>
</tr>
<tr>
<td>plotBoxplot</td>
<td>Boxplot_spectra_X.png</td>
<td>Plots</td>
<td>This set of files contain a boxplot for each significant feature found in the analysis.</td>
</tr>
<tr>
<td>Validation</td>
<td>Confusion_Tables</td>
<td>Folder</td>
<td>Folder where the Confusion matrices for every iteration step are saved.</td>
</tr>
<tr>
<td></td>
<td>Boxplot.Classes.Classification.png</td>
<td>Plot</td>
<td>Boxplot showing the classification ratio for each class and classifier.</td>
</tr>
<tr>
<td></td>
<td>Boxplot.Overall_Classification.png</td>
<td>Plot</td>
<td>Boxplot showing the classification ratio for classifier regardless of the classes.</td>
</tr>
<tr>
<td></td>
<td>ClassificationTable.Class_X.csv</td>
<td>Tables</td>
<td>Table showing the classification ratios for each classifier and for class X. There one of these tables for each class in the data.</td>
</tr>
<tr>
<td></td>
<td>ClassificationTable.csv</td>
<td>Table</td>
<td>Table showing the overall classification ratios for each classifier regardless of the classes.</td>
</tr>
</tbody>
</table>
Table 4: Correspondence between the necessary arguments of the MAITbuilder and the MAIT functions that can be launched. Given a function, the arguments not mentioned in the should be considered as optional for that function. The argument significantFeatures is a flag that, if it is set to TRUE, the provided features are considered to be statistically significant. A field labelled with an asterisk refers to an optional argument.

<table>
<thead>
<tr>
<th>MAIT function to be launched</th>
<th>Necessary arguments of the MAITbuilder function</th>
</tr>
</thead>
<tbody>
<tr>
<td>spectralSigFeatures()</td>
<td>classes, data</td>
</tr>
<tr>
<td>Biotransformations()</td>
<td>masses, significantFeatures=TRUE, spectraID*</td>
</tr>
<tr>
<td>identifyMetabolites()</td>
<td>masses, significantFeatures=TRUE, spectraID*</td>
</tr>
<tr>
<td>Validation()</td>
<td>classes, data, significantFeatures=TRUE</td>
</tr>
<tr>
<td>Plot functions (plotBoxplot, plotHeatmap, plotPCA, plotPLS)</td>
<td>classes, data, significantFeatures=TRUE</td>
</tr>
</tbody>
</table>

5.5. External Peak Data

MAIT supports importing external peak data through a function called MAITbuilder. This function allows the user to create a MAIT object from a wide variety of data. Table 4 shows the correspondence between the arguments that the user needs to provide to the MAITbuilder function and the function that the user wants to run. An important point of the MAITbuilder function is the spectraID argument. Whereas this argument is not necessary to run any of the functions related to the statistical processing, it has a big impact on the annotation functions (i.e. Biotransformations and identifyMetabolites). The reason is that, if no spectral information is provided and the flag named spectraEstimation is set to FALSE, the MAITbuilder function considers the provided data as being all in separate spectra (one peak/one spectrum). Therefore the annotation functions will not find any annotation for the provided data. Nevertheless, if the spectraEstimation flag is set to TRUE, MAIT uses a retention time window (defined by the argument rtRange) and a correlation threshold value (defined by corThresh) to estimate a peak grouping into spectra for the provided data.

6. Using MAIT

The data files for this example are a subset of the data used in reference (Saghatelian, Trauger, Want, Hawkins, Siuzdak, and Cravatt 2004), which are freely distributed through the faahKO package Smith (2012). In these data there are 2 classes of mice: a group where the fatty acid amide hydrolase gene has been suppressed (class knockout or KO) and a group of wild type mice (class wild type or WT). There are 6 spinal cord samples in each class. In the following, the MAIT package will be used to read and analyse these samples using the main functions discussed in Section 5. The significant features related to each class will be found using statistical tests and analysed through the different plots that MAIT produces.
6.1. Data Import

Each sample class file should be placed in a directory with the class name. All the class folders should be placed under a directory containing only the folders with the files to be analysed. In this case, 2 classes are present in the data. An example of correct file distribution using the example data files is shown in Figure 2.

6.2. Peak Detection

Once the data is placed in 2 subdirectories of a single folder, the function `sampleProcessing()` is run to detect the peaks, group the peaks across samples, perform the retention time correction and carry out the peak filling process. As function `sampleProcessing()` uses the `XCMS` package to perform these 4 processing steps, this function exposes `XCMS` parameters that might be modified to improve the peak detection step. A project name should be defined because all the tables and plots will be saved in a folder using that name. For example, typing `project = "project_Test"`, the output result folder will be "Results_project_Test".

By choosing "MAIT_Demo" as the project name, the peak detection stage can be launched by typing:

```
R> MAIT <- sampleProcessing(dataDir = "Dataxcms", project = "MAIT_Demo", snThres = 2, rtStep = 0.03)
```

```
```
... 

575:8302 590:8792 
Peak detection done 
262 325 387 450 512 575 
Retention Time Correction Groups: 7 

Warning: Span too small, resetting to 0.8 

Retention time correction done 
262 325 387 450 512 575 
Peak grouping after samples done 
ko15 
. . . 
Peak missing integration done 

After having launched the `sampleProcessing` function, peaks are detected, they are grouped across samples and their retention time values are corrected. A short summary in the R session can be retrieved by typing the name of the `MAIT-class` object.

R> MAIT 
A MAIT object built of 12 samples 
The object contains 6 samples of class KO 
The object contains 6 samples of class WT 

The result is a `MAIT-class` object that contains information about the peaks detected, their class names and how many files each class contains. A longer summary of the data is retrieved by performing a summary of a `MAIT-class` object. In this longer summary version, further information related to the input parameters of the whole analysis is displayed. This functionality is especially useful in terms of traceability of the analysis.

R> summary(MAIT) 
A MAIT object built of 12 samples 
The object contains 6 samples of class KO 
The object contains 6 samples of class WT 

Parameters of the analysis: 

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataDir</td>
</tr>
<tr>
<td>&quot;Data&quot;</td>
</tr>
</tbody>
</table>
6.3. Peak Annotation

The next step in the data processing is the first peak annotation step, which is performed through the \texttt{peakAnnotation()} function. If the input parameter \texttt{adductTable} is not set, then the default \texttt{MAIT} table for positive polarisation will be selected. However, if the \texttt{adductTable} parameter is set to "negAdducts", the default \texttt{MAIT} table for negative fragments will be chosen instead. \texttt{peakAnnotation} function also creates an output table (see Table 3) containing the peak mass (in charge/mass units), the retention time (in minutes) and the spectral ID number for all the peaks detected. A call of the function \texttt{peakAnnotation} may be:

\begin{verbatim}
R> MAIT <- peakAnnotation(MAIT.object = MAIT, corrBetSamp = 0.75, perfwhm = 0.6)
\end{verbatim}

WARNING: No input adduct/fragment table was given. Selecting default MAIT table for positive polarity...
Set adductTable equal to negAdducts to use the default MAIT table for negative polarity
Start grouping after retention time.
Created 1037 pseudospectra.
Spectrum build after retention time done
Generating peak matrix!
Run isotope peak annotation
% finished: 10 20 30 40 50 60 70 80 90 100
Found isotopes: 15
Isotope annotation done
Start grouping after correlation.
Generating EIC's..

Calculating peak correlations in 1037 Groups...
% finished: 10 20 30 40 50 60 70 80 90 100

Calculating peak correlations across samples.
% finished: 10 20 30 40 50 60 70 80 90 100

Calculating isotope assignments in 1037 Groups...
% finished: 10 20 30 40 50 60 70 80 90 100
Calculating graph cross linking in 1037 Groups...
% finished: 10 20 30 40 50 60 70 80 90 100
New number of ps-groups: 2398
xsAnnotate has now 2398 groups, instead of 1037
Spectrum number increased after correlation done
Generating peak matrix for peak annotation!
Found and use user-defined ruleset!
Calculating possible adducts in 2398 Groups...
% finished: 10 20 30 40 50 60 70 80 90 100
Adduct/fragment annotation done

Because the parameter adductTable was not set in the peakAnnotation call, a warning was shown informing that the default MAIT table for positive polarisation mode was selected. The xsAnnotated object that contains all the information related to peaks, spectra and their annotation is stored in the MAIT object. It can be retrieved by typing:

R> rawData(MAIT)
$xsaFA
An "xsAnnotate" object!
With 2398 groups (pseudospectra)
With 12 samples and 2640 peaks
Polarity mode is set to: positive
Using automatic sample selection
Annotated isotopes: 15
Annotated adducts & fragments: 16
Memory usage: 7.07 MB

6.4. Statistical Analysis

Following the first peak annotation stage, we want to know which features are different between classes. Consequently, we run the function spectralSigFeatures().

R> MAIT<-spectralSigFeatures(MAIT.object = MAIT, pvalue = 0.05, p.adj = "none", scale = FALSE)
It is worth mentioning that by setting the `scale` parameter to `TRUE`, the data will be scaled to have unit variance. The parameter `p.adj` allows for using the multiple testing correction methods included in the function `p.adjust` of the package `stats`. A summary of the statistically significant features is created and saved in a table called `significantFeatures.csv` (see Table 3). It is placed inside the Tables subfolder located in the project folder. This table shows characteristics of the statistically significant features, such as their P-value, the peak annotation or the expression of the peaks across samples. This table can be retrieved at any time from the `MAIT`-class objects by typing the instruction:

```r
R> signTable <- sigPeaksTable(MAIT.object = MAIT, printCSVfile = FALSE)
R> head(signTable)
```

| mz     | mzmin | mzmax | rt   | rtmin | rtmax | npeaks | KO  | WT  | ko15 | ...
|--------|-------|-------|------|-------|-------|--------|-----|-----|------|-------
| 610    | 300.2 | 300.2 | 56.36| 56.18 | 56.56 | 17     | 6   | 3   | 4005711.4 | ...
| 762    | 326.2 | 326.2 | 56.92| 56.79 | 57.00 | 9      | 5   | 2   | 3184086.4 | ...
| 885    | 348.2 | 348.2 | 56.95| 56.79 | 57.15 | 14     | 4   | 2   | 320468.2  | ...
| 1760   | 495.3 | 495.3 | 56.93| 56.82 | 57.05 | 11     | 3   | 4   | 110811.4  | ...
| 935    | 356.2 | 356.3 | 63.77| 63.58 | 63.92 | 9      | 4   | 4   | 962224.6  | ...
| 1259   | 412.2 | 412.3 | 68.61| 68.44 | 68.81 | 16     | 4   | 3   | 113096.3  | ...

<table>
<thead>
<tr>
<th>isotopes</th>
<th>adduct</th>
<th>pcgroup</th>
<th>P.adjust</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>610</td>
<td>27</td>
<td>0.01748294</td>
<td>0.01748294</td>
<td>...</td>
</tr>
<tr>
<td>762</td>
<td>[M+H]+</td>
<td>325.202</td>
<td>0.01991433</td>
<td>0.01991433</td>
</tr>
<tr>
<td>885</td>
<td>[M+Na]+</td>
<td>325.202</td>
<td>0.16856322</td>
<td>0.16856322</td>
</tr>
<tr>
<td>1760</td>
<td>31</td>
<td>0.96828618</td>
<td>0.96828618</td>
<td>...</td>
</tr>
<tr>
<td>935</td>
<td>74</td>
<td>0.03310409</td>
<td>0.03310409</td>
<td>...</td>
</tr>
<tr>
<td>1259</td>
<td>81</td>
<td>0.02240898</td>
<td>0.02240898</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Median Class KO</th>
<th>Median Class WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>610 2769931.356</td>
<td>115642.29</td>
</tr>
<tr>
<td>762 2353947.791</td>
<td>43006.61</td>
</tr>
<tr>
<td>885 40384.825</td>
<td>0.00</td>
</tr>
<tr>
<td>1760 6531.515</td>
<td>15969.26</td>
</tr>
<tr>
<td>935 848999.980</td>
<td>16836.67</td>
</tr>
<tr>
<td>1259 215979.768</td>
<td>34607.95</td>
</tr>
</tbody>
</table>

The number of significant features can be retrieved from the `MAIT`-class object as follows:

```r
R> MAIT
```

A MAIT object built of 12 samples and 2640 peaks.
No peak aggregation technique has been applied
106 of these peaks are statistically significant
The object contains 6 samples of class KO
The object contains 6 samples of class WT

By default, when using two classes, the statistical test applied by `MAIT` is the Welch's test. Nevertheless, when having two classes, `MAIT` also supports applying the Student's t-test and
the non-parametric test Mann-Whitney test. For using the Student’s t-test on the data, the
call to the \texttt{spectralSigFeatures} function should be as:

\begin{verbatim}
R> MAIT_Student <- spectralSigFeatures(MAIT.object = MAIT, pvalue = 0.05,
p.adj = "none", scale = FALSE, var.equal = TRUE)
R> MAIT_Student
A MAIT object built of 12 samples and 2640 peaks.
No peak aggregation technique has been applied
148 of these peaks are statistically significant
The object contains 6 samples of class KO
The object contains 6 samples of class WT
\end{verbatim}

If we want to apply the Mann-Whitney test, in this case is necessary to add some jitter noise
in our data. The reason is that the Mann-Whitney test has ties when the data has values
equal to zero. Adding a small noise to the data solves this issue. \texttt{MAIT} supports using jitter
noise through the flag \texttt{jitter} and the parameter \texttt{jitter.amount}:

\begin{verbatim}
R> MAIT_MW <- spectralSigFeatures(MAIT.object = MAIT, pvalue = 0.05,
p.adj = "none", scale = FALSE, parametric = FALSE, jitter = TRUE)
\end{verbatim}

As an example of its modularity, \texttt{MAIT} supports applying user-defined statistical tests on the
data. To use a user-defined test in \texttt{MAIT}, the output of the function should give the p-value
of the test given a numeric vector (i.e. the variable values) and a factor vector (the classes
of the samples in the parameter \texttt{group}). In the following example, let us suppose that we
want to know whether a certain metabolite is present/absent in a particular group contrast.
To that end, we will define a special exact Fisher’s test function after applying a threshold
intensity value. If the intensity of the peak for a particular sample is above this threshold
value, the peak is labelled as "present". If the intensity value is below the threshold, the peak
is labelled as "absent". If all the labels for a particular peak are the same, the function will
not compute the Fisher’s test and will throw an \texttt{NA} as a p-value. The function of the Fisher’s
test, can be defined as follows:

\begin{verbatim}
R> ftest <- function(x,group){
    threshold<-100
    x[x>threshold]<="present"
    x[!x>threshold]<="absent"
    x<-as.factor(x)
    if(length(summary(x))==1){
        out<-NA
    }else{
        out<-fisher.test(x=x,y=group)$p.value
    }
    return(out)}
\end{verbatim}

And the call to the \texttt{spectralSigFeatures} in this case:
R> MAIT_Fisher <- spectralSigFeatures(MAIT.object = MAIT, test.fun = ftest, namefun = "fisher's test")
R> MAIT_Fisher
A MAIT object built of 12 samples and 2640 peaks.
No peak aggregation technique has been applied
18 of these peaks are statistically significant
The object contains 6 samples of class KO
The object contains 6 samples of class WT

R> head(sigPeaksTable(MAIT_Fisher))

<table>
<thead>
<tr>
<th></th>
<th>mz</th>
<th>mzmin</th>
<th>mzmax</th>
<th>rt</th>
<th>rtmin</th>
<th>rtmax</th>
<th>npeaks</th>
<th>KO</th>
<th>WT</th>
<th>ko15</th>
<th>ko16</th>
<th>ko18</th>
<th>ko19</th>
<th>ko21</th>
<th>ko22</th>
<th>wt15</th>
<th>wt16</th>
<th>wt18</th>
<th>wt19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>686</td>
<td>314.20</td>
<td>314.3</td>
<td>58.34</td>
<td>58.26</td>
<td>58.52</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>53657.59</td>
<td>44311.386</td>
<td>46921.83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>743</td>
<td>323.10</td>
<td>323.2</td>
<td>58.36</td>
<td>58.10</td>
<td>58.70</td>
<td>32</td>
<td>5</td>
<td>6</td>
<td>93579.73</td>
<td>163605.100</td>
<td>269543.46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1879</td>
<td>512.10</td>
<td>512.1</td>
<td>58.36</td>
<td>58.30</td>
<td>58.36</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>20781.25</td>
<td>9272.833</td>
<td>15164.85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2398</td>
<td>572.10</td>
<td>572.2</td>
<td>58.35</td>
<td>58.23</td>
<td>58.54</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>17548.34</td>
<td>0.000</td>
<td>10066.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2425</td>
<td>574.15</td>
<td>574.3</td>
<td>58.36</td>
<td>58.17</td>
<td>58.54</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>0.00</td>
<td>0.000</td>
<td>7615.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2484</td>
<td>582.10</td>
<td>582.2</td>
<td>58.36</td>
<td>58.88</td>
<td>58.75</td>
<td>32</td>
<td>5</td>
<td>6</td>
<td>67578.36</td>
<td>19928.601</td>
<td>20647.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All the peaks in the sigPeaksTable are found to be significant for the user-defined Fisher’s exact test (note that the column named Fisher.test in the sigPeaksTable refers to the Fisher
LSD test performed after an ANOVA test and not to the user-defined Fisher’s exact test). This means that if the peak mass related to the fragmentation of the metabolite we are looking for is found in this table, the metabolite would statistically show a different absence/presence behaviour across the classes (WT/KO).

On the other hand, in the call to the `spectralSigFeatures` function, the argument `test.fun` contains the function of the user-defined test and the argument `namefun` is an optional parameter that contains the name of the user-defined function. This name will appear in the `parameters` slot of the `MAIT`-class object:

```R
R> summary(MAIT_Fisher)
A MAIT object built of 12 samples and 2640 peaks.
No peak aggregation technique has been applied
18 of these peaks are statistically significant
The object contains 6 samples of class KO
The object contains 6 samples of class WT

Parameters of the analysis:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataDir</td>
<td>&quot;Data&quot;</td>
</tr>
<tr>
<td>snThres</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>Sigma</td>
<td>&quot;2.12332257516562&quot;</td>
</tr>
<tr>
<td>mzSlices</td>
<td>&quot;0.3&quot;</td>
</tr>
<tr>
<td>retcorrMethod</td>
<td>&quot;loess&quot;</td>
</tr>
<tr>
<td>groupMethod</td>
<td>&quot;density&quot;</td>
</tr>
<tr>
<td>bwGroup</td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td>mzWidGroup</td>
<td>&quot;0.25&quot;</td>
</tr>
<tr>
<td>filterMethod</td>
<td>&quot;matchedFilter&quot;</td>
</tr>
<tr>
<td>rtStep</td>
<td>&quot;0.03&quot;</td>
</tr>
<tr>
<td>nSlaves</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>project</td>
<td>&quot;MAIT_Demo&quot;</td>
</tr>
<tr>
<td>ppm</td>
<td>&quot;10&quot;</td>
</tr>
<tr>
<td>minfrac</td>
<td>&quot;0.5&quot;</td>
</tr>
<tr>
<td>fwhm</td>
<td>&quot;30&quot;</td>
</tr>
<tr>
<td>family1</td>
<td>&quot;gaussian&quot;</td>
</tr>
<tr>
<td>family2</td>
<td>&quot;symmetric&quot;</td>
</tr>
<tr>
<td>span</td>
<td>&quot;0.2&quot;</td>
</tr>
<tr>
<td>centWave peakwidth1</td>
<td>&quot;5&quot;</td>
</tr>
<tr>
<td>centWave peakwidth2</td>
<td>&quot;20&quot;</td>
</tr>
<tr>
<td>corrWithSamp</td>
<td>&quot;0.7&quot;</td>
</tr>
<tr>
<td>corrBetSamp</td>
<td>&quot;0.75&quot;</td>
</tr>
<tr>
<td>perfwhm</td>
<td>&quot;0.6&quot;</td>
</tr>
<tr>
<td>sigma</td>
<td>&quot;6&quot;</td>
</tr>
<tr>
<td>peakAnnotation pvalue</td>
<td>&quot;0.05&quot;</td>
</tr>
<tr>
<td>calcIso</td>
<td>&quot;TRUE&quot;</td>
</tr>
<tr>
<td>calcCiS</td>
<td>&quot;TRUE&quot;</td>
</tr>
<tr>
<td>calcCaS</td>
<td>&quot;TRUE&quot;</td>
</tr>
</tbody>
</table>
```
The multiple test corrections are also implemented in this case by changing the p.adj argument of the function:

```r
R> MAIT_Fisher <- spectralSigFeatures(MAIT.object = MAIT, test.fun = ftest, namefun = "fisher's test", p.adj = "fdr")
```

Warning message:
```
In spectralSigFeatures(MAIT.object = MAIT, test.fun = ftest, namefun = "fisher's test", : No significative features found with the selected parameters.
```

In this particular case, a warning is thrown as no significant features were found with a false discovery rate-adjusted p-value lower or equal 0.05.

### 6.5. Statistical Plots

Out of 2,402 features, 106 were found to be statistically significant. At this point, several MAIT functions can be used to extract and visualise the results of the analysis. Functions `plotBoxplot`, `plotHeatmap`, `plotPCA` and `plotPLS` automatically generate boxplots, heat maps PCA score plot and PLS score plot files in the project folder when they are applied to a MAIT object (see Table 3).

```r
R> plotBoxplot(MAIT)
R> plotHeatmap(MAIT)
R> MAIT<-plotPCA(MAIT)
R> MAIT<-plotPLS(MAIT)
```

The `plotPCA` and `plotPLS` functions produce MAIT objects with the corresponding PCA and PLS models saved inside. The models, loadings and scores can be retrieved from the MAIT objects by using the functions `model`, `loadings` and `scores`:

```r
R> PLSmodel <- model(x=MAIT, type = "PLS")
R> PCAmodel <- model(x=MAIT, type = "PCA")
R> PLSscores <- scores(x=MAIT,model="PLS")
R> PCAscores <- scores(x=MAIT,model="PCA")
R> PLSlotloadings <- loadings(x=MAIT,model="PLS")
R> PCAloloadings <- loadings(x=MAIT,model="PCA")
```

R> PLSscores
24

MAIT (Metabolite Automatic Identification Toolkit)

Comp 1
1  8.460117
2  8.238226
3  7.465394
4  6.341839
5  4.958885
6  5.887925
7  -6.577803
8  -6.570983
9  -6.660059
10 -6.363424
11 -7.427228
12 -7.752889

attr(,"class")
[1] "scores"

R> PCAscores[,1:3]
   PC1   PC2   PC3
[1,] -8.758728 0.92480221 -6.1406083
[2,] -8.348530 -0.86569846  0.1783953
[3,] -7.570347 0.32825445 -1.6159867
[4,] -6.209758 -0.01281555  3.1104855
[5,] -4.632576 -0.80459247  5.6779015
[6,] -5.757966 -0.47710433  0.8561668
[7,] 6.483476  7.10158291  0.9827710
[8,] 6.508645  0.44504996 -1.2287543
[9,] 6.568818  3.66149693 -0.2422269
[10,] 6.311563 -1.97819990 -0.8625683
[11,] 7.518147 -5.26076372 -0.8812214
[12,] 7.887257 -3.06201203  0.1656458

R> head(matrix(PLSloadings))
   [,1]
 [1,] 0.11179158
 [2,] 0.10718688
 [3,] 0.10167223
 [4,] 0.10124325
 [5,] -0.09481443
 [6,] 0.10828112

R> head(PCAloadings[,1:3])
   PC1   PC2   PC3
 [1,] -0.1129682 0.008376894 -0.14442144
 [2,] -0.1080615 -0.002674411 -0.14786276
 [3,] -0.1027608 -0.006700719 -0.10304058
 [4,] -0.1009138 -0.010796632  0.09038020
 [5,]  0.0950440 -0.212358347 -0.06243794
All the output figures are saved in their corresponding subfolders contained in the project folder. The names of the folders for the boxplots, heat maps and score plots are Boxplots, Heatmaps, PCA_Scoreplots and PLS_Scoreplots respectively. Figures 3 and 4 depict a heat map, a PCA score plot and a PLS score plot created when functions plotHeatmap, plotPCA and plotPLS were launched. Inside the R session, the project folder is recovered by typing:

R> resultsPath(MAIT)

6.6. Biotransformations

Before identifying the metabolites, peak annotation can be improved using the function Biotransformations to make interpreting the results easier. The MAIT package uses a default biotransformations table, but another table can be defined by the user and introduced by using the bioTable function input variable. The biotransformations table that MAIT uses is saved inside the file MAITtables.RData, under the name biotransformationsTable.

R> MAIT <- Biotransformations(MAIT.object = MAIT, peakPrecision = 0.005, adductAnnotation=FALSE)

WARNING: No input biotransformations table was given. Selecting default MAIT table for biotransformations...
WARNING: No input adduct/fragment table was given. Selecting default MAIT table for positive polarity...
Set adductTable equal to negAdducts to use the default MAIT table for negative polarity

% Annotation in progress: 10 20 30 40 60 70 80 90 100

The Biotransformations function can also annotate adducts by setting the flag adductAnnotation as TRUE. This is useful when analysing peak data that come from an external source (i.e. peaks and spectra have not been detected by MAIT).

Building a user-defined biotransformations table from the MAIT default table or adding a new biotransformation is straightforward. For example, let’s say we want to add a new adduct called "custom_biotrans" whose mass loss is 105.

R> data(MAITtables)
R> myBiotransformation<-c("custom_biotrans",105.0)
R> myBiotable<-biotransformationsTable
R> myBiotable[,1]<-as.character(myBiotable[,1])
R> myBiotable<-rbind(myBiotable,myBiotransformation)
R> myBiotable[,1]<-as.factor(myBiotable[,1])
R> tail(myBiotable)

<table>
<thead>
<tr>
<th>NAME</th>
<th>MASSDIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom_biotrans</td>
<td>105.0</td>
</tr>
</tbody>
</table>
Figure 3: Heat map created by the function `plotHeatmap`. Row numbers refer to spectra numbers.
Figure 4: PCA and PLS score plots (left and right plots respectively) generated by functions `plotPCA` and `plotPLS`. The PLS decomposition in this case has just one principal component.

<table>
<thead>
<tr>
<th>Metabolite</th>
<th>Mass Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>glucuronide conjugation</td>
<td>176.0321</td>
</tr>
<tr>
<td>hydroxylation + glucuronide</td>
<td>192.0270</td>
</tr>
<tr>
<td>GSH conjugation</td>
<td>305.0682</td>
</tr>
<tr>
<td>2x glucuronide conjugation</td>
<td>352.0642</td>
</tr>
<tr>
<td>[C13]</td>
<td>1.0034</td>
</tr>
<tr>
<td>custom_biotrans</td>
<td>105.0</td>
</tr>
</tbody>
</table>

To build an entire new biotransformations table, you only need to follow the format of the `biotransformationsTable`, which means writing the name of the biotransformations as factors in the `NAME` field of the data frame and their corresponding mass losses in the `MASSDIFF` field.

### 6.7. Metabolite Identification

Once the biotransformations annotation step is finished, the significant features have been enriched with a more specific annotation. The annotation procedure performed by the `Biotransformations()` function never replaces the peak annotations already done by other functions. `MAIT` considers the peak annotations to be complementary; therefore, when new annotations are detected, they are added to the current peak annotation and the identification function may be launched to identify the metabolites corresponding to the statistically significant features in the data.

```r
R> MAIT <- identifyMetabolites(MAIT.object = MAIT, peakTolerance = 0.005, polarity="positive")
```

**WARNING:** No input database table was given.
Selecting default MAIT database...
MAIT (Metabolite Automatic Identification Toolkit)

Metabolite identification initiated

% Metabolite identification in progress: 10 20 30 40 50 60 70
80 90 100
Metabolite identification finished

By default, the function `identifyMetabolites()` looks for the peaks of the significant features in the MAIT default metabolite database. The input parameter `peakTolerance` defines the tolerance between the peak and a database compound to be considered a possible match. It is set to 0.005 mass/charge units by default. The argument `polarity`, refers to the polarity in which the samples were taken (positive or negative). It is set to "positive" by default but it should be adjusted changed to "negative" if the samples were recorded in negative polarisation mode. To check the results easily, function `identifyMetabolites()` creates a table containing the significant feature characteristics and the possible metabolite identifications.

Such a table is recovered from the MAIT-class object using the instruction:

R> metTable <- metaboliteTable(MAIT)
R> head(metTable)

<table>
<thead>
<tr>
<th>Query Mass</th>
<th>Database Mass (neutral mass)</th>
<th>rt</th>
<th>Isotope</th>
<th>Adduct</th>
<th>Name</th>
<th>spectra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300.2</td>
<td>56.36</td>
<td>Unknown</td>
<td>Unknown</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>588.2</td>
<td>46.65</td>
<td>Unknown</td>
<td>Unknown</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>537.4</td>
<td>64.41</td>
<td>Unknown</td>
<td>Unknown</td>
<td>1869</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>451.2</td>
<td>61.88</td>
<td>Geranylgeranyl-PP</td>
<td>1891</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>325.2</td>
<td>60.95</td>
<td>Unknown</td>
<td>Unknown</td>
<td>1901</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>395.1</td>
<td>51.19</td>
<td>Unknown</td>
<td>Unknown</td>
<td>1921</td>
<td></td>
</tr>
</tbody>
</table>

Biofluid ENTRY p.adj p Fisher.Test Mean Class KO Mean Class WT
<table>
<thead>
<tr>
<th>Biofluid</th>
<th>ENTRY</th>
<th>p adj</th>
<th>p Fisher.Test</th>
<th>Mean Class KO</th>
<th>Mean Class WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>0.017482939</td>
<td>0.017482939</td>
<td>NA</td>
<td>2258350.1365</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>0.193607894</td>
<td>0.193607894</td>
<td>NA</td>
<td>1998.5050</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>0.024657677</td>
<td>0.024657677</td>
<td>NA</td>
<td>521.9275</td>
</tr>
<tr>
<td>Not Available</td>
<td>HMDB04486</td>
<td>0.003172073</td>
<td>0.003172073</td>
<td>NA</td>
<td>8853.1464</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>0.019582285</td>
<td>0.019582285</td>
<td>NA</td>
<td>7781.1248</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>0.025496645</td>
<td>0.025496645</td>
<td>NA</td>
<td>1463.7786</td>
</tr>
</tbody>
</table>

Median Class KO Median Class WT KO WT ko15 ko16 ko18 ko19
<table>
<thead>
<tr>
<th>Median Class KO</th>
<th>Median Class WT KO WT</th>
<th>ko15</th>
<th>ko16</th>
<th>ko18</th>
<th>ko19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2769931.3564</td>
<td>115642.2922</td>
<td>6</td>
<td>3</td>
<td>400571.400</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
<td>10033.2150</td>
<td>2</td>
<td>4</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>3751.3050</td>
<td>1</td>
<td>3</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>9644.3125</td>
<td>835.6261</td>
<td>5</td>
<td>0</td>
<td>10878.315</td>
</tr>
<tr>
<td>5</td>
<td>7676.3250</td>
<td>1773.4658</td>
<td>5</td>
<td>6</td>
<td>9563.384</td>
</tr>
<tr>
<td>6</td>
<td>900.5959</td>
<td>6702.1125</td>
<td>0</td>
<td>4</td>
<td>0.000</td>
</tr>
<tr>
<td>ko21</td>
<td>ko22</td>
<td>wt15</td>
<td>wt16</td>
<td>wt18</td>
<td>wt19</td>
</tr>
<tr>
<td>1</td>
<td>57169.450</td>
<td>832329.600</td>
<td>192385.450</td>
<td>90436.332</td>
<td>48410.145</td>
</tr>
<tr>
<td>2</td>
<td>2837.345</td>
<td>9153.685</td>
<td>40378.565</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>3131.565</td>
<td>3306.845</td>
<td>0.000</td>
<td>4255.525</td>
</tr>
<tr>
<td>4</td>
<td>8338.320</td>
<td>9654.485</td>
<td>1671.252</td>
<td>3877.383</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>6814.010</td>
<td>7867.225</td>
<td>17009.985</td>
<td>18556.947</td>
<td>27223.175</td>
</tr>
<tr>
<td>6</td>
<td>3386.308</td>
<td>0.000</td>
<td>4895.743</td>
<td>9045.700</td>
<td>11105.240</td>
</tr>
</tbody>
</table>
This table provides useful results about the analysis of the samples, such as the P-value of the statistical test, its adduct or isotope annotation and the name of any possible hit in the database. Note that if no metabolite has been found in the database for a certain feature, it is labelled as "unknown" in the table. The table also includes the median and mean values per class and feature.

6.8. Validation

Finally, we will use the function Validation() to check the predictive value of the significant features. All the information related to the output of the Validation() function is saved in the project directory in a folder called "Validation". Two boxplots showing the overall and per class classification ratios are created, along with every confusion matrix corresponding to each iteration (see Table 3).

R> MAIT <- Validation(Iterations = 20, trainSamples= 3, MAIT.object = MAIT)

Iteration 1 done
Iteration 2 done
Iteration 3 done
...
Iteration 19 done
Iteration 20 done

A summary of a MAIT object, which includes the overall classification values, can be accessed:

R> summary(MAIT)

A MAIT object built of 12 samples and 2640 peaks. No peak aggregation technique has been applied 106 of these peaks are statistically significant
The object contains 6 samples of class KO
The object contains 6 samples of class WT
The Classification using 3 training samples and 20 Iterations gave the results:

<table>
<thead>
<tr>
<th>KNN</th>
<th>PLSDA</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Parameters of the analysis:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataDir</td>
<td>&quot;Data&quot;</td>
</tr>
<tr>
<td>snThres</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>Sigma</td>
<td>&quot;2.1232257616562&quot;</td>
</tr>
<tr>
<td>mzSlices</td>
<td>&quot;0.3&quot;</td>
</tr>
<tr>
<td>retcorrMethod</td>
<td>&quot;loess&quot;</td>
</tr>
<tr>
<td>groupMethod</td>
<td>&quot;density&quot;</td>
</tr>
<tr>
<td>bwGroup</td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td>mzWidGroup</td>
<td>&quot;0.25&quot;</td>
</tr>
<tr>
<td>filterMethod</td>
<td>&quot;matchedFilter&quot;</td>
</tr>
<tr>
<td>rtStep</td>
<td>&quot;0.03&quot;</td>
</tr>
<tr>
<td>nSlaves</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>project</td>
<td>&quot;MAIT_Demo&quot;</td>
</tr>
<tr>
<td>ppm</td>
<td>&quot;10&quot;</td>
</tr>
</tbody>
</table>
minfrac "0.5"
fwhm "30"
family1 "gaussian"
family2 "symmetric"
span "0.2"
centWave peakwidth1 "5"
centWave peakwidth2 "20"
corrWithSamp "0.7"
corrBetSamp "0.75"
perfwhm "0.6"
sigma "6"
peakAnnotation pvalue "0.05"
calcIso "TRUE"
calcCiS "TRUE"
calcCaS "TRUE"
graphMethod "hcs"
annotateAdducts "TRUE"
peakAggregation method "None"
peakAggregation PCAscale "FALSE"
peakAggregation PCAcenter "FALSE"
peakAggregation scale "FALSE"
peakAggregation RemoveOnePeakSpectra "FALSE"
Welch pvalue "0.05"
Welch p.adj "none"
peakPrecision "0.005"
Biotransformations adductAnnotation "0"
peakTolerance "0.005"
polarity "positive"
Validation Iterations "20"
Validation trainSamples "3"
Validation PCAscale "0"
Validation PCAcenter "1"
Validation RemoveOnePeakSpectra "0"
Validation tuneSVM "0"
Validation scale "1"
PCA data logarithm "FALSE"
PCA data centered "TRUE"
PCA data scaled "TRUE"

It is also possible to gather the classification ratios per class, classifier used and iteration number by using the function classifRatioClasses():

R> classifRatioClasses(MAIT)

The classification ratios are 100% in all the iterations; the set of significant features separates the samples belonging to these classes.

6.9. Using External Peak Data

Taking advantage of the modularised design of MAIT, it is possible to use the function MAITbuilder to import peak data and analyse it using the MAIT statistical functions. As stated in section 5.5, there are certain arguments that should be provided depending on which function is wanted to be launched. In this section we will show an example of this data importation procedure using the same
data that we have been using in the tutorial so far. Let’s say we have a peak table recorded in positive polarisation mode with the peak masses and retention time values such as:

```R
> peaks <- scores(MAIT)
> masses <- getPeaklist(MAIT)$mz
> rt <- getPeaklist(MAIT)$rt/60
```

We want to perform an annotation stage and metabolite identification on these data. To that end, we can launch the function `MAITbuilder` to build a `MAIT-class` object with the data in the table:

```R
> importMAIT <- MAITbuilder(data = peaks, masses = masses, rt = rt,
significantFeatures = TRUE, spectraEstimation = TRUE, rtRange=0.2,corThresh=0.7)
```

We have selected the option `spectraEstimation` as `TRUE` because we do not know the grouping of the peaks into spectra. As we want to annotate and identify all the peaks in the data frame, we set the flag `significantFeatures` to `TRUE`. At this point, we can launch the Biotransformations function:

```R
> importMAIT <- Biotransformations(MAIT.object = importMAIT, adductAnnotation = TRUE,
peakPrecision = 0.005, adductTable = NULL)
```

We set the `adductAnnotation` flag to `TRUE` as we want to perform an adduct annotation step. The parameter `adductTable` set to `NULL` implies that a positive polarisation adduct annotation stage will be performed. To run a negative annotation, the argument should be set to `negAdducts`. The metabolite identification stage is launched as in the previous case:

```R
> importMAIT <- identifyMetabolites(MAIT.object = importMAIT, peakTolerance=0.005,
polarity="positive")
```

The annotation of the Biotransformations and the adducts is given in the `Adduct` field of the metabolite table. The identification procedure can be performed for LC/MS data gathered in negative polarisation mode by setting `polarity = "negative"`. If the class information is also introduced in the `MAITbuilder`, it is also possible to launch the computation of statistical tests (through function `spectralSigFeatures`), the validation and the functions regarding the statistical plots and models.

### 7. Conclusions

**MAIT** package is a new R package that analyses LC/MS metabolomic data files. The package provides functions yielding a programmable environment that is especially focused on performing an end-to-end metabolomic analysis. Special emphasis is given to peak annotation and statistical result validation using a predictive approach. **MAIT** also supports peak aggregation techniques to improve the predictive power of the features. The package is capable of producing a set of post-processing plots, such as PCA score plots, and summary tables to evaluate the results of the analysis. In short, **MAIT** is an easy, quick-to-use package for performing a complete automatic analysis of LC/MS metabolomic data files.

### 8. Acknowledgements

This research was supported by Spanish national grants AGL2009-13906-C02-01/ALI, AGL2010-10084-E, the CONSOLIDER INGENIO 2010 Programme and FUN-C-FOOD (CSD2007-063) under
the MICINN, as well as Merck Serono 2010 Research Grants (Fundación Salud 2000). R. Llorach thanks the MICINN and The European Social Funds for their financial contribution to the R. L. Ramón y Cajal contract (Ramon y Cajal Programme, MICINN-RC). This work has been partially supported by the Spanish Ministerio de Ciencia y Tecnología through the TEC2010-20886-C02-02 and TEC2010-20886-C02-01 grants, and the Ramon y Cajal programme. A. Perera is part of the 2009SGR-1395 consolidated research group of the Generalitat de Catalunya, Spain. CIBER-BBN is an initiative of the Spanish ISCIII. F. Fernández-Albert thanks EVALXARTA-UB and Agencia de Gestió d’Ajuts Universitaris I de Recerca, AGAUR (Generalitat de Catalunya) for their financial support.

References


Affiliation:

Francesc Fernández-Albert and Alexandre Perera
Department d’Enginyeria de Sistemes, Automàtica i Informàtica Industrial
Universitat Politècnica de Catalunya
Barcelona, Spain
E-mail: francesc.fernandez.albert@upc.edu
Alexandre.Perera@upc.edu

Francesc Fernández-Albert, Rafael Llorach and Cristina Andrés-Lacueva
Nutrition and Food Science Department, XaRTA INSA, INGENIO-CONSOLIDER Program, FUN-C-Food CSD2007-063
Avinguda Joan XXIII sn, 08028 Barcelona
Pharmacy School
University of Barcelona, Spain
Barcelona, Spain
E-mail: raffallorach@ub.edu
candres@ub.edu