Chapter 4

Processing and Visualization of Metabolomics Data Using R

Stephen C. Grace and Dane A. Hudson

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/65405

Abstract

Metabolomics provides a wealth of information about the biochemical status of cells, tissues, and other biological systems. However, for many researchers, processing the large quantities of data generated in typical metabolomics experiments poses a formidable challenge. Robust computational tools are required for all data processing steps, from handling raw data to high level statistical analysis and interpretation. This chapter describes several established methods for processing and analyzing metabolomics data within the R statistical programming environment. The focus is on processing LCMS data but the methods can be applied virtually to any analytical platform. We provide a step-by-step workflow to demonstrate how to integrate, analyze, and visualize LCMS-based metabolomics data using computational tools available in R. These concepts and methods will allow specialists and nonspecialists alike to develop and evaluate their own data more critically.

Keywords: liquid chromatography mass spectrometry (LCMS), R, data processing, multivariate analysis, data visualization

1. Introduction

Metabolomics is a rapidly growing discipline focusing on the global study of small molecule metabolites in biological systems. Through the characterization of metabolite dynamics, interactions, and responses to genetic or environmental perturbations, metabolomics can provide a comprehensive picture of both baseline physiology and global biochemical responses to genetic, abiotic, and biotic factors [1].

As the diversity in abundance and chemical properties of metabolites varies greatly in organisms, a range of analytical techniques must be utilized to survey the entire metabolome. A number of methods have been developed for the extraction, detection, identification, and quantification of the metabolome [2]. Mass spectrometry coupled with gas chromatography (GC-MS) or liquid chromatography (LC-MS) are the most common analytical platforms, although capillary electrophoresis mass spectrometry (CE-MS) and nuclear magnetic resonance (NMR) are also widely used in metabolomics research [3–6].

Since metabolomics experiments typically produce large amounts of data, sophisticated bioinformatic tools are needed for efficient and high-throughput data processing to remove systematic bias and to explore biologically significant findings. Both multivariate statistical analysis and data visualization play a critical role in extracting relevant information and interpreting the results of metabolomics experiments.

The data generated in a metabolomics experiment generally can be represented as a matrix of intensity values containing N observations (samples) of K variables (peaks, bins, etc.). Additional information, such as experimental group, genotype, time point, gender, etc., is also required for some statistical procedures. For multivariate analysis, very few mathematical constraints are placed on the values contained in the data matrix. Therefore, a common set of statistical tools can be used to analyze metabolomics data of almost any type. However, as discussed below, multiple preprocessing steps are often necessary to yield interpretable results [7, 8].

The focus of this chapter is on describing methods for processing and visualizing metabolomics data obtained by liquid chromatography mass spectrometry (LCMS). LCMS is the most widely used method in metabolomics research due to its dynamic range, coverage, ease of sample preparation, and high information content [3–5]. We present a standard workflow for handling LCMS data, from raw data processing to downstream statistical analysis using open source tools available within the R software environment.

1.1. What is R?

R is a software environment for statistical computing, data analysis, and graphics, which has become an essential tool in all areas of bioinformatics research. A major advantage of R over commercial software is that it is open source and free to all users. The base distribution of R and a large number of user contributed packages are available under the terms of the Free Software Foundation's GNU General Public License in source code form. There are versions of R for Unix, Windows, and Macintosh at the official CRAN website (http://cran.r-project.org/).

In addition to being a popular language for performing high level statistics, R has a wide array of graphical tools that make it an ideal environment for exploratory data analysis and generating publication quality figures. All work is done using the command line-based text functions with user-defined scripts. Although R can be challenging for new users, it is quite flexible once the basic commands, functions, and data structures have been learned. A detailed description of every function with examples can be obtained by typing *help* followed by the name of the

function, i.e., *help(plot)*. In addition, there are ample online resources to help users learn the basics of R as well as solve a wide range of common data analysis problems [9–11].

R has a powerful set of functions for creating graphics, from fairly simple graphs using base graphics commands to highly sophisticated graphs using the one of several advanced graphics packages [12]. The focus of this chapter is on using the ggplot2 package, a high-level platform for creating graphics that is especially powerful for working with high-dimensional data [13]. The basic idea in ggplot2 is to build graphs by adding successive layers that include visual representations as well as statistical summaries of the data [14]. A layer is defined as an R data frame or matrix, a specification mapping columns of that frame into aesthetic properties, a statistical approach to summarize the rows of that frame, a geometric object to visually represent that summary, and an optional position adjustment to move overlapping geometric objects. This approach allows great flexibility in producing highly customizable graphs by combining layers that describe single or multiple data objects. We will demonstrate these ideas throughout this chapter.

1.2. Open source tools for metabolomics

A number of free software tools are available for processing, visualization, and statistical analysis of metabolomics data. Some of the more popular platforms are presented in **Table 1**.

XCMS is a powerful R-based software for LCMS data processing. As with any R-based package, it is command line driven and requires some background knowledge of the R programming language. XCMS uses nonlinear retention time correction, matched filtration, peak detection, and peak matching to extract relevant information from raw LCMS data [15]. Peak detection parameters can be optimized to process the raw data in an appropriate and efficient manner. As shown in the following sections, XCMS can be combined with base R functions and additional R packages to provide a complete solution to LCMS data processing needs. Statistical analysis and data visualization can all be incorporated into the scripts to quickly process the large amounts of data from start to finish.

XCMS online is a web-based version of XCMS that provides many of the advantages of the traditional R package without the use of a command line-based environment [16]. It allows limited control over processing parameters and gives interactive graphs of univariate and multivariate analyses [17].

MetaboAnalyst is a popular web-based resource that provides an easy to use, comprehensive interface for metabolomics data analysis [18]. It includes a variety of data preprocessing and statistical tools for univariate and multivariate analysis and generates high resolution, interactive graphics. Depending on the type of data being analyzed, it can also be used for biomarker analysis, enrichment analysis, pathway analysis, and more [19].

Haystack is a web server-based processing tool that uses mass bins to filter and extract information from raw LCMS data [20]. Haystack also provides graphical tools to visualize raw and processed data and incorporates some exploratory statistical analysis tools. Because extracted features are based on mass bins, missing values due to redundant or missing peaks

are absent from the processed data. Processed data files are downloadable in .csv format, which can be imported into analysis software of the user's preference.

Platform	Description	Advantages	Disadvantages	Reference
XCMS	R-based platform for raw LCMS data processing and visualization	-Adjustable parameters -Streamlined workflow	-Requires knowledge of R language -Command line based	[15]
XCMS online	Web-based graphical user interface version of XCMS	-Cloud storage and sharing -Relatively easy to use	-Not as customizable as the R version	[16, 17]
MetaboAnalyst	Online statistical analysis	-Easy to use -Wide variety of statistical tests available -Interactive plots	-Relies on pre-processed data -Limited options for customizing graphics	[18, 19]
Haystack	Raw data processing and visualization using mass bins	-Unbiased -No zero values -Not dependent on quality of chromatography	-Graphics not customizable -Does not take into account peak retention time	[20]
MZmine 2	Raw data processing and visualization	-Java based -User friendly -Project batching	-Limited options for customizing graphics -Numerous options can be overwhelming	[21]
MET-IDEA	Raw data processing for GCMS and LCMS data	-Works well with very large data sets -Optional manual integration	-Aimed more for GCMS data -Low-quality graphics	[22]

Table 1. Open source and web-based platforms for metabolomics data analysis.

MZmine 2 is a Java-based platform that allows for flexible MS data processing through a user-friendly graphical interface and customizable parameters for data processing and visualization [21].

Metabolomics Ion-Based Data Extraction Algorithm (MET-IDEA) is a large-scale metabolomics data processing program generally used for GCMS data but can also be used for LCMS data. It performs peak alignment, annotation, and integration of hyphenated mass spectrometry data and allows visualization of integrated peaks along with their accompanying mass spectra [22].

1.3. Data processing for metabolomics

Robust computational tools are essential to analyze and interpret metabolomics data. The first step in data processing, especially in untargeted metabolomics, is to convert the raw data into

a numerical format that can be used for downstream statistical analysis. For LCMS data, this involves multiple steps, including filtering, feature detection, alignment, and normalization [23, 24]. Filtering methods aim to remove effects like measurement or baseline noise. Feature detection is used to identify measured ions from the raw signal. Alignment methods cluster measurements from across different samples and normalization removes unwanted systematic variation between samples.

Once the data have been converted to a numerical matrix, statistical tools are used to reveal patterns in the data, determine class membership, and identify relevant biological features. Univariate methods are often used as a first step to obtain a rough ranking of potentially important features before applying more sophisticated statistical techniques [25]. Familiar examples include fold change differences, *t*-tests, and volcano plots.

Multivariate methods treat multiple, often correlated, variables simultaneously, and attempt to model relationships between variables and observations [25–27]. Well known examples include principal component analysis (PCA) and partial least squares (PLS). PCA is an unsupervised method meaning that only the data matrix itself is used to model the data. Since class membership is not considered, PCA provides an unbiased summary of the data structure. For exploratory studies where differences between experimental groups may be unknown or unpredictable, it is appropriate to apply PCA as a first step to reveal patterns in the data and relationships between groups.

A shortcoming of PCA is that it can only reveal group differences when within-group variation is sufficiently less than between-group variation [26]. To overcome this problem, supervised forms of discriminant analysis such as PLS that rely on class membership are also routinely applied in metabolomics studies [28]. The primary goal of these methods is to identify class differences from a multivariate data set and to identify biologically important features that account for these differences. Often, the results of a PCA are used to formulate a hypothesis that PLS or other supervised methods can test or verify in more detail.

In the following sections, we will provide an overview of the data processing steps used in a typical metabolomics experiment. We do not aim to provide a thorough description of the statistical methods but rather introduce the basic concepts behind these methods and demonstrate how to perform them in the R computing environment.

2. Experimental methods

To illustrate the concepts and methods presented in this chapter, we produced a data set from two tomato varieties harvested at two developmental stages using untargeted LCMS. The varieties used were "Manapal" and its nearly isogenic counterpart, the *h*igh *p*igment-2 *dark green* (hp-2dg) mutant. Hp-2dg plants have a mutation in the tomato homolog of the DEETIOLATED-1 gene involved in light-mediated signal transduction and plant photomorphogenesis [29]. These plants display a number of interesting traits, including shorter stature, slower growth rates, darker foliage, and elevated levels of certain metabolites such as

flavonoids and carotenoids [30–32]. Since many of these phytochemicals are important for human health, there is great interest in understanding the molecular mechanisms that underlie the altered phenotype of the hp-2dg mutation.

Fully expanded fruits were harvested at green and red stages, lyophilized, and stored at −80°C until analysis. Samples from 10 individual fruits were extracted in 80% methanol and analyzed using an Agilent 1100 HPLC/MSD-VI Ion Trap mass spectrometer with an electrospray ionization (ESI) source. Chromatograms were saved in netCDF format using the instrument software.

The data were analyzed using R as described in the following sections. A summary of the data processing workflow is presented in **Figure 1**. The full source code for all procedures is available online at https://github.com/ualr-Rgroup/metabolomics-in-r.

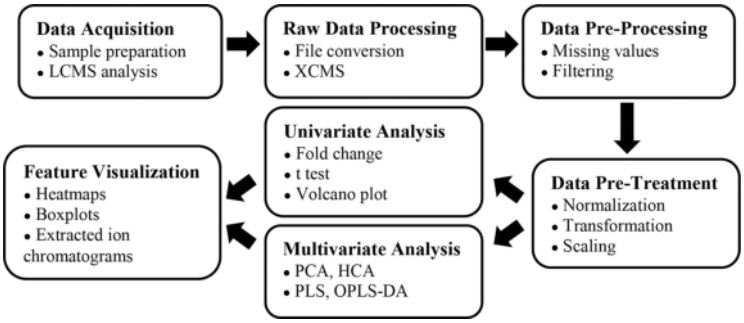

Figure 1. Summary of metabolomic data processing workflow.

2.1. Data processing with XCMS

While web-based tools such as MetaboAnalyst and XCMS online are inarguably convenient, learning data analysis procedures in R gives researchers much greater flexibility not only in processing and analyzing their data but also in creating high-quality custom graphics. Here, we demonstrate how to perform raw data processing in R using the XCMS package. XCMS is a powerful and flexible software package that has gained widespread use for untargeted metabolomic studies [15]. It is available through Bioconductor and can be installed in R using the following commands:

```
source("http://bioconductor.org/biocLite.R")
1.1.1.1. biocLite("xcms")
```

2.2. Data import and visualization

XCMS requires data in an open access nonproprietary format such as Network Common Data Form (NetCDF) or mzXML. File conversion often can be done within the operating software

http://dx.doi.org/10.5772/65405

of the instrument. A popular online tool, ProteoWizard (http://proteowizard.sourceforge.net), is also available that can be used to convert raw LCMS data into an open data format. The converted data files are placed in a subdirectory named "cdf" within the R working directory. From here the data can be imported, processed, and visualized.

An LCMS data file is a series of successively recorded mass spectra over a range of *m*/*z* values. The total intensity of all ions at each time point is known as the total ion chromatogram (TIC). The *xcmsRaw* function is used to read data files into R's memory environment. The *plotTIC* function can be used to produce a TIC or base peak chromatogram (BPC). This function can also be used to obtain the numerical data for custom plotting. The following example demonstrates how to do this in R:

```
library(xcms) # load the xcms library
cdffiles <-list.files("./cdf", recursive=T, full=T) # define data
xr1 <- xcmsRaw(cdffiles[6]) # extract raw data
t1 <- xr1@scantime # extract time vector
int1 <- xr1@tic # extract TIC intensity vector
s1 <- rep("HG5", 1896) # create sample vector
g1 <- rep("hp-2dg Green", 1896) # create group vector
tic1 <- data.frame(t1, int1, s1, g1)
```

This script loads the xcms package, defines the raw data files, and creates an xcmsRaw object (xr1) from file number 6. This object stores information in the raw data that can be extracted and combined into a data frame. We add factor columns for sample and group that will be used to make a custom plot with ggplot2.

This process is repeated for additional raw data files. The individual data frames are combined into a single data frame with the *rbind* function.

```
tic.data <- rbind(tic1, tic2, tic3, tic4)
```

Once the data have been combined into a single data frame, a multipanel plot can be produced using the facet_wrap function in ggplot2. The result is shown in **Figure 2**. Variation in peak profiles can be readily observed between the different groups.

2.3. Raw data processing

XCMS uses several algorithms to process LCMS data. The first step is to filter and detect ion peaks using the *xcmsSet* method. The peak detection algorithm is based on cutting the data into slices of predefined mass widths and then finding peaks in the chromatographic time domain by applying a Gaussian model peak matched filter. Although the default arguments for the *xcmsSet* method may provide acceptable results in some cases, it is recommended that the parameters used for peak selection be optimized for this step. Without optimization, common problems that may arise include oversampling, i.e., assigning the same metabolite to

multiple peaks, and missing values, i.e., failure to detect certain peaks in some samples that can interfere with downstream statistical analysis [8].

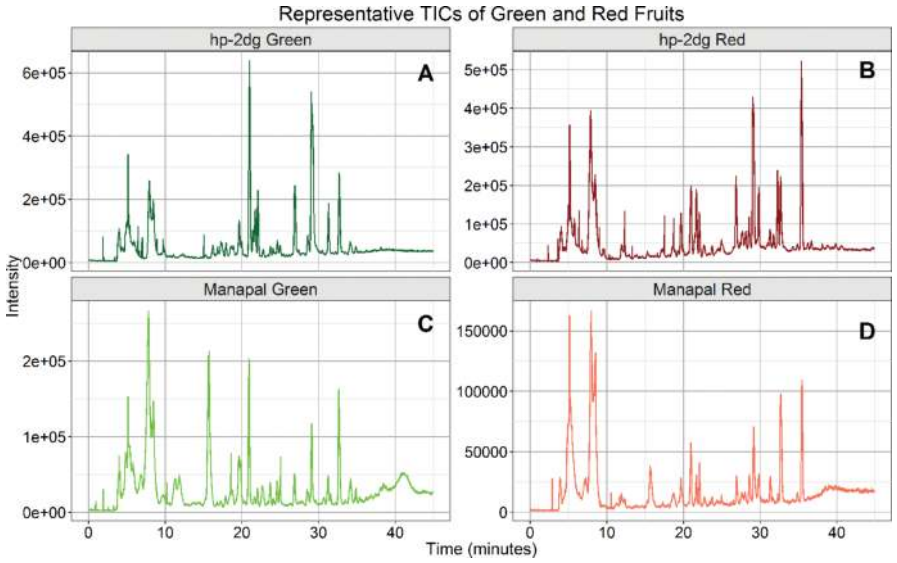

Figure 2. Representative total ion chromatograms (TICs) of green fruits (A, C) and red fruits (B, D) of the hp-2dg (A, B) and Manapal (C, D) tomato varieties.

The next step after filtration and peak identification is matching peaks across samples. Peaks representing the same analyte across samples must be placed into groups. This is accomplished with the *group* function, which returns a new xcmsSet object. After matching peaks into groups, XCMS can use those groups to identify and correct correlated drifts in retention time using the *retcor* function. The aligned peaks can then be used for a second pass of peak grouping which will be more accurate than the first.

After the second pass of peak grouping, there will still be missing peaks from some of the samples. This can occur because peaks were missed during peak identification or because an analyte was not present or below the detection limit in a sample. Missing values can be problematic for statistical methods that require a fully defined data matrix. Those missing data points can be filled in by re-reading the raw data files and integrating them in the regions of missing peaks using the *fillPeaks* function.

XCMS can generate a report showing fold change differences in analyte intensities and their statistical significance using the *diffreport* function. However, we recommend obtaining the raw peak integration results using the *groupval* function. This function returns a numerical matrix in which each row represents a peak defined by its mass and retention time and each column represents a different sample. In an example used here, 308 peaks were identified with 1.4% missing values. This matrix provides the starting point for downstream statistical analysis.

http://dx.doi.org/10.5772/65405

```
# Step 1.  Create xcmsSet object
xset <- xcmsSet(cdffiles, snthresh = 20, step = 0.5)
# Step 2.  Peak Alignment
xsg <- group(xset)
# Step 3.  Retention Time Correction
xsg2 <- retcor(xsg, method="obiwarp", plottype="deviation")
# Step 4.  Regroup and change bandwidth as desired
xsg3 <- group(xsg2, bw=10)
# Step 5.  Fill in missing peaks
xsg4 <- fillPeaks(xsg3)
# Step 6.  Get peak intensity matrix
results <- groupval(xsg4, "medret", "into")
# Step 7.  Save results as .csv file
write.csv(results, file="xcmsresults.csv")
```

3. Descriptive statistics

Once the raw data have been processed in XCMS, it is often useful to obtain descriptive statistics for each variable. This can be accomplished in R by creating a function to calculate the mean, median, standard deviation, standard error, and coefficient of variation for each variable. The *apply* function executes these operations on each row and returns the results into a data frame.

```
sumstats <- function(z) {
       Mean <- apply(z, 1, mean)
       Median <- apply(z, 1, median)
       SD <- apply(z, 1, sd)
       SE <- apply(z, 1, function(x) sd(x)/sqrt(length(x)))
       CV <- apply(z, 1, function(x) sd(x)/mean(x))
       result <- data.frame(Mean, Median, SD, SE, CV)
       return(result)
       }
```

This function creates a new data object containing descriptive statistics for each variable that can be used to rank variables according to mean or median intensity or to assess the degree of dispersion for each variable.

4. Data preprocessing

Before higher order statistical methods can be applied, it is often necessary to "clean up" the data to improve the analysis. Typical steps include (1) imputation of missing values, (2) transformation, (3) scaling, and (4) normalization.

4.1. Imputation of missing values

A common phenomenon in metabolomics measurements is the occurrence of missing values, i.e., empty cells where a respective metabolite peak has not been assigned any numerical value. As many multivariate methods require a fully defined matrix or become computationally inefficient for incomplete data, estimation of missing values is an important step in the preparation of the data [8].

Even after using the *fillPeaks* function in XCMS, there are still typically a large number of missing values. There are several strategies for dealing with these, including removing variables with missing values that exceed a certain threshold. However, these are often interesting features that are important in discriminating experimental groups. An alternative and widely used approach is imputation, where missing values are replaced with a small value with the assumption that the feature in question is below the limit of detection in those samples where XCMS fails to detect a peak.

The following function can be used to find the minimum nonzero value in a set of numbers and then divide that value by 2. We can then use the *apply* function to replace the missing values in each row of the matrix using this function.

```
replacezero <- function(x) "[<-"(x, !x | is.na(x), min(x[x > 0],
na.rm = TRUE) / 2)
newdata <- apply(data, 1, replacezero)
```

4.2. Log transformation

Since metabolomics studies are generally concerned with relative changes in metabolite levels, a log or other suitable transformation is normally applied before performing higher order statistical analysis. A log transformation helps to remove heteroscedasticity from the data and correct for a skewed data distribution [7]. This operation is easily performed in R using the *log* function. The default option is to compute the natural logarithm. However, the general form *log* (*x*, base) computes logarithms with any desired based. The base 2 log transformation is commonly used in metabolomics studies. Note that the *log* function will return NA for any zero values in the data matrix.

4.3. Normalization and scaling

The purpose of data normalization is to reduce systematic variation and to separate biological variation from nonbiological variation introduced by the experimental process. This is often necessary to improve the results of higher order statistical analysis [7, 23, 25].

Normalization can be sample wise or feature wise or both. Sample wise normalization makes the samples more comparable to each other. Common approaches include normalization to constant sum, to a reference sample or feature, or sample specific normalization such as dry weight or tissue volume.

http://dx.doi.org/10.5772/65405

Feature wise normalization involves centering the data around the mean combined with various types of scaling. Centering focuses the data on the amount of variation instead of the mean intensity. Scaling involves dividing each variable by a factor that approximates the amount of data dispersion. The most common scaling approach is known as unit scaling or autoscaling where each variable is mean centered and then divided by the standard deviation. After autoscaling all variables become equally important and are analyzed on the basis of correlations instead of covariances. A disadvantage of autoscaling is that it tends to inflate the importance of small variables which are more likely to contain measurement errors [7]. Other scaling operations include Pareto scaling, which uses the square root of the standard deviation as the scaling factor, vast scaling, which uses the standard deviation and the coefficient of variation as scaling factors, and range scaling, where the range is used as the scaling factor [7].

The *scale* function in R automatically performs centering and autoscaling. Other scaling procedures can also be carried out using the custom functions. For example, the following function can be used for Pareto scaling which is recommended for metabolomics data.

```
paretoscale <- function(z) {
       rowmean <- apply(z, 1, mean) # row means
       rowsd <- apply(z, 1, sd)  # row standard deviation
       rowsqrtsd <- sqrt(rowsd) # sqrt of sd
       rv <- sweep(z, 1, rowmean,"-")  # mean center
       rv <- sweep(rv, 1, rowsqrtsd, "/")  # divide by sqrtsd
       return(rv)
       }
```

5. Principal component analysis

Principal component analysis (PCA) is the foundation for many multivariate techniques that seek to describe a set of observations based on a large number of variables [25, 26]. The core idea of PCA is to reduce the dimensionality of the data, i.e., the number of variables while retaining as much of the variation as possible. Using PCA, it is possible to extract and display the systematic variation in the data and identify related groups, trends, and outliers.

PCA returns two important types of information: a scores matrix and a loadings matrix. The scores matrix contains the coordinates of the samples (i.e., observations) for each principal component and provides a summary of the observations in a lower dimensional space. The first principal component describes the largest variation in the data matrix, the second component describes the second largest, and so on. All PCA components are mutually orthogonal, meaning they are uncorrelated. Generally, most of the variation is captured in the first two or three principal components. Therefore, a scatter plot of the first two score vectors usually provides a good summary of all the samples and can reveal if there are differences between the groups as well as outliers.

Analogous to the scores matrix, the loadings matrix describes the relationships among the measured variables for each principal component. A scatterplot of the first two loading vectors can reveal the influence (weights) of individual variables in the model. An important aspect of PCA is that directions in the scores plot correspond to directions in the loadings plot, and so a comparison of these two plots can be used to identify, which variables (loadings) are most important for separating the different samples (scores) [28].

When there are more than two experimental classes, it is generally appropriate to use multivariate methods such as PCA to find patterns in the data [27]. The primary goal of these methods is to determine if the classes can be predicted from the variables (class assignment) and to identify which variables are important in predicting class membership.

There are several ways to perform PCA in R. Here, we will demonstrate the procedure using the *prcomp* function, which comes with the built-in R stats package. This method uses singular value decomposition (SVD) to calculate eigenvalues, which is the standard approach in PCA. The following syntax is used.

```
prcomp(data, retx = TRUE, center = TRUE, scale = FALSE)
```

where "data" is a dataframe or matrix containing the data, retx is a logical value that indicates whether the scores will be returned, center is a logical value indicating whether the variables should be mean centered, and scale is a logical value indicating whether the variables should be scaled to unit variance.

After missing values have been replaced, the data are log transformed and Pareto scaled. Note that the Pareto scale function must first be defined as above.

```
logdata <- log(newdata, 2)
pareto.logdata <- paretoscale(logdata)
```

Now, we perform PCA. The center and scale options are set to FALSE since these operations have already been performed with the *paretoscale* function.

```
pca <- prcomp(t(pareto.logdata), center=F, scale=F)
```

The *t* function is used here to transpose the matrix so that each row represents an observation (sample) and each column represents a variable (peak). This is necessary if we want the scores matrix to correspond to samples and the loadings matrix to correspond to variables.

The *prcomp* function returns several outputs that can be accessed by the *summary* command.

http://dx.doi.org/10.5772/65405

```
pcaresults <- summary(pca)
```

This returns a list that contains the standard deviations (eigenvalues) and proportion of the total variance for each principal component, the scores matrix, and the loadings matrix. We can extract each of these outputs into a new data frame and save the results to file for later use.

```
scree.data <- as.data.frame(pcaresults$importance)
score.data <- as.data.frame(pcaresults$x)
loadings.data <- as.data.frame(pcaresults$rotation)
write.csv(scree.data, "pca_scree.csv")
write.csv(score.data, "pca_scores.csv")
write.csv(loadings.data, "pca_loadings.csv")
```

5.1. Scores plot

The results of a PCA can be easily visualized using the base graphics functions in R. However, it is often desirable to produce a high-quality figure with custom formatting using ggplot2. To do this, we first import the scores matrix from the PCA. Since the first two principal components capture most of the variance, we will subset the data to include only those values.

```
data <- read.csv("pca_scores.csv", header=T)
data <- data[, c(1:3)] # subset columns 1-3
```

In order to map individual samples to their respective groups, we need to add a new column to the data frame indicating the group to which each sample belongs. To do this, we first create a vector of group names and then add the vector to the data frame with the *cbind* function. We can then generate the scores plot with ggplot2.

```
ggplot(data, aes(PC1, PC2)) +
geom_point(aes(shape=Group)) +
geom_text(aes(label=data$X)) +
stat_ellipse(aes(fill=Group))
```

This script defines the data and adds layers for data points, text labels, and confidence ellipses. The resulting plot is shown in **Figure 3**.

The scores plot shows that the two genotypes are well separated along the first PC axis while the developmental stage (green versus red) is separated along the second PC axis. There is more variation among the hp-2dg green samples than among the other groups.

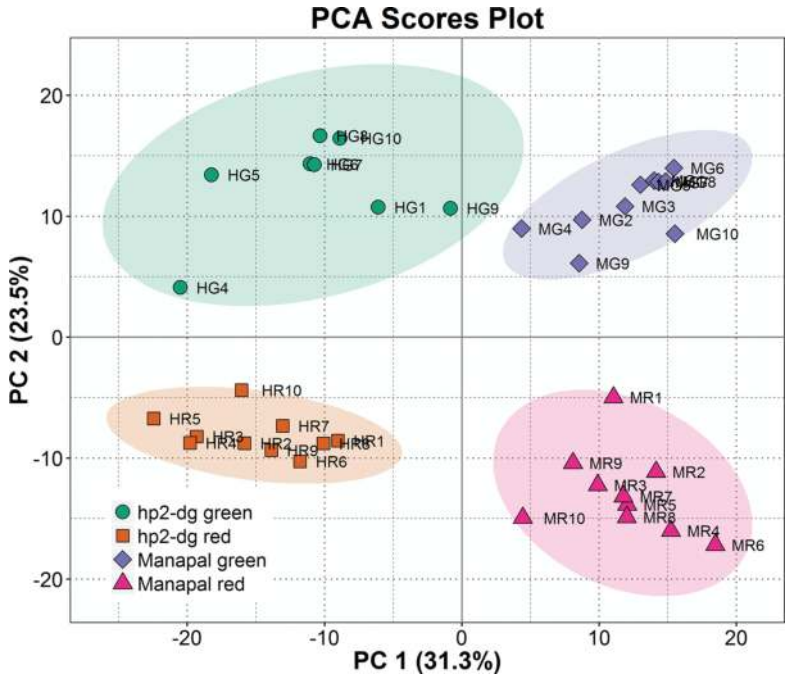

Figure 3. PCA scores plot.

5.2. Loadings plot

We can create a loadings plot using a similar approach. Since there are a large number of variables, we would also like to know which ones have the largest influence on the PCA. Variables with high loadings (positive and negative) are more likely to be important for discriminating groups that are separated in the scores plot.

One of the major advantages of R is that it has many powerful and flexible functions for subsetting data. One approach might be to identify the maximum and minimum loadings using the *range* function and then subset the data based on a percentage of these values. Alternatively, we can make a plot of PC1 versus PC2 loadings and visually inspect the data for high and low values. The *subset* function can be used to select rows that meet certain criteria. In this example, 0.09 and −0.09 were selected for threshold values.

```
significant.loadings <- subset(loadings, PC1 > 0.09 | PC1 < -0.09 |
                                         PC2 > 0.09 | PC2 < -0.09)
```

For plotting in ggplot2, it is generally recommended to add factor columns to the data frame for the purpose of mapping aesthetics to variables. A factor is a categorical value in R with

http://dx.doi.org/10.5772/65405

predefined levels. We can use the *ifelse* function to specify the factor level in the new columns much in the same way as the *subset* function was used to create a new data frame. Loadings above and below the threshold values are marked for subsetting in this way.

```
loadings$pc1.change <-
  ifelse(loadings$PC1 > 0.09,"UP",
         ifelse(loadings$PC1 < -0.09,"DOWN",
                "nochange"))
```

With the added factors, we can make the plot in ggplot2 and indicate significant loadings with different colors. The grid package provides several options for adding text annotations. The resulting plot is shown in **Figure 4**.

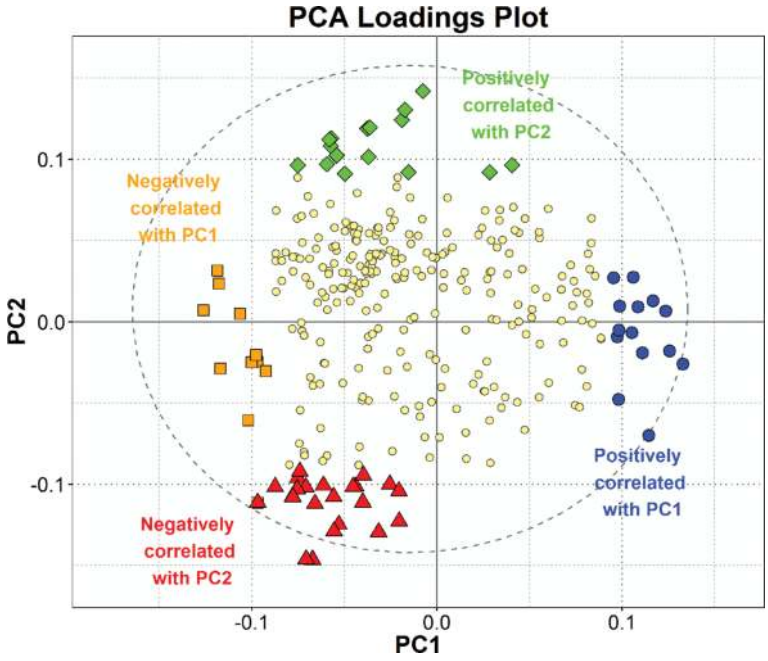

Figure 4. PCA loadings plot.

Since the scores matrix and the loadings matrix share similar geometric properties, the direction of the loadings indicate those variables that have the greatest influence on class separation. Based on these criteria, 64 potentially significant peaks were identified out of the original 308 (**Figure 4**). Separation along the PC1 axis identified features that show high variation by genotype while separation along the PC2 axis identified features that show high variation by developmental stage.

It should be emphasized that since PCA is an exploratory method, the interpretation of PCA results for the purpose of inferring biological importance must be done with caution. Potentially interesting features must be further analyzed to assess their biological significance. This can be done using boxplots, heatmaps, or other suitable graphical displays and rechecking the raw data by generating extracted ion chromatograms.

The first step in this process is to subset the original data to include only those variables of interest, i.e., the colored symbols in **Figure 4**. This can be done in R out using the extremely useful *merge* function.

```
data.sub <- merge(newdata, significant.loadings, by="row.names")
```

This command creates a new data frame containing the peak intensity values for the 64 variables with high PCA loadings that can be further analyzed as described below.

6. Partial least squares-discriminant analysis (PLS-DA)

Partial least squares-discriminant analysis (PLS-DA) is a supervised method that uses multiple linear regression to find the direction of maximum covariance between a data set and class labels [28]. Supervised techniques can be very helpful for highlighting sample/group differences when PCA results are masked by high levels of spectral noise, strong batch effects, or high within group variation [26]. PLS-DA sharpens the separation between groups of observations by rotating PCA components such that a maximum separation among classes is obtained and identifies variables that carry most of the class separating information. However, contrary to PCA, supervised methods like PLS-DA aggressively overfit models to the data, almost always yielding scores in which classes are separated [26, 27]. As a result, these methods can generate excellent class separation even with random data. For this reason results of these types of tests should be critically checked and properly cross-validated.

The pls, plsdepot, and muma packages can all be used for partial least squares analysis in R [33, 34]. We will demonstrate how to perform an extension of the PLS method known as OPLS-DA using the muma package below.

7. Heatmaps

Heatmaps are an effective tool for displaying feature variation among groups of samples [35]. The basic concept of a heatmap is to represent relationships among variables as a color image. Rows and columns typically are reordered so that variables and/or samples with similar profiles are closer to one another, making these profiles more visible. Each value in the data matrix is displayed as a color, making it possible to view the patterns graphically.

Heatmaps use an agglomerative hierarchical clustering algorithm to order and display the data as a dendrogram. Two important factors to consider when constructing a heatmap are the type of distance measure and the agglomeration method used. For details on the various methods available see [35].

The *heatmap.2* function in the gplots package can be used to construct a heatmap that is easily customizable and include options for both the distance and agglomeration methods, as well as scaling options for rows or columns. Unless the actual numerical values in the data matrix have an explicit meaning, row scaling is usually advisable [35].

A heatmap showing the scaled data from the 64 loadings extracted by PCA is shown in **Figure 5**. Four well-defined clusters are evident that correlate well with the four different experimental groups. These variables form a starting point for further experiments and analyses.

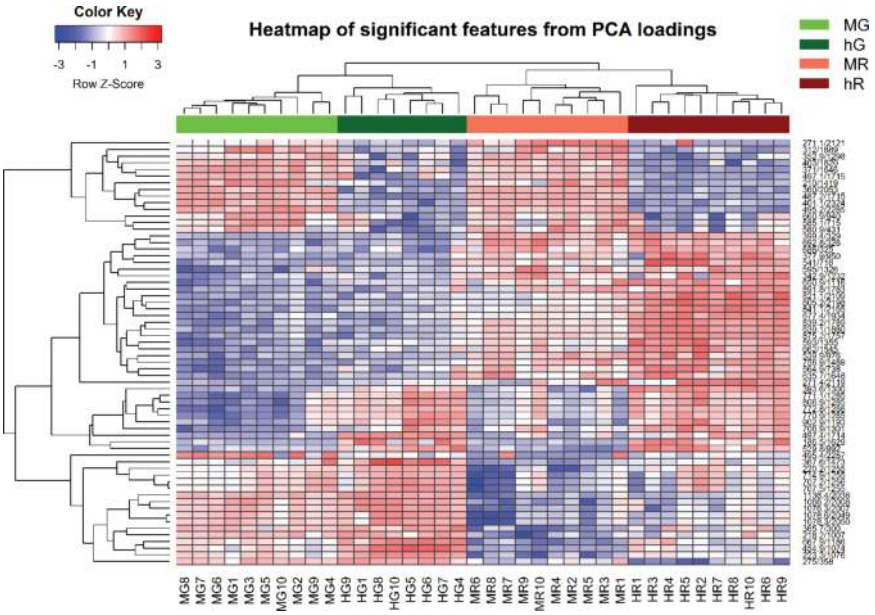

Figure 5. Heatmap of significant features obtained from PCA loadings.

8. Boxplots

Boxplots are another good way to visualize and compare features among different samples. A boxplot graphically depicts a vector through its five-number summary. The edges of the box represent the lower and upper quartiles while the whiskers represent the minimum and

maximum values. The median is displayed as a line within the box. Outliers are represented as symbols outside of the whiskers.

A simple boxplot can be generated from any numeric vector using the *boxplot* function in R. However, a more customizable boxplot can be created using the ggplot2 package. **Figure 6** shows boxplots for four significant features from the PCA results. The data first were log transformed and Pareto scaled to show relative differences.

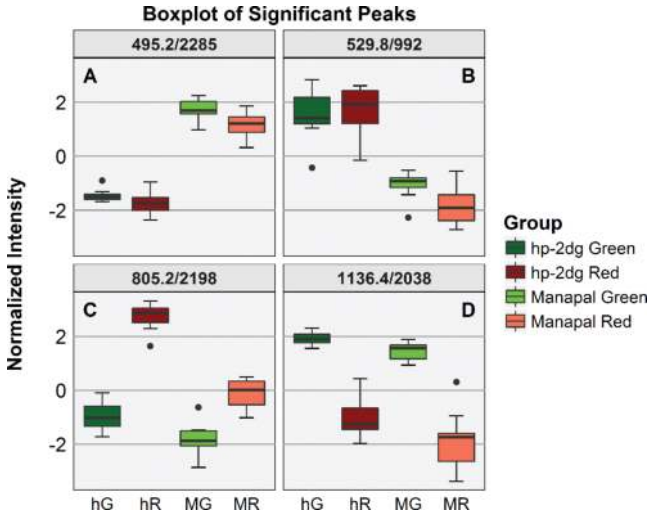

Figure 6. Boxplot of four significant peaks identified from PCA loadings.

The 495.2/2285 peak, which had a high positive PC1 loading, was significantly higher in the Manapal strain, whereas the 529.8/992 peak, which had a high negative PC1 loading, was significantly higher in the hp-2dg strain. The 1136.4/2038 peak, which had the highest positive loading for PC2, was significantly higher in green fruits of both varieties. Interestingly, the 805.2/2198 peak, which had a high negative loadings for both PC1 and PC2, was only significant in red fruits of the hp-2dg strain.

9. Extracted ion chromatograms

While statistical procedures provide important clues about potentially significant variables, a critical but often overlooked step in analyzing metabolomics data is reinspecting the raw data to assess the validity of these results. Not only can this provide confirmation of meaningful features but it can also reveal false positives caused by scaling artifacts or spurious peak assignment, which is common in XCMS-processed data.

LCMS ion peaks can be visualized through extracted ion chromatograms (EICs). An EIC is essentially a "slice" of the raw data that covers a specific *m*/*z* and time range. XCMS automat-

http://dx.doi.org/10.5772/65405

ically generates EICs for peaks that show high significance, but these are low quality "snapshot" images. However, the *plotEIC* function in XCMS can be used to extract the numerical data for any EIC of interest. The following commands describe how to obtain EIC data for all samples in a data set and generate an EIC plot for grouped samples.

We first create a list of xcmsRaw objects from the raw data files with the *lapply* function.

```
cdffiles <-list.files("./cdf", recursive = TRUE, full=T)
dat.raw <- lapply(cdffiles, xcmsRaw)
```

Next, we set the upper and lower limits for *m/z* and time. For this example, we will look at the 805.2/2198 peak since PCA and boxplots indicate that this peak was highly significant in red fruits of the hp-2dg strain.

```
mrange <- c(804.5, 805.5)
trange <- c(2000, 2400)
```

We can use the *lapply* function again to create an EIC for all samples. The data are then merged into a data frame for custom plotting in ggplot2.

```
eicraw <- lapply(dat.raw, plotEIC, mzrange=mrange, rtrange=trange)
eic.df <- do.call(rbind, lapply(eicraw, data.frame))
```

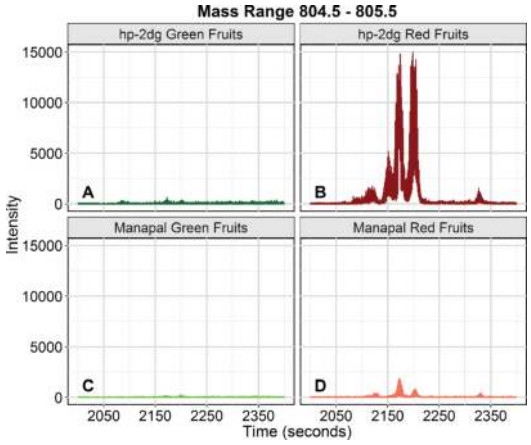

Figure 7. Extracted ion chromatograms for the *m/z* range 804.5-805.5 in hp-2dg green fruits (A), hp-2dg red fruits (B), Manapal green fruits (C) and Manapal red fruits (D). Each panel represents 10 samples.

The results are shown in **Figure 7**. The grouped EIC data clearly show that this feature is completely absent in green fruits and is very low in red fruits of Manapal. In contrast, there are several large peaks over this mass range in red fruits of hp-2dg, indicating this feature is a class-specific biomarker.

10. Volcano plots

A metabolomics experiment often involves a comparison of two groups, e.g., a treatment group versus a control. It is customary in such cases to use univariate methods to obtain a summary of the data and identify potentially important variables before applying multivariate methods [27]. A common tool to identify discriminatory features is to construct a volcano plot. This type of plot displays the fold change differences and the statistical significance for each variable. The log of the fold change is plotted on the *x*-axis so that changes in both directions (up and down) appear equidistant from the center. The *y*-axis displays the negative log of the *p*-value from a two-sample *t*-test. Data points that are far from the origin, i.e., near the top of the plot and to the far left or right, are considered important variables with potentially high biological relevance.

The steps required to construct a volcano plot can be carried out using several base R functions. The fold change is typically calculated as the ratio of the two means. We can use the *apply* function to determine the means for each variable.

```
group1mean <- apply(data[1:10], 1, FUN=mean)
group2mean <- apply(data[11:20], 1, FUN=mean)
```

We then divide the means of each variable to obtain the ratio and take the logarithm so that changes in both directions appear equidistant from the center.

```
FC <- group1mean/group2mean
log2FC <- log(FC,2)
```

The *t.test* function in the R stats package returns the *p*-value for an unpaired *t*-test of two independent samples. The default option is Welch's *t*-test, which assumes unequal variance. Note that data preprocessing steps, such as sum normalization and log transformation, are usually applied to make the samples more comparable and to reduce heteroscedasticity.

```
pvalue <- apply(log.data, 1, function(x)
          {t.test(x[1:10], x[10:20])$p.value } )
```

It is recommended to use a multiple testing correction when performing *t*-test on multiple variables. There *p.adjust* function in R provides several options for this, including the family

http://dx.doi.org/10.5772/65405

wise error rate (FWER), also known as the Bonferroni correction, and the false discovery rate (FDR), also known as the Benjamini-Hochberg correction. The false discovery rate is a less stringent condition than the family-wise error rate, so this method is preferred when one is interested in having more true positives.

```
pvalue.BHcorr <- p.adjust(pvalue, method = "BH")
```

We take the negative log10 values so that variables with low adjusted *p*-values (i.e., high significance) appear near the top of the plot.

```
pvalue.BHcorr.neglog <- -log10(pvalue.BHcorr)
```

Finally, the data are merged into a single data frame that can be plotted in ggplot2.

```
volcano.data <- data.frame(log2FC, pvalue.BHcorr.neglog)
```

A volcano plot comparison of the two tomato genotypes in green and red fruits is shown in **Figure 8**. Significant variables are shown as colored symbols. We selected rather conservative cutoff values of 2 and −2 for the log2 fold change (fold change >4) and 2 for the –log FDR adjusted *p*-value ($p < 0.01$) to highlight those features that showed the largest differences. In green fruits, this led to the identification of 38 metabolite peaks that were significantly higher in Manapal and 20 peaks that were higher in hp-2dg, while in red fruits, 40 metabolite peaks were higher in Manapal while 24 were higher in hp-2dg. As with the PCA loadings, these variables can be explored further with heatmaps, boxplots, EICs, etc.

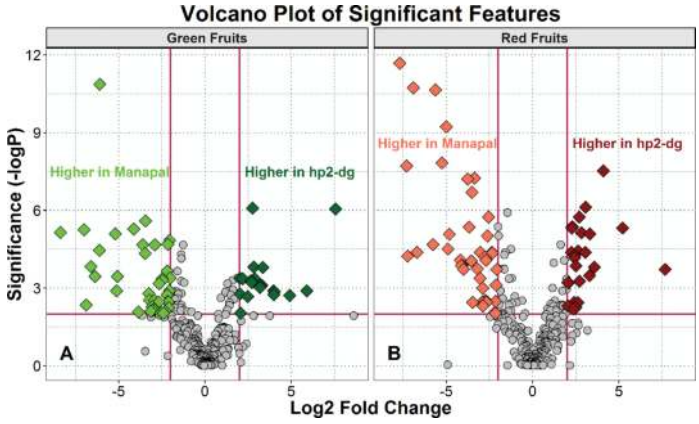

Figure 8. Volcano plot analysis of Manapal versus hp-2dg in green (A) and red (B) fruits.

The R package muma (Metabolomic Univariate and Multivariate Analysis) has a more sophisticated procedure for testing significance and returning *p*-values for a volcano plot [34]. Briefly, Shapiro Wilk's test for normality is performed to assess whether each variable has a normal distribution and to decide whether to perform a parametric test (Welch's *t*-test) or a nonparametric test (Wilcoxon-Mann Whitney test). The analysis returns fold change differences and a merged set of *p*-values from both tests and also applies a multiple testing correction that the user can specify. Finally, a volcano plot is generated highlighting significant variables based on the corrected *p*-values.

11. Orthogonal projection to latent structures-discriminant analysis (OPLS-DA)

An extension of the PLS technique known as orthogonal projection to latent structure is another very useful tool for analyzing metabolomics data. Like PLS this is a supervised method that pairs a data matrix ***X*** with a corresponding matrix ***Y*** containing sample information. The basic concept in OPLS is to separate the systematic variation in ***X*** into two parts, one that is correlated to ***Y*** and one that is not correlated (orthogonal) with ***Y*** [28]. Only the ***Y***-predictive variation is used to model the data. When working with discrete variables such as class labels the method is called OPLS discriminant analysis. The main advantages of OPLS-DA over PCA are better class discrimination and more robust identification of important features. The OPLS-DA algorithm normally is applied when there are only two classes comprising ***Y***.

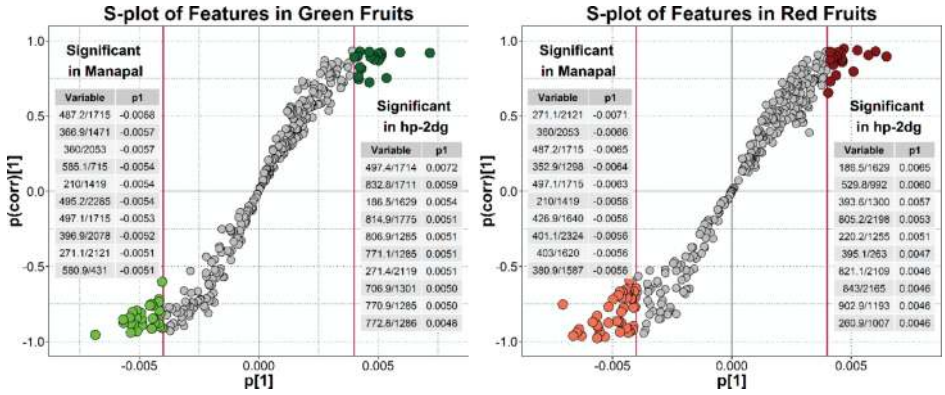

Figure 9. S-plots from OPLS-DA modeling of green and red fruits in Manapal and hp-2dg varieties.

The muma package can be used to perform OPLS-DA in R. A numerical class vector must be added to represent the ***Y*** matrix, i.e., the control group is given a value of 1 and the experimental group is given a value of 2. The data frame is saved in the working directory, and the analysis is carried out with two simple functions.

```
explore.data(file="logdata.csv", scaling="pareto")
oplsda(scaling="pareto")
```

This method automatically creates a new folder in the working directory that contains the OPLS-DA results in both numerical and graphical formats. The numerical data can be merged into a new data frame for custom plotting in ggplot2.

The loadings from an OPLS-DA model are displayed by means of an "S-plot" where the modeled covariance $p[1]$ is plotted on the x-axis and the correlation profile $p(\text{corr})[1]$ is plotted on the y-axis. Variables with higher $p[1]$ values in both positive and negative directions have a larger impact on the variance between the groups, whereas variables with higher $p(\text{corr})[1]$ values have more reliability. Therefore, data points that fall in the upper right and lower left quadrants have a high impact on the model and represent possible class-specific biomarkers.

Figure 9 shows S-plots from an OPLS-DA model of the two tomato varieties in green and red fruits. Variables with $|p[1]| > 0.004$ are highlighted, and the top 10 for each class are listed in tabular form on the graph. In general, there was good agreement between the OPLS-DA and volcano plot results for identifying significant variables.

12. Metabolite identification

One of the major challenges in LCMS-based metabolomics is metabolite annotation, i.e., identifying biological molecules from mass spectral data. Although metabolic profiling approaches that do not assign observed features to known metabolites can provide a powerful means of classifying and directly comparing samples, metabolite identification remains a crucial step for obtaining mechanistic insights into cellular processes. However, the complexity of the metabolome combined with the fact that many metabolites have not been structurally identified means that untargeted metabolomic studies typically yield a large number of unknown peaks.

The accurate identification of metabolites usually requires the ability to match candidate spectra with standard compounds run under the same conditions. Ideally, an orthogonal descriptor such as retention index is used for further validation. However, the lack of readily available standards remains a major obstacle in this regard, particularly in plant phytochemical studies [36]. Consequently, a number of strategies are being brought forward to assist in the chemical identification of unknown metabolites, including the development comprehensive mass spectral libraries [37–39], searchable databases [40–42], and information networks that integrate genomic, transcriptomic, and metabolomic data [43–45]. The construction, maintenance, and integration of these resources are crucial to the advancement of the field of metabolomics.

13. Conclusions

Untargeted metabolomics has become an increasingly powerful tool to investigate biological problems in agriculture, medicine, and a number of other fields. Therefore, efficient processing methods must be developed and refined to enable robust interpretation of metabolomics data. Method development and new software tools have helped address these challenges over the last decade. However, since improvements are still required at the various stages of data processing, establishing and refining new methods will continue to be important in the future of metabolomics research.

In this chapter, we have presented an overview of several common methods used for processing and analyzing LCMS-based metabolomics data and how to carry out these methods in the R programming environment. Although a variety of open source and web-based tools are available to support metabolomics data analysis, the ability to tailor the data processing workflow to one's own needs and generate custom graphics in R offers major advantages.

As the data sets used in all scientific disciplines get ever larger and more complex, it is becoming critical for scientists to be knowledgeable about how to use high-level languages such as R, which allow for easy and intuitive data manipulation. Along with powerful statistical capabilities, graphical tools make R an ideal environment for exploratory data analysis and provide exceptional flexibility for preparing high-quality publication-ready figures. Nevertheless, many technical and methodological issues must still be addressed to create analytical platforms that readily answer biological questions efficiently.

Acknowledgements

The authors like to thank Arthur Colvis for help with the LCMS experiments.

Author details

Stephen C. Grace* and Dane A. Hudson

*Address all correspondence to: scgrace@ualr.edu

Department of Biology, University of Arkansas, Little Rock, AR, USA

References

[1] Fiehn O. Metabolomics: the link between genotypes and phenotypes. Plant Mol Biol. 2002;48:155–171. DOI: 10.1023/A:1013713905833

[2] Kim HK, Verpoorte R. Sample preparation for plant metabolomics. Phytochem. Anal. 2010;12:4–13. DOI: 10.1002/pca.1188

[3] Pitt JJ. Principles and applications of liquid chromatography-mass spectrometry in clinical biochemistry. Clin. Biochem. Rev. 2009;30(1):19–34.

[4] Allwood JW, Goodacre R. An introduction to liquid chromatography-mass spectrometry instrumentation applied in plant metabolomic analyses. Phytochem. Anal. 2010;21:33–47. DOI: 10.1002/pca.1187

[5] Want EJ, Nordstrom A, Morita H, Siuzdak G. From exogenous to endogenous: the inevitable imprint of mass spectrometry in metabolomics. J. Proteome Res. 2007;6:459–468. DOI: 10.1021/pr060505+

[6] Schripsema J. Application of NMR in plant metabolomics: techniques, problems, and prospects. Phytochem. Anal. 2010;21:14–21. DOI: 10.1002/pca.1185

[7] van den Berg RA, Hoefsloot H, Westerhuis JA, Smilde AK, van der Werf MJ. Centering, scaling, and transformations: improving the biological information content of metabolomics data. BMC Genomics. 2006;7:142. DOI: 10.1186/1471-2164-7-142

[8] Steuer R, Morgenthal K, Weckwerth W, Selbig J. A gentle guide to the analysis of metabolomic data. In: Weckwerth W. Metabolomics: Methods and Protocols. Methods Mol. Biol. 2007;358:105–126. DOI: 10.1007/978-1-59745-244-1_7

[9] J.M. Quick. R Tutorial Series [Internet]. 2010. Available from: http://rtutorialseries.blogspot.com/

[10] Kabacoff RI. Quick-R: accessing the power of R [Internet]. 2014. Available from: http://www.statmethods.net/index.html

[11] Chang W. Cookbook for R [Internet]. Available from: http://www.cookbook-r.com/

[12] Sanderson A. R plot and graph gallery [Internet]. 2013. Available from: http://www.sr.bham.ac.uk/~ajrs/R/r-gallery.html

[13] Ross Z. Beautiful plotting in R: a ggpot2 cheatsheet [Internet]. 2016. Available from: http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/

[14] Software and Programmer Efficiency Research Group. ggplot2 Quick Reference [Internet]. 2015. Available from: http://sape.inf.usi.ch/quick-reference/ggplot2

[15] Smith CA, Want EJ, O'Maille G, Abagyan R, Siuzdak G. XCMS: processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching and identification. Anal. Chem. 2006;78:779–787. DOI: 10.1021/ac051437y

[16] Tautenhahn R, Patti GJ, Rinehart D, Siuzdak G. XCMS Online: a web-based platform to process untargeted metabolomic data. Anal. Chem. 2012;84:5035–5039. DOI: 10.1021/ac300698c

[17] Gowda H, Ivanisevic J, Johnson CH, Kurczy ME, Benton HP, Rinehart D, Siuzdak G. Interactive XCMS Online: simplifying advanced metabolomic data processing and

subsequent statistical analyses. Anal. Chem. 2014;86(14):6931–6939. DOI: 10.1021/ac500734c

[18] Xia J, Wishart DS. Web-based inference of biological patterns, functions and pathways from metabolomic data using MetaboAnalyst. Nat. Protocol. 2011;6:743–760. DOI: 10.1038/nprot.2011.319

[19] Xia J, Sinelnikov I, Han B, Wishart DS. MetaboAnalyst 3.0 - making metabolomics more meaningful. Nucleic Acids Res. 2015;43:251–257. DOI: 10.1093/nar/gkv380

[20] Grace SC, Embry S, Luo H. Haystack, a web-based tool for metabolomics research. BMC Bioinformatics. 2014;15(Suppl 11):S12. DOI: 10.1186/1471-2105-15-S11-S12

[21] Pluskal T, Castillo S, Villar-Briones A, Oresic M. MZmine 2: Modular framework for processing visualizing, and analyzing mass spectrometry-based molecular profile data. BMC Bioinformatics. 2010;11:395. DOI: 10.1186/1471-2105-11-395

[22] Lei Z, Li H, Chang J, Zhao PX, Sumner LW. MET-IDEA version 2.06: improved efficiency and additional functions for mass spectrometry-based metabolomics data processing. Metabolomics. 2012;8(S1):105–110. DOI: 10.1007/s11306-012-0397-5

[23] Katajamaa M, Orešič M. Data processing for mass spectrometry-based metabolomics. J. Chromatogr. A. 2007;1158:318–328. DOI: 10.1016/j.chroma.2007.04.021

[24] Sugimoto M, Kawakami M, Robert M, Soga T, Tomita M. Bioinformatics tools for mass spectroscopy-based metabolomic data processing and analysis. Curr. Bioinformatics. 2012;7:96–108. DOI: 10.2174/157489312799304431

[25] Scholz M, Selbig J. Visualization and analysis of molecular data. In: Weckwerth W. Metabolomics: methods and protocols. Meth. Mol. Biol. 2007;358:87–104. DOI: 10.1007/978-1-59745-244-1_6

[26] Worley B, Powers R. Multivariate analysis in metabolomics. Curr. Metabolomics. 2013;1:92–107. DOI: 10.2174/2213235X11301010092

[27] Saccenti E, Hoefsloot H, Smilde AK, Westerhuis JA, Hendriks MM. Reflections on univariate and multivariate analysis of metabolomics data. Metabolomics. 2014;10:361–374. DOI: 10.1007/s11306-013-0598-6

[28] Trygg J, Lundstedt T. Chemometrics techniques for metabonomics. The Handbook of Metabonomics and Metabolomics. J.C. Lindon, J.K. Nicholson, E. Holmes (eds.). Amsterdam, The Netherlands, Elsevier B.V. 2007; pp. 171–199.

[29] Levin I, Frankel P, Gilboa N, Tanny S, Lalazar A. The tomato dark green mutation is a novel allele of the tomato homolog of the DEETIOLATED1 gene. Theor. Appl. Genet. 2003;106:454–460. DOI: 10.1007/s00122-002-1080-4

[30] Peters JL, van Tuinen A, Adams P, Kendrick RE, Koornneef M. High pigment mutants of tomato exhibit high sensitivity for phytochrome action. Plant Physiol. 1989;134:661–666. DOI: 10.1016/S0176-1617(89)80024-0

[31] Mustilli AC, Fenzi F, Ciliento R, Alfano F, Bowler C. Phenotype of the tomato high pigment-2 mutant is caused by a mutation in the tomato homolog of DEETIOLATED1. Plant Cell. 1999;11:145–157.

[32] Bino RJ, Ric de Vos CH, Lieberman M, Hall RD, Bovy A, Jonker HH, Tikunov Y, Lommen A, Moco S, Levin I. The light-hyperresponsive high pigment-2dg mutation of tomato: alterations in the fruit metabolome. New Phytol. 2005;166:427–438. DOI: 10.1111/j.1469-8137.2005.01362.x

[33] Mevik BH, Wehrens R. The pls package: principal component and partial least squares regression in R. J. Stat. Softw. 2007;18:1–23. DOI: 10.18637/jss.v018.i02

[34] Gaude E, Chignola F, Spiliotopoulos D, Mari S, Spitaleri A, Ghitti M. muma, an R package for metabolomics univariate and multivariate statistical analysis. Curr. Metabolomics. 2015;1:180–189. DOI: 10.2174/2213235X11301020005

[35] Key M. A tutorial in displaying mass spectrometry-based proteomic data using heatmaps. BMC Bioinformatics. 2012;13(Suppl 16):S10. DOI: 10.1186/1471-2105-13-S16-S10

[36] Yang Z, Nakabayashi R, Okazaki Y, Mori T, Takamatsu S, Kitanaka S, Kikuchi J, Saito K. Toward better annotation in plant metabolomics: isolation and structure elucidation of 36 specialized metabolites from *Oryza sativa* (rice) by using MS/MS and NMR analyses. Metabolomics. 2014;10:543–555. DOI: 10.1007/s11306-013-0619-5

[37] Smith CA, O'Maille G, Want EJ, Qin C, Trauger SA, Brandon TR, Custodio DE, Abagyan R, Siuzdak G. METLIN: a metabolite mass spectral database. Ther. Drug Monit. 2005;27:747–751.

[38] Brown M, Dunn WB, Dobson P, Patel Y, Winder CL, Francis-McIntyre S, Begley P, Carroll K, Broadhurst D, Tseng A, Swainston N, Spasic I, Goodacre R, Kell DB. Mass spectrometry tools and metabolite-specific databases for molecular identification in metabolomics. Analyst. 2009;134:1322–1332. DOI: 10.1039/B901179J

[39] Kind T, Wohlgemuth G, Lee DY, Lu Y, Palazoglu M, Shahbaz S, Fiehn O. FiehnLib: mass spectral and retention index libraries for metabolomics based on quadrupole and time-of-flight gas chromatography/mass spectrometry. Anal. Chem. 2009;81:10038–10048. DOI: 10.1021/ac9019522

[40] Wishart DS, Knox C, Guo AC, Eisner R, Young N, Gautam B, Hau DD, Psychogios N, Dong E, Bouatra S, Mandal R, Sinelnikov I, Xia J, Jia L, Cruz JA, Lim E, Sobsey CA, Shrivastava S, Huang P, Liu P, Fang L, Peng J, Fradette R, Cheng D, Tzur D, Clements M, Lewis A, De Souza A, Zuniga A, Dawe M, Xiong Y, Clive D, Greiner R, Nazyrova A, Shaykhutdinov R, Li L, Vogel HJ, Forsythe I. HMDB: a knowledgebase for the human metabolome. Nucleic Acids Res. 2009;37:D603–D610. DOI: 10.1093/nar/gkn810

[41] Sawada Y, Nakabayashi R, Yamada Y, Suzuki M, Sato M, Sakata A, Akiyama K, Sakurai T, Matsuda F, Aoki T, Hirai MY, Saito K. RIKEN tandem mass spectral database

(ReSpect) for phytochemicals: A plant-specific MS/MS-based data resource and database. Phytochemistry. 2012;82:38–45. DOI: 10.1016/j.phytochem.2012.07.007

[42] Udayakumar M, Prem Chandar D, Arun N, Mathangi J, Hemavathi K, Seenivasagam R. (2012). PMDB: Plant metabolome database-a metabolomic approach. Med. Chem. Res. 2012;21(1):47–52. DOI: 10.1007/s00044-010-9506-z

[43] Bais P, Moon SM, He K, Leitao R, Dreher K, Walk T, Sucaet Y, Barkan L, Wohlgemuth G, Roth MR, Wurtele ES, Dixon P, Fiehn O, Lange BM, Shulaev V, Sumner LW, Welti R, Nikolau BJ, Rhee SY, Dickerson JA. PlantMetabolomics.org: a web portal for plant metabolomics experiments. Plant Physiol. 2010;152:1807–1816. DOI: 10.1104/pp.109.151027

[44] Sucaet Y, Wang Y, Li J, Wurtele ES. MetNet Online: a novel integrated resource for plant systems biology. BMC Bioinformatics. 2012;13:267. DOI: 10.1186/1471-2105-13-267

[45] King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, Ebrahim A, Palsson BO, Lewis NE. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res. 2016;44(D1):D515–D522. DOI: 10.1093/nar/gkv1049