MetPlast Tutorial

Lucio D'Andrea

8/3/2021

Introduction

Plants as sessile organisms are unable to evade unfavorable growing conditions by simply moving away. Hence, they evolved a unique phenotypic plasticity that allows them to better adapt or survive to challenging environments. Metabolic plasticity is the ability of plants to biosynthesize a myriad of specialized compounds that allows them to cope with changes in their immediate surroundings. Thus, specialized metabolites are involved in a wide variety of ecological processes such as herbivorous attack, interaction with neighboring plants, as well as dealing with changes in light, or temperature conditions.

The evolution of plant metabolic plasticity has been mainly driven by gene duplication, (or even whole genome duplication) followed by neo-functionalization. Gene duplication has been proven to shape the evolution of several specialized metabolic pathways. However, the effect of WGD on the metabolic plasticity remains to be elucidated. Possibly, the duplication of the whole genome allowed plants to screen a wider phenotypic space under stress conditions, promoting innovation, rapid adaptation and ultimately, speciation.

Artificial selection processes have also influenced plant metabolic repertoire. Domestication, i.e., the process of selecting plants to increase their suitability to human requirements, as well as crop improvement has caused genetic bottlenecks and massive reduction of the allelic diversity. Thus, artificial selection has introduced quantitative changes in various nutrition compounds. For instance, studies on tomato domestication have shown a major reduction in the levels of the anti-nutritional steroidal glycoalkaloids in ripe fruits. Although, both natural and artificial selection have been pointed as major forces shaping the biosynthesis and accumulation of several specialized metabolites, the evaluation of their effects on the metabolome and metabolic plasticity are in their infancy.

Information theory provides a statistical framework that allows to quantify and evaluate metabolic plasticity. Metabolome diversity and specialization can be calculated based on the Shannon entropy of the metabolic frequency distribution. Shannon entropy is a useful parameter, that measures the information held in a set of data. Thus, its calculation can be used to estimate different parameters associated with a given metabolome: (1) Hj index, metabolome diversity; (2) (Delta)j index, metabolic profile specialization; (3) Si index, metabolic specificity of individual metabolites. The individual calculation of these parameters was successfully applied on LC-MS/MS data to understand the dynamics of different plant species' metabolomes.

Here, we present MetPlast, an R-package that integrates the calculation, and visualization of Shannon-based metabolic plasticity parameters. We evaluate the effect of crop domestication by comparing the proposed parameters between the domesticated Solanum lycopersicum, the semi-domesticated S. lycopersicum var. cerisiforme and the wild relative S. pimpinellifolium.

Upload MetPlast packages

Install and load the MetPlast package from GitHub:

```
# devtools::install_github ("danlucio86/MetPlast")

library("MetPlast")
```

```
## Loading required package: dlookr
```

```
##
## Attaching package: 'dlookr'
```

```
## The following object is masked from 'package:base':
##
##     transform
```

```
## Loading required package: ggplot2
```

```
## Loading required package: ggfortify
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ tibble  3.1.6     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.4     ✓ stringr 1.4.0
## ✓ readr   2.1.1     ✓ forcats 0.5.1
## ✓ purrr   0.3.4
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x tidyr::extract() masks dlookr::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

Upload external packages

This package has several external dependencies that are automatically uploaded with MetPlast package:

1. dlookr
2. ggplot2
3. ggfortify
4. tidyverse
5. data.table
6. gridExtra

Data

Uploading

This package takes as a data set a data frame containing the standardized quantity of different metabolites -measured as peak's intensity- in a given set of samples. It needs to be tidy in order to have the first column the "Compounds", and all the samples as columns. It is important to named the first column as "Compounds".

A data set extract from Zhug et al 2018 can be easily uploaded from the folder Data/Data.rda.

```
Data <- MetPlast::Data
```

This data set contains the metabolic profile of red-fruited tomato populations including three different species: S. lycopersicum, S. lycopercicum var. cersiforme, and S. pimpinellifolium. The data set contains 301 different accessions (2 biological replicates each), expanding across different geographical distribution, passport information, between others. Thus, although through out the tutorial the samples are visualize based ONLY on the species identity, it is absolutely essential to keep in mind the bio-geographical diversity among them.

When the user desires to upload its own data set, it can be use the following commands:

```
##Data <- read.csv2(file = "Test.csv", header = TRUE, row.names = "Compounds")

library(rmarkdown)
paged_table(head(Data))
```

	S..lycopersicum <dbl>	S..lycopersicum.1 <dbl>	S..lycopersicum.2 <dbl>	S..lycopersicum.3 <dbl>
SlFM0001	9561.45	290809.910	7753.55	3.000000e-10
SlFM0002	7465.25	7567.723	15637.75	1.105186e+04
SlFM0003	7248.95	64658.588	1070.10	5.109610e+03
SlFM0004	7178.50	1528529.230	714.00	1.706112e+02
SlFM0006	209964.25	165790.076	173637.20	4.654161e+04
SlFM0007	272210.25	170302.838	485893.45	2.953438e+05

6 rows | 1-6 of 403 columns

Replacing NAs values

When comparing the set of metabolites from different samples, it is pretty usual that some metabolites are sample-specific. Hence, it is expected to have missing values (NAs). In order to deal with this, The NAs values are replaced by the minimum value found in the data set divided by 1e6. The Test.csv provided data set has already been tidy-up.

In case a new data set is used, the following command can be use:

```
Data[is.na(Data)] <- (min(Data, na.rm=TRUE))/1000000

library(rmarkdown)
paged_table(head(Data))
```

	S..lycopersicum <dbl>	S..lycopersicum.1 <dbl>	S..lycopersicum.2 <dbl>	S..lycopersicum.3 <dbl>
SlFM0001	9561.45	290809.910	7753.55	3.000000e-10
SlFM0002	7465.25	7567.723	15637.75	1.105186e+04
SlFM0003	7248.95	64658.588	1070.10	5.109610e+03
SlFM0004	7178.50	1528529.230	714.00	1.706112e+02
SlFM0006	209964.25	165790.076	173637.20	4.654161e+04
SlFM0007	272210.25	170302.838	485893.45	2.953438e+05

6 rows | 1-6 of 403 columns

Metabolic Parameters

MetPlast is structured in four set of functions:

1. MetPlast_int.R : Includes 7 internal functions supports the rest of the functions provided by the package. These functions are only accessible for developers.
2. MetPlast_ind.R : Includes 7 functions that performs individual tasks. While MetDiv(), Si_index(), and Dj_index() calculate and generate data frames with the indicated metabolic parameters; MetDiv_plot(), MetDiv_Supp_plot(), Si_index_plot(), and Dj_index_plot() assist their plotting.
3. MetPlast_sum.R: Includes 5 functions that allows the user to support and summaries, the MetPlast_ind calculations. While MetStats() calculates specific statistical parameters related with MetDiv(); Dj_index_weight() (and its graphical support Dj_index_weight_plot()) allows the user to get a better understanding of Dj_index(). Finally, MetPar_df(), and MetPar_plot() integrates and plot all variables(i.e. samples, or species)-related information, respectively.
4. MetPlast_pipe.R: Pipe functions are shortcuts that integrates several functions provided by MetPlast_ind.R and MetPlast_sum.R. (1) MetDiv_pipe() integrates MetDiv(), MetDiv_plot(), and MetDiv_Supp_plot(). (2) MetSpec_pipe() integrates Si_index(), Si_index_plot(), Dj_index, and Dj_index_plot(). (3) MetliteSpec_pipe() integrates Dj_index_weight(), and Dj_index_weight_plot(). MetliteSpec_pipe() also extracts and plots the metabolites with the highest Pij*Si value per sample. (4) MetPar_pipe() integrates MetPar_df(), and MetPar_plot().

Metabolic Diversity related functions

MetDiv() calculates METabolic Diversity (Hj) index based on Shannon entropy. The metabolic profile diversity is defined as the Shannon entropy using metabolite frequency distribution in a sample (Hj). Hj can take any value between zero when only one metabolite is detected up to log2(m), where all m metabolites are detected and accumulates at the same frequency: 1/m.

```
MetDiv_df <- MetDiv (Data)
```

```
print(head((MetDiv_df)))
```

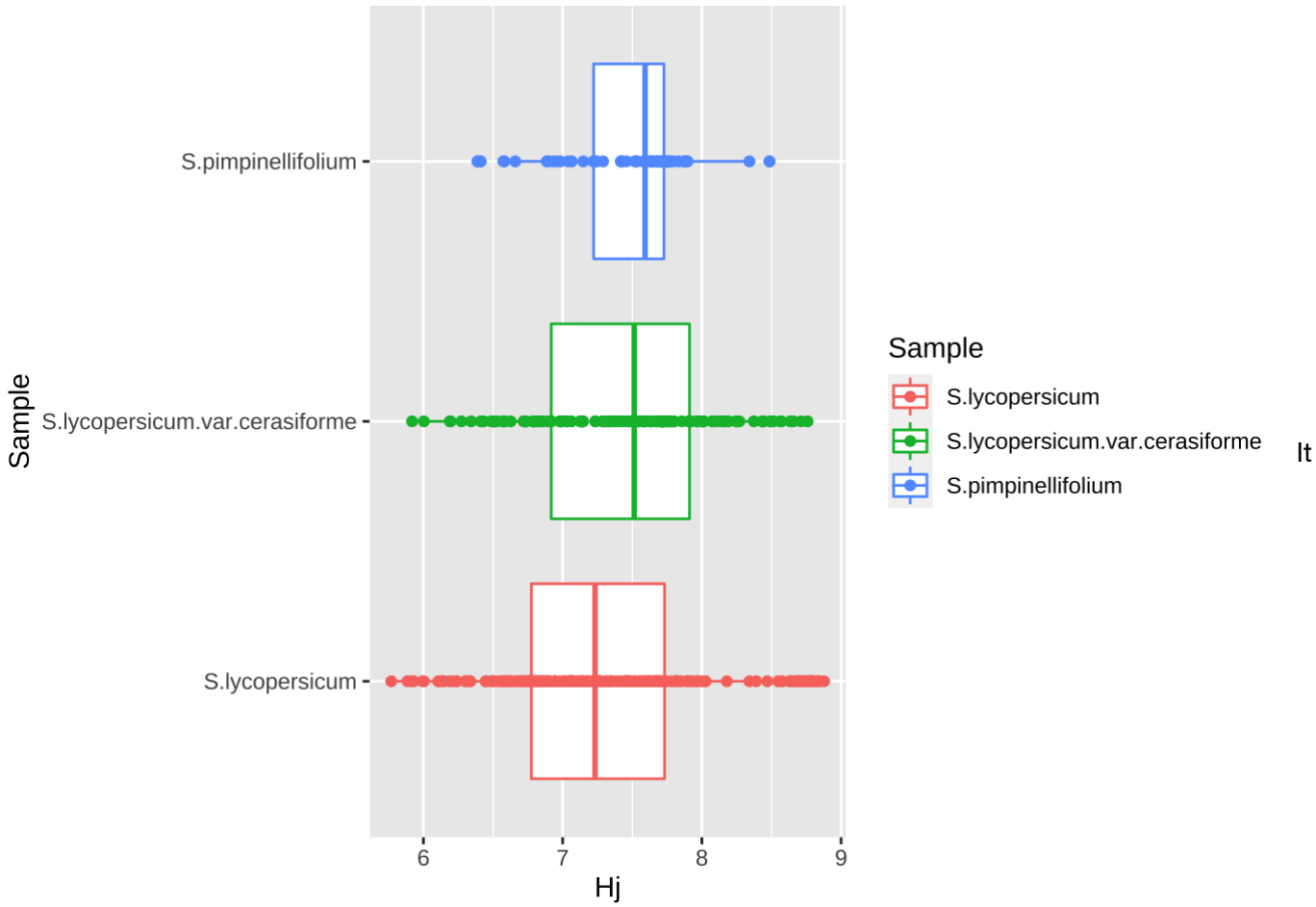
```
##            Sample       Hj numb_peaks
## 1: S.lycopersicum 7.660288        948
## 2: S.lycopersicum 7.634306        976
## 3: S.lycopersicum 7.597500        963
## 4: S.lycopersicum 7.505589        936
## 5: S.lycopersicum 8.006721        965
## 6: S.lycopersicum 7.106506        949
```

In our example the Hj factor can range between 0 - when only one metabolite is detected- and log2(980) = 9,93 - when all the detected metabolites accumulates at the same frequency.

The data in the data frame can then be visualize:

```
MetDiv_plot <- MetDiv_plot(MetDiv_df =  MetDiv_df)

plot(MetDiv_plot)
```



can be observe that the domesticated S. lycopersicum Hj indexes display a higher variance, ranging from 5.76 to 8.87, compare with S. lycopersicum var. cersiformis (5.91 - 8.75) and the wild relative S. pimpinellifolium

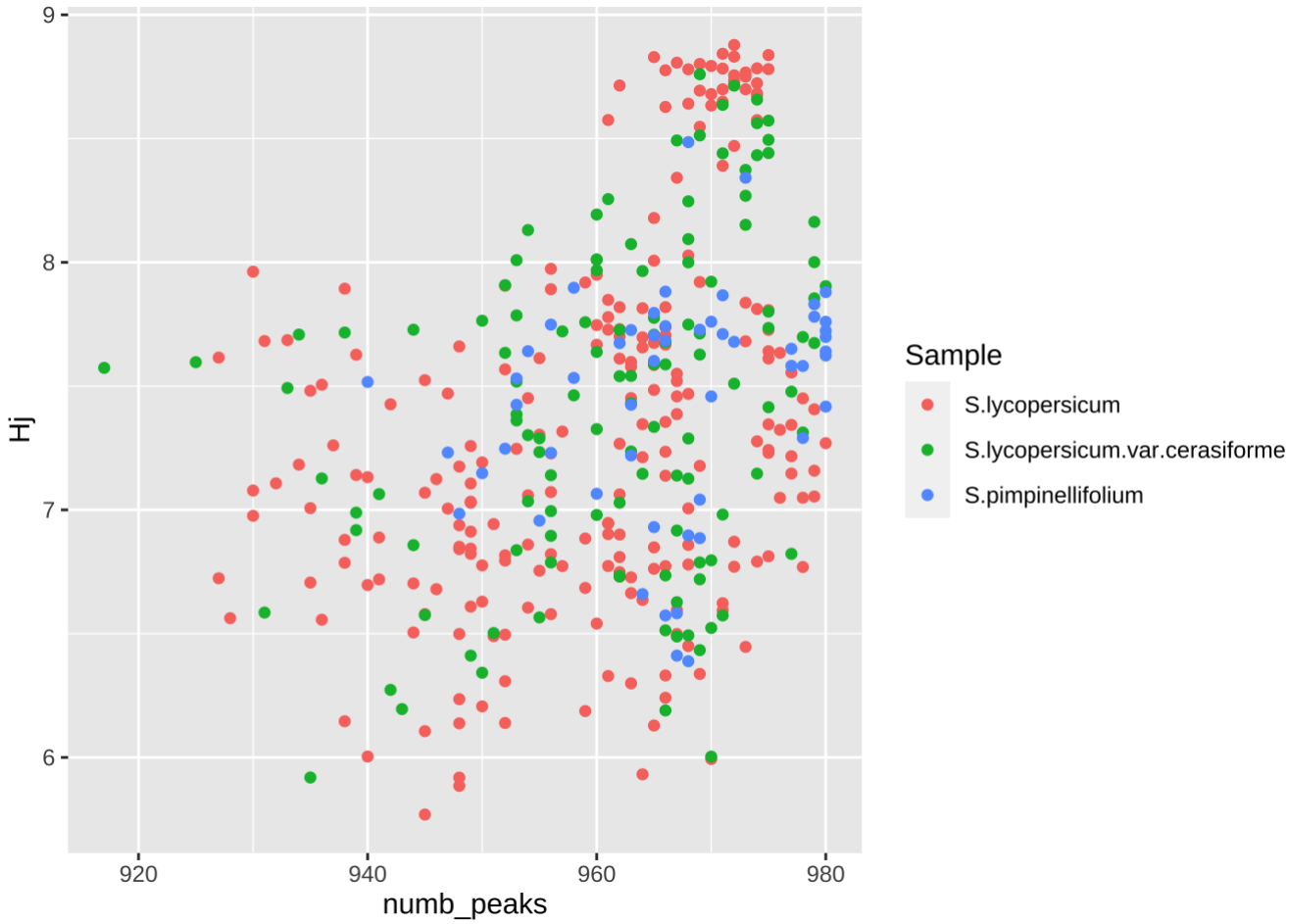
(6.38 - 8.48). These differences might be due to technical and biological reasons, such as sampling, accessions and the domestication process. Additionally, a few outliers can be observed.

The median values show that there is a higher Hj median value when the degree of domestication is lower. Possibly indicating that in average and considering this data set, it is more likely to get a more diverse metabolome as the degree of domestication is lower.

```
MetDiv_Supp_plot_1 <- MetDiv_Supp_plot(MetDiv_df = MetDiv_df, y_element = Hj)
```

```
## [1] "This function creates a plot with number of peaks per sample as x element"
```

```
plot(MetDiv_Supp_plot_1)
```

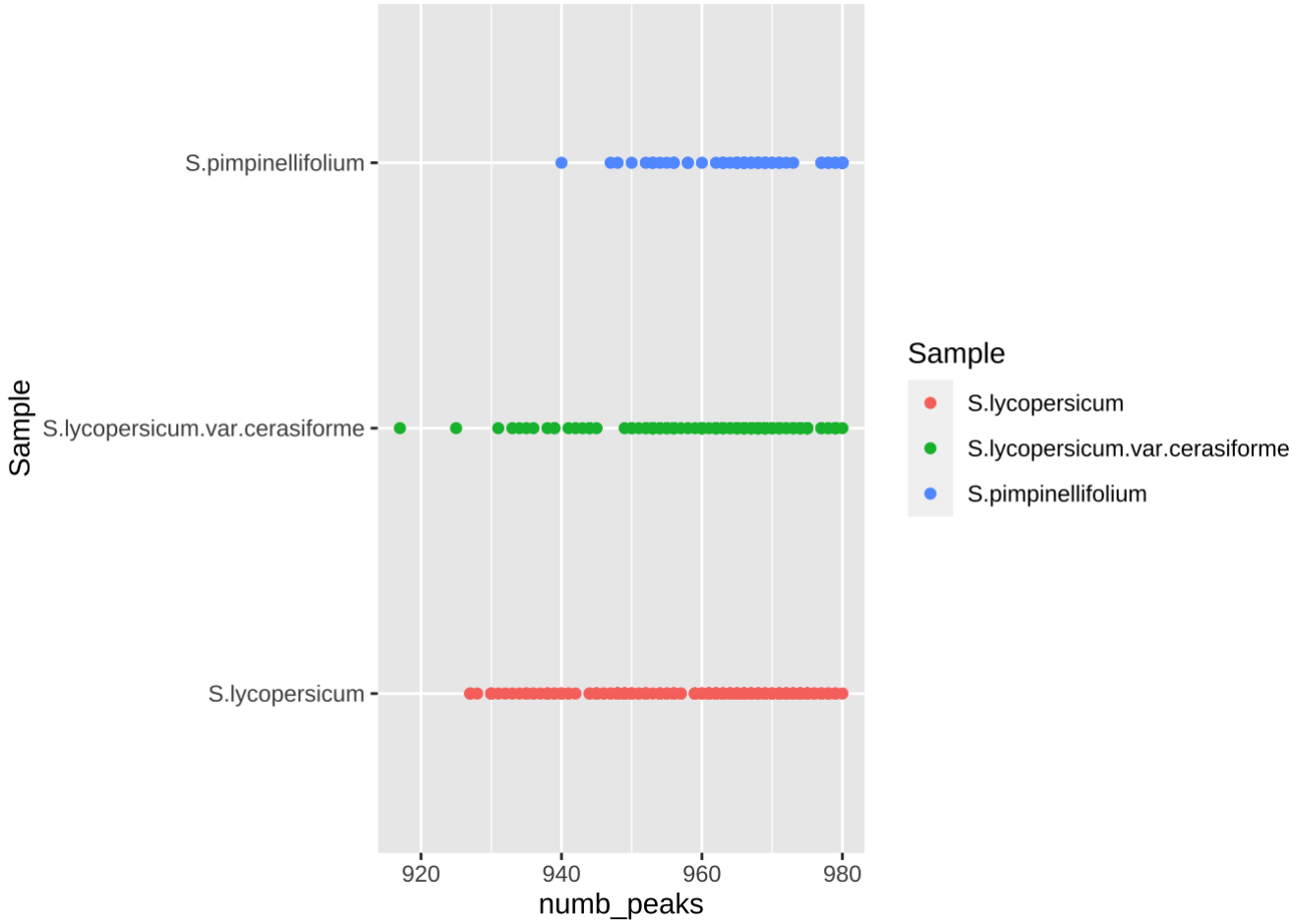


The Hj factors depends mostly on two different parameters: (a) The number of peaks, (b) the frequency of each peak in the whole data set. This plot shows to what extend the Hj increases based on the number of peak in the species under evaluation. In general, it can be expected to observe a curve where the Hj values reaches a plateau.

```
MetDiv_Supp_plot_2 <- MetDiv_Supp_plot(MetDiv_df = MetDiv_df, y_element = Sample)
```

```
## [1] "This function creates a plot with number of peaks per sample as x element"
```

```
plot(MetDiv_Supp_plot_2)
```



As mention before he Hj factors depends mostly on two different parameters including the number of peaks. In this plot we can observe that the variation on the compounds detected in the different species have a similar behavior as the Hj factor. Hence, we can infer that the higher level of metabolic diversity in the less domesticated species, could be related with a lost in the capacity of synthesizing certain compounds.

Summary: These differences might be related with the geographical origins, consumption type or improvement status in the S. lycopersicum and S.lycopersicum.var.cerasiformis. To evaluate this possibility we can include categorical data. As an example please, the user can upload the file categories.csv, and store it as a vector called “categories”, or simply upload it from the file data/categories.rda.ç

```
categories <- MetPlast::categories
```

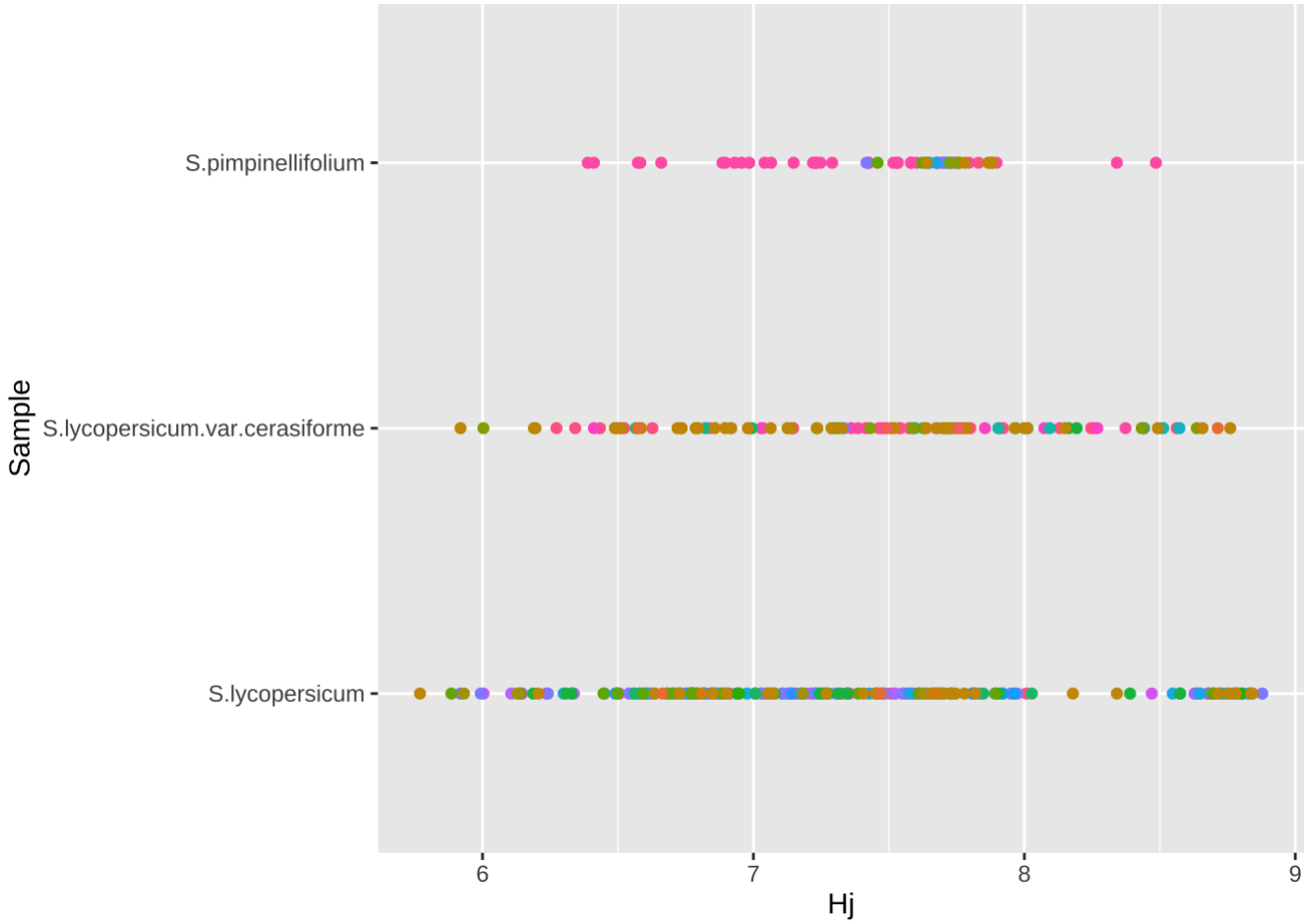
```
## categories <- read.csv2(file="categories.csv")
```

Then, it can be added to the data frame generated by MetDiv()

```
Data_categ <- cbind (MetDiv_df, categories)
```

Finally, the same plots generated by MetDiv() can be generated using the package ggplot2. As an example:

```
ggplot(Data_categ, aes(Hj, Sample, color = Categories)) + geom_point() + theme(legen
d.position = "none")
```



This analysis shows that the categorical data used explain at some extend the high variance observe within some species.

Metabolic Specialization related functions

Metabolite Specialization Index (Si)

Metabolic specificity (Si) is defined as the specificity of a particular metabolite (i) among a set of samples (j).

```
Si_df <- Si_index(Data)
```

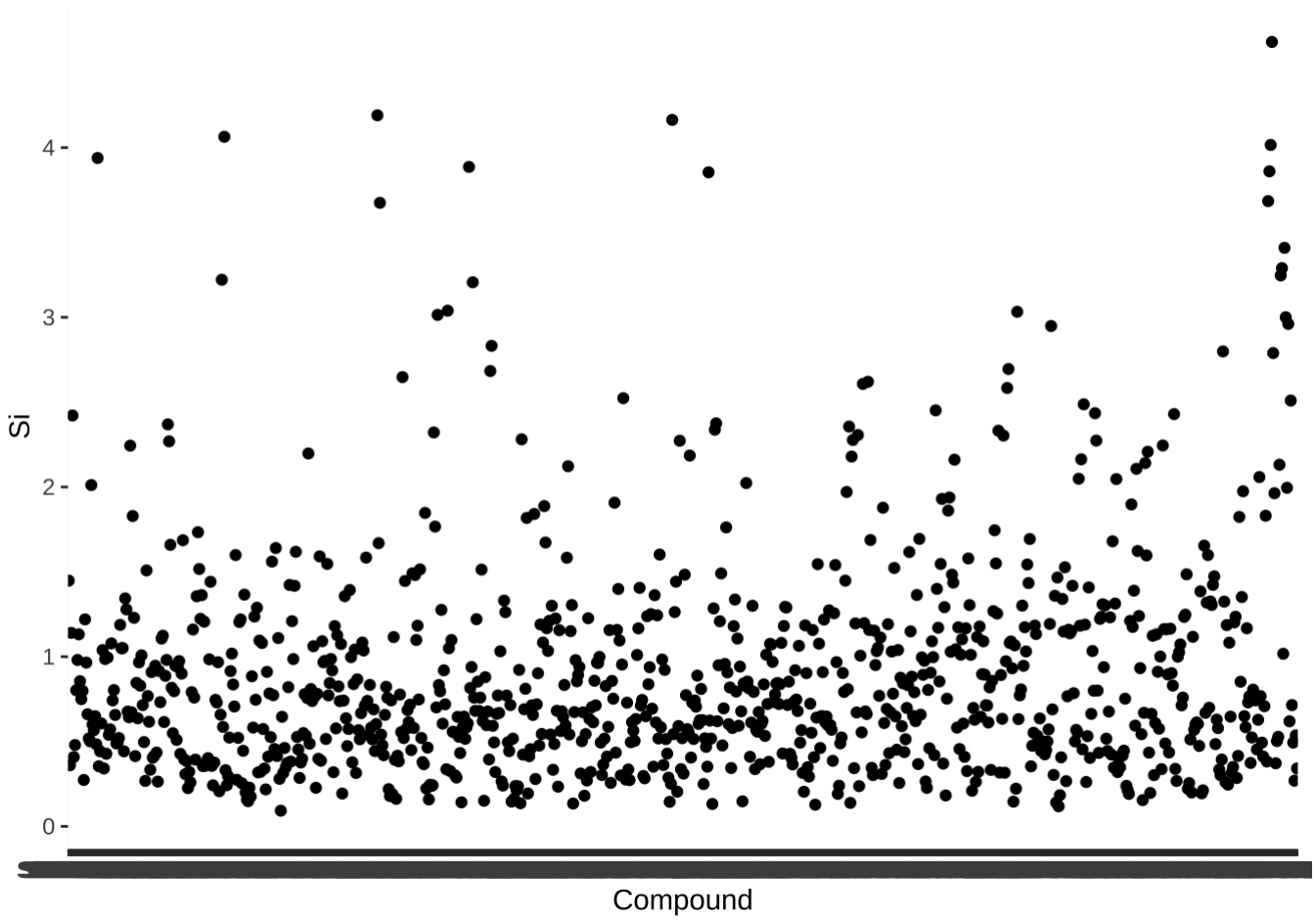
```
print(head((Si_df)))
```

```
##    Compound        Si
## 1: SlFM0001 1.4480607
## 2: SlFM0002 0.3598622
## 3: SlFM0003 1.1401669
## 4: SlFM0004 2.4211992
## 5: SlFM0006 0.4082558
## 6: SlFM0007 0.4792751
```

Si will be zero if the metabolite accumulates at the same frequency in all samples and will be maximum with log2(m), i.e. if the metabolite exclusively accumulated in a single sample. In our example, theoretically range between 0 and log2(980)=9.93 These values indicate how specific a metabolite is in a given data set. A closer look to the values, shows that Si ranges from 0.092 (m = SlFM0252) to 4,62 (m = SlFM1980), indicating that the latter was detected in a fewer samples compare with SlFM0252.

```
Si_plot <- Si_index_plot(Si_df = Si_df)

plot(Si_plot)
```



This dot plot is a visual representation of the Si_df data frame, where each metabolite specificity is quantified (Si). It can be observe that the vast majority of the compounds has a Si value between 0 and 1, indicating that most the metabolites accumulates at average levels in the different samples.

Metabolome Specialization Index (δj)

METabolome SPECialization (δj) index. Metabolome specialization δj is measured for each jth sample, as the average of the metabolite specificity.

```
Dj_df <- Dj_index(Data = Data)
```

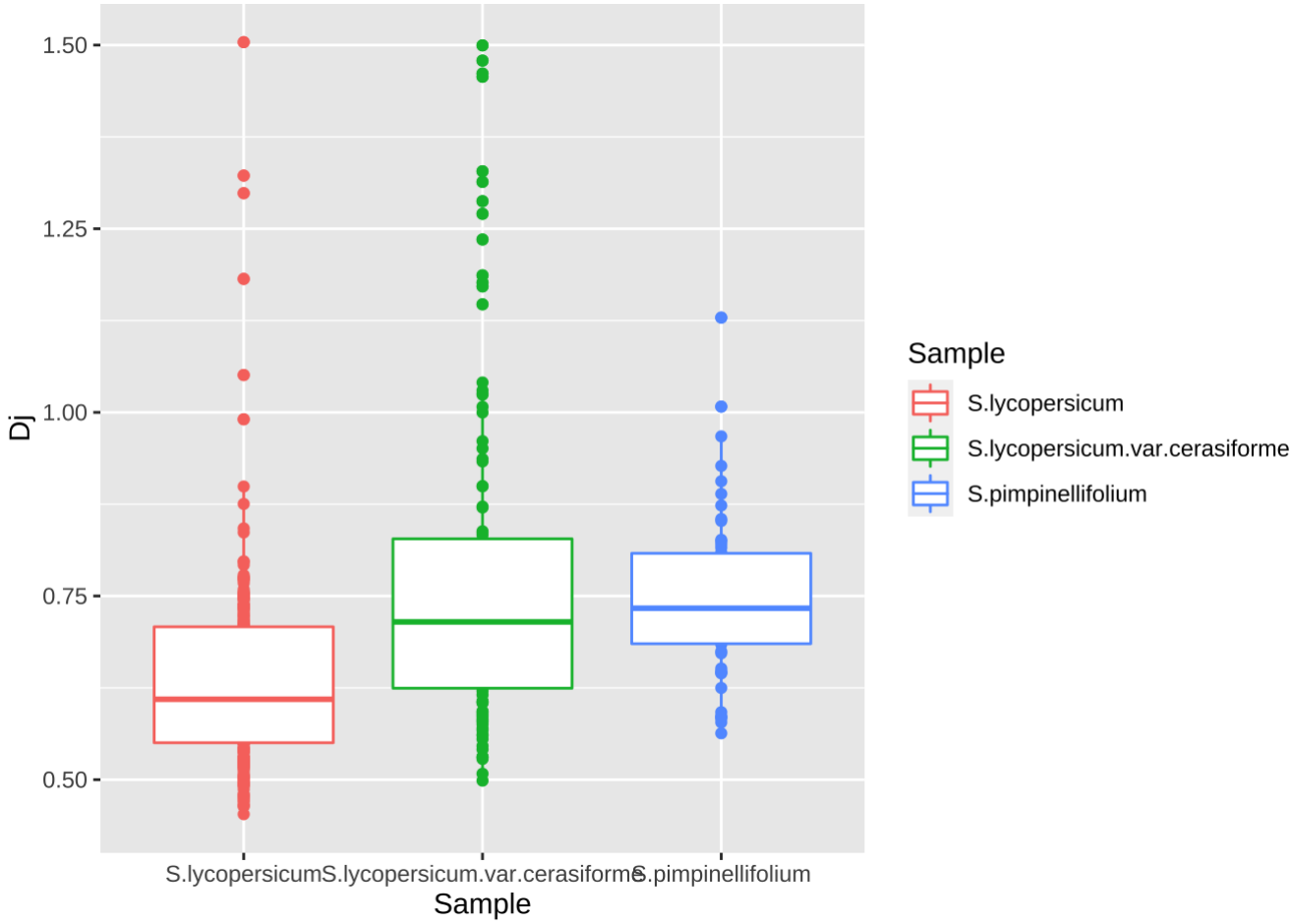
```
print(head((Dj_df)))
```

```
##            Sample        Dj
## 1: S.lycopersicum 0.5904608
## 2: S.lycopersicum 0.6552455
## 3: S.lycopersicum 0.6507864
## 4: S.lycopersicum 0.5651353
## 5: S.lycopersicum 0.7064863
## 6: S.lycopersicum 0.5713923
```

Dj_index() allows the user to obtain a data frame with each sample metabolome specialization value (Dj -δj-). These values indicate how specialized a metabolome is, across a given data set. δj varies from 0 if all metabolites that accumulates in the sample are completely unspecific (Si = 0 for all) up to a maximum of log2(m), when all metabolites accumulating in a sample are not synthesized anywhere else.

```
Dj_plot <- Dj_index_plot(Dj_df = Dj_df)

plot(Dj_plot)
```



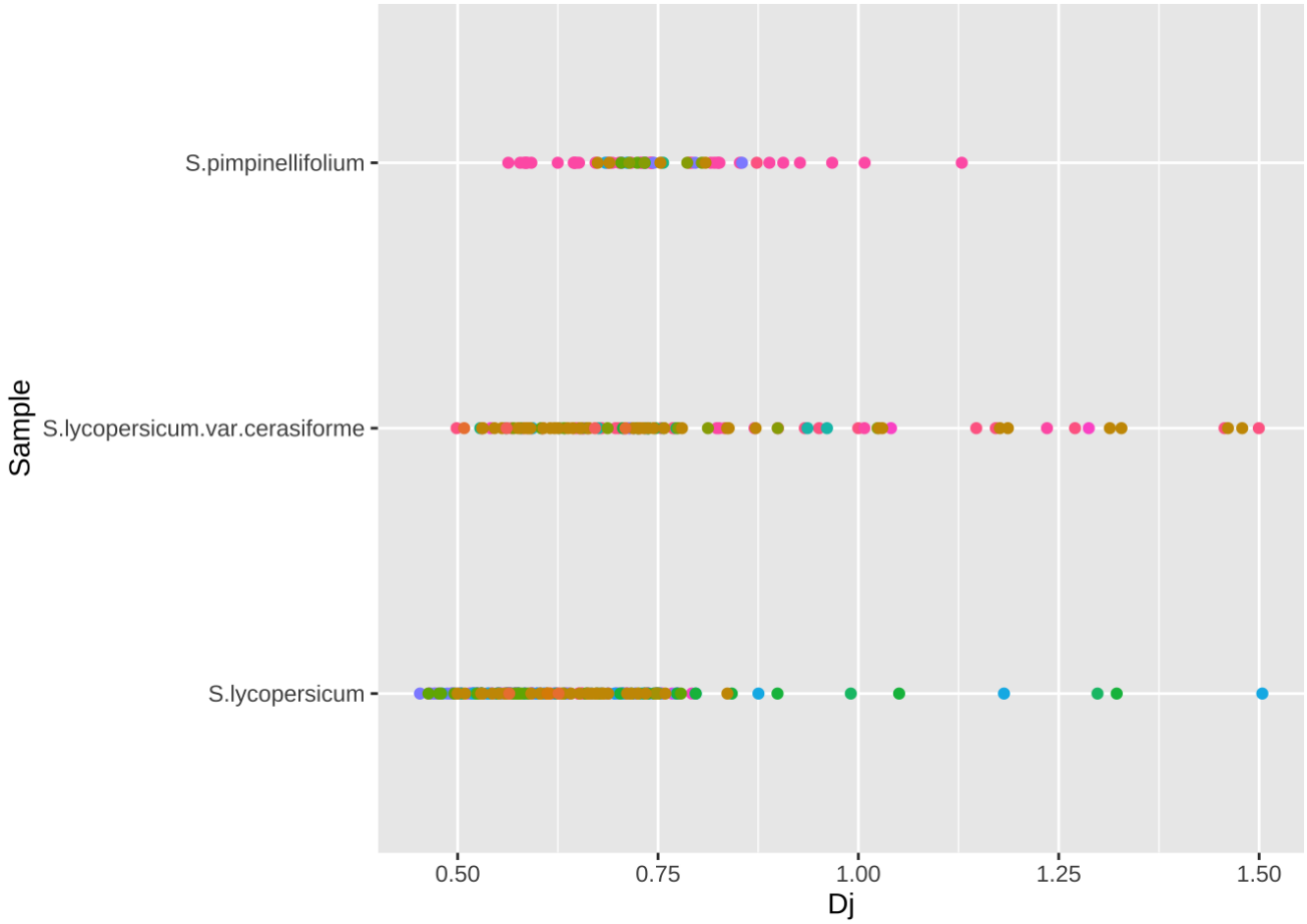
This box plot is a visual representation of the Dj_index() data frame, where the metabolome specialization (Dj) index of each species are calculated. In our example, there is a higher variation in the domesticated or semi-domesticated species compared with the wild relative S. pimpinellifolium. Thus, we might need to consider some categorical data better describing the samples to confidently assess differences in the specialization levels:

Then, we can add the previosly uploaded categorical data in the Dj_df data frame:

```
Dj_categ <- cbind (Dj_df, categories)
```

Finally, the similar plots generated by Dj_index_plot() can be generated using the package ggplot2. As an example:

```
ggplot(Dj_categ, aes(Dj, Sample, color = Categories)) + geom_point() + theme(legend.p
osition = "none")
```



Metabolite Specialization Analysis

Dj_index_weight() calculates the contribution of the metabolite specialization factor (Pij.Si) to the Metabolome Specialization index.

We defined the Metabolite Specialization factor (Pij.Si) as the product of the metabolite specilization index (Si) and the frequency of the metabolite in a given sample (Pij).

```
Dj_weight_df <- Dj_index_weight(Data = Data, Si_df = Si_df)
```

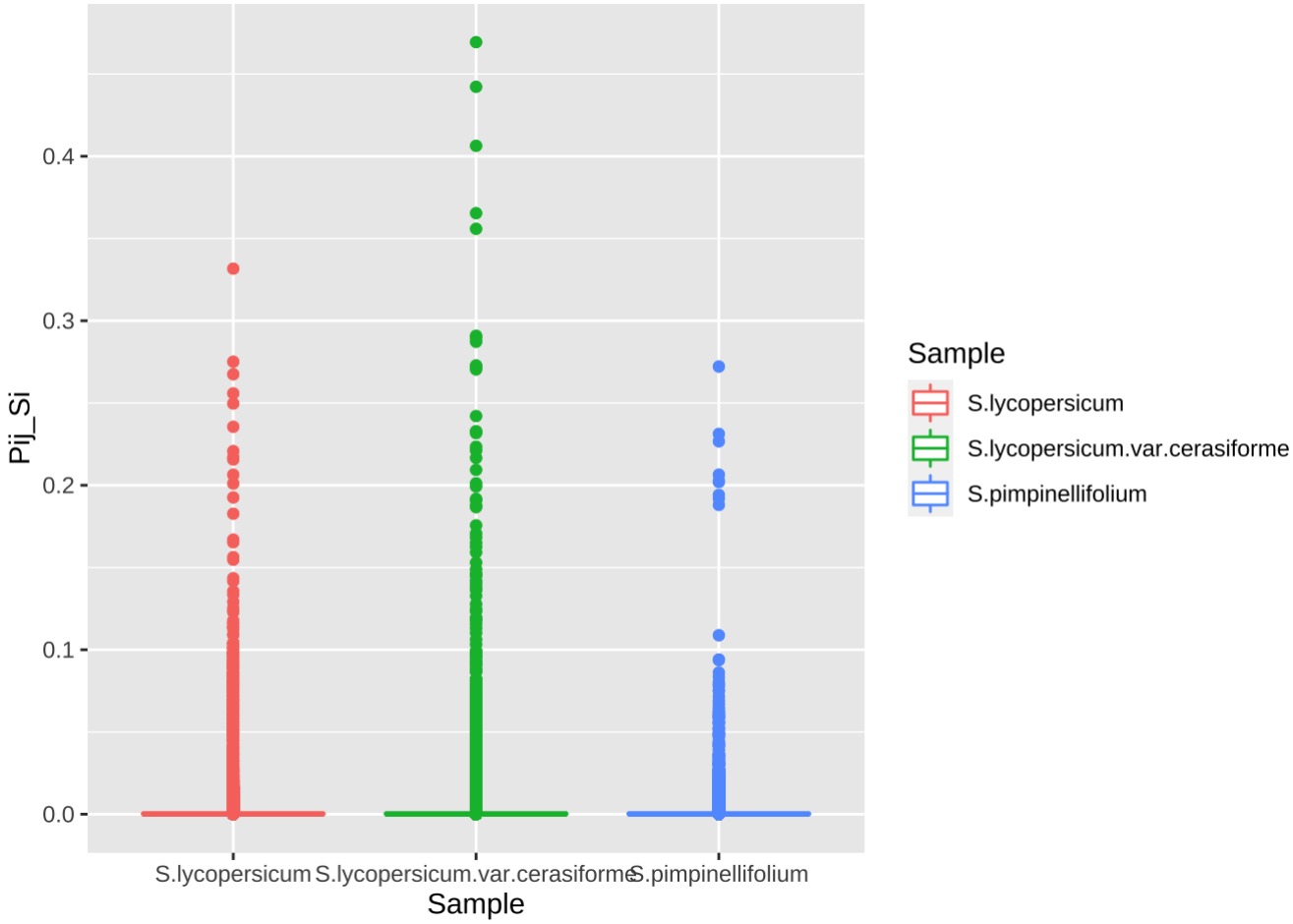
```
print(head((Dj_index_weight)))
```

```
##
## 1 function (Data, Si_df)
## 2 {
## 3     print("THIS PARAMETER CAN NOT BE EXTRAPOLATED TO OTHER DATA SETS OR SUBSET
S")
## 4     if (nrow(Data) < ncol(Data)) {
## 5         warning("MetPlast functions takes as Data argument a data frame with com
pounds as observations -rows- and individual samples as variables -columns-")
## 6     }
```

This information can be easly plot using the function:

```
Dj_weight_plot <- Dj_index_weight_plot (Dj_weight_df = Dj_weight_df)

plot(Dj_weight_plot)
```



This box plot depicting the Pij*Si value of each species shows the contribution of each metabolite in the specialization index of each metabolome. This plot not only provides a general overview of each metabolite contribution to the Dj, but also facilitate the visualization of some important features such as clustering. The most obvious case happens in S. pimpinellifolium, where two set of metabolite specialization clusters can be observe. Hence, it can be infer that the high Dj value is mostly due to an small amount of highly specific metabolites.

Metabolic Diversity Statistics

MetPlast package calculates two statistical parameters using the MetStats() function. MetStats() generates a table summarizing the Hj, number of peaks, and the divergence associated to each Hj (HRj) and the Kullback–Leibler divergence (Divj).

```
Stats <- MetStats(Data = Data, MetDiv_df= MetDiv_df)
```

```
print(head((Stats)))
```

```
##           Sample       Hj numb_peaks      HRj      Divj
## 1 S.lycopersicum 7.660288        948 8.138887 0.4785987
## 2 S.lycopersicum 7.634306        976 8.198506 0.5642006
## 3 S.lycopersicum 7.597500        963 7.998197 0.4006972
## 4 S.lycopersicum 7.505589        936 8.107272 0.6016825
## 5 S.lycopersicum 8.006721        965 8.464697 0.4579765
## 6 S.lycopersicum 7.106506        949 7.602010 0.4955047
```

The data frame includes two specific statistical parameters:

a. HRj measures the divergence with respect to the whole average metabolome. HRj will be equal to or larger than the corresponding Hj.

b. Divj is define as the Kullback–Leibler divergence of the sample j. Divj measures how much a given sample j departs from the corresponding metabolome distribution of the whole system.

Metabolic Plasticity Parameters Summary

This function generates a table summarizing the Hj, number of peaks, Dj and statistical parameters per species.

It uses the data frames generated by MetDiv() and MetSpec(), a generates a new data frame that allows the pairwise comparison of the different parameters.

It returns a list with two objects explained below.

```
MetPar <- MetPar_df(Stats_df = Stats, Dj_df = Dj_df)
```

```
print(head((MetPar)))
```

```
##           Sample       Hj numb_peaks      HRj      Divj        Dj
## 1 S.lycopersicum 7.660288        948 8.138887 0.4785987 0.5904608
## 2 S.lycopersicum 7.634306        976 8.198506 0.5642006 0.6552455
## 3 S.lycopersicum 7.597500        963 7.998197 0.4006972 0.6507864
## 4 S.lycopersicum 7.505589        936 8.107272 0.6016825 0.5651353
## 5 S.lycopersicum 8.006721        965 8.464697 0.4579765 0.7064863
## 6 S.lycopersicum 7.106506        949 7.602010 0.4955047 0.5713923
```

This data frame can be use to evaluate the statistical significance of the observed differences:

1. To evaluate the Hj differences among species:

```
# Statistical Analysis

## ANOVA analysis

library (ggpubr)

compare_means(Hj ~ Sample, data = MetPar)
```

```
## # A tibble: 3 × 8
##   .y.   group1            group2               p p.adj p.format p.signif method
##   <chr> <chr>             <chr>            <dbl> <dbl> <chr>    <chr>    <chr>
## 1 Hj    S.lycopersicum    S.lycopersicum.… 0.0902  0.18 0.090    ns       Wilco…
## 2 Hj    S.lycopersicum    S.pimpinellifol… 0.0424  0.13 0.042    *        Wilco…
## 3 Hj    S.lycopersicum.v… S.pimpinellifol… 0.949   0.95 0.949    ns       Wilco…
```

2. To evaluate the Dj differences among species:

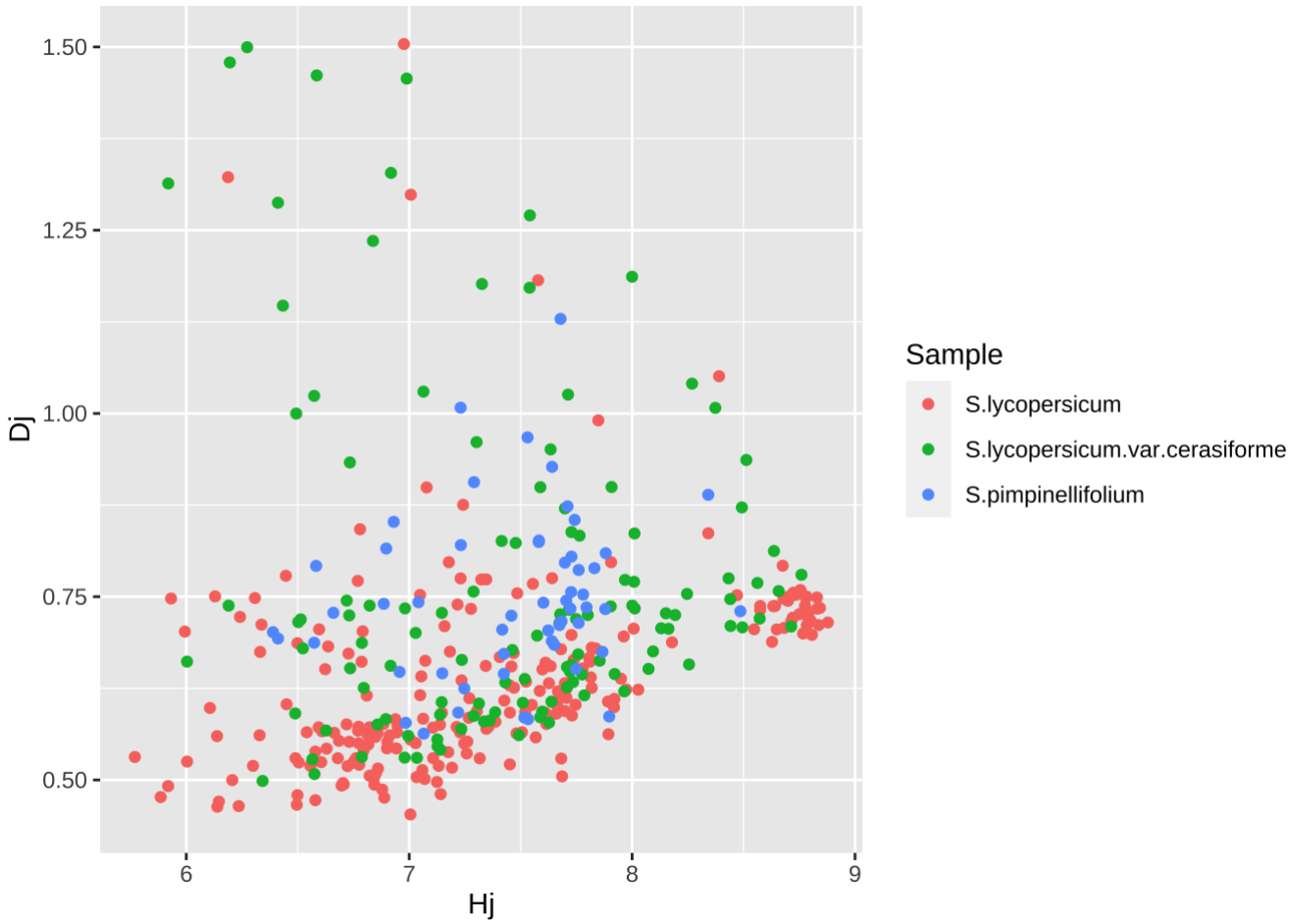
```
compare_means(Dj ~ Sample, data = MetPar)
```

```
## # A tibble: 3 × 8
##   .y.   group1          group2                p    p.adj p.format p.signif method
##   <chr> <chr>           <chr>             <dbl>    <dbl> <chr>    <chr>    <chr>
## 1 Dj    S.lycopersicum  S.lycopersicu… 1.41e-10  3.7e-10 1.4e-10  ****     Wilco…
## 2 Dj    S.lycopersicum  S.pimpinellif… 1.22e-10  3.7e-10 1.2e-10  ****     Wilco…
## 3 Dj    S.lycopersicu… S.pimpinellif… 2.84e- 1  2.8e- 1 0.28     ns       Wilco…
```

MetPar_plot() allows to graphically explore the dependency of Dj and the other variables in the MetPar_df() data frame:

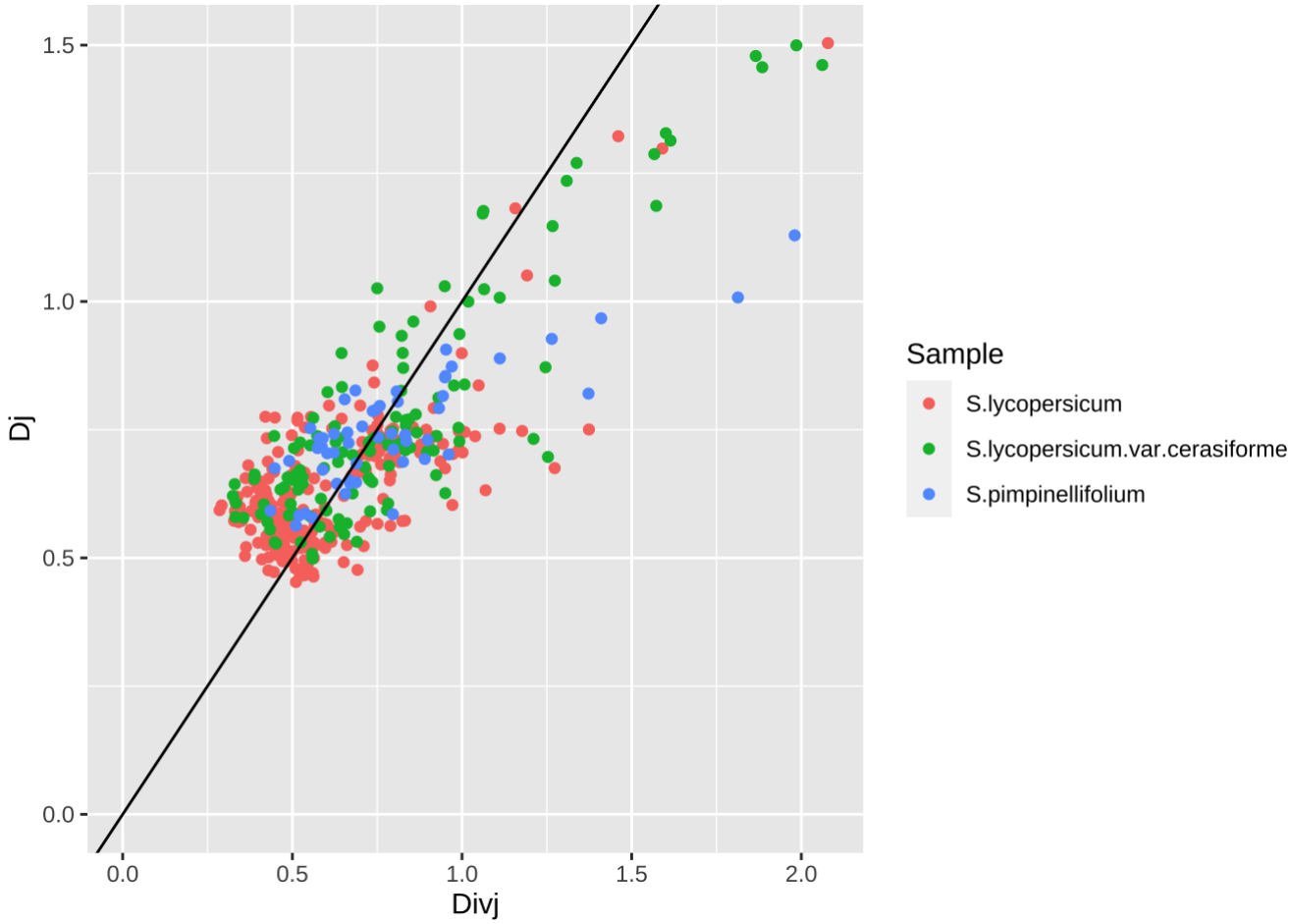
```
Dj_vs_Hj_plot <- MetPar_plot(MetPar_df = MetPar, x_element = Hj, ref_line = "FALSE")
```

```
## [1] "DEFAULT, ref_line=FALSE. Please, for x=y reference line, select = TRUE"
```



This dot plot shows the dependency between the metabolic diversity (Hj) and the metabolome specialization (Dj) in each sample. Although, there is not a clear trend, this plot suggests that samples that belongs to different species show a different relationship between Hj and Dj.

```
Divj_vs_Hj_plot <- MetPar_plot(MetPar_df = MetPar, x_element = Divj, ref_line = TRUE)
```



A scatter plot showing Metabolome Specialization (Dj) vs Divergence (Divj). This plot allows to observe the different specialization strategies of each sample. Samples with Dj > Divj are above the black line that marks Dj=Divj, whereas samples with Dj < Divj are below that line. Samples with Dj > Divj have a specialization strategy that consists mainly of accumulating highly specialized metabolites, whereas samples with Dj < Divj achieve their specialization by accumulating at higher or lower levels metabolites that are, on average, accumulating in the whole system. The distance of each point (sample) to the line Divj = Dj denotes how extreme is the specialization strategy.

Pipe functions

Pipe functions integrate the individual functions previously exposed in this tutorial. Here, we are going to show how to use them, but for the sake of simplicity the results obtained from them are not going to be printed.

```
MetDiv <- MetDiv_pipe(Data = Data)
```

```
MetSpec <- MetSpec_pipe(Data = Data)
```

```
MetliteSpec_pipe <- MetliteSpec_pipe(Data = Data)
```

```
MetPar_pipe <- MetPar_pipe(Stats_df = Stats, Dj_df = Dj_df)
```