LipidSigR tutorial

Wei-Chung Cheng [1] ***and Wen-Jen, Lin*** [2]

[1]China Medical University
[2]China Medical University

April 11, 2022

Abstract

"LipidSigR" is an R package developed based on LipidSig web-based tool http://www.chenglab.cmu.edu.tw/lipidsig/. This package integrates a comprehensive analysis for streamlined data mining of lipidomic datasets. We provide 4 main analysis workflows, which is "Profiling", "Differential expression", "Machine learning", and "Correlation". Each section provides unique aspects to analyze the lipidome profiling data based on different characteristics including lipid class, chain length, unsaturation, hydroxyl group, and fatty acid composition.

Package

LipidSigR 0.99.0

Contents

Introduction of LipidSigR

Lipidomics technology provides a fast and high-throughput screening to identify thousands of lipid species in cells, tissues, or other biological samples and has been broadly used in several areas of studies.

"LipidSigR" is an R package developed based on LipidSig web-based tool (http://www.chenglab.cmu.edu.tw/lipidsig/) [1]. This package integrates a comprehensive analysis for streamlined data mining of lipidomic datasets. We provide 4 main analysis workflows, which is **"Profiling"**, **"Differential expression"**, **"Machine learning"**, and **"Correlation"**. Each section provides unique aspects to analyze the lipidome profiling data based on different characteristics including lipid class, chain length, unsaturation, hydroxyl group, and fatty acid composition.

The usage of the 4 sections is introduced briefly as below.

1. **Profiling**: The profiling section provides an overview of comprehensive analyses for you to efficiently examine data quality, the clustering of samples, the correlation between lipid species, and the composition of lipid characteristics.
2. **Differential expression**: The differential expression section integrates many useful lipid-focused analyses for identifying significant lipid species or lipid characteristics.
3. **Machine learning**: The machine learning section provides a broad variety of feature selection methods and classifiers to build binary classification models. Furthermore, the subsequent analyses can help users to evaluate the learning algorithm's performance and to explore important lipid-related variables.
4. **Correlation**: The correlation section illustrates and compares the relationships between different clinical phenotypes and lipid features.

The functions and usage of the above 4 sections are orderly described in the following sections, which is Profiling in Section 1, Differential expression in Section 2, Machine learning in Section 3, and Correlation in Section 4.

Installation

This section briefly describes the procedures of running the **LipidSigR** package on your system. We assume that you have already installed the R program (see the R project at http://www.r-project.org) and are familiar with it. You need to have R 4.0.0 or a later version installed for running LipidSigR.

The **LipidSigR** package is available at the Bioconductor repository (http://www.bioconductor.org). To install our package, first, you need to install the core Bioconductor packages. If you have already installed the Bioconductor packages on your system, you can skip the following step below.

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install()
```

Once the core Bioconductor packages have been installed, we can begin the installation of the LipidSigR package.

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("LipidSigR")
```

After conducting the above step, now you can load in our package and start using it!

```
> library(LipidSigR)
```

1 Profiling

On the first step of analyzing lipid data, we have to take an overview of the data. In this section, you can get comprehensive analyses to explore the quality and the clustering of samples, the correlation between lipids and samples, and the expression and composition of lipids.

1.1 Input data

First, we have to read the input data needed for the profiling section. We have to prepare lipid expression data and lipid characteristics (optional) as the input data of `exp_data` and `lipid_char_table`. Please note that `lipid_char_table` only be used in Section 1.5.

```
> # clears all objects from workspace
> rm(list = ls())
> # lipid expression data
> data("profiling_exp_data")
> exp_data <- profiling_exp_data
> head(exp_data[, 1:5], 5)

     feature control_01 control_02 control_03 control_04
1 Cer 38:1;2  0.1167960  0.1638070  0.1759450  0.1446540
2 Cer 40:1;2  0.7857833  0.9366095  0.8944465  0.8961396
3 Cer 40:2;2  0.1494030  0.1568970  0.1909800  0.1312440
4 Cer 42:1;2  1.8530153  2.1946591  2.6377576  2.3418783
5 Cer 42:2;2  1.3325520  1.2514943  1.9466750  1.2948319

> # lipid characteristics table (only use in Section 3.5)
> data("profiling_lipid_char_table")
> lipid_char_table <- profiling_lipid_char_table
> head(lipid_char_table[, 1:4], 5)

     feature class structural_category functional_category
1 Cer 38:1;2   Cer                  SL                 MEM
2 Cer 40:1;2   Cer                  SL                 MEM
3 Cer 40:2;2   Cer                  SL                 MEM
4 Cer 42:1;2   Cer                  SL                 MEM
5 Cer 42:2;2   Cer                  SL                 MEM
```

After importing the input data, sometimes, we may need to conduct data processing before analysis. Here, we provide the `data_process` function for data processing, including removing features with missing values, missing values imputation, percentage transformation, log10 transformation, etc.

```
> # lipid expression data
> head(exp_data[, 1:5], 5)

     feature control_01 control_02 control_03 control_04
1 Cer 38:1;2  0.1167960  0.1638070  0.1759450  0.1446540
2 Cer 40:1;2  0.7857833  0.9366095  0.8944465  0.8961396
3 Cer 40:2;2  0.1494030  0.1568970  0.1909800  0.1312440
4 Cer 42:1;2  1.8530153  2.1946591  2.6377576  2.3418783
5 Cer 42:2;2  1.3325520  1.2514943  1.9466750  1.2948319
```

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, zero2what='min',
+                                     xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE, trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # exp_data after data processing
> head(exp_transform_table[, 1:5], 5)

     feature control_01 control_02 control_03 control_04
1 Cer 38:1;2  -2.820387  -2.705816  -2.735971  -2.769140
2 Cer 40:1;2  -1.992512  -1.948590  -2.029794  -1.977095
3 Cer 40:2;2  -2.713455  -2.724534  -2.700360  -2.811391
4 Cer 42:1;2  -1.619936  -1.578781  -1.560113  -1.559906
5 Cer 42:2;2  -1.763130  -1.822719  -1.692055  -1.817257
```

1.2 Cross-sample variability

Now, let's start with a simple view of sample variability to compare the amount/expression difference of lipid between samples (i.e., patients vs. control).

```
> # conduct profiling
> profiling_result <- exp_profiling(exp_data)
```

After conduct the above code, you will get a list `profiling_result` with three types of distribution plots.

```
> # view result: histograms (number of expressed lipids)
> profiling_result$i.expr.lip
```

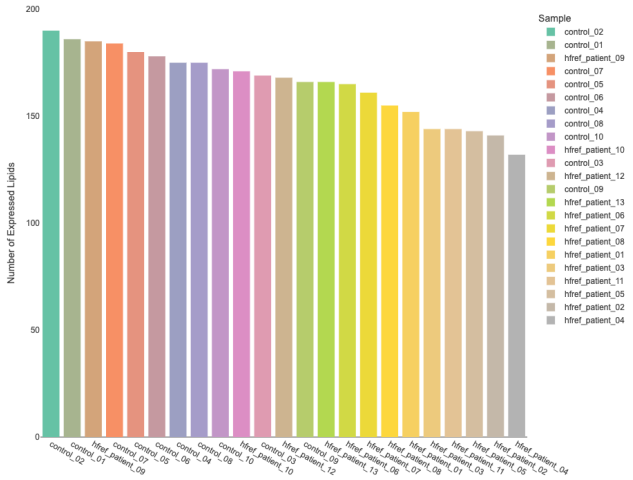

Figure 1: Histogram of number of expressed lipids
The histogram overviews the total number of lipid species over samples. From the plot, we can discover the number of lipid species present in each sample.

```
> # view result: histogram (total amount of lipid)
> profiling_result$i.p.amount
```

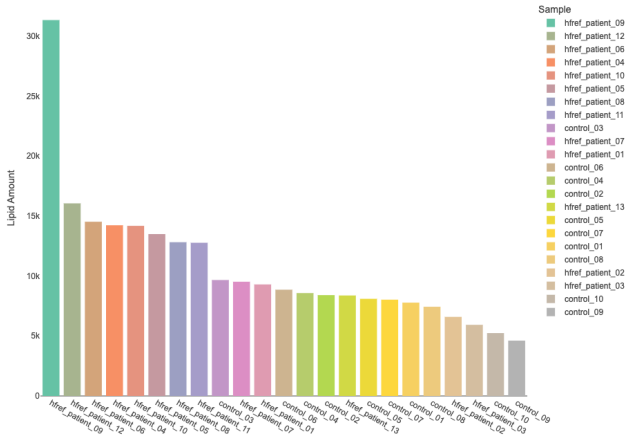

Figure 2: Histogram of lipid amount
The histogram describes the variability of the total lipid amount between samples.

```
> # view result: density plot of expression distribution
> profiling_result$p.hist.value
```

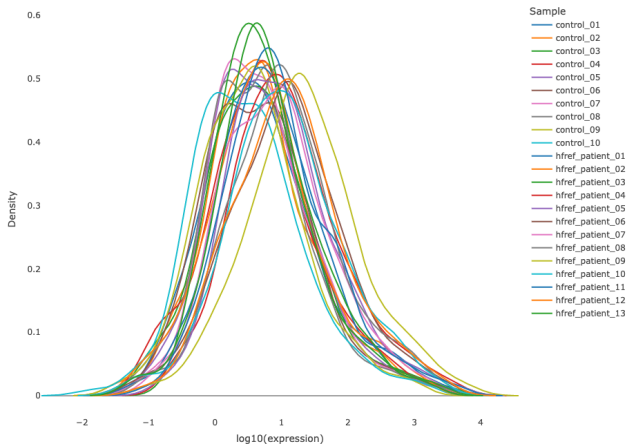

Figure 3: Density plot of expression distribution
The density plot uncovers the distribution of lipid expression in each sample (line). All expression was log10 transformed. From this plot, we can have a deeper view of the distribution between samples.

1.3 Dimensionality reduction

Dimensionality reduction is commonly used when dealing with large numbers of observations and/or large numbers of variables in lipids analysis. It transforms data from a high-dimensional space into a low-dimensional space so that it retains vital properties of the original data and is close to its intrinsic dimension.

Here we provide 3 dimensionality reduction methods and 4 clustering methods. As for the number of groups shown on the PCA, t-SNE, and UMAP plot, it can be defined by users (default: 2 groups).

- **Dimensionality reduction methods**: PCA, t-SNE, UMAP.
- **Clustering methods**: K-means, partitioning around medoids (PAM), Hierarchical clustering, and DBSCAN

1.3.1 PCA

PCA (Principal component analysis) is an unsupervised linear dimensionality reduction and data visualization technique for high dimensional data, which tries to preserve the global structure of the data. Scaling (by default) indicates that the variables should be scaled to have unit variance before the analysis takes place, which removes the bias towards high variances. In general, scaling (standardization) is advisable for data transformation when the variables in the original dataset have been measured on a significantly different scale. As for the centering options (by default), we offer the option of mean-centering, subtracting the mean of each variable from the values, making the mean of each variable equal to zero. It can help users to avoid the interference of misleading information given by the overall mean.

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, zero2what='min',
+                                     xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
```

```
+                                    pct_transform=TRUE,
+                                    data_transform=TRUE, trans_type='log',
+                                    centering=FALSE, scaling=FALSE)
> # conduct PCA
> PCA_result <- PCA(exp_transform_table,
+                   group_info = NULL, sig_feature = NULL,
+                   scaling=TRUE, centering=TRUE, cluster_method='kmeans',
+                   group_num=2, var1 = NULL, var2 = NULL,
+                   insert_ref_group=NULL, ref_group=NULL,
+                   n_PC=c(1,2), top_n_feature=10)
> # view result: PCA prcomp
> head(PCA_result[[1]], 1)

$sdev
 [1] 6.641716e+00 5.272274e+00 4.247168e+00 3.711370e+00 3.058461e+00
 [6] 2.732843e+00 2.623854e+00 2.476808e+00 2.307884e+00 2.086144e+00
[11] 1.981612e+00 1.923971e+00 1.848639e+00 1.791394e+00 1.756612e+00
[16] 1.722894e+00 1.518288e+00 1.451361e+00 1.379832e+00 1.304074e+00
[21] 1.181540e+00 1.018145e+00 4.131560e-15
```

```
>   # view result: PCA plot
>   PCA_result[[4]]
```

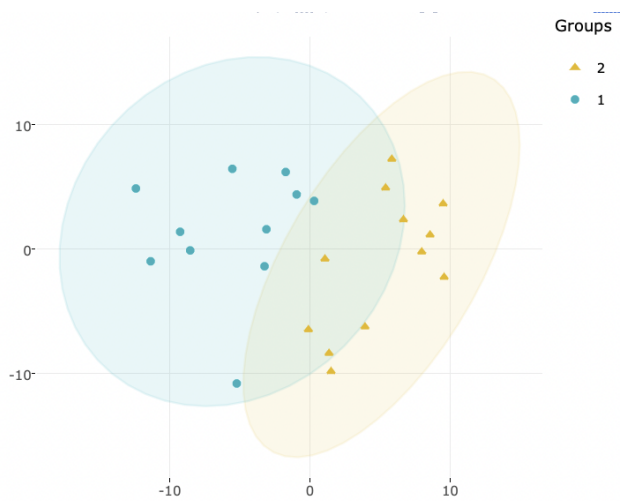

Figure 4: PCA plot

Accompanying with the PCA plot, we offer scree plot criterion, which is a common method for determining the number of PCs to be retained. The "elbow" of the graph indicates all components to the left of this point can explain most variability of the samples.

```
> # view result: scree plot of top 10 principle components
> PCA_result[[5]]
```

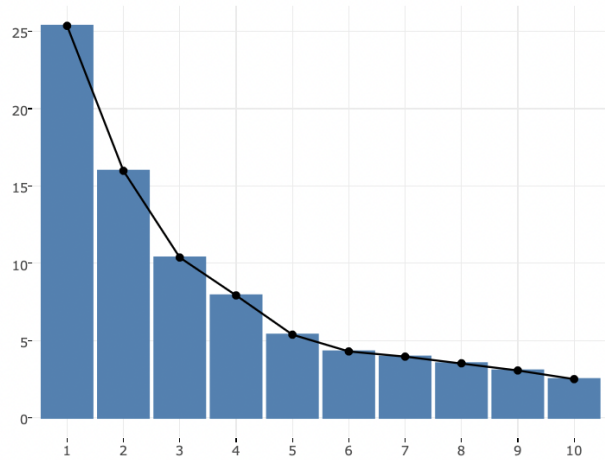

Figure 5: Scree plot

Next, the two data frames related to PCA show the contribution to each principal component in each sample and the contribution of each feature (lipid species).

```
> # view result: data frame of PCA rotated data
> head(PCA_result[[2]][,1:5], 5)

           sample_name group       PC1        PC2       PC3
control_01  control_01     2 -1.738538  6.1654860 2.8800338
control_02  control_02     2 -0.957185  4.3614691 0.6953459
control_03  control_03     1  7.959509 -0.2488075 4.2278920
control_04  control_04     2 -3.112118  1.5662358 6.7091316
control_05  control_05     1  5.826969  7.2008185 0.1542630

> # view result: data frame of PCA contribution table
> head(PCA_result[[3]][,1:5], 5)

              feature          PC1        PC2         PC3       PC4
Cer 38:1;2 Cer 38:1;2 0.0001235637 0.53728920 0.762186877 0.5195321
Cer 40:1;2 Cer 40:1;2 0.0500511640 0.76262932 0.009689057 1.4135372
Cer 40:2;2 Cer 40:2;2 0.0154321655 0.35529119 0.036215184 3.2121693
Cer 42:1;2 Cer 42:1;2 0.0175217782 0.61677615 0.633822909 1.8038403
Cer 42:2;2 Cer 42:2;2 0.0004789253 0.06235655 0.002076345 0.6206153
```

Following is the correlation circle plot that reveals the relationships between all variables.

```
> # view result: correlation circle plot of PCA variables
> PCA_result[[7]]
```

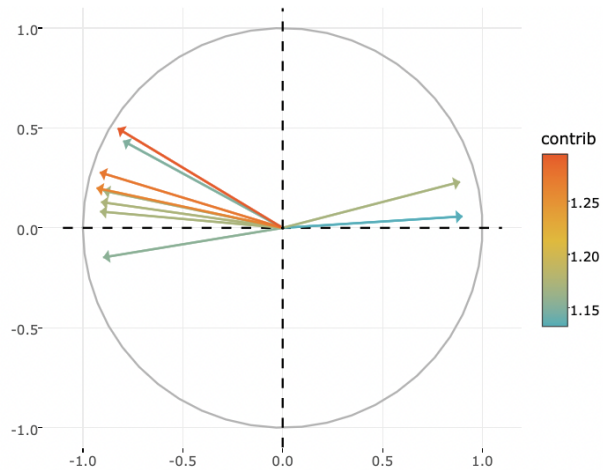

Figure 6: Correlation circle plot
The correlation circle plot showing the correlation between a feature (lipid species) and a principal component (PC) used as the coordinates of the variable on the PC [2]. The positively correlated variables are in the same quadrants while negatively correlated variables are on the opposite sides of the plot origin. The closer a variable to the edge of the circle, the better it represents on the factor map.

Lastly, we can have a closer look at the contribution of top 10 features to a user-defined principal component (e.g., PC1, PC2, or PC1+PC2). Therefore, in the histogram, we can find out the features (lipid species) that contribute more to the user-defined principal component.

```
> # view result: bar plot of contribution of top 10 features
> PCA_result[[6]]
```

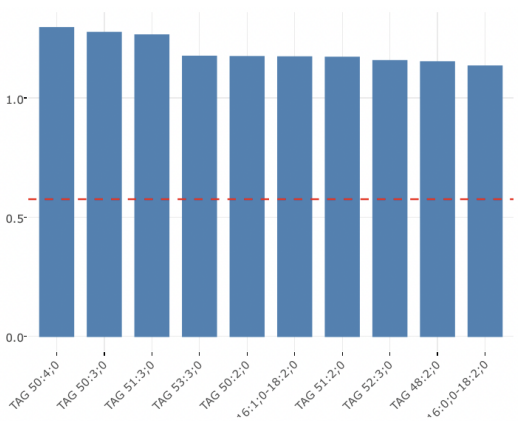

Figure 7: Bar plot of contribution of top 10 features

1.3.2 t-SNE

t-SNE (t-Distributed Stochastic Neighbour Embedding) is an unsupervised non-linear dimensionality reduction technique that tries to retain the local structure(cluster) of data when visualising the high-dimensional datasets. Package *Rtsne* is used for calculation, and PCA is applied as a pre-processing step. In t-SNE, `perplexity` and `max_iter` are adjustable for users. The `perplexity` may be considered as a knob that sets the number of effective nearest

neighbours, while `max_iter` is the maximum number of iterations to perform. The typical perplexity range between 5 and 50, but if the t-SNE plot shows a 'ball' with uniformly distributed points, you may need to lower your perplexity [3].

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, zero2what='min',
+                                     xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE, trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # conduct t-SNE
> tsne_result <- tsne(exp_transform_table, group_info = NULL,
+                     sig_feature = NULL, pca=TRUE, perplexity=5,
+                     max_iter=500, cluster_method='kmeans',
+                     group_num=2, var1 = 'euclidean', var2 = NULL,
+                     insert_ref_group = NULL, ref_group = NULL)

Performing PCA
Read the 23 x 23 data matrix successfully!
Using no_dims = 2, perplexity = 5.000000, and theta = 0.000000
Computing input similarities...
Symmetrizing...
Done in 0.00 seconds!
Learning embedding...
Iteration 50: error is 55.322047 (50 iterations in 0.00 seconds)
Iteration 100: error is 56.226075 (50 iterations in 0.00 seconds)
Iteration 150: error is 53.924249 (50 iterations in 0.00 seconds)
Iteration 200: error is 60.472140 (50 iterations in 0.00 seconds)
Iteration 250: error is 57.829684 (50 iterations in 0.00 seconds)
Iteration 300: error is 1.095317 (50 iterations in 0.00 seconds)
Iteration 350: error is 0.840229 (50 iterations in 0.00 seconds)
Iteration 400: error is 0.561340 (50 iterations in 0.00 seconds)
Iteration 450: error is 0.484735 (50 iterations in 0.00 seconds)
Iteration 500: error is 0.470855 (50 iterations in 0.00 seconds)
Fitting performed in 0.00 seconds.

> # view result: data frame of t-SNE data
> head(tsne_result[[1]], 5)

  sample_name group     tsne1      tsne2
1  control_01     1 -38.87708 -31.926484
2  control_02     1 -35.09291  -7.900503
3  control_03     1 -26.54940 -65.136507
4  control_04     1 -55.59101 -36.572127
5  control_05     1 -25.11773 -26.169460
```

```
> # view result: t-SNE plot
> tsne_result[[2]]
```

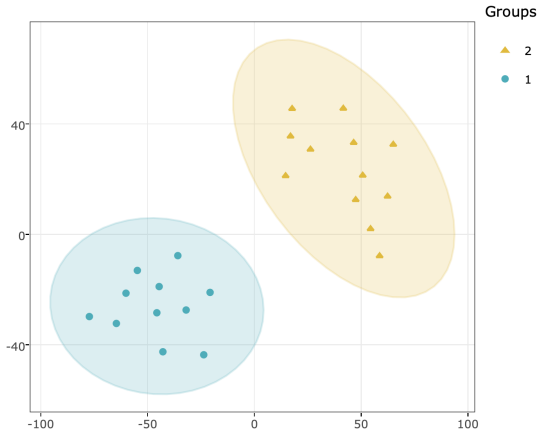

Figure 8: t-SNE plot

1.3.3 UMAP

UMAP (Uniform Manifold Approximation and Projection) using a nonlinear dimensionality reduction method, Manifold learning, which effectively visualizing clusters or groups of data points and their relative proximities. Both tSNE and UMAP are intended to predominantly preserve the local structure that is to group neighbouring data points which certainly delivers a very informative visualization of heterogeneity in the data. The significant difference with t-SNE is scalability, which allows UMAP eliminating the need for applying pre-processing step (such as PCA). Besides, UMAP applies Graph Laplacian for its initialization as tSNE by default implements random initialization. Thus, some people suggest that the key problem of tSNE is the Kullback-Leibler (KL) divergence, which makes UMAP superior over t-SNE. Nevertheless, UMAP's cluster may not good enough for multi-class pattern classification [4].

The type of distance metric to find nearest neighbors the size of the local neighborhood (as for the number of neighboring sample points) are set by parameter `metric` and `n_neighbors`. Larger values lead to more global views of the manifold, while smaller values result in more local data being preserved. Generally, values are set in the range of 2 to 100. (default: 15).

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, trans_type='log',
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, replace_NA=TRUE,
+                                     zero2what='min', xmin=0.5,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE, centering=FALSE,
+                                     data_transform=TRUE, scaling=FALSE )
> # conduct UMAP
> UMAP_result <- UMAP(exp_transform_table, group_info=NULL,
+                     sig_feature=NULL, n_neighbors=15,
+                     scale=TRUE, metric='euclidean', group_num=2,
+                     cluster_method='kmeans', var1=NULL, var2=NULL,
+                     insert_ref_group=NULL, ref_group=NULL)
> # view result: data frame of UMAP data
> head(UMAP_result[[1]], 5)
```

```
            sample_name group     UMAP-1    UMAP-2
control_01  control_01     2  1.0678994 0.8424913
control_02  control_02     2  0.7339923 1.0167486
control_03  control_03     1 -1.5019214 0.5647976
control_04  control_04     2  0.7466738 0.2757284
control_05  control_05     1 -0.2138493 1.3673412
```

```
> # view result: UMAP plot
> UMAP_result[[2]]
```

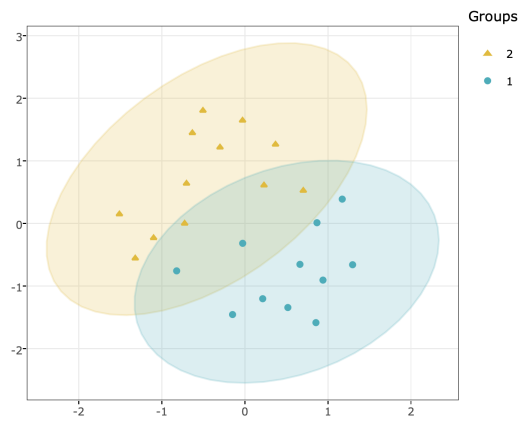

Figure 9: UMAP plot

1.4 Correlation heatmap

The correlation heatmap illustrates the correlation between samples or lipid species and also depicts the patterns in each group. The correlation is calculated by the method defined by parameter `corr_method`, and the correlation coefficient is then clustered depending on method defined by parameter `distfun` and the distance defined by parameter `hclustfun`. Two heatmaps will be shown by lipid species and by samples. Please note that if the number of lipids or samples is over 50, the names of lipids/samples will not be shown on the heatmap.

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50, replace_zero=TRUE,
+                               zero2what='min', xmin=0.5, replace_NA=TRUE,
+                               NA2what='min', ymin=0.5, pct_transform=TRUE,
+                               data_transform=TRUE, trans_type='log',
+                               centering=FALSE, scaling=FALSE)
> # correlation calculation
> corr_result <- corr_heatmap(exp_transform, corr_method="pearson",
+                             distfun="maximum", hclustfun="average")
> # view result: matrix of correlation coefficients
> head(corr_result$sample_corr_coef[, 1:5], 5)

            control_01 control_02 control_03 control_04 control_05
control_01  1.0000000  0.8436678  0.8369328  0.8540800  0.8734609
```

```
control_02  0.8436678  1.0000000  0.7457617  0.8255156  0.8040823
control_03  0.8369328  0.7457617  1.0000000  0.7739019  0.7849288
control_04  0.8540800  0.8255156  0.7739019  1.0000000  0.7201342
control_05  0.8734609  0.8040823  0.7849288  0.7201342  1.0000000

> # view result: matrix of correlation p-value
> head(corr_result$sample_corr_p[, 1:4], 5)

             control_01   control_02   control_03   control_04
control_01 0.000000e+00 2.473172e-48 6.844569e-47 1.056963e-50
control_02 2.473172e-48 0.000000e+00 3.718806e-32 1.367375e-44
control_03 6.844569e-47 3.718806e-32 0.000000e+00 5.897612e-36
control_04 1.056963e-50 1.367375e-44 5.897612e-36 0.000000e+00
control_05 1.203055e-55 1.079608e-40 1.344742e-37 4.167638e-29

> # view result: matrix of reorder sample correlation
> head(corr_result$reorder_sample_corr_coef[, 1:3], 5)

                 hfref_patient_10 hfref_patient_04 hfref_patient_05
hfref_patient_13        0.6976086        0.6809920        0.7187572
hfref_patient_12        0.7310794        0.7062071        0.7171976
hfref_patient_11        0.5647891        0.7577328        0.7851855
hfref_patient_10        1.0000000        0.6060374        0.6578909
hfref_patient_09        0.6871129        0.7057743        0.6171108

> # view result: matrix of correlation coefficients between lipids
> head(corr_result$lipids_corr_coef[, 1:5], 5)

           Cer 38:1;2 Cer 40:1;2 Cer 40:2;2 Cer 42:1;2 Cer 42:2;2
Cer 38:1;2 1.00000000  0.2676729  0.2647730  0.3129075 0.01703923
Cer 40:1;2 0.26767290  1.0000000  0.5201040  0.7863653 0.43877001
Cer 40:2;2 0.26477303  0.5201040  1.0000000  0.4727582 0.42078183
Cer 42:1;2 0.31290745  0.7863653  0.4727582  1.0000000 0.43310999
Cer 42:2;2 0.01703923  0.4387700  0.4207818  0.4331100 1.00000000

> # view result: matrix of correlation p-value between lipids
> head(corr_result$lipid_corr_p[, 1:4], 5)

              Cer 38:1;2   Cer 40:1;2    Cer 40:2;2    Cer 42:1;2
Cer 38:1;2 7.469380e-166 2.169008e-01  2.221124e-01  1.460114e-01
Cer 40:1;2  2.169008e-01 0.000000e+00  1.095933e-02  8.635027e-06
Cer 40:2;2  2.221124e-01 1.095933e-02 7.469380e-166  2.271903e-02
Cer 42:1;2  1.460114e-01 8.635027e-06  2.271903e-02 7.639373e-161
Cer 42:2;2  9.384919e-01 3.621739e-02  4.556576e-02  3.897876e-02
```

```
> # view result: sample-sample heatmap
> corr_result$sample_hm
```

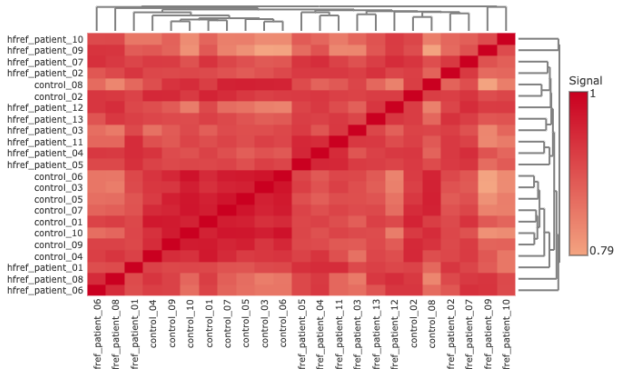

Figure 10: Heatmap of sample to sample correlations
The heatmap reveals sample to sample correlations. Correlations between lipid species are colored from strong positive correlations (red) to no correlation (white).

```
> # view result: lipid-lipid correlations heatmap
> corr_result$lipids_hm
```

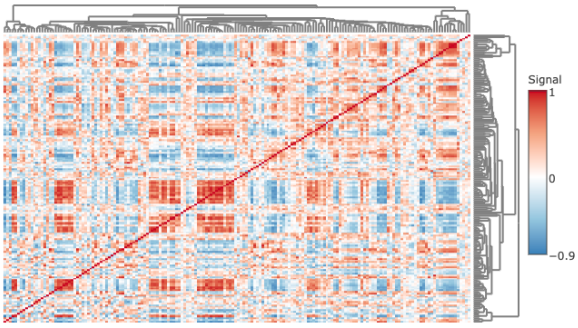

Figure 11: Heatmap of lipid to lipid correlations
The heatmap illustrates the lipid to lipid correlations. Correlations between lipid species are colored from strong positive correlation (red) to no correlation (white), to negative correlation (blue).

1.5 Lipid characteristics

Now, we are going to take a view of lipid expression over specific lipid characteristics. First, lipids are classified by characteristics selected from the 'Lipid characteristics' table. Here, we select "class" as the selected lipid characteristic. The results will be showed by two plots.

```
> # lipid characteristic
> char_var <- colnames(lipid_char_table)[-1]
> # calculate lipid expression of selected characteristic
> compo_result <- exp_compo_by_lipidinfo(exp_data, lipid_char_table, char_var[1])
```

```
> # view result: bar plot
> compo_result$p.barplot.p
```

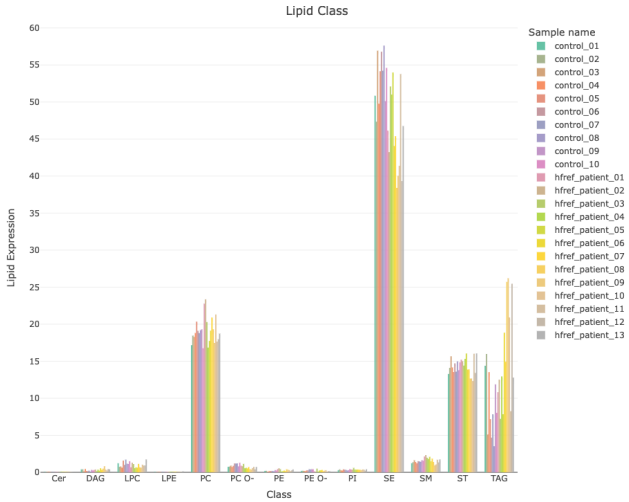

Figure 12: Bar plot classified by selected characteristic
The bar plot depicts the expression level of each sample within each group (e.g., PE, PC) of selected characteristics (e.g., class).

```
> # view result: stacked horizontal bar chart
> compo_result$p.compos
```

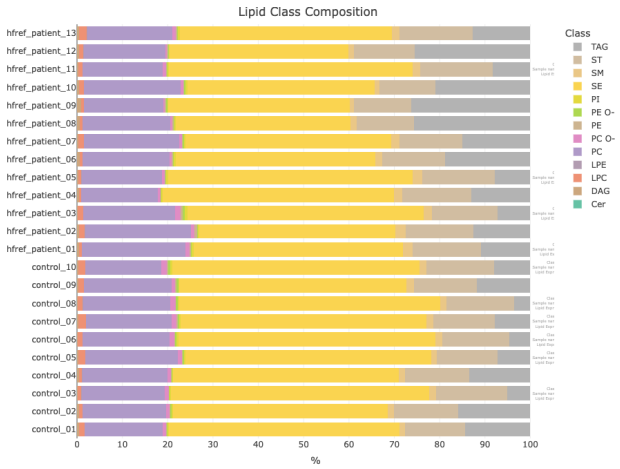

Figure 13: Lipid class composition
The stacked horizontal bar chart illustrates the percentage of characteristics in each sample. The variability of percentage between samples can also be obtained from this plot.

2 Differential expression

After overviewing the lipid data, then we move on to differential expression to identify the significant lipid species and lipid characteristics. Differential Expression is divided into two main analyses, **'Lipid species analysis'** and **'Lipid characteristics analysis'**. Further analysis and visualization methods can also be conducted based on the results of differential expressed analysis.

- **Lipid species analysis**: The lipid species analysis explores the significant lipid species based on differentially expressed analysis. Data are analyzed based on each lipid species. Further analysis and visualization methods, include
  1. dimensionality reduction,
  2. hierarchical clustering,
  3. characteristics association,
  4. enrichment.
- **Lipid characteristics analysis**: The lipid characteristics analysis explores the significant lipid characteristics. Lipid species are categorized and summarized into a new lipid expression table according to a selected lipid characteristic. The expression of all lipid species of the same categories are summed up, then conduct differential expressed analysis. Further analysis and visualization methods include
  1. dimensionality reduction,
  2. hierarchical clustering.

2.1 Input data

First, we have to read the input data needed for the differential expression section. We have to prepare lipid expression data (`exp_data`), lipid characteristics table (`lipid_char_table`), and a table of one clinical term (or one set of clinical terms) from demographic data (`group_info`) as input data.

```
> # clears all objects from workspace
> rm(list = ls())
> # lipid expression data
> data("DE_exp_data")
> exp_data <- DE_exp_data
> head(exp_data[, 1:5], 5)

     feature control_01 control_02 control_03 control_04
1 Cer 38:1;2  0.1167960  0.1638070  0.1759450  0.1446540
2 Cer 40:1;2  0.7857833  0.9366095  0.8944465  0.8961396
3 Cer 40:2;2  0.1494030  0.1568970  0.1909800  0.1312440
4 Cer 42:1;2  1.8530153  2.1946591  2.6377576  2.3418783
5 Cer 42:2;2  1.3325520  1.2514943  1.9466750  1.2948319

> # lipid characteristics table
> data("DE_lipid_char_table")
> lipid_char_table <- DE_lipid_char_table
> head(lipid_char_table[, 1:4], 5)
```

```
     feature class structural_category functional_category
1 Cer 38:1;2   Cer                  SL                 MEM
2 Cer 40:1;2   Cer                  SL                 MEM
3 Cer 40:2;2   Cer                  SL                 MEM
4 Cer 42:1;2   Cer                  SL                 MEM
5 Cer 42:2;2   Cer                  SL                 MEM

> # group information table
> data("DE_group_info")
> group_info <- DE_group_info
> head(group_info, 5)

  sample_name label_name group pair
1  control_01      ctrl1  ctrl   NA
2  control_02      ctrl2  ctrl   NA
3  control_03      ctrl3  ctrl   NA
4  control_04      ctrl4  ctrl   NA
5  control_05      ctrl5  ctrl   NA
```

After importing the input data, sometimes, we may need to conduct data processing before analysis. Here, we provide the `data_process` function for data processing, including removing features with missing values, missing values imputation, percentage transformation, log10 transformation, etc.

```
> # lipid expression data
> head(exp_data[, 1:5], 5)

     feature control_01 control_02 control_03 control_04
1 Cer 38:1;2  0.1167960  0.1638070  0.1759450  0.1446540
2 Cer 40:1;2  0.7857833  0.9366095  0.8944465  0.8961396
3 Cer 40:2;2  0.1494030  0.1568970  0.1909800  0.1312440
4 Cer 42:1;2  1.8530153  2.1946591  2.6377576  2.3418783
5 Cer 42:2;2  1.3325520  1.2514943  1.9466750  1.2948319

> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, zero2what='min',
+                                     xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE, trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> head(exp_transform_table[, 1:5], 5)

     feature control_01 control_02 control_03 control_04
1 Cer 38:1;2  -2.820387  -2.705816  -2.735971  -2.769140
2 Cer 40:1;2  -1.992512  -1.948590  -2.029794  -1.977095
3 Cer 40:2;2  -2.713455  -2.724534  -2.700360  -2.811391
4 Cer 42:1;2  -1.619936  -1.578781  -1.560113  -1.559906
5 Cer 42:2;2  -1.763130  -1.822719  -1.692055  -1.817257
```

2.2 Lipid species analysis

Now, let's start with the analysis of lipid species.

2.2.1 Differentially expressed analysis

For lipid species analysis section, differentially expressed analysis is performed to figure out significant lipid species. In short, samples will be divided into two groups (independent) according to the input "Group Information" table.

```
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data, exclude_var_missing=TRUE,
+                                       missing_pct_limit=50, replace_zero=TRUE,
+                                       zero2what='min', xmin=0.5,
+                                       replace_NA=TRUE, NA2what='min',
+                                       ymin=0.5, pct_transform=TRUE,
+                                       data_transform=FALSE, trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # conduct differentially expressed analysis of lipid species
> DE_species_result <- DE_species_2(exp_transform_non_log,
+                                   data_transform=TRUE,
+                                   group_info = group_info,
+                                   paired=FALSE, test='t.test',
+                                   adjust_p_method='BH', sig_stat='p.adj',
+                                   sig_pvalue=0.05, sig_FC=2)
> # view result: data frame of lipid species
> head(DE_species_result$DE_species_table_all[, 1:5], 5)

# A tibble: 5 × 5
# Groups:   feature [5]
  feature    mean_ctrl mean_exp method    FC
  <chr>          <dbl>    <dbl> <chr>  <dbl>
1 Cer 38:1;2   0.00177  0.00146 t-test 0.826
2 Cer 40:1;2   0.00969  0.00865 t-test 0.893
3 Cer 40:2;2   0.00191  0.00234 t-test 1.22
4 Cer 42:1;2   0.0234   0.0198  t-test 0.849
5 Cer 42:2;2   0.0164   0.0172  t-test 1.05

> # view result: data frame of significant lipid species
> head(DE_species_result$DE_species_table_sig [, 1:5], 5)

# A tibble: 5 × 5
# Groups:   feature [5]
  feature           mean_ctrl mean_exp method     FC
  <chr>                 <dbl>    <dbl> <chr>   <dbl>
1 DAG 16:0;0-18:1;0   0.0326   0.0713  t-test  2.19
2 PC 18:0;0-18:1;0    0.00893  0.328   t-test 36.7
3 PC 18:0;0-18:3;0    0.0170   0.00754 t-test  0.443
4 PC 18:1;0-15:0;0    0.0375   0.0939  t-test  2.50
5 PC 20:4;0-18:2;0    0.0875   0.0117  t-test  0.134
```

The above `DE_species_table_sig` data frame will be used in the following analyses in Section 2.2.2, Section 2.2.3, Section 2.2.4, and Section 2.2.5.

```
> # view result: lollipop chart
> DE_species_result$DE_species_dotchart_sig
```

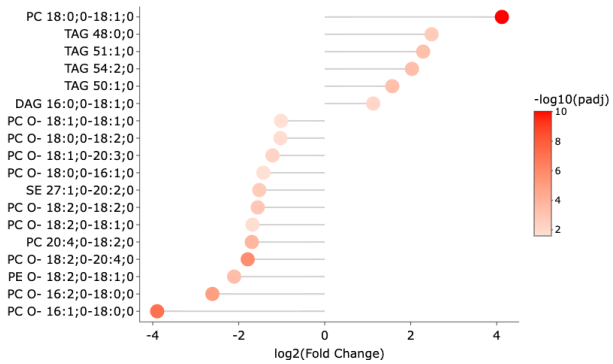

Figure 14: Lollipop chart of lipid species analysis
The lollipop chart reveals the lipid species that pass chosen cut-offs. The x-axis shows log2 fold change while the y-axis is a list of lipids species. The color of the point is determined by `-log10(adj_value/p-value)`.

```
> # view result: MA plot
> DE_species_result$DE_species_maplot
```

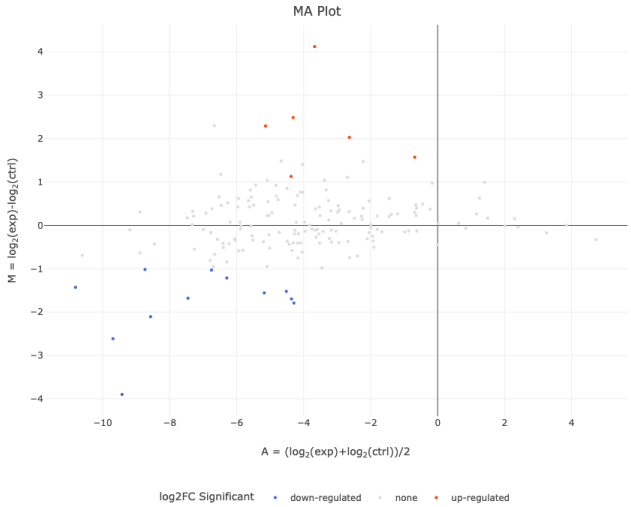

Figure 15: MA plot
The MA plot indicates three groups of lipid species, up-regulated(red), down-regulated(blue), and non-significant(grey).

```
> # view result: MA plot
> DE_species_result$DE_species_volcano
```

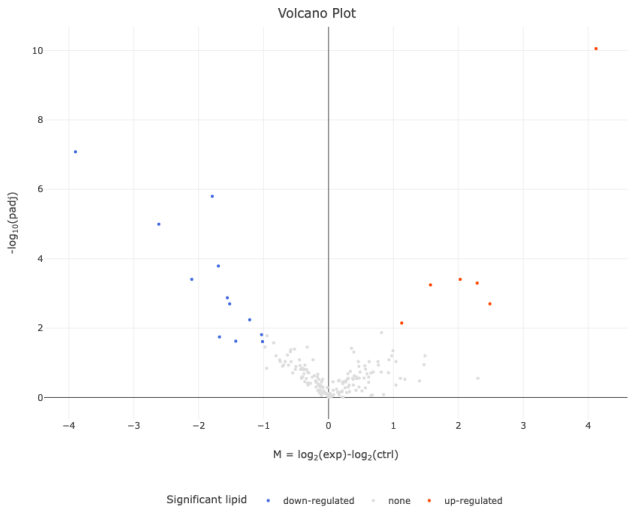

Figure 16: Volcano plot
The volcano plot illustrates a similar concept to the MA plot. These points visually identify the most biologically significant lipid species (red for up-regulated, blue for down-regulated, and grey for non-significant).

2.2.2 Dimensionality reduction

Dimensionality reduction is common when dealing with large numbers of observations and/or large numbers of variables in lipids analysis. It transforms data from a high-dimensional space into a low-dimensional space to retain vital properties of the original data and close to its intrinsic dimension.

Here, we provide 4 dimensionality reduction methods, namely, PCA, t-SNE, UMAP, and PLS-DA. For the detailed information of the former three methods, please refer to Section 1.3.

- Note: The input data of this section should be by filtered by function `DE_species_2`.

1. **PCA (Principal component analysis)**
   PCA is an unsupervised linear dimensionality reduction and data visualization technique for high dimensional data, which tries to preserve the global structure of the data. For detailed information of PCA, please refer to Section 1.3.1.

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, zero2what='min',
+                                     xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
```

```
+                                        replace_zero=TRUE,
+                                        zero2what='min', xmin=0.5,
+                                        replace_NA=TRUE, NA2what='min',
+                                        ymin=0.5, pct_transform=TRUE,
+                                        data_transform=FALSE,
+                                        trans_type='log',
+                                        centering=FALSE, scaling=FALSE)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform=TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test='t.test',
+                                      adjust_p_method='BH',
+                                      sig_stat='p.adj', sig_pvalue=0.05,
+                                      sig_FC=2)$DE_species_table_sig
> # conduct PCA
> DEspec_PCA <- PCA(exp_transform_table,
+                   group_info = group_info,
+                   sig_feature = DE_species_table_sig$feature,
+                   scaling=TRUE, centering=TRUE, group_num=2,
+                   cluster_method='kmeans', var1=NULL, var2=NULL,
+                   insert_ref_group=NULL, ref_group=NULL,
+                   n_PC=c(1,2), top_n_feature=10)
> # view result: PCA prcomp
> head(DEspec_PCA[[1]], 1)

$sdev
 [1] 3.32398872 1.30509519 1.10856378 0.95885503 0.85550667 0.70613194
 [7] 0.65384878 0.57403502 0.52170035 0.49204203 0.41594593 0.37011897
[13] 0.32974367 0.28151197 0.23796390 0.18860151 0.08400909 0.02121482

> # view result: data frame of PCA rotated data
> head(DEspec_PCA[[2]][,1:4], 5)

           sample_name group       PC1         PC2
control_01  control_01     1 -2.984733 -0.09429360
control_02  control_02     1 -1.675546 -0.44762370
control_03  control_03     1 -4.235340 -0.04884014
control_04  control_04     1 -2.395611 -0.47465428
control_05  control_05     1 -4.666318 -0.16614533

> # view result: data frame of PCA contribution table
> head(DEspec_PCA[[3]][,1:3], 5)

                            feature      PC1        PC2
DAG 16:0;0-18:1;0 DAG 16:0;0-18:1;0 5.626008  5.5175090
PC 18:0;0-18:1;0   PC 18:0;0-18:1;0 7.500202  0.5292732
PC 18:0;0-18:3;0   PC 18:0;0-18:3;0 4.415798  3.4265727
PC 18:1;0-15:0;0   PC 18:1;0-15:0;0 2.837259 25.6616361
PC 20:4;0-18:2;0   PC 20:4;0-18:2;0 4.908783  1.6798059
```

```
> # view result: PCA plot
> DEspec_PCA[[4]]
```

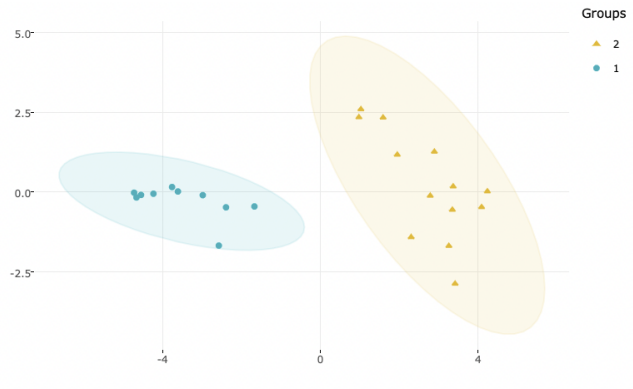

Figure 17: PCA Results - PCA plot

```
> # view result: scree plot
> DEspec_PCA[[5]]
```

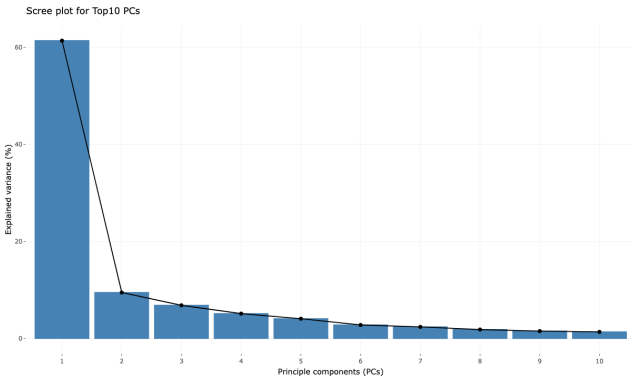

Figure 18: PCA Results - scree plot

```
> # view result: bar plot of contribution of top 10 features
> DEspec_PCA[[6]]
```

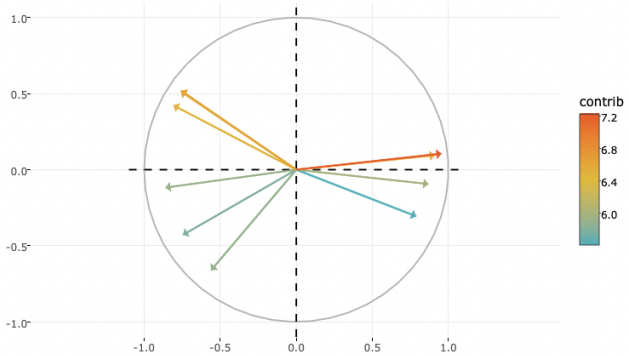

Figure 19: PCA Results - bar plot

```
> # view result: correlation circle plot of variables
> DEspec_PCA[[7]]
```

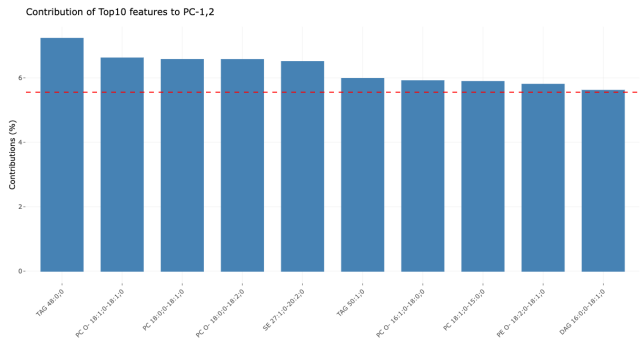

Figure 20: PCA Results - correlation circle plot

2. **t-SNE (t-Distributed Stochastic Neighbour Embedding)**
   t-Distributed Stochastic Neighbour Embedding (t-SNE) is an unsupervised non-linear dimensionality reduction technique that tries to retain the local structure(cluster) of data when visualising the high-dimensional datasets. For detailed information of t-SNE, please refer to Section 1.3.2.

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, xmin=0.5,
+                                     zero2what='min', replace_NA=TRUE,
+                                     ymin=0.5, NA2what='min',
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE,
+                                       zero2what='min', xmin=0.5,
+                                       replace_NA=TRUE, NA2what='min',
+                                       ymin=0.5, pct_transform=TRUE,
+                                       data_transform=FALSE,
+                                       trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform=TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test='t.test',
+                                      adjust_p_method='BH',
+                                      sig_stat='p.adj', sig_pvalue=0.05,
```

```
+                                       sig_FC=2)$DE_species_table_sig
> # conduct t-SNE
> DEspec_tsne <- tsne(exp_transform_table,
+                     group_info = group_info,
+                     sig_feature=DE_species_table_sig$feature,
+                     pca=TRUE, perplexity=5, max_iter=500,
+                     cluster_method='kmeans', group_num=2,
+                     var1 = 'euclidean', var2 = NULL,
+                     insert_ref_group=NULL, ref_group=NULL)

Performing PCA
Read the 23 x 18 data matrix successfully!
Using no_dims = 2, perplexity = 5.000000, and theta = 0.000000
Computing input similarities...
Symmetrizing...
Done in 0.00 seconds!
Learning embedding...
Iteration 50: error is 60.729137 (50 iterations in 0.00 seconds)
Iteration 100: error is 54.232235 (50 iterations in 0.00 seconds)
Iteration 150: error is 51.124733 (50 iterations in 0.00 seconds)
Iteration 200: error is 59.209555 (50 iterations in 0.00 seconds)
Iteration 250: error is 52.892927 (50 iterations in 0.00 seconds)
Iteration 300: error is 1.283966 (50 iterations in 0.00 seconds)
Iteration 350: error is 0.786141 (50 iterations in 0.00 seconds)
Iteration 400: error is 0.184900 (50 iterations in 0.00 seconds)
Iteration 450: error is 0.140192 (50 iterations in 0.00 seconds)
Iteration 500: error is 0.091942 (50 iterations in 0.00 seconds)
Fitting performed in 0.00 seconds.

> # view result: data frame of t-SNE data
> head(DEspec_tsne[[1]], 5)

  sample_name group    tsne1      tsne2
1  control_01     2 72.01415   9.815317
2  control_02     2 81.78349  37.753103
3  control_03     2 78.25941 -10.161240
4  control_04     2 76.46956  25.341892
5  control_05     2 76.38731  -1.062356
```

```
> # view result: t-SNE plot
> DEspec_tsne[[2]]
```

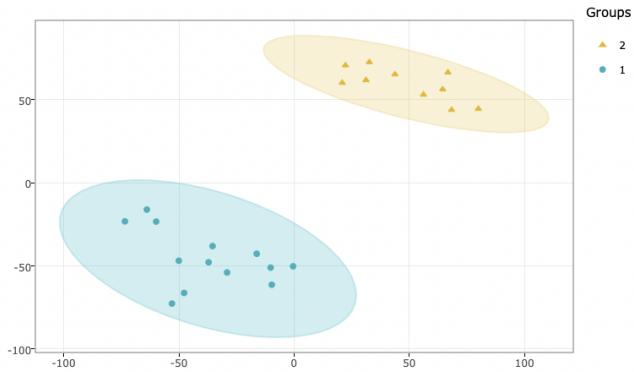

Figure 21: t-SNE plot

3. **UMAP (Uniform Manifold Approximation and Projection)**
   UMAP using a nonlinear dimensionality reduction method, Manifold learning, which effectively visualizing clusters or groups of data points and their relative proximities. For detailed information of UMAP, please refer to Section 1.3.3.

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE, xmin=0.5,
+                                     zero2what='min', replace_NA=TRUE,
+                                     ymin=0.5, NA2what='min',
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE,
+                                       zero2what='min', xmin=0.5,
+                                       replace_NA=TRUE, NA2what='min',
+                                       ymin=0.5, pct_transform=TRUE,
+                                       data_transform=FALSE,
+                                       trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform=TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test='t.test',
+                                      adjust_p_method='BH',
+                                      sig_stat='p.adj', sig_pvalue=0.05,
+                                      sig_FC=2)$DE_species_table_sig
> # conduct UMAP
> DEspec_UMAP <- UMAP(exp_transform_table, group_num=2,
```

```
+                     group_info=group_info, n_neighbors=15,
+                     sig_feature=DE_species_table_sig$feature,
+                     metric='euclidean', scale=TRUE, var1=NULL,
+                     cluster_method='kmeans', var2=NULL,
+                     insert_ref_group=NULL, ref_group=NULL)
> # view result: data frame of UMAP data
> head(DEspec_UMAP[[1]], 5)

           sample_name group   UMAP-1    UMAP-2
control_01  control_01     1 5.172147 -3.850589
control_02  control_02     1 4.440248 -3.897364
control_03  control_03     1 5.463851 -2.657048
control_04  control_04     1 4.440672 -3.460473
control_05  control_05     1 5.851358 -3.362325
```

```
> # view result: UMAP plot
> DEspec_UMAP[[2]]
```

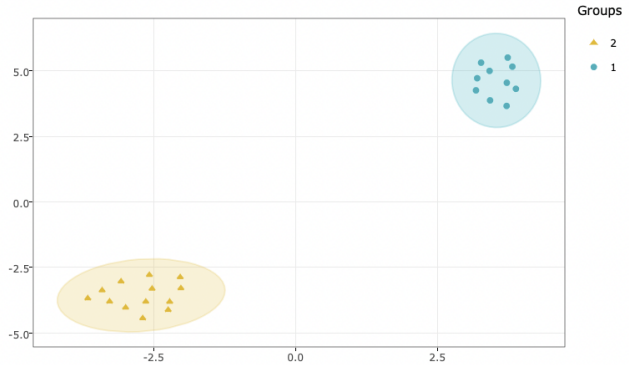

Figure 22: UMAP plot

4. **PLS-DA**

```
> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, replace_zero=TRUE,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     zero2what='min',
+                                     xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log', centering=FALSE,
+                                     scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE,
```

```
+                                        zero2what='min', xmin=0.5,
+                                        replace_NA=TRUE, NA2what='min',
+                                        ymin=0.5, pct_transform=TRUE,
+                                        data_transform=FALSE,
+                                        trans_type='log',
+                                        centering=FALSE, scaling=FALSE)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform=TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test='t.test',
+                                      adjust_p_method='BH',
+                                      sig_stat='p.adj', sig_pvalue=0.05,
+                                      sig_FC=2)$DE_species_table_sig
> # conduct PLSDA
> DEspec_PLSDA <- PLSDA(exp_transform_table, scaling=TRUE,
+                       group_info=group_info, ncomp=2,
+                       sig_feature=DE_species_table_sig$feature,
+                       cluster_method='group_info',
+                       group_num=NULL, var1=NULL, var2=NULL,
+                       insert_ref_group=NULL, ref_group=NULL)
> # view result: data frame of sample variate
> head(DEspec_PLSDA[[1]], 5)

           sample_name group   PLSDA1     PLSDA2
control_01  control_01  ctrl 3.035767 -0.6536566
control_02  control_02  ctrl 1.793747 -1.3015973
control_03  control_03  ctrl 4.172360  0.5023834
control_04  control_04  ctrl 2.481938 -0.9827351
control_05  control_05  ctrl 4.620765  0.3085736

> # view result: data frame of sample loading
> head(DEspec_PLSDA[[2]], 5)

                      PLSDA1      PLSDA2
DAG 16:0;0-18:1;0 -0.2092640 -0.28753561
PC 18:0;0-18:1;0  -0.2848519  0.09685029
PC 18:0;0-18:3;0   0.2259545 -0.14248077
PC 18:1;0-15:0;0   0.1925862 -0.20567347
PC 20:4;0-18:2;0   0.2653050 -0.43831973
```

```
> # view result: PLS-DA sample plot
> DEspec_PLSDA[[3]]
```

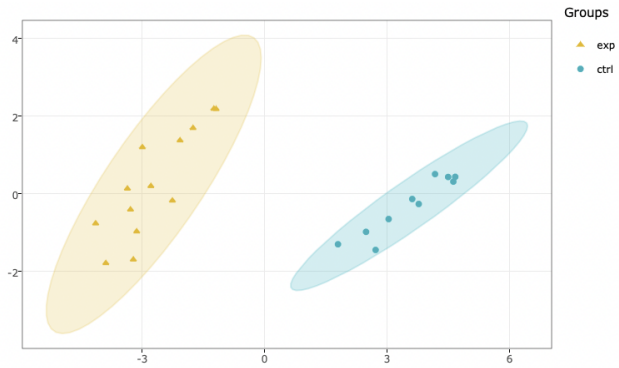

Figure 23: Sample plot

```
> # view result: PLS-DA variable plot
> DEspec_PLSDA[[4]]
```

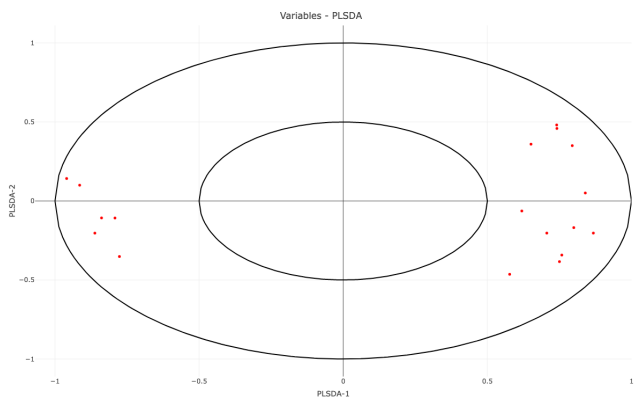

Figure 24: Loading plot
In the PLS-DA loading plot, the distance to the center of the variables indicates the contribution of the variable. The value of the x-axis reveals the contribution of the variable to PLS-DA-1, whereas the value of the y-axis discloses the contribution of the variable to PLS-DA-2.

2.2.3 Hierarchical clustering

Based on the results of differentially expressed analysis, we further take a look at differences of lipid species between the control group and the experimental group. Lipid species derived from two groups are clustered and visualized on heatmap by hierarchical clustering.

The top of the heatmap is grouped by sample group (top annotation) while the side of the heatmap (row annotation) can be chosen from `lipid_char_table`, such as class, structural category, functional category, total length, total double bond (totaldb), hydroxyl group number (totaloh), fatty acid length (FA_length), the double bond of fatty acid(FA_db), hydroxyl group number of fatty acid(FA_oh).

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50, replace_zero=TRUE,
+                               zero2what='min', xmin=0.5, replace_NA=TRUE,
```

```
+                                NA2what='min', ymin=0.5, pct_transform=TRUE,
+                                data_transform=TRUE, trans_type='log',
+                                centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data,
+                                        exclude_var_missing=TRUE,
+                                        missing_pct_limit=50,
+                                        replace_zero=TRUE, zero2what='min',
+                                        xmin=0.5, replace_NA=TRUE,
+                                        NA2what='min', ymin=0.5,
+                                        pct_transform=TRUE,
+                                        data_transform=FALSE,
+                                        trans_type='log',
+                                        centering=FALSE, scaling=FALSE)
> # filter lipid_char_table according to exp_transform
> lipid_char_filter <- lipid_char_table %>%
+   filter(feature %in% exp_transform$feature)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                       data_transform=TRUE,
+                                       group_info = group_info,
+                                       paired=FALSE, test='t.test',
+                                       adjust_p_method='BH',
+                                       sig_stat='p.adj', sig_pvalue=0.05,
+                                       sig_FC=2)$DE_species_table_sig
> # get lipid characteristics
> char_var <- colnames(lipid_char_filter)[-1]
> # conduct hierarchical clustering
> DEspec_Hcluster<- Hclustering(exp_transform, DE_species_table_sig,
+                                group_info, lipid_char_filter,
+                                char_var = char_var[1],
+                                distfun='pearson', hclustfun='complete')
> # view result: matrix of all lipid species heatmap
> head(DEspec_Hcluster$all.lipid.data[, 1:4], 5)

                          ctrl3     ctrl6      ctrl8    ctrl10
PE 18:1;0-18:1;0    -0.57045553 0.2205936  0.1879053 0.3535505
PE 18:1;0-18:0;0    -1.07097214 0.7258493  0.7137582 0.7962292
PE 16:0;0-18:1;0    -1.75847725 0.1293000 -1.6431183 0.3493462
PE O- 16:1;0-20:4;0 -0.05404226 0.3758311  0.4160057 0.6231943
PE 16:0;0-20:4;0    -0.10148198 0.2187453  0.2718418 0.3603043

> # view result: matrix of significant lipid species heatmap
> head(DEspec_Hcluster$sig.lipid.data[, 1:4], 5)

                        ctrl9     ctrl2     ctrl3     ctrl6
PC O- 18:2;0-20:4;0 0.9932150 0.9468314 0.7946772 0.8697414
PC 18:0;0-18:3;0    0.9539972 0.8653070 0.7014008 0.6180106
PC O- 18:2;0-18:2;0 0.4391563 0.7028138 0.6068341 0.6941454
PC O- 18:0;0-16:1;0 0.8721702 0.3950769 1.2761599 1.4822756
PE O- 18:2;0-18:1;0 1.1039591 0.8097932 0.7735702 0.7746229
```

```
> # view result: heatmap of all lipid species
> DEspec_Hcluster$all.lipid
```

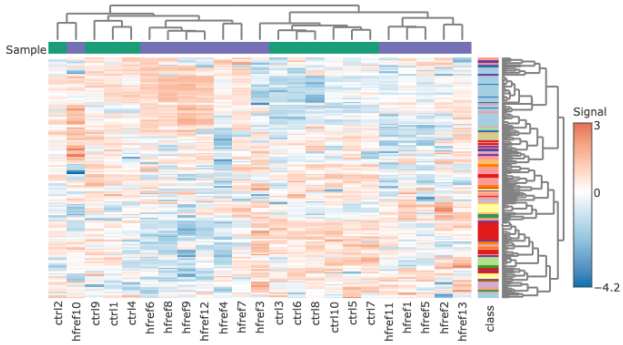

Figure 25: Heatmap of all lipid species

```
> # view result: heatmap of significant lipid species
```

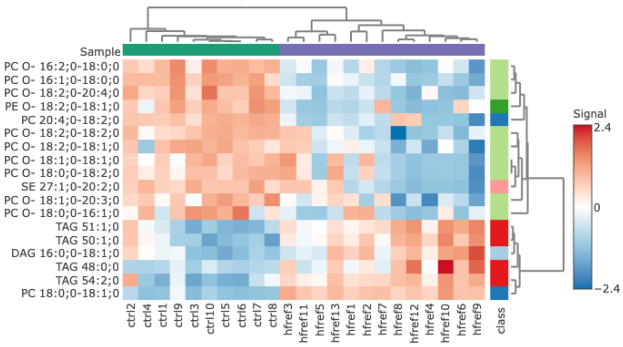

Figure 26: Heatmap of significant lipid species
Through heatmap, we can discover the difference between the two groups by observing the distribution of lipid species.

2.2.4 Characteristics analysis

The characteristics analysis visualizes the difference between control and experimental groups of significant lipid species categorized based on different lipid characteristics from `lipid_char_table`, such as class, structural category, functional category, total length, total double bond (totaldb), hydroxyl group number (totaloh), fatty acid length (FA_length), the double bond of fatty acid (FA_db), hydroxyl group number of fatty acid (FA_oh).

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50,
+                               replace_zero=TRUE, zero2what='min',
+                               xmin=0.5, replace_NA=TRUE,
+                               NA2what='min', pct_transform=TRUE,
+                               data_transform=TRUE, trans_type='log',
+                               ymin=0.5, centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data, exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE,
+                                       zero2what='min', xmin=0.5,
+                                       replace_NA=TRUE, NA2what='min',
+                                       ymin=0.5, pct_transform=TRUE,
+                                       data_transform=FALSE,
+                                       trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # filter lipid_char_table according to exp_transform
> lipid_char_filter <- lipid_char_table %>%
+   filter(feature %in% exp_transform_non_log$feature)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform=TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test='t.test',
+                                      adjust_p_method='BH',
+                                      sig_stat='p.adj', sig_pvalue=0.05,
+                                      sig_FC=2)$DE_species_table_sig
> # get lipid characteristics
> char_var <- colnames(lipid_char_filter)[-1]
> # conduct Sig_lipid_feature
> sig_feature_result <- Sig_lipid_feature(DE_species_table_sig,
+                                         lipid_char_filter,
+                                         char_var[1], sig_FC=2)
```

```
> # view result: bar chart
> sig_feature_result$barPlot
```

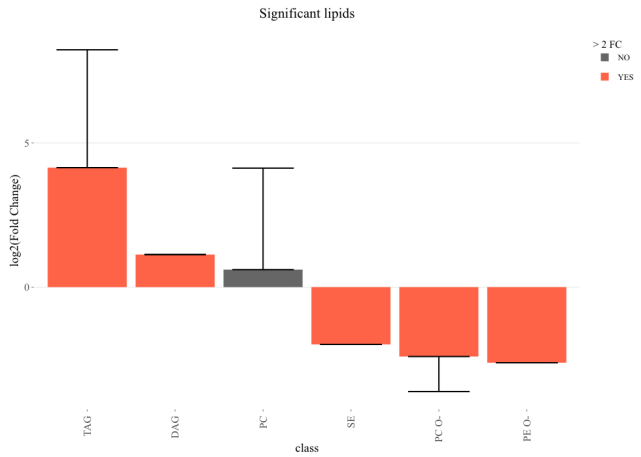

Figure 27: The bar chart of significant groups
The bar chart shows the significant groups (values) with mean fold change over 2 in the selected characteristics by colors (red for significant and black for insignificant).

```
> # view result: lollipop plot
> sig_feature_result$lolipop
```

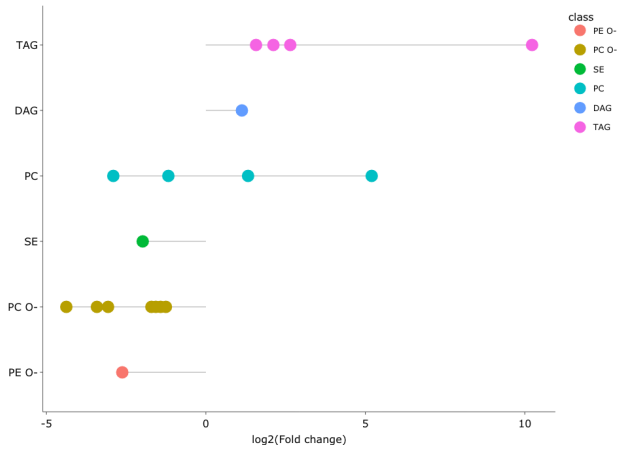

Figure 28: The lollipop chart of all significant groups
The lollipop chart compares multiple values simultaneously and it aligns the log2(fold change) of all significant groups (values) within the selected characteristics.

```
> # view result: word cloud
> sig_feature_result$word
```

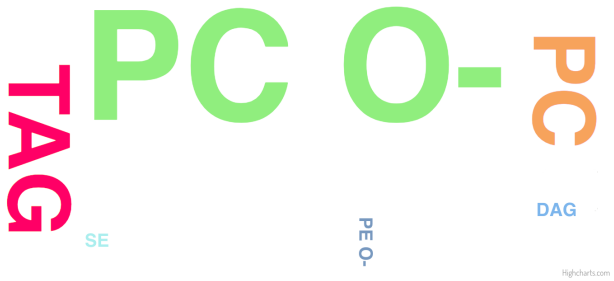

Figure 29: Word cloud with the count of each group
The word cloud shows the count of each group(value) of the selected characteristics.

2.2.5 Enrichment

Enrichment analysis assist us to determine whether significant lipid species are enriched in the categories of the selected characteristics.

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50, replace_zero=TRUE,
+                               zero2what='min', xmin=0.5, replace_NA=TRUE,
+                               NA2what='min', ymin=0.5, pct_transform=TRUE,
+                               data_transform=TRUE, trans_type='log',
+                               centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE, zero2what='min',
+                                       xmin=0.5, replace_NA=TRUE,
+                                       NA2what='min', ymin=0.5,
+                                       pct_transform=TRUE, data_transform=FALSE,
+                                       trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # filter lipid_char_table according to exp_transform
> lipid_char_filter <- lipid_char_table %>%
+   filter(feature %in% exp_transform$feature)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform = TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test='t.test',
+                                      adjust_p_method='BH',
+                                      sig_stat='p.adj', sig_pvalue=0.05,
+                                      sig_FC=2)$DE_species_table_sig
> # get lipid characteristics
> char_var <- colnames(lipid_char_filter)[-1]
> # conduct enrichment analysis
> enrich_result <- Enrichment(DE_species_table_sig,
+                             lipid_char_table = lipid_char_filter,
```

```
+                                char_var=char_var[1], sig_pvalue=0.05)
> # view result: summary data frame of enrichment
> head(enrich_result$enrich_char_table[, 1:5], 5)

# A tibble: 5 × 5
  condition characteristic sig.count total.count  p.value
  <chr>     <fct>              <int>       <int>    <dbl>
1 UP        DAG                    1           7 0.275
2 UP        PC                     2          36 0.452
3 UP        TAG                    4          38 0.0516
4 DOWN      PC                     2          36 0.702
5 DOWN      PC O-                  7          26 0.000620
```

- Note: A lipid species may have more than one fatty acid attached; thus, if the selected lipid characteristics are FA-related terms, we decompose lipid species into FA and do the enrichment instead of counting species.

```
> # view result: enrichment plot
> enrich_result$enrich_char_barplot
```

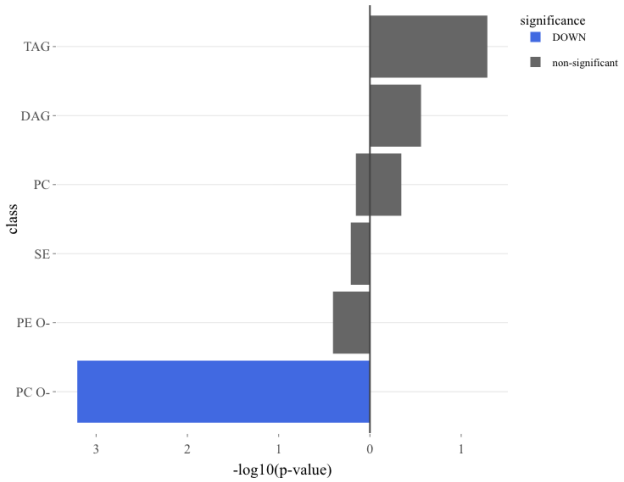

Figure 30: Enrichment plot for up/down/non-significant groups
The enrichment summary table and plot are classified into up/down/non-significant groups by log2 fold change of significant lipid species. Each group (value) of the selected characteristics will have a value of significant count and p-value within a summary table.

Then, only when we select 'class' aslipid characteristic, KEGG pathway analysis can be executed.The names of significant lipid classes in enrichment analysis will be matched to our mapping table and find corresponding KEGG compound ID and KEGG pathways.

Here, we provide data on lipid-related pathways and genes. For conducting pathway analysis, we will need the following two data, `DE.lipid.gene.path` and `DE.pathway.gene.list`.

- **DE.lipid.gene.path**: A data frame of pathway information, which can be retrieved and reorganized from KEGG or other databases. It includes the related pathways of lipids, ID, pathway ID, pathway name, gene ID, and gene name.
- **DE.pathway.gene.list**: A list that comprises the genes included in each pathway for compiling the significant network of the selected lipid class.

```
> # data frame of pathway information
> data("DE_lipid_gene_path")
> DE.lipid.gene.path <- DE_lipid_gene_path
> head(DE.lipid.gene.path[, 1:4], 5)

                                        key_name   DB ID_type     ID
1                        (3R)-3-Hydroxyacyl-ACP KEGG KEGG_ID C01271
2                        (3R)-3-Hydroxyacyl-ACP KEGG KEGG_ID C01271
3                        (3R)-3-Hydroxyacyl-ACP KEGG KEGG_ID C01271
4 (3S)-3-Hydroxyacyl-CoA; (S)-3-Hydroxyacyl-CoA KEGG KEGG_ID C00640
5 (3S)-3-Hydroxyacyl-CoA; (S)-3-Hydroxyacyl-CoA KEGG KEGG_ID C00640

> # list of genes included in each pathway
> data("DE_pathway_gene_list")
> DE.pathway.gene.list <- DE_pathway_gene_list
> head(DE.pathway.gene.list, 2)

$`hsa:00010`
 [1] "ENTREZID:10327"  "ENTREZID:124"    "ENTREZID:125"    "ENTREZID:126"
 [5] "ENTREZID:127"    "ENTREZID:128"    "ENTREZID:130"    "ENTREZID:130589"
 [9] "ENTREZID:131"    "ENTREZID:160287" "ENTREZID:1737"   "ENTREZID:1738"
[13] "ENTREZID:2023"   "ENTREZID:2026"   "ENTREZID:2027"   "ENTREZID:217"
[17] "ENTREZID:218"    "ENTREZID:219"    "ENTREZID:220"    "ENTREZID:2203"
[21] "ENTREZID:221"    "ENTREZID:222"    "ENTREZID:223"    "ENTREZID:224"
[25] "ENTREZID:226"    "ENTREZID:229"    "ENTREZID:230"    "ENTREZID:2538"
[29] "ENTREZID:2597"   "ENTREZID:26330"  "ENTREZID:2645"   "ENTREZID:2821"
[33] "ENTREZID:3098"   "ENTREZID:3099"   "ENTREZID:3101"   "ENTREZID:387712"
[37] "ENTREZID:3939"   "ENTREZID:3945"   "ENTREZID:3948"   "ENTREZID:441531"
[41] "ENTREZID:501"    "ENTREZID:5105"   "ENTREZID:5106"   "ENTREZID:5160"
[45] "ENTREZID:5161"   "ENTREZID:5162"   "ENTREZID:5211"   "ENTREZID:5213"
[49] "ENTREZID:5214"   "ENTREZID:5223"   "ENTREZID:5224"   "ENTREZID:5230"
[53] "ENTREZID:5232"   "ENTREZID:5236"   "ENTREZID:5313"   "ENTREZID:5315"
[57] "ENTREZID:55276"  "ENTREZID:55902"  "ENTREZID:57818"  "ENTREZID:669"
[61] "ENTREZID:7167"   "ENTREZID:80201"  "ENTREZID:83440"  "ENTREZID:84532"
[65] "ENTREZID:8789"   "ENTREZID:92483"  "ENTREZID:92579"  "ENTREZID:9562"

$`hsa:00020`
 [1] "ENTREZID:1431"  "ENTREZID:1737"  "ENTREZID:1738"  "ENTREZID:1743"
 [5] "ENTREZID:2271"  "ENTREZID:3417"  "ENTREZID:3418"  "ENTREZID:3419"
 [9] "ENTREZID:3420"  "ENTREZID:3421"  "ENTREZID:4190"  "ENTREZID:4191"
[13] "ENTREZID:47"    "ENTREZID:48"    "ENTREZID:4967"  "ENTREZID:50"
[17] "ENTREZID:5091"  "ENTREZID:5160"  "ENTREZID:5161"  "ENTREZID:5162"
[21] "ENTREZID:55753" "ENTREZID:6389"  "ENTREZID:6390"  "ENTREZID:6391"
[25] "ENTREZID:6392"  "ENTREZID:8801"  "ENTREZID:8802"  "ENTREZID:8803"
[29] "ENTREZID:5105"  "ENTREZID:5106"
```

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50, replace_zero=TRUE,
+                               zero2what='min', xmin=0.5, replace_NA=TRUE,
+                               ymin=0.5, NA2what='min', data_transform=TRUE,
+                               pct_transform=TRUE, trans_type='log',
```

```
+                                centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data, replace_zero=TRUE,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       zero2what='min', xmin=0.5,
+                                       replace_NA=TRUE, ymin=0.5,
+                                       NA2what='min', centering=FALSE,
+                                       pct_transform=TRUE, scaling=FALSE,
+                                       data_transform=FALSE, trans_type='log')
> # filter lipid_char_table according to exp_transform
> lipid_char_filter <- lipid_char_table %>%
+   filter(feature %in% exp_transform$feature)
> # filter significant lipid
> DE_species_table_sig <- DE_species_2(exp_transform_non_log,
+                                      data_transform=TRUE,
+                                      group_info = group_info,
+                                      paired=FALSE, test = 't.test',
+                                      adjust_p_method = 'BH',
+                                      sig_stat = 'p.adj', sig_pvalue=0.05,
+                                      sig_FC=2)$DE_species_table_sig
> # get lipid characteristics
> char_var <- colnames(lipid_char_filter)[-1]
> # get enrich characteristic table
> enrich_char_table <- Enrichment(DE_species_table_sig,
+                                 lipid_char_table=lipid_char_filter,
+                                 char_var = char_var[1],
+                                 sig_pvalue=0.05)$enrich_char_table
> # filter "class" of significant characteristics
> sig_enrich_class <- enrich_char_table %>% filter(significant == 'YES') %>%
+   distinct(characteristic) %>% .$characteristic
> # set output path
> dir.create(file.path(getwd(),"pathview_result"), recursive=TRUE,
+            showWarnings=TRUE)
> outPath <- file.path(getwd(), "pathview_result")
> # conduct pathview analysis
> path_result <- pathview_function(sig_enrich_class, path=outPath,
+                                  DE.lipid.gene.path, DE.pathway.gene.list)
```

```
> # view result: data frame of pathway information
> path_result

  key_name   DB     ID   path_id              path_name
2    PC O- KEGG C05212 hsa:00565 Ether lipid metabolism

> # view result: pathview plot
> showimage::show_image(file.path(outPath,"hsa00565.pathview.png"))
```

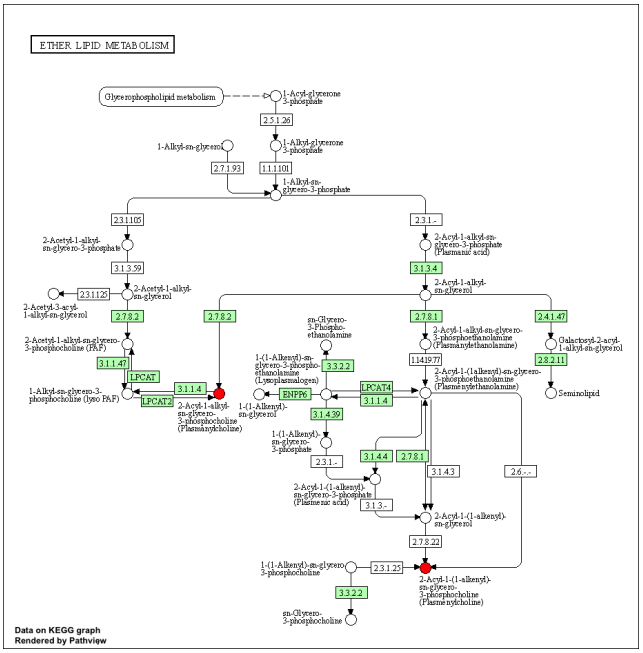

Figure 31: An example of KEGG pathway

2.3 Lipid characteristics analysis

After lipid species analysis, now let's move on to another main analysis of the Differential expression section – **'Lipid Characteristics Analysis'**. The massive degree of structural diversity of lipids contributes to the functional variety of lipids. The characteristics can range from subtle variance (i.e. the number of a double bond in the fatty acid) to major change (i.e. diverse backbones). In this section, lipid species are categorized and summarized into a new lipid expression table according to two selected lipid characteristics, then conducted differential expressed analysis. Samples are divided into two groups based on the input 'Group Information' table.

2.3.1 Differentially expressed analysis

In differentially expressed analysis, we are going to conduct two procedures of analysis - first is **'Characteristics'** and then **'Subgroup of characteristics'**. **'Characteristics'** is based on the first selected 'characteristics' while **'Subgroup of characteristics'** is the subgroup analysis of the previous section.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # aggregated(sum) expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[4])
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data_Spe2Char,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE,
```

```
+                                        zero2what='min', xmin=0.5,
+                                        replace_NA=TRUE, NA2what='min',
+                                        ymin=0.5, pct_transform=TRUE,
+                                        data_transform=FALSE,
+                                        trans_type='log',
+                                        centering=FALSE, scaling=FALSE)
> # conduct deferentially expressed of lipid characters
> DE_char_result <- DE_char_2(exp_transform_non_log, data_transform=TRUE,
+                              group_info = group_info, paired=FALSE,
+                              sig_pvalue=0.05, sig_FC=2,
+                              insert_ref_group=NULL, ref_group=NULL)
> # view result: data frame of expression
> head(DE_char_result$DE_char_exp_data[, 1:5], 5)

  totallength  control_01  control_02  control_03  control_04
1          14 0.007876691 0.005188040 0.002007919 0.002262715
2          16 0.614686226 0.371303913 0.440362404 0.360532543
3          18 0.499222531 0.340597404 0.336071017 0.233617647
4          20 0.138269347 0.045272091 0.047786065 0.057898780
5          22 0.021905258 0.004618018 0.011122576 0.007643132

> # view result: data frame of statistical analysis
> head(DE_char_result$DE_char_table_all[, 1:5], 5)

  totallength        method anova_pvalue post_hoc_test   mean_ctrl
1          14 two-way anova 4.344857e-05        t.test 0.005739745
2          16 two-way anova 4.344857e-05        t.test 0.551457134
3          18 two-way anova 4.344857e-05        t.test 0.527247670
4          20 two-way anova 4.344857e-05        t.test 0.096587704
5          22 two-way anova 4.344857e-05        t.test 0.014645490

> # view result: data frame with value of the continuous lipid characteristics
> head(DE_char_result$DE_char_combined_table[, 1:5])

        totallength control_01 control_02 control_03 control_04
1 totallength_index   41.41296   41.25918   40.19358   41.21941

> # view result: data frame with statistics of t.test
> head(DE_char_result$DE_char_combine_result_table[, 1:5])

        totallength method mean_ctrl   sd_ctrl mean_exp
1 totallength_index t.test  40.68546 0.4939548 41.02902
```

```
> # view result: bar plot of split_class
> DE_char_result$DE_char_barplot
```

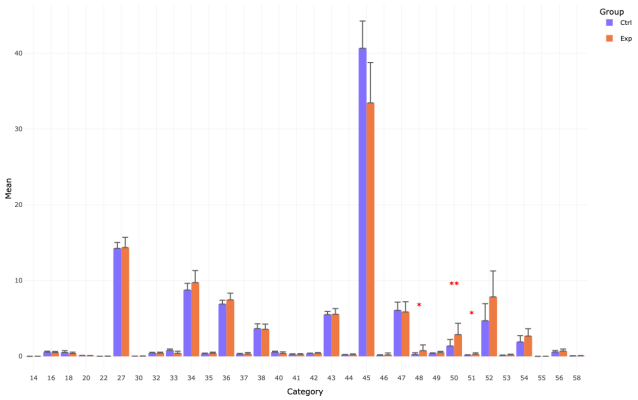

Figure 32: The results of 'Characteristics' analysis in the first section - bar plot

```
> # view result: sqrt-scaled bar plot of split_class
> DE_char_result$DE_char_barplot_sqrt
```

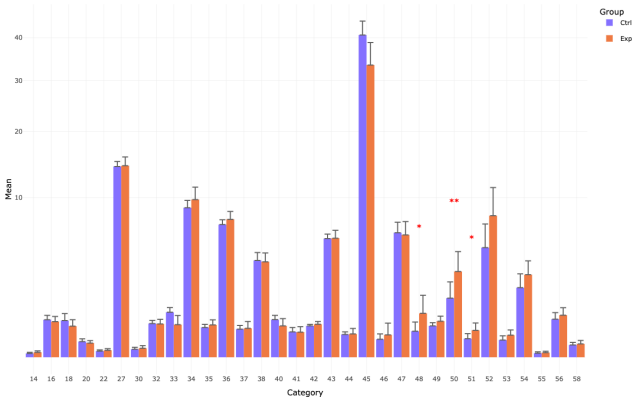

Figure 33: The results of 'Characteristics' analysis in the first section - sqrt-scaled bar plot

```
> # view result: line plot of split_class
> DE_char_result$DE_char_trendplot
```

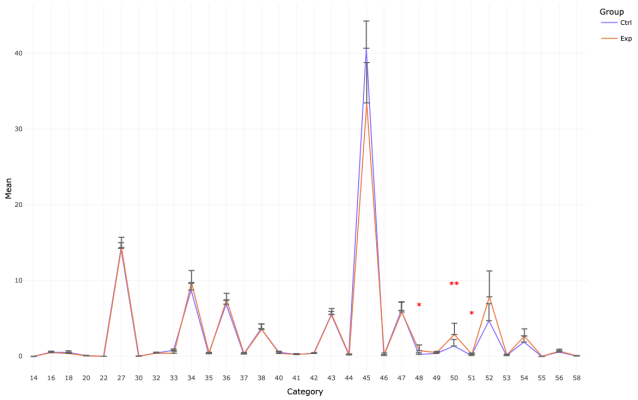

Figure 34: The results of 'Characteristics' analysis in the first section - line plot

```
> # view result: sqrt-scaled line plot of split_class
> DE_char_result$DE_char_trendplot_sqrt
```

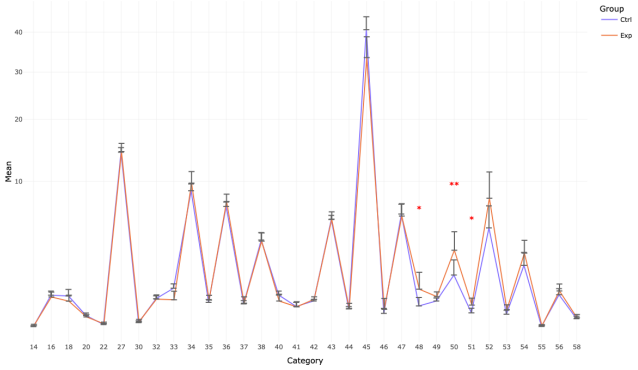

Figure 35: The results of 'Characteristics' analysis in the first section - sqrt-scaled line plot

```
> # view result: box plot of split_class
> DE_char_result$DE_char_boxplot
```

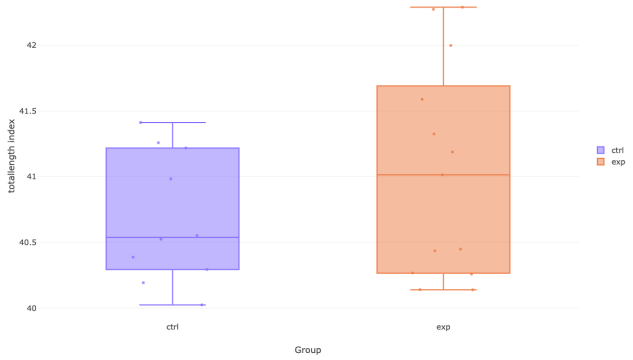

Figure 36: The results of 'Characteristics' analysis in the first section - box plot

In the **'Subgroup of characteristics'**, besides the selected characteristic in first section defined by parameter `char_var`, we can further choose another characteristic by parameter `split_var` (e.g. class). And then, analyzed results from the previous section are categorized by one of the subgroups (e.g. PC) of the selected characteristic.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # subgroup deferentially expressed of lipid characters
> DE.sub.char.2 <- DE_sub_char_2(exp_data, data_transform=TRUE,
+                                lipid_char_table=lipid_char_table,
+                                split_var = char_var[1],
+                                char_var = char_var[4],
+                                group_info = group_info,
+                                paired=FALSE, sig_pvalue=0.05,
+                                sig_FC=2, exclude_var_missing=TRUE,
+                                missing_pct_limit=50,
+                                replace_zero=TRUE, zero2what='min',
+                                xmin=0.5, replace_NA=TRUE,
+                                NA2what='min', ymin=0.5,
+                                pct_transform=TRUE, trans_type='log',
+                                centering=FALSE, scaling=FALSE)
> # get class of characteristics
> char.class <- unique(DE.sub.char.2[[2]][1])
> # visualize subgroup deferentially expressed of lipid characters
> sub_char_result <- DE_sub_char_plot_2(DE.sub.char.2[[2]],
+                                       DE.sub.char.2[[3]],
+                                       group_info=group_info,
+                                       char_var=char_var[4],
+                                       split_var=char_var[1],
+                                       split_class=char.class[5,],
+                                       insert_ref_group=NULL, ref_group=NULL)
```

- *Note: The star above the bar shows the significant difference of the specific subgroup of the selected characteristic between control and experimental groups.*

```
> # view result: bar plot of split_class
> sub_char_result[[1]]
```

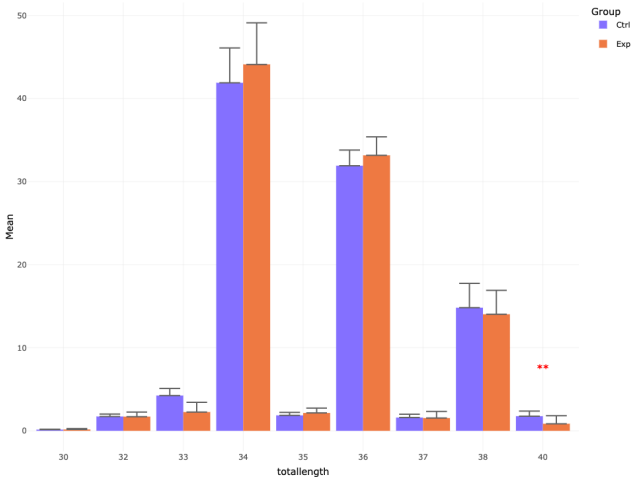

Figure 37: The results of 'Subgroup of characteristics' analysis in the second section - bar plot

```
> # view result: sqrt-scaled bar plot of split_class
> sub_char_result[[4]]
```

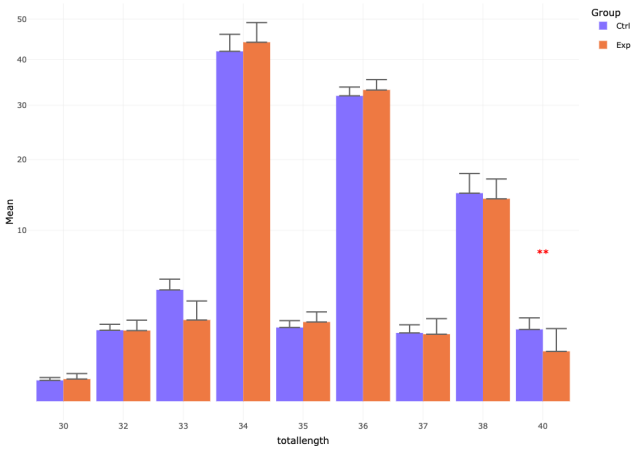

Figure 38: The results of 'Subgroup of characteristics' analysis in the second section - sqrt-scaled bar plot

```
> # view result: line plot of split_class
> sub_char_result[[2]]
```

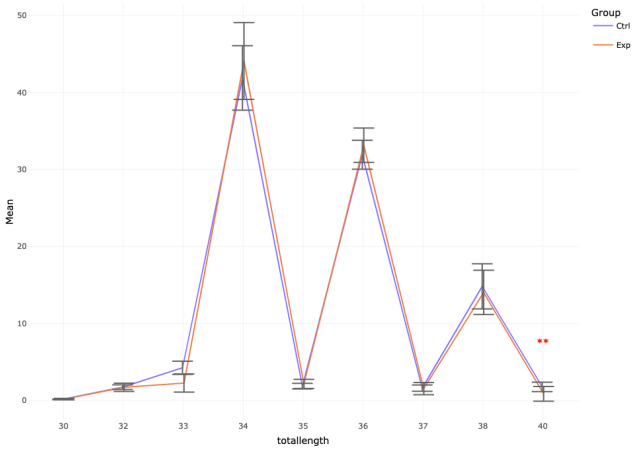

Figure 39: The results of 'Subgroup of characteristics' analysis in the second section - line plot

```
> # view result: sqrt-scaled line plot of split_class
> sub_char_result[[5]]
```

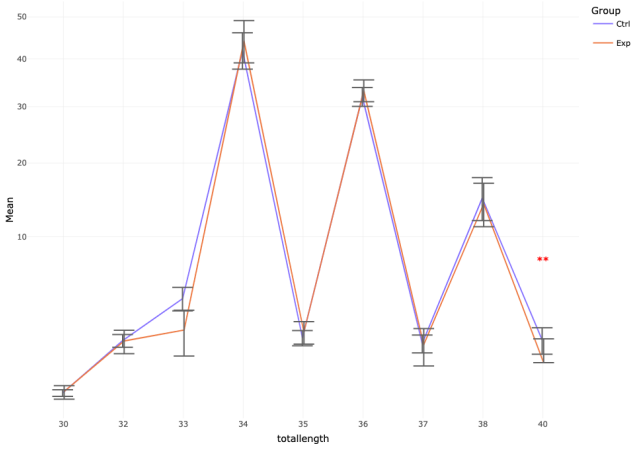

Figure 40: The results of 'Subgroup of characteristics' analysis in the second section - sqrt-scaled line plot

```
> # view result: box plot of split_class
> sub_char_result[[3]]
```

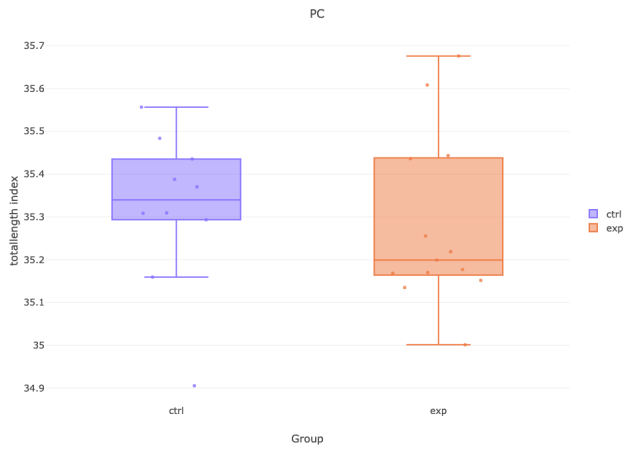

Figure 41: The results of 'Subgroup of characteristics' analysis in the second section - box plot

2.3.2 Dimensionality reduction

Dimensionality reduction is common when dealing with large numbers of observations and/or large numbers of variables in lipids analysis. It transforms data from a high-dimensional space into a low-dimensional space to retain vital properties of the original data and close to its intrinsic dimension. Here, we provide 4 dimensionality reduction methods, namely, PCA, t-SNE, UMAP, and PLS-DA. For the detailed information of the former three methods, please refer to Section 1.3.

1. **PCA (Principal component analysis)**
   PCA is an unsupervised linear dimensionality reduction and data visualization technique for high dimensional data, which tries to preserve the global structure of the data. For detailed information of PCA, please refer to Section 1.3.1.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # sum expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[4])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE,
+                                     zero2what='NA', xmin=0.5,
+                                     replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data_Spe2Char,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
```

```
+                                        replace_zero=TRUE,
+                                        zero2what='min', xmin=0.5,
+                                        replace_NA=TRUE, NA2what='min',
+                                        ymin=0.5, pct_transform=TRUE,
+                                        data_transform=FALSE,
+                                        trans_type='log',
+                                        centering=FALSE, scaling=FALSE)
> # filter significant lipid characteristics
> DE_char_table_sig <- DE_char_2(exp_transform_non_log,
+                                data_transform=TRUE,
+                                group_info = group_info,
+                                paired=FALSE, sig_pvalue=0.05,
+                                sig_FC=2)$DE_char_table_all %>%
+                                filter(sig=="yes")
> # conduct PCA
> DEchar_PCA <- PCA(exp_transform_class, group_info = group_info,
+                   sig_feature = DE_char_table_sig[,1],
+                   scaling=TRUE, centering=TRUE,
+                   cluster_method='kmeans',
+                   group_num=2, var1 = NULL, var2 = NULL,
+                   insert_ref_group=NULL, ref_group=NULL,
+                   n_PC=c(1,2), top_n_feature=10)
> # view result: PCA prcomp
> head(DEchar_PCA[[1]], 2)

$sdev
[1] 1.6953108 0.2786763 0.2196837

$rotation
          PC1        PC2        PC3
48 -0.5744300  0.8103231 -0.1157871
50 -0.5780849 -0.5017458 -0.6434819
51 -0.5795240 -0.3027006  0.7566534

> # view result: data frame of PCA rotated data
> head(DEchar_PCA[[2]][,1:5], 5)

           sample_name group        PC1         PC2         PC3
control_01  control_01     2 -0.5466139 -0.12594152  0.12327983
control_02  control_02     2 -1.3869805  0.13744289 -0.03504498
control_03  control_03     1  2.5648862  0.01778046 -0.11647110
control_04  control_04     2 -0.3094606 -0.25223171 -0.02916247
control_05  control_05     1  1.8308181  0.23114389  0.20304546

> # view result: data frame of PCA contribution table
> head(DEchar_PCA[[3]])

   feature      PC1       PC2       PC3
48      48 32.99698 65.662350  1.340665
50      50 33.41821 25.174885 41.406901
51      51 33.58480  9.162764 57.252434
```

```
> # view result: PCA plot
> DEchar_PCA[[4]]
```

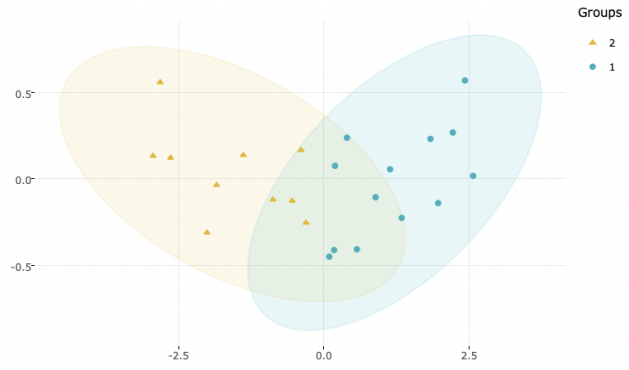

Figure 42: Results of PCA - PCA plot

```
> # view result: scree plot
> DEchar_PCA[[5]]
```

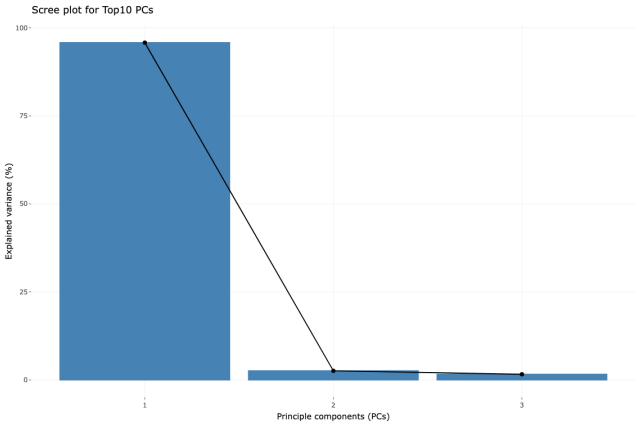

Figure 43: Results of PCA - scree plot

```
> # view result: correlation circle plot of variables
> DEchar_PCA[[7]]
```

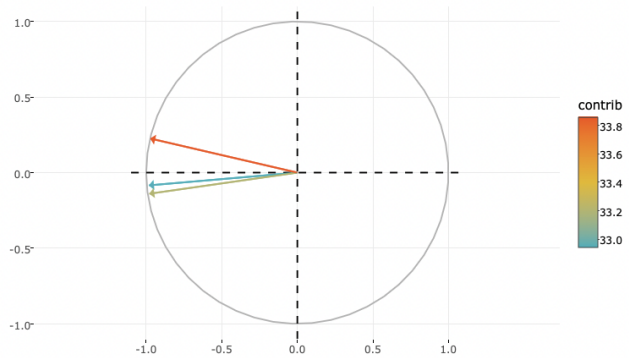

Figure 44: Results of PCA - bar plot

```
> # view result: bar plot of features contribution
> DEchar_PCA[[6]]
```

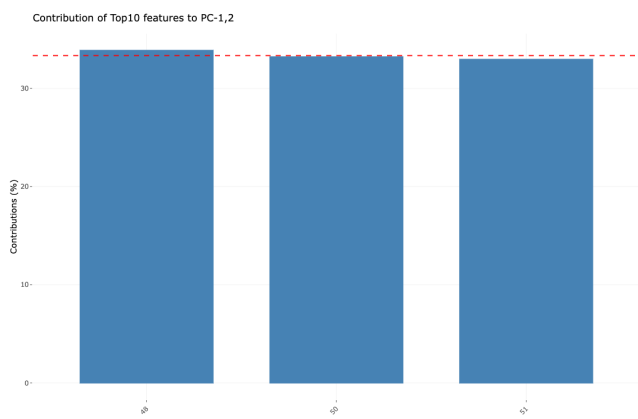

Figure 45: Results of PCA - correlation circle plot

2. **t-SNE (t-distributed stochastic neighbour embedding)**
   t-Distributed Stochastic Neighbour Embedding (t-SNE) is an unsupervised non-linear dimensionality reduction technique that tries to retain the local structure(cluster) of data when visualising the high-dimensional datasets. For detailed information of t-SNE, please refer to Section 1.3.2.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # sum expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[4])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE,
+                                     zero2what='NA', xmin=0.5,
```

```
+                                       replace_NA=TRUE,
+                                       NA2what='min', ymin=0.5,
+                                       pct_transform=TRUE,
+                                       data_transform=TRUE,
+                                       trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data_Spe2Char,
+                                         exclude_var_missing=TRUE,
+                                         missing_pct_limit=50,
+                                         replace_zero=TRUE,
+                                         zero2what='min', xmin=0.5,
+                                         replace_NA=TRUE, NA2what='min',
+                                         ymin=0.5, pct_transform=TRUE,
+                                         data_transform=FALSE,
+                                         trans_type='log',
+                                         centering=FALSE, scaling=FALSE)
> # filter significant lipid characteristics
> DE_char_table_sig <- DE_char_2(exp_transform_non_log,
+                                data_transform=TRUE,
+                                group_info = group_info,
+                                paired=FALSE, sig_pvalue=0.05,
+                                sig_FC=2)$DE_char_table_all %>%
+                                filter(sig=="yes")
> # conduct t-SNE
> DEchar_tsne <- tsne(exp_transform_class, group_info = group_info,
+                     sig_feature=DE_char_table_sig[,1],
+                     pca=TRUE, perplexity=5, max_iter=500,
+                     cluster_method='kmeans', group_num=2,
+                     var1 = 'euclidean', var2=NULL,
+                     insert_ref_group=NULL, ref_group=NULL)

Performing PCA
Read the 23 x 3 data matrix successfully!
Using no_dims = 2, perplexity = 5.000000, and theta = 0.000000
Computing input similarities...
Symmetrizing...
Done in 0.00 seconds!
Learning embedding...
Iteration 50: error is 56.742924 (50 iterations in 0.00 seconds)
Iteration 100: error is 54.269117 (50 iterations in 0.00 seconds)
Iteration 150: error is 60.322952 (50 iterations in 0.00 seconds)
Iteration 200: error is 58.386650 (50 iterations in 0.00 seconds)
Iteration 250: error is 54.387840 (50 iterations in 0.00 seconds)
Iteration 300: error is 1.080389 (50 iterations in 0.00 seconds)
Iteration 350: error is 0.207586 (50 iterations in 0.00 seconds)
Iteration 400: error is 0.122523 (50 iterations in 0.00 seconds)
Iteration 450: error is 0.117896 (50 iterations in 0.00 seconds)
Iteration 500: error is 0.116646 (50 iterations in 0.00 seconds)
Fitting performed in 0.00 seconds.

> # view result: data frame of t-SNE data
```

```
> head(DEchar_tsne[[1]], 5)

  sample_name group     tsne1      tsne2
1  control_01     1 -26.22530   8.596066
2  control_02     1 -44.06333  26.878557
3  control_03     2  70.26458 -27.635304
4  control_04     1 -18.26301   4.684444
5  control_05     2  53.05343 -33.340064
```

```
> # view result: t-SNE plot
> DEchar_tsne[[2]]
```

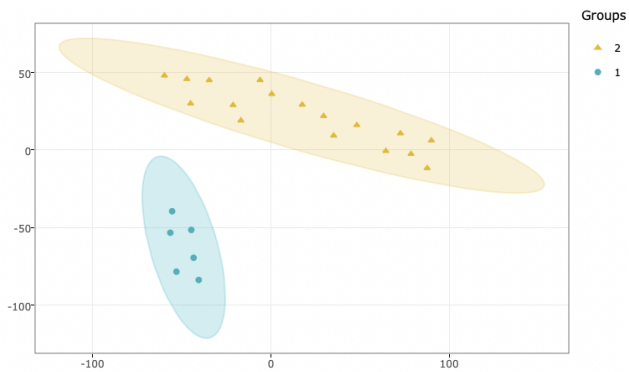

Figure 46: t-SNE plot

3. **UMAP (Uniform Manifold Approximation and Projection)**
   UMAP using a nonlinear dimensionality reduction method, Manifold learning, which effectively visualizing clusters or groups of data points and their relative proximities. For detailed information of UMAP, please refer to Section 1.3.3.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # sum expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[4])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE,
+                                     zero2what='NA', xmin=0.5,
+                                     replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data_Spe2Char,
+                                       exclude_var_missing=TRUE,
```

```
+                                        missing_pct_limit=50,
+                                        replace_zero=TRUE,
+                                        zero2what='min', xmin=0.5,
+                                        replace_NA=TRUE, NA2what='min',
+                                        ymin=0.5, pct_transform=TRUE,
+                                        data_transform=FALSE,
+                                        trans_type='log',
+                                        centering=FALSE, scaling=FALSE)
> # filter significant lipid characteristics
> DE_char_table_sig <- DE_char_2(exp_transform_non_log,
+                                data_transform=TRUE,
+                                group_info = group_info,
+                                paired=FALSE, sig_pvalue=0.05,
+                                sig_FC=2)$DE_char_table_all %>%
+                                filter(sig=="yes")
> # conduct UMAP
> DEchar_UMAP <- UMAP(exp_transform_class, group_info = group_info,
+                     sig_feature = DE_char_table_sig[,1],
+                     n_neighbors=15, scale=TRUE, group_num=2,
+                     metric = 'euclidean',
+                     cluster_method = 'kmeans',
+                     var1=NULL, var2=NULL,
+                     insert_ref_group=NULL, ref_group=NULL)
> # view result: data frame of UMAP data
> head(DEchar_UMAP[[1]], 5)

           sample_name group     UMAP-1     UMAP-2
control_01  control_01     2  0.4889651 -0.9097222
control_02  control_02     2  1.5379871 -1.1903843
control_03  control_03     1 -1.7016882  2.0996615
control_04  control_04     2  0.5782412 -0.1531183
control_05  control_05     1 -1.2176780  1.5065559
```

```
> # view result: UMAP plot
> DEchar_UMAP[[2]]
```

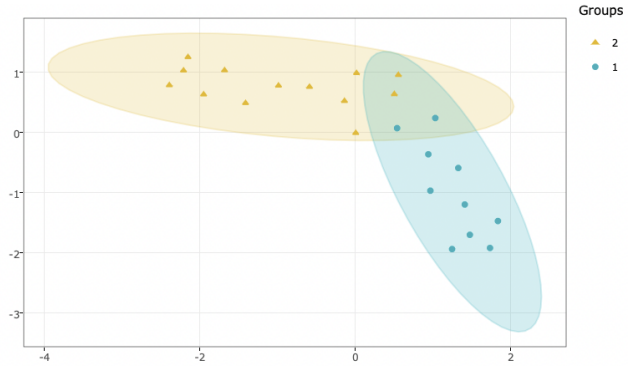

Figure 47: UMAP plot

4. **PLS-DA**

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # sum expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[4])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50,
+                                     replace_zero=TRUE,
+                                     zero2what='NA', xmin=0.5,
+                                     replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=TRUE,
+                                     trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data_Spe2Char,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE,
+                                       zero2what='min', xmin=0.5,
+                                       replace_NA=TRUE, NA2what='min',
+                                       ymin=0.5, pct_transform=TRUE,
+                                       data_transform=FALSE,
+                                       trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # filter significant lipid characteristics
> DE_char_table_sig <- DE_char_2(exp_transform_non_log,
+                                data_transform=TRUE,
+                                group_info = group_info,
+                                paired=FALSE, sig_pvalue=0.05,
+                                sig_FC=2)$DE_char_table_all %>%
+                                filter(sig=="yes")
> # conduct PLSDA
> DEchar_PLSDA <- PLSDA(exp_transform_class, group_info = group_info,
+                       sig_feature=DE_char_table_sig[,1],
+                       ncomp=2, scaling=TRUE, cluster_method='group_info',
+                       group_num = NULL, var1=NULL, var2=NULL,
+                       insert_ref_group=NULL, ref_group=NULL)
> # view result: data frame of sample variate
> head(DEchar_PLSDA[[1]], 5)

           sample_name group     PLSDA1      PLSDA2
control_01  control_01  ctrl -0.5475526  0.03467860
control_02  control_02  ctrl -1.3706971 -0.09832257
control_03  control_03  ctrl  2.5472937  0.05654781
control_04  control_04  ctrl -0.3303837  0.22597494
```

```
control_05  control_05  ctrl  1.8520391 -0.30215285
> # view result: data frame of sample loading
> head(DEchar_PLSDA[[2]])
       PLSDA1     PLSDA2
48 -0.5109755 -0.6621155
50 -0.6520544  0.7141024
51 -0.5601153 -0.2272903
```

```
> # view result: PLS-DA sample plot
> DEchar_PLSDA[[3]]
```

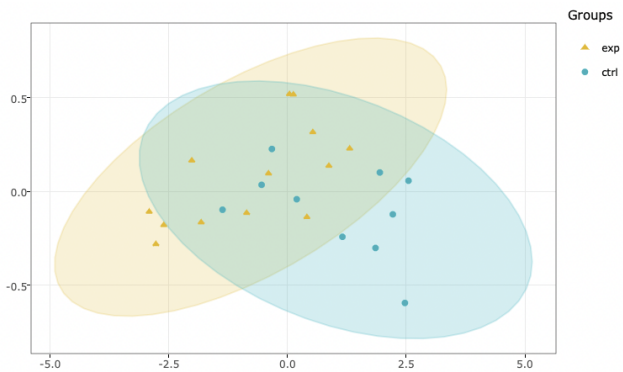

Figure 48: PLS-DA sample plot

```
> # view result: PLS-DA loading plot
> DEchar_PLSDA[[4]]
```

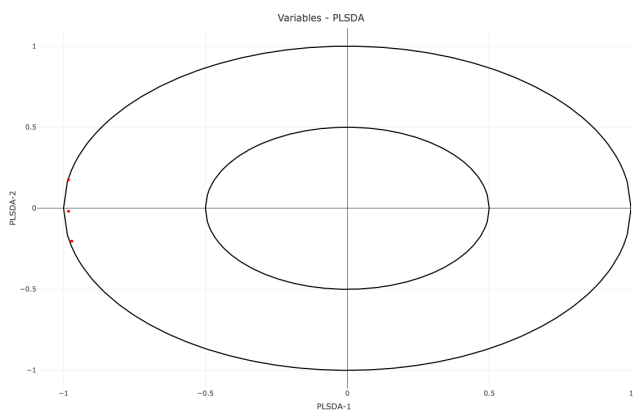

Figure 49: PLS-DA loading plot
In the PLS-DA loading plot, the distance to the center of the variables indicates the contribution of the variable. The value of the x-axis reveals the contribution of the variable to PLS-DA-1, whereas the value of the y-axis discloses the contribution of the variable to PLS-DA-2.

2.3.3 Hierarchical clustering

A new lipid expression table summed up from species is clustered and shown on the heatmap using hierarchical clustering.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # sum expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[4])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50, replace_zero=TRUE,
+                                     replace_NA=TRUE,zero2what='NA',
+                                     NA2what='min', pct_transform=TRUE,
+                                     xmin=0.5, data_transform=TRUE,
+                                     ymin=0.5, trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # data processing of exp_data (without log10 transformation)
> exp_transform_non_log <- data_process(exp_data_Spe2Char,
+                                       exclude_var_missing=TRUE,
+                                       missing_pct_limit=50,
+                                       replace_zero=TRUE, zero2what='min',
+                                       xmin=0.5, replace_NA=TRUE, NA2what='min',
+                                       ymin=0.5, pct_transform=TRUE,
+                                       data_transform=FALSE, trans_type='log',
+                                       centering=FALSE, scaling=FALSE)
> # filter significant lipid characteristics
> DE_char_table_sig <- DE_char_2(exp_transform_non_log,
+                                data_transform=TRUE, group_info = group_info,
+                                paired=FALSE, sig_pvalue=0.05, sig_FC=2
+                               )$DE_char_table_all %>% filter(sig=="yes")
> # conduct hierarchical clustering
> DEchar_Hcluster <- Hclustering(exp_transform_class, DE_char_table_sig,
+                                group_info, lipid_char_table=NULL,
+                                char_var=NULL, distfun='pearson',
+                                hclustfun='complete')
> # view result: matrix of all lipid species heatmap
> head(DEchar_Hcluster$all.lipid.data[, 1:5], 5)

      hfref3     ctrl5     ctrl7       ctrl6      ctrl8
36 1.8556987 0.5180388 0.5170678 -0.34186785 -0.2904855
14 0.5235378 0.5388503 0.4404471 -1.39383685  0.9335044
18 0.4053947 1.4803137 1.9614334  0.20197549  0.6085228
16 0.7343398 1.2327273 0.8511100  0.03909701  0.3186890
20 0.2614331 1.0882464 1.3247222  0.05499599  0.5322051

> # view result: matrix of significant lipid species heatmap
> head(DEchar_Hcluster$sig.lipid.data[, 1:5])

        ctrl8      ctrl5  hfref12     ctrl3    hfref13
51 -1.4046357 -0.9773353 1.589734 -1.579923 -0.1388107
50 -1.8327292 -1.3049998 1.381710 -1.416696  0.4152979
```

```
48 -0.9584274 -0.8878857 1.599940 -1.445454  0.4108188
```

```
> # view result: heatmap of all lipid species
> DEchar_Hcluster$all.lipid
```

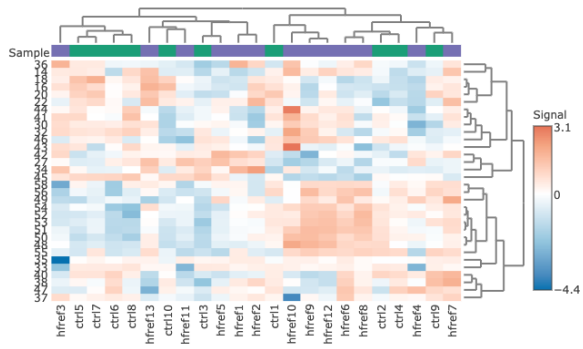

Figure 50: Heatmap of all lipid species

```
> # view result: heatmap of significant lipid species
> DEchar_Hcluster$sig.lipid
```

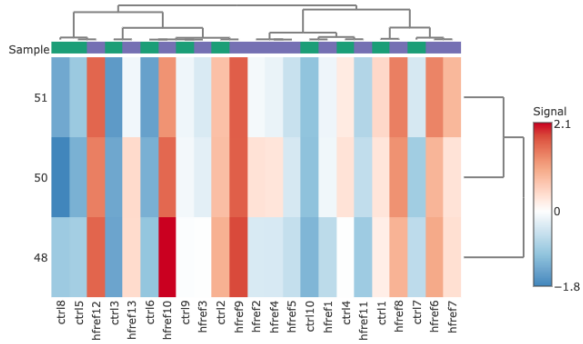

Figure 51: Heatmap of all lipid species

Through heatmap of lipid characteristic expression differences between the control group and the experimental group, we may discover the difference between the two groups by observing the distribution of lipid characteristic expression. Columns are all samples and rows are the significant characteristic group (value) selected from the first 'Characteristics' section (Section 2.3.1).

3 Machine learning analysis

After identifying significant lipid species and lipids characteristics, now we are going to conduct machine learning for feature selection and then view the importance of each feature. Lipid species and lipid characteristics data will be combined to predict the binary outcome using various machine learning methods and select the best feature combination to explore further relationships. For cross-validation, Monte-Carlo cross-validation (MCCV) is executed to evaluate the model performance and to reach statistical significance.

Monte-Carlo cross-validation is a model validation technique that we used to create multiple random splits of the dataset into training and validation data, which prevent an unnecessary large model and thus prevent over-fitting for the calibration model [5]. With MCCV, we can conduct a split-sample CV multiple times and aggregate the results from each to quantify predictive performance for a candidate mode. For each CV, data is randomly split into training and testing data. The training data is used to select the top 2, 3, 5, 10, 20, 50, and 100 important features for model training. Then, the model will be validated by testing data. If the data are less than 100 features, the total feature number is set as the maximum. The proportion of data used for testing and the times of cross-validation (CV) and can be defined by the parameters `split_prop` and `nfold`. *(Note: The more cross-validation times are, the longer it takes to calculate the results.)*

Feature selection methods are aimed to rank the most significant variables to a model to predict the target variable. Our platform provides two categories of feature selection methods: the univariate and the multivariate analysis. Univariate analysis, including p-value, p-value*Fold Change or ROC, compares each feature between two groups and picks top N features based on -log10(p-value), –log10(p-value)*Fold change or Area Under Curve (AUC), respectively according to the user-selected ranking methods. On the other hand, for multivariate analysis, we offer Random Forest, Linear SVM (e1071), Lasso (glmnet), Ridge (glmnet), and ElasticNet (glmnet). Random Forest (ranger) uses built-in feature importance results, while others rank the features according to the absolute value of their coefficients in the algorithm. *(Note: The names in the bracket are the packages we adopt.)*

Here, we provide eight feature ranking methods and six classification methods for training and selecting the best model.

- **Feature ranking methods:** p-value, p-value*FC, ROC, Random Forest, Linear SVM, Lasso, Ridge, ElasticNet.
- **Classification methods:** Random Forest, Linear SVM, Lasso, Ridge, ElasticNet, XGBoost.

Furthermore, we will conduct a series of consequent analyses to evaluate the methods and visualize the results of machine learning, including ROC/PR curve, model predictivity, sample probability, feature importance, and network.

3.1 Input data

First, we have to read the input data needed for the machine learning section. We have to prepare lipid expression data (`exp_data`), lipid characteristics table (`lipid_char_table`), and a condition table of sample names and clinical conditions (`condition_table`) as input data.

```
> # clears all objects from workspace
> rm(list = ls())
> # lipid expression data
```

```
> data("ML_exp_data")
> exp_data <- ML_exp_data
> head(exp_data[, 1:5], 5)

    feature ACH_000973 ACH_000070 ACH_000411 ACH_001306
1 C14.0.LPC   6.073383   5.986797   5.813777   5.693855
2 C16.1.LPC   6.065947   5.740817   5.849202   5.612929
3 C16.0.LPC   6.080238   5.588912   5.807984   5.578419
4 C18.2.LPC   5.463088   5.985183   6.082293   6.006846
5 C18.1.LPC   6.182150   5.486523   5.925475   5.665173

> # lipid characteristics table
> data("ML_lipid_char_table")
> lipid_char_table <- ML_lipid_char_table
> head(lipid_char_table[, 1:4], 5)

    feature class structural_category functional_category
1 C14.0.LPC   LPC                 GPL                 LYS
2 C16.1.LPC   LPC                 GPL                 LYS
3 C16.0.LPC   LPC                 GPL                 LYS
4 C18.2.LPC   LPC                 GPL                 LYS
5 C18.1.LPC   LPC                 GPL                 LYS

> # condition table
> data("ML_condition_table")
> condition_table <- ML_condition_table
> head(condition_table, 5)

  sample_name group
1  ACH_000294     0
2  ACH_000167     0
3  ACH_000402     0
4  ACH_000732     0
5  ACH_000324     0
```

After importing the input data, sometimes, we may need to conduct data processing before analysis. Here, we provide the `ML_data_process` function for data processing, including removing features with missing values, missing values imputation, percentage transformation, log10 transformation, etc.

```
> # lipid expression data
> head(exp_data[, 1:5], 5)

    feature ACH_000973 ACH_000070 ACH_000411 ACH_001306
1 C14.0.LPC   6.073383   5.986797   5.813777   5.693855
2 C16.1.LPC   6.065947   5.740817   5.849202   5.612929
3 C16.0.LPC   6.080238   5.588912   5.807984   5.578419
4 C18.2.LPC   5.463088   5.985183   6.082293   6.006846
5 C18.1.LPC   6.182150   5.486523   5.925475   5.665173

> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50, replace_zero=TRUE,
+                                     zero2what='min', xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
```

```
+                                    pct_transform=TRUE,
+                                    data_transform=TRUE, trans_type='log',
+                                    centering=FALSE, scaling=FALSE)
> # exp_data after data processing
> head(exp_transform_table[, 1:5], 5)

    feature ACH_000973 ACH_000070 ACH_000411 ACH_001306
1 C14.0.LPC 0.06386282 0.04626122 0.04606746 0.02961827
2 C16.1.LPC 0.06333076 0.02804039 0.04870571 0.02340142
3 C16.0.LPC 0.06435273 0.01639396 0.04563451 0.02072300
4 C18.2.LPC 0.01787035 0.04614412 0.06567643 0.05285837
5 C18.1.LPC 0.07157169 0.00836389 0.05433226 0.02742505
```

3.2 ROC/PR curve

The ROC and Precision-Recall (PR) curves are very common methods to evaluate the diagnostic ability of a binary classifier. The mean AUC and 95% confidence interval of the ROC and PR curve are calculated from all CV runs in each feature number. Theoretically, the higher the AUC is, the better the model performs. PR curve is more sensitive to data with highly skewed datasets (i.e., rare positive samples), and offers a more informative view of an algorithm's performance [6]. A random classifier yields a ROC-AUC about 0.5 and a PR-AUC close to a positive sample proportion. On the contrary, both AUC equal to 1 represents perfect performance in two methods.

Speaking of interpreting plots, the ROC curve is created with 'sensitivity' (proportion of positive samples that are correctly classified) as y-axis and '1-specificity' (proportion of negative samples that are correctly classified) as x-axis based on different thresholds whereas the PR curve is a similar graph with 'precision' (proportion of positive samples out of those that are predicted positive) on the y-axis and 'recall' (=sensitivity) on the x-axis. Generally, a better model shows a ROC curve approaching the left upper corner and a PR curve around the right upper corner.

To combine the testing results from all CV runs, 300 thresholds are evenly distributed from 0 to 1. The thresholds are then calculated the corresponding sensitivity, specificity, precision, and recall with predicted probabilities and true labels of testing samples in each CV. These values are then averaged to plot a final ROC and PR curve.

Now, we are going to conduct calculation for plotting ROC curves first, and then the PR curves.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]],ranking_method='Random_forest',
```

```
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # ROC curves
> ROC_result <- ROC_plot_all(ML_output[[3]], ML_output[[5]], feature_n=10)
> # view result: ROC data frame of 10 features
> head(ROC_result[[3]][, 1:5], 5)

  ranking_method     ML_method cv_fold feature_num sensitivity
1  Random_forest Random_forest       1          10           1
2  Random_forest Random_forest       1          10           1
3  Random_forest Random_forest       1          10           1
4  Random_forest Random_forest       1          10           1
5  Random_forest Random_forest       1          10           1

> # view result: data frame of ROC values
> head(ROC_result[[1]][, 1:5], 5)

# A tibble: 5 × 5
# Groups:   feature_num [1]
  ranking_method ML_method     cv_fold feature_num threshold
  <chr>          <chr>         <chr>         <dbl>     <dbl>
1 Random_forest  Random_forest mean              2   0
2 Random_forest  Random_forest mean              2   0.00334
3 Random_forest  Random_forest mean              2   0.00669
4 Random_forest  Random_forest mean              2   0.0100
5 Random_forest  Random_forest mean              2   0.0134
```

```
> # view result: ROC curve plot
> ROC_result[[2]]
```

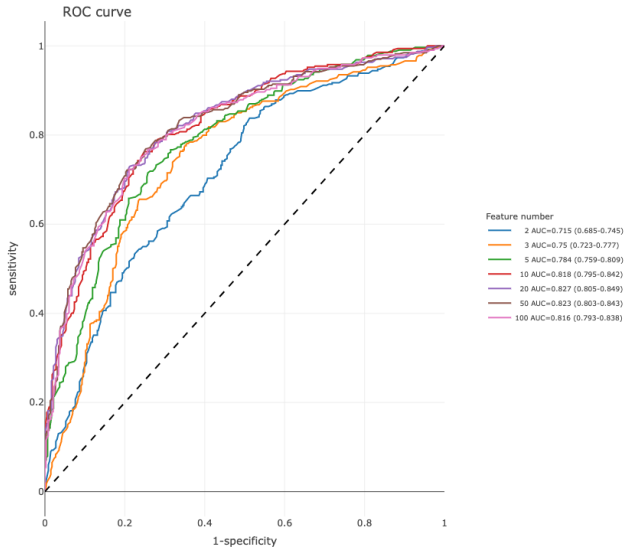

Figure 52: ROC curve plot
The plot shows the average ROC curve for different feature numbers with their mean AUC and 95% confidence interval.

```
> # view result: average ROC curve plot of 10 features
> ROC_result[[4]]
```

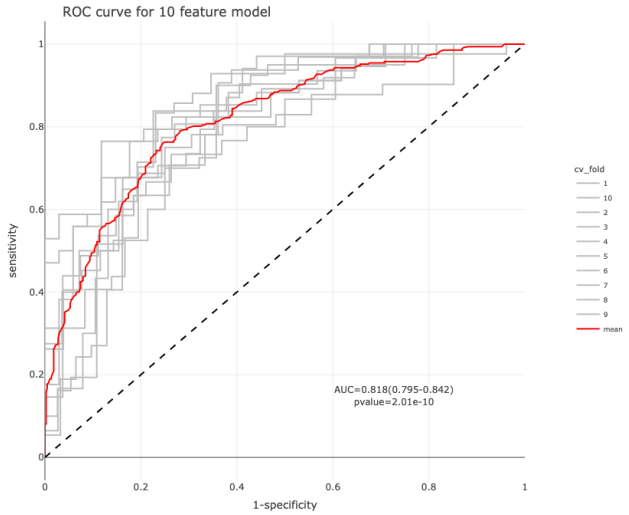

Figure 53: average ROC curve plot of 10 features
The plot displays average ROC curves of user-defined features. Each CV is in grey, and the red line is the average of those cross-validations (CVs) for the ROC curves.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> ## data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
```

```
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]],ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # PR curves
> PR_result <- PR_plot_all(ML_output[[4]], ML_output[[5]], feature_n=10)
> # view result: data frame of precision and recall values
> head(PR_result[[1]][, 1:5], 5)

# A tibble: 5 × 5
# Groups:   feature_num [1]
  ranking_method ML_method     cv_fold feature_num threshold
  <chr>          <chr>         <chr>         <dbl>     <dbl>
1 Random_forest  Random_forest mean              2   0
2 Random_forest  Random_forest mean              2   0.00334
3 Random_forest  Random_forest mean              2   0.00669
4 Random_forest  Random_forest mean              2   0.0100
5 Random_forest  Random_forest mean              2   0.0134

> # view result: data frame of PR values
> head(PR_result[[3]][, 1:5], 5)

  ranking_method     ML_method cv_fold feature_num precision
1  Random_forest Random_forest       1          10         1
2  Random_forest Random_forest       1          10         1
3  Random_forest Random_forest       1          10         1
4  Random_forest Random_forest       1          10         1
5  Random_forest Random_forest       1          10         1
```

```
> # view result: PR curve plot
> PR_result[[2]]
```

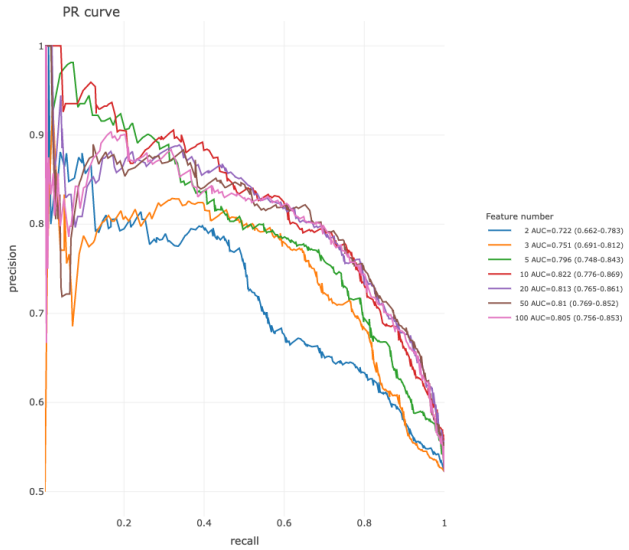

Figure 54: PR curve plot
The plot shows the average PR curve for different feature numbers with their mean AUC and 95% confidence interval.

```
> # view result: average PR curve plot of 10 features
> PR_result[[4]]
```

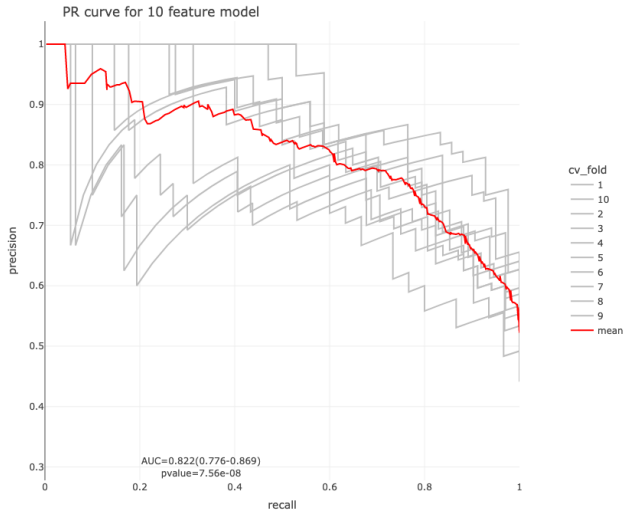

Figure 55: average PR curve plot of 10 features
The plot displays the average PR curves of user-defined features. Each CV is in grey, and the red line is the average of those cross-validations (CVs) for the PR curves.

3.3 Model performance

After constructing the model, it is necessary to evaluate the performance of our model. Here, we provide many useful indicators to evaluate model performance. For each feature number, we calculate and plot the average value and 95% confidence interval of accuracy, sensitivity

(recall), specificity, positive predictive value (precision), negative predictive value, F1 score, prevalence, detection rate, detection prevalence, and balanced accuracy in all CV runs with confusion matrix function in carat package. All these indicators can be described in terms of true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

Here, all the provided evaluation indicators are listed below. We can define the evaluation method by the parameter `method`.

- Sensitivity = Recall = $\frac{TP}{(TP+FN)}$
- Specificity = $\frac{TN}{(FP+TN)}$
- Prevalence = $\frac{(TP+FN)}{(TP+FP+FN+TN)}$
- Positive predictive value (PPV) = Precision = $\frac{TP}{(TP+FP)}$
- Negative predictive value (NPV) = $\frac{TN}{(FN+TN)}$
- Detection rate = $\frac{TP}{(TP+FP+FN+TN)}$
- Detection prevalence = $\frac{(TP+FP)}{(TP+FP+FN+TN)}$
- Balanced accuracy = $\frac{(Sensitivity+Specificity)}{2}$
- F1 score = $\frac{2\times Precision\times Recall}{(Precision+Recall)}$

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # conduct model evaluation
> evaluate_result <- evalution_plot(ML_output[[2]], method='Accuracy')
> # view result: data frame of model evaluation information
> head(evaluate_result[[1]][, 1:5], 5)
```

```
# A tibble: 5 × 5
# Groups:   feature_num [1]
  ranking_method ML_method     cv_fold feature_num index
  <chr>          <chr>           <int>       <dbl> <chr>
1 Random_forest  Random_forest       1           2 Accuracy
2 Random_forest  Random_forest       2           2 Accuracy
3 Random_forest  Random_forest       3           2 Accuracy
4 Random_forest  Random_forest       4           2 Accuracy
5 Random_forest  Random_forest       5           2 Accuracy
```

```
> # view result: model performance plot
> evaluate_result[[2]]
```

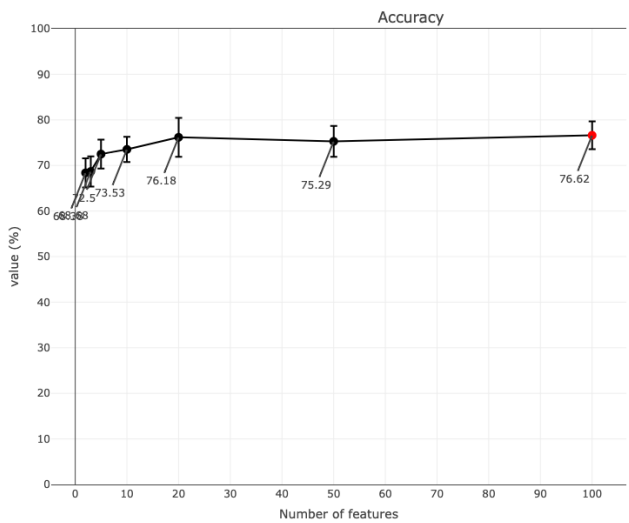

Figure 56: Model performance (Accuracy)
The evaluation plot shows the model performance of accuracy. The highest value is marked in red.

3.4 Predicted probability

The average predicted probabilities of each sample in testing data from all CV runs assist us to explore those incorrect or uncertain labels.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
```

```
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # compute and visualize the average predicted probabilities
> prob_result <- probability_plot(ML_output[[1]], feature_n=10)
> # view result: data frame of confusion matrix
> head(prob_result[[1]][, 1:5], 5)

# A tibble: 5 × 5
# Groups:   ID, feature_num [5]
  ranking_method ML_method     feature_num ID         true_label
  <chr>          <chr>               <dbl> <chr>           <int>
1 Random_forest  Random_forest          10 ACH_000819          0
2 Random_forest  Random_forest          10 ACH_000627          0
3 Random_forest  Random_forest          10 ACH_000732          0
4 Random_forest  Random_forest          10 ACH_000984          1
5 Random_forest  Random_forest          10 ACH_000324          0

> # view result: data frame of predicted probability and labels
> head(prob_result[[4]][, 1:5], 5)

# A tibble: 5 × 5
# Groups:   ID, feature_num [5]
  ranking_method ML_method     feature_num ID         true_label
  <chr>          <chr>               <dbl> <chr>           <int>
1 Random_forest  Random_forest           2 ACH_000819          0
2 Random_forest  Random_forest           2 ACH_000627          0
3 Random_forest  Random_forest           2 ACH_000732          0
4 Random_forest  Random_forest           2 ACH_000984          1
5 Random_forest  Random_forest           2 ACH_000324          0
```

```
> # view result: the distribution of predicted probabilities
> prob_result[[2]]
```

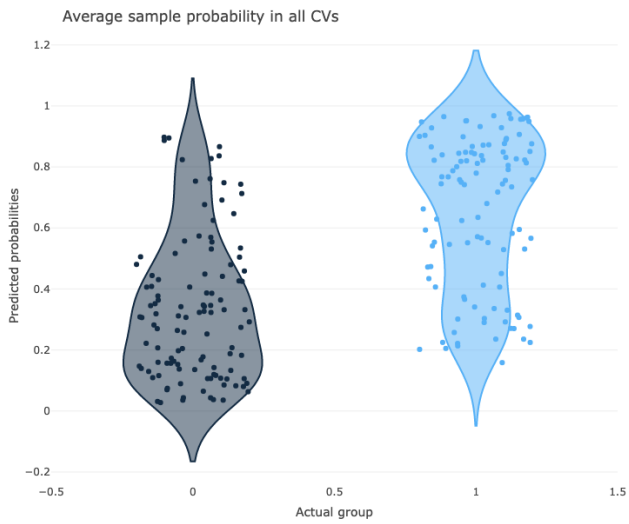

Figure 57: Probability plot
In the plot of average sample probability in all CVs' distribution, each point represents a sample, which is the mean of the prediction from all models of all Cross Validations. The y-axis is the predicted probabilities of the samples, which means the probability that the prediction value of each machine learning model is one. To depict in detail, the blue group of samples shows the probabilities that the true value is one and the prediction value of the sample is also one. The black group illustrates the probabilities that the true value is zero and the prediction value of the sample is also one. Hence, the Black group should be as close to zero, whereas the blue group should be as close to one as possible.

```
> # view result: confusion matrix of sample number and proportion
> prob_result[[3]]
```

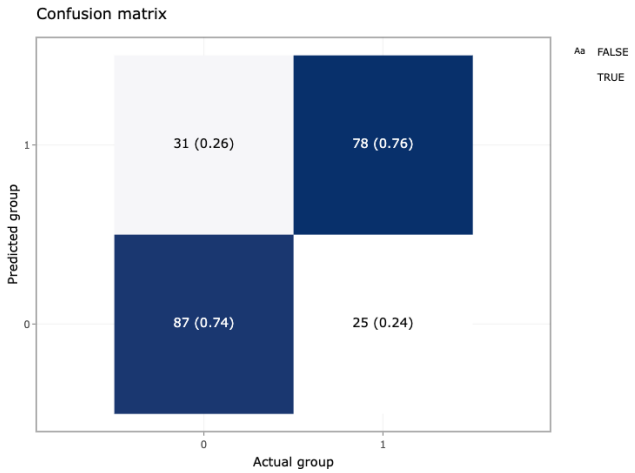

Figure 58: Confusion matrix
In the confusion matrix, the y-axis indicates the predicted class, the x-axis is the actual class. Therefore, the upper left is a true positive; the upper right is a false positive; the lower left is a false negative; the lower right is a true negative. The numbers are the count and the number in the bracket is the percentage.

3.5 Feature importance

After building a high-accuracy model, now we are going to explore the contribution of each feature. Two methods, **'Algorithm-based'** and **'SHAP analysis'** are provided to rank and visualize the feature importance.

3.5.1 Algorithm-based

In the **'Algorithm-based'** part, when we set a certain feature number by parameter `feature_n`, the selected frequency and the average feature importance of the top 10 features from all CV runs will be displayed. For a Linear SVM, Lasso, Ridge, or ElasticNet model, the importance of each feature depends on the absolute value of their coefficients in the algorithm, while Random Forest and XGBoost use built-in feature importance results.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # compute and rank the contribution of each feature
> feature_result <- feature_plot(ML_output[[6]], ML_output[[7]],
+                                feature_n=10, nfold=10)
> # view result: data frame of the selected frequency
> head(feature_result[[1]][, 1:5], 5)

  ranking_method     ML_method feature_num    feature sele_freq
1  Random_forest Random_forest          10   C22.0.SM       1.0
2  Random_forest Random_forest          10 C36.4.PC.B       1.0
3  Random_forest Random_forest          10   C38.5.PC       1.0
4  Random_forest Random_forest          10   C38.6.PC       1.0
5  Random_forest Random_forest          10  C18.0.LPE       0.9

> # view result: data frame of feature importance
> head(feature_result[[3]][, 1:5], 5)
```

```
# A tibble: 5 × 5
# Groups:   feature [5]
  ranking_method ML_method     feature_num feature    importance
  <chr>          <chr>               <dbl> <chr>           <dbl>
1 Random_forest  Random_forest          10 C38.6.PC         5.40
2 Random_forest  Random_forest          10 C38.5.PC         4.67
3 Random_forest  Random_forest          10 C36.4.PC.B       4.58
4 Random_forest  Random_forest          10 C18.0.LPE        4.13
5 Random_forest  Random_forest          10 C22.0.SM         3.44
```

```
> # view result: selected frequency plot
> feature_result[[2]]
```

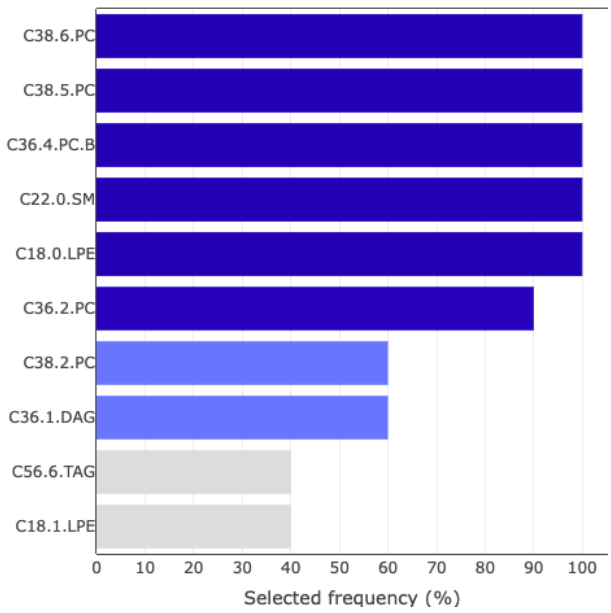

Figure 59: Feature importance (Algorithm-based)-selected frequency

```
> # view result: feature importance plot
> feature_result[[4]]
```

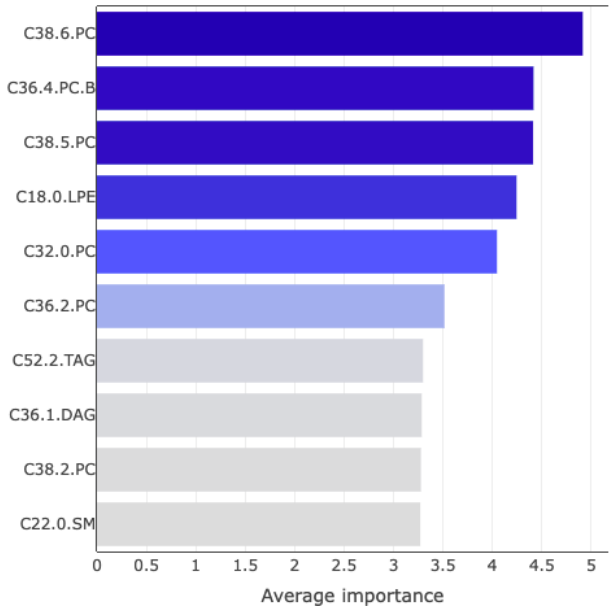

Figure 60: Feature importance (Algorithm-based)-feature importance

3.5.2 SHAP analysis

Shapley Additive exPlanations (SHAP) approach on the basis of Shapley values in game theory has recently been introduced to explain individual predictions of any machine learning model. More detailed information can be found in the paper 'A Unified Approach to Interpreting Model Predictions' (2017) [7].

The analysis is based on the result of ROC-AUC and PR-AUC. We can decide the feature number by parameter `feature_n`. According to the defined feature number, the corresponding best model in all CVs will be used to compute approximate Shapley values of each feature for all samples with *fastshap* package in R.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
```

```
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"
> # conduct SHAP
> SHAP_output <- SHAP(ML_data[[2]], best_model=ML_output[[8]],
+                     best_model_feature=ML_output[[9]],
+                     ML_method='Random_forest', feature_n=10, nsim=5)
```

```
> # view result: SHAP feature importance plot
> SHAP_output[[3]]
```

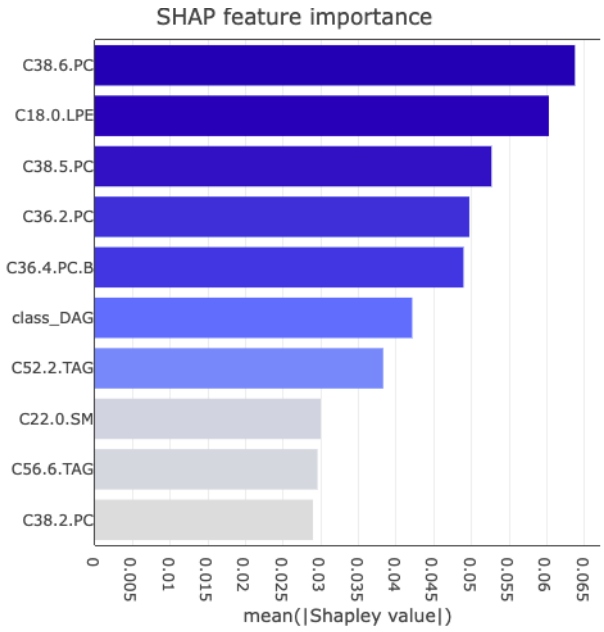

Figure 61: SHAP feature importance plot
The top 10 features are ranked and demonstrated according to the average absolute value of shapely values from all samples.

```
> # view result: SHAP summary plot
> SHAP_output[[4]]
```

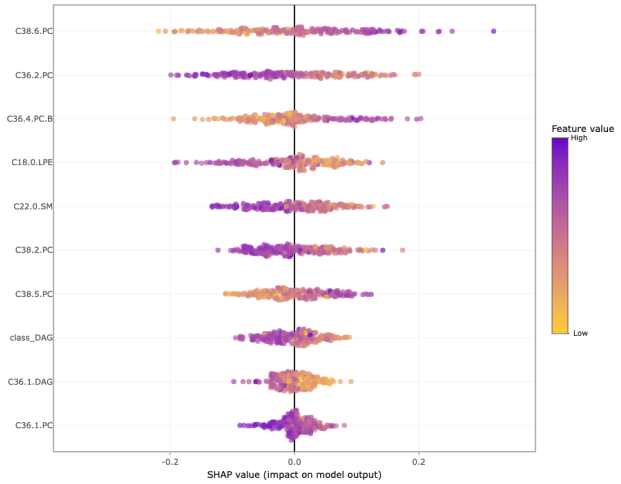

Figure 62: SHAP summary plot
The SHAP summary plot illustrates the distribution of all shapely values for each feature. It uses sina plot to present important features by binary patterns. The color exemplifying the value of the feature from low (yellow) to high (purple) indicates the variable is high/low for that observation. The x-axis presents whether the impact is positive or negative on quality rating (target variable). In the summary plot, the relationship between the value of a feature and the influence on the prediction is shown.

Next, we are going to visualize the SHAP feature importance of N samples.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, pct_transform=TRUE,
+                            missing_pct_limit=50, replace_zero=TRUE,
+                            zero2what='min', NA2what='min',
+                            xmin=0.5, replace_NA=TRUE, ymin=0.5,
+                            data_transform=TRUE, centering=FALSE,
+                            trans_type='log', scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # conduct SHAP
> SHAP_output <- SHAP(ML_data[[2]], best_model=ML_output[[8]],
```

```
+                     best_model_feature=ML_output[[9]], nsim=5,
+                     ML_method='Random_forest', feature_n=10)
> # visualize SHAP feature importance of 10 samples
> SHAP_sample_result <- SHAP_sample(SHAP_output[[2]], n_sample=10)
```

```
> # view result: SHAP feature importance plot of 10 samples
> SHAP_sample_result
```

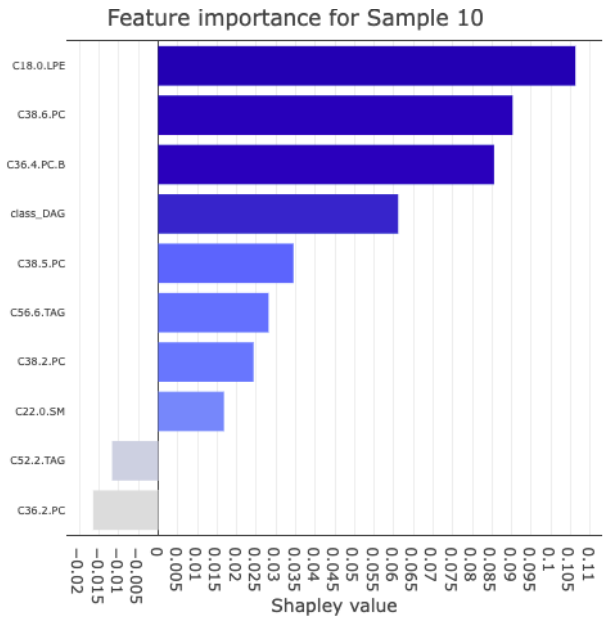

Figure 63: SHAP feature importance of 10 samples

Lastly, we build the SHAP force plot and dependence plot with different parameter sets.

The SHAP force plot stacks these Shapley values and shows how the selected features affect the final output for each sample. We can decide the number of top features to be shown by parameter `topN_feature`, and the number of groups the samples to be clustered into by parameter `group_num`.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
```

```
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # conduct SHAP
> SHAP_output <- SHAP(ML_data[[2]], best_model=ML_output[[8]],
+                     best_model_feature=ML_output[[9]],
+                     ML_method='Random_forest', feature_n=10, nsim=5)
> # visualize each predictor's attributions
> SHAP_force_result <- SHAP_forceplot(SHAP_output[[1]], topN_feature=10,
+                                     cluster_method="ward.D", group_num=10)
> # view result: data frame of force plot information
> head(SHAP_force_result[[1]][, 1:5], 5)

  Sample ID Group Sorted ID    C38.6.PC  C36.4.PC.B
1        11     5         1 0.121646667 -0.04350000
2        32     5         2 0.114500000 -0.05056667
3       127     5         3 0.024766667 -0.09206667
4       120     5         4 0.003893333 -0.09528667
5        31     5         5 0.068886667 -0.03732000
```

```
> # view result: SHAP force plot
> SHAP_force_result[[2]]
```

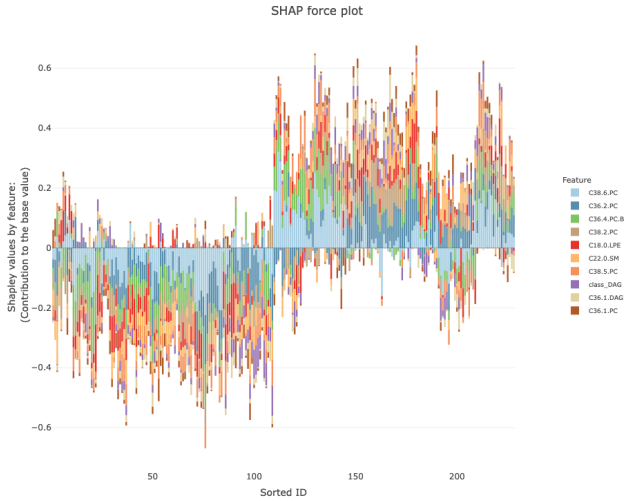

Figure 64: SHAP force plot
The colors of the bars are filled according to the features.

As to the SHAP dependence plot, it allows us to explore how the model output varies by a feature value. It reveals whether the link between the target and the variable is linear, monotonic, or more complex.

We can define the x-axis, y-axis, and color of the plot. Generally, the x-axis represents the value of a certain feature while the y-axis is the corresponding Shapley value. The color parameter can be assigned to check if a second feature has an interaction effect with the feature we are plotting.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)

[1] "CV fold 1 done"
[1] "CV fold 2 done"
[1] "CV fold 3 done"
[1] "CV fold 4 done"
[1] "CV fold 5 done"
[1] "CV fold 6 done"
[1] "CV fold 7 done"
[1] "CV fold 8 done"
[1] "CV fold 9 done"
[1] "CV fold 10 done"

> # conduct SHAP
> SHAP_output <- SHAP(ML_data[[2]], best_model=ML_output[[8]],
+                     best_model_feature=ML_output[[9]], nsim=5,
+                     ML_method='Random_forest', feature_n=10)
> # visualize SHAP values against feature values for each variable
> SHAP_depend_result <- SHAP_dependence_plot(SHAP_output[[2]], x="C38.6.PC",
+                                            y="C38.6.PC", color_var="C38.6.PC")
>
```

```
> # view result: SHAP dependence plot
> SHAP_depend_result
```

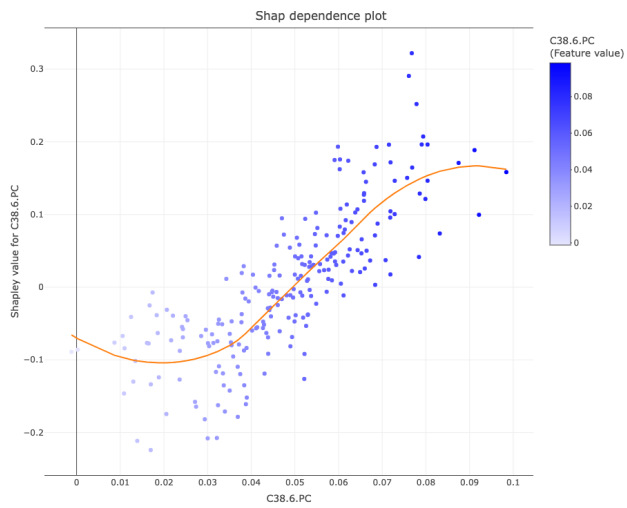

Figure 65: SHAP dependence plot

3.6 Network

A correlation network helps us interrogate the interaction of features in a machine learning model. We can decide on an appropriate feature number according to previous cross-validation results. Then, the features in the best model (based on ROC-AUC + PR-AUC) are used to compute the correlation coefficients between each other.

To build a network, nodes (features) are filled based on feature importance whereas line width represents the value of the correlation coefficient. Two methods, 'Algorithm-based' and 'SHAP analysis', can be selected to evaluate feature importance. The detailed information about them can be found in the Feature importance section (Section 3.5). A plus or minus are assigned to feature importance in SHAP analysis based on the direction of feature values and Shapley values of samples.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # data processing of machine learning
> ML_data <- ML_data_process(exp_data, group_info = condition_table,
+                            lipid_char_table, char_var[1],
+                            exclude_var_missing=TRUE, missing_pct_limit=50,
+                            replace_zero=TRUE, zero2what='min', xmin=0.5,
+                            replace_NA=TRUE, NA2what='min', ymin=0.5,
+                            pct_transform=TRUE, data_transform=TRUE,
+                            trans_type='log', centering=FALSE, scaling=FALSE)
> # conduct machine learning
> ML_output <- ML_final(ML_data[[2]], ranking_method='Random_forest',
+                       ML_method='Random_forest', split_prop=0.3, nfold=10)
> # select feature importance from model of ML_output
> model_net <- model_for_net(ML_data[[2]], ML_method='Random_forest',
+                            varimp_method='Algorithm-based', ML_output[[8]],
+                            ML_output[[9]], feature_num=10, nsim=5)
> # compute correlation coefficients and visualize correlation network
> cor_net_result <- cor_network(ML_data[[1]], lipid_char_table,
+                               model_net[[2]], model_net[[3]],
```

```
+                                cor_method='pearson', edge_cutoff=0)
```

```
> # view result: the network of feature importance
> cor_net_result
```

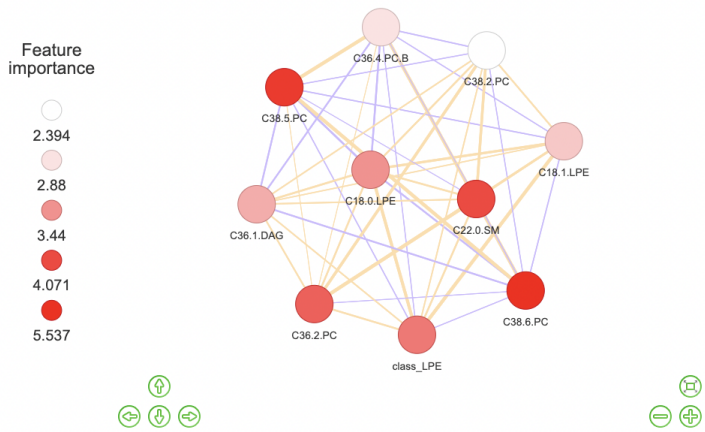

Figure 66: The network of feature importance

4 Correlation analysis

In the final part, we are going conduct a comprehensive correlation analysis to interrogate the clinical features that connect to lipids species and other mechanistically relevant lipid characteristics. Correlation analysis between lipids and clinical features is broadly used in many fields of study, such as Bowler RP et al. discovering that sphingomyelins are strongly associated with emphysema and glycosphingolipids are associated with COPD exacerbations [8].

The correlation analysis can be conducted by **"lipid species"** or **"lipid characteristics"**. In lipid species analysis, data are analyzed by lipid species. As to lipid characteristics analysis, the expression of all lipid species in the same categories of a selected characteristic is summed up for analysis.

This section is designed for continuous clinical data. Two correlation analyses are accessible, **'Correlation Coefficient'** and **'Linear Regression'**. A heatmap will be shown once the correlation analysis is completed, it depicts the pattern between lipid species/lipid characteristics and clinical features.

The available clustering methods are as follows.

- Distance measurement: Pearson, Spearman, Kendall, Euclidean, Maximum, Manhattan, Canberra, Binary, and Minkowski.
- Clustering method: median, average, single, complete, Ward.D, Ward.D2, WPGMA, and UPGMC

4.1 Input data

First, we have to read the input data needed for the correlation analysis section. We have to prepare lipid expression data (`exp_data`), lipid characteristics table (`lipid_char_table`), a condition table of sample names and clinical conditions (`condition_table`), and an adjusted table with additional variables for adjusting confounding effects (`adjusted_table`) as input data.

```
> # clears all objects from workspace
> rm(list = ls())
> # lipid expression data
> data("corr_exp_data")
> exp_data <- corr_exp_data
> head(exp_data[, 1:5], 5)

         feature sample1 sample2 sample3 sample4
1 SM d18:1 N14:1   1.440   1.359   1.035   0.398
2 SM d18:1 N14:0   1.088   0.934   1.017   0.519
3 SM d18:1 N16:1   1.142   0.898   1.032   0.695
4 SM d18:1 N16:0   0.883   0.875   0.952   0.778
5 SM d18:1 N18:1   1.184   1.057   0.838   0.923

> # lipid characteristics table
> data("corr_lipid_char_table")
> lipid_char_table <- corr_lipid_char_table
> head(lipid_char_table, 5)
```

```
         feature          class
1 SM d18:1 N14:1 Sphingomyelins
2 SM d18:1 N14:0 Sphingomyelins
3 SM d18:1 N16:1 Sphingomyelins
4 SM d18:1 N16:0 Sphingomyelins
5 SM d18:1 N18:1 Sphingomyelins

> # condition table (clinical factor)
> data("corr_condition_table")
> condition_table <- corr_condition_table
> head(condition_table,  5)

  sample_name FEV1_FVC Emphysema Exacerbations
1     sample1     0.29   11.0172             6
2     sample2     0.57    2.3615             0
3     sample3     0.79    0.9829             0
4     sample4     0.39   34.9931             0
5     sample5     0.37   26.8499             0

> # adjusted table
> data("corr_adjusted_table")
> adjusted_table <- corr_adjusted_table
> head(adjusted_table, 5)

  sample_name  Age Sex Smoking   BMI FEV1
1     sample1 48.7   0       0 36.49 14.9
2     sample2 70.1   0       0 27.92 66.7
3     sample3 49.6   0       0 23.85 99.9
4     sample4 54.1   1       0 28.44 31.2
5     sample5 70.5   0       0 23.23 31.6
```

After importing the input data, sometimes, we may need to conduct data processing before analysis. Here, we provide the `data_process` function for data processing, including removing features with missing values, missing values imputation, percentage transformation, log10 transformation, etc.

```
> # lipid expression data
> head(exp_data[, 1:5], 5)

         feature sample1 sample2 sample3 sample4
1 SM d18:1 N14:1   1.440   1.359   1.035   0.398
2 SM d18:1 N14:0   1.088   0.934   1.017   0.519
3 SM d18:1 N16:1   1.142   0.898   1.032   0.695
4 SM d18:1 N16:0   0.883   0.875   0.952   0.778
5 SM d18:1 N18:1   1.184   1.057   0.838   0.923

> # data processing of exp_data
> exp_transform_table <- data_process(exp_data, exclude_var_missing=TRUE,
+                                     missing_pct_limit=50, replace_zero=TRUE,
+                                     zero2what='min', xmin=0.5,
+                                     replace_NA=TRUE, NA2what='min',
+                                     ymin=0.5, pct_transform=TRUE,
+                                     data_transform=TRUE, trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
```

```
> # exp_data after data processing
> head(exp_transform_table[, 1:5], 5)

         feature   sample1   sample2    sample3     sample4
1 SM d18:1 N14:1 0.3653555 0.3045099 0.16812157 -0.16985602
2 SM d18:1 N14:0 0.2436219 0.1416374 0.16050218 -0.05457173
3 SM d18:1 N16:1 0.2646591 0.1245668 0.16686092  0.07224572
4 SM d18:1 N16:0 0.1529537 0.1132985 0.13181817  0.12124051
5 SM d18:1 N18:1 0.2803447 0.1953655 0.07642524  0.19546261
```

4.2 Lipid species analysis

The following correlation analysis is conducted after lipids classified by lipid species.

4.2.1 Correlation

The Correlation Coefficient gives a summary view that tells us whether a relationship exists between clinical features and lipid species, how strong that relationship is, and whether the relationship is positive or negative. We can decide the cut-offs for the correlation coefficient and the p-value by parameter `sig_cor_coef` and `sig_pvalue`. The rule of thumb in medical research recommended by Mukaka for interpreting the size of a correlation coefficient is provided below [9].

Size of Correlation	Interpretation
0.90 to 1.00 (-.90 to -1.00)	Very high positive (negative) correlation
0.70 to .90 (-.70 to -.90)	High positive (negative) correlation
0.50 to .70 (-.50 to -.70)	Moderate positive (negative) correlation
0.30 to .50 (-.30 to -.50)	Low positive (negative) correlation
0.00 to .30 (.00 to -.30)	negligible correlation

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50,
+                               replace_zero=TRUE, zero2what='min',
+                               xmin=0.5, replace_NA=TRUE,
+                               NA2what='min', ymin=0.5,
+                               pct_transform=TRUE, data_transform=TRUE,
+                               trans_type='log', centering=FALSE, scaling=FALSE)
> # compute correlation coefficient and visualize by heatmap
> COspec_clinCor <- Clin_Cor_heatmap(exp_transform, condition_table,
+                                    test = 'pearson', adjust_p_method = 'BH',
+                                    sig_stat = 'p.adj', sig_pvalue=1,
+                                    sig_cor_coef=0, heatmap_col='statistic',
+                                    distfun='spearman', hclustfun='average')
> # view result: data frame of clinical features and lipid species
> head(COspec_clinCor$Cor_table_all[, 1:5], 5)

  clin_factor         feature  method     cor_coef  statistic
1    FEV1_FVC Cer d18:1 N16:0 pearson  0.060762016  0.6860207
```

```
2    FEV1_FVC Cer d18:1 N18:0 pearson -0.071961883 -0.8130772
3    FEV1_FVC Cer d18:1 N22:0 pearson  0.037532293  0.4232657
4    FEV1_FVC Cer d18:1 N23:0 pearson  0.009129345  0.1028868
5    FEV1_FVC Cer d18:1 N24:0 pearson  0.148356369  1.6905996

> # view result: data frame of significant clinical features and lipid species
> head(COspec_clinCor$Cor_table_sig[, 1:5], 5)

  clin_factor         feature  method     cor_coef  statistic
1    FEV1_FVC Cer d18:1 N16:0 pearson  0.060762016  0.6860207
2    FEV1_FVC Cer d18:1 N18:0 pearson -0.071961883 -0.8130772
3    FEV1_FVC Cer d18:1 N22:0 pearson  0.037532293  0.4232657
4    FEV1_FVC Cer d18:1 N23:0 pearson  0.009129345  0.1028868
5    FEV1_FVC Cer d18:1 N24:0 pearson  0.148356369  1.6905996

> # view result: clinical features and lipid species correlation reorder matrix
> head(COspec_clinCor$Cor_reorder_mat[, 1:2])

              SM d18:1 N25:1 SM d18:1 N18:0
Exacerbations     -1.8776181     -1.0411487
Emphysema          0.4659626      0.4403503
FEV1_FVC          -0.5799724     -0.6708284
```

```
> # view result: heatmap of clinical features and lipid species
> COspec_clinCor$Cor_table_plot
```

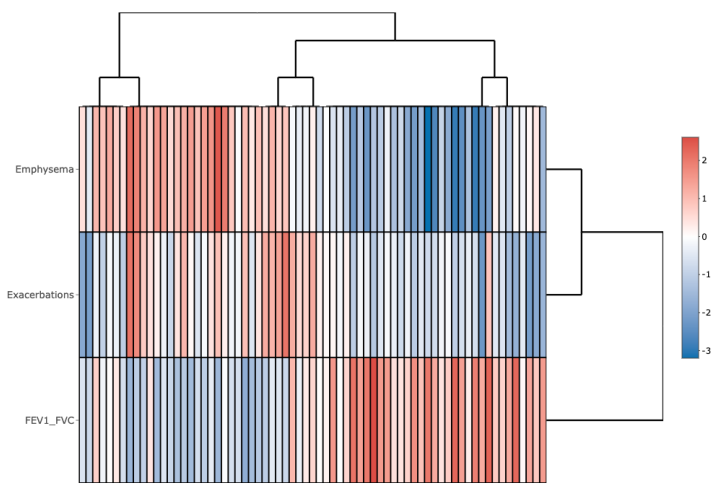

Figure 67: Correlation coefficient for lipid species analysis
Only the variables that pass the defined cut-offs for p-value and the correlation coefficient are shown on the heatmap. The rows of the heatmap are clinical features and the columns are the lipid species.

4.2.2 Linear regression

Linear regression is a statistical technique that uses several explanatory variables to predict the outcome of a continuous response variable, allowing us to estimate the associations between lipid levels and clinical features. In multiple linear regression analysis, additional variables

from the 'adjusted table' are added to the algorithm for adjusting the confounding effect. Once the calculation completes, a beta coefficient and t statistic (p-value) will be assigned to each lipid species, which can be chosen for clustering.

```
> # data processing of exp_data
> exp_transform <- data_process(exp_data, exclude_var_missing=TRUE,
+                               missing_pct_limit=50, replace_zero=TRUE,
+                               zero2what='min', xmin=0.5, replace_NA=TRUE,
+                               NA2what='min', ymin=0.5, pct_transform=TRUE,
+                               data_transform=TRUE, trans_type='log',
+                               centering=FALSE, scaling=FALSE)
> # compute linear regression and visualize by heatmap
> COspec_clin_LR <- Clin_LR_heatmap(exp_transform, condition_table,
+                                   adjusted_table, adjust_p_method = 'BH',
+                                   sig_stat = 'p.adj', sig_pvalue = 1,
+                                   distfun='spearman', hclustfun='centroid',
+                                   heatmap_col='beta_coef')
> # view result: data frame of statistical results
> head(COspec_clin_LR$LR_table_all[, 1:4], 5)

  clin_factor         feature            method     beta_coef
1    FEV1_FVC Cer d18:1 N16:0 Linear Regression  0.0097237802
2    FEV1_FVC Cer d18:1 N18:0 Linear Regression  0.0007913872
3    FEV1_FVC Cer d18:1 N22:0 Linear Regression -0.0006561572
4    FEV1_FVC Cer d18:1 N23:0 Linear Regression  0.0001037899
5    FEV1_FVC Cer d18:1 N24:0 Linear Regression  0.0071430053

> # view result: data frame of significant statistical results
> head(COspec_clin_LR$LR_table_sig[, 1:4], 5)

  clin_factor         feature            method     beta_coef
1    FEV1_FVC Cer d18:1 N16:0 Linear Regression  0.0097237802
2    FEV1_FVC Cer d18:1 N18:0 Linear Regression  0.0007913872
3    FEV1_FVC Cer d18:1 N22:0 Linear Regression -0.0006561572
4    FEV1_FVC Cer d18:1 N23:0 Linear Regression  0.0001037899
5    FEV1_FVC Cer d18:1 N24:0 Linear Regression  0.0071430053

> # view result: matrix of heatmap
> head(COspec_clin_LR$LR_reorder_mat[, 1:2])

              trihexcer d18:1 N24:0 GM3 d18:1 N22:0
Exacerbations           0.222523503     0.073865440
Emphysema               0.217745884     0.009443763
FEV1_FVC                0.001855383    -0.002877643
```

```
> # view result: heatmap of linear regression
> COspec_clin_LR$LR_table_plot
```

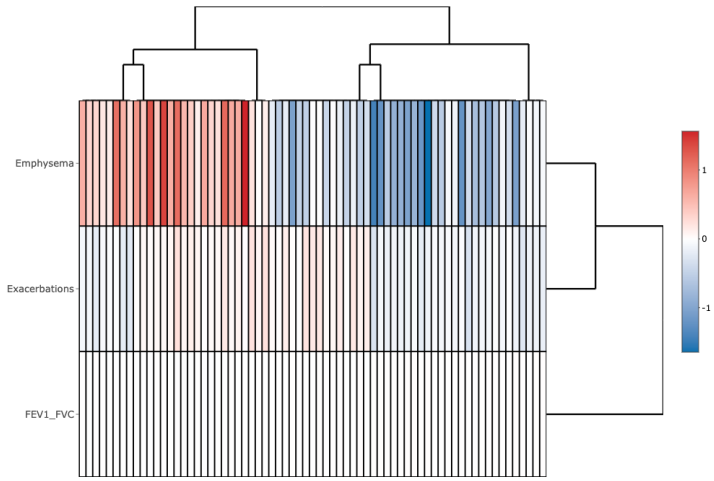

Figure 68: The heatmap of linear regression for lipid species analysis
Only the variables that pass the user-defined cut-offs for p-value and the correlation coefficient are shown on the heatmap. The rows of the heatmap are clinical features and the columns are the lipid species.

4.3 Lipid characteristics analysis

The following correlation analysis is conducted after lipids are classified by lipid characteristics from the 'Lipid characteristics' table.

4.3.1 Correlation

his section provides the correlation of lipid characteristics analysis. The Correlation Coefficient gives a summary view that tells us whether a relationship exists between clinical features and lipid species, how strong that relationship is, and whether the relationship is positive or negative. We can decide the cut-offs for the correlation coefficient and the p-value by parameter `sig_cor_coef` and `sig_pvalue`. The rule of thumb in medical research recommended by Mukaka for interpreting the size of a correlation coefficient is provided below [9].

Size of Correlation	Interpretation
0.90 to 1.00 (-.90 to -1.00)	Very high positive (negative) correlation
0.70 to .90 (-.70 to -.90)	High positive (negative) correlation
0.50 to .70 (-.50 to -.70)	Moderate positive (negative) correlation
0.30 to .50 (-.30 to -.50)	Low positive (negative) correlation
0.00 to .30 (.00 to -.30)	negligible correlation

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # aggregated(sum) expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[1])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
```

```
+                                    missing_pct_limit=50, replace_zero=TRUE,
+                                    zero2what='NA', xmin=0.5, replace_NA=TRUE,
+                                    NA2what='min', ymin=0.5,
+                                    pct_transform=TRUE,
+                                    data_transform=FALSE, trans_type='log',
+                                    centering=FALSE, scaling=FALSE)
> # compute correlation coefficient and visualize by heatmap
> COchar_clinCor <- Clin_Cor_heatmap(exp_transform_class, condition_table,
+                                   test = 'pearson', adjust_p_method = 'BH',
+                                   sig_stat = 'p.adj', sig_pvalue=1,
+                                   sig_cor_coef=0, heatmap_col='statistic',
+                                   distfun='spearman', hclustfun='average')
> # view result: data frame of clinical features and lipid characteristics
> head(COchar_clinCor$Cor_table_all[, 1:5], 5)

  clin_factor                class  method     cor_coef  statistic
1    FEV1_FVC Ceramide-1-phosphate pearson -0.009467447 -0.1066975
2    FEV1_FVC            Ceramides pearson  0.039164206  0.4416971
3    FEV1_FVC   Dihexosylceramides pearson -0.118348019 -1.3431539
4    FEV1_FVC         Gangliosides pearson  0.203704899  2.3448027
5    FEV1_FVC Monohexosylceramides pearson -0.126905025 -1.4418042

> # view result: data frame of significant clinical features and lipid characteristics
> head(COchar_clinCor$Cor_table_sig[, 1:5], 5)

  clin_factor                class  method     cor_coef  statistic
1    FEV1_FVC Ceramide-1-phosphate pearson -0.009467447 -0.1066975
2    FEV1_FVC            Ceramides pearson  0.039164206  0.4416971
3    FEV1_FVC   Dihexosylceramides pearson -0.118348019 -1.3431539
4    FEV1_FVC         Gangliosides pearson  0.203704899  2.3448027
5    FEV1_FVC Monohexosylceramides pearson -0.126905025 -1.4418042

> # view result: clinical features and lipid characteristics correlation reorder matrix
> head(COchar_clinCor$Cor_reorder_mat[, 1:2])

              Monohydroxylated ceramides Trihexosylceramides
Exacerbations                   1.807396            1.524813
Emphysema                       1.234428            1.761442
FEV1_FVC                       -1.617080           -1.217313
```

```
> # view result: heatmap of clinical features and lipid characteristics
> COchar_clinCor$Cor_table_plot
```

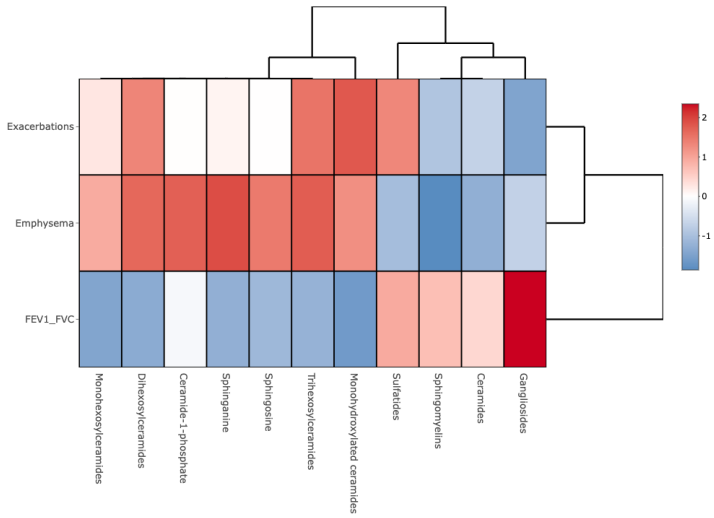

Figure 69: Correlation coefficient for lipid characteristics analysis
Only the variables that pass the defined cut-offs for p-value and the correlation coefficient are shown on the heatmap. The rows of the heatmap are clinical features and the columns are lipid characteristics.

4.3.2 Linear regression

This section provides the linear regression of lipid characteristics analysis. Linear regression is a statistical technique that uses several explanatory variables to predict the outcome of a continuous response variable, allowing us to estimate the associations between lipid levels and clinical features. Lipids are classified and summed by the user-selected lipid characteristics (e.g., class), then implementing univariate or multivariate linear regression analysis with whether 'adjusted table' is provided. Each component in the selected characteristics will be assigned a beta coefficient and t statistic (p-value), which can be chosen for clustering.

```
> # get lipid characteristics
> char_var <- colnames(lipid_char_table)[-1]
> # aggregated(sum) expression data by selected characteristics
> exp_data_Spe2Char <- Species2Char(exp_data, lipid_char_table,
+                                   char_var = char_var[1])
> # data processing of exp_data_Spe2Char
> exp_transform_class <- data_process(exp_data_Spe2Char,
+                                     exclude_var_missing=TRUE,
+                                     missing_pct_limit=50, replace_zero=TRUE,
+                                     zero2what='NA', xmin=0.5, replace_NA=TRUE,
+                                     NA2what='min', ymin=0.5,
+                                     pct_transform=TRUE,
+                                     data_transform=FALSE, trans_type='log',
+                                     centering=FALSE, scaling=FALSE)
> # compute linear regression and visualize by heatmap
> COchar_clin_LR <- Clin_LR_heatmap(exp_transform_class, condition_table,
+                                   adjusted_table, adjust_p_method = 'BH',
+                                   sig_stat = 'p.adj', sig_pvalue = 1,
+                                   distfun='spearman', hclustfun='centroid',
+                                   heatmap_col='beta_coef')
> # view result: data frame of statistical results
```

```
> head(COchar_clin_LR$LR_table_all[, 1:4], 5)

  clin_factor                class            method     beta_coef
1    FEV1_FVC Ceramide-1-phosphate Linear Regression  0.0122194546
2    FEV1_FVC            Ceramides Linear Regression  0.0040750490
3    FEV1_FVC   Dihexosylceramides Linear Regression -0.0059228358
4    FEV1_FVC         Gangliosides Linear Regression  0.0053379332
5    FEV1_FVC Monohexosylceramides Linear Regression -0.0009121046

> # view result: data frame of significant statistical results
> head(COchar_clin_LR$LR_table_sig[, 1:4], 5)

  clin_factor                class            method     beta_coef
1    FEV1_FVC Ceramide-1-phosphate Linear Regression  0.0122194546
2    FEV1_FVC            Ceramides Linear Regression  0.0040750490
3    FEV1_FVC   Dihexosylceramides Linear Regression -0.0059228358
4    FEV1_FVC         Gangliosides Linear Regression  0.0053379332
5    FEV1_FVC Monohexosylceramides Linear Regression -0.0009121046

> # view result: matrix of heatmap
> head(COchar_clin_LR$LR_reorder_mat[, 1:2])

              Ceramide-1-phosphate Gangliosides
Exacerbations           0.00715838 -0.082293578
Emphysema               0.31282832  0.548877182
FEV1_FVC                0.01221945  0.005337933
```

```
> # view result: heatmap of linear regression
> COchar_clin_LR$LR_table_plot
```

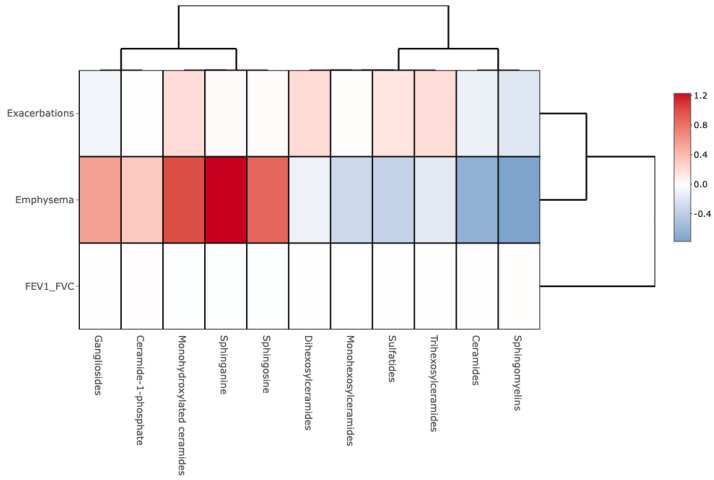

Figure 70: The heatmap of linear regression for lipid characteristics analysis
Only the variables that pass the user-defined cut-offs for p-value and the correlation coefficient are shown on the heatmap. The rows of the heatmap are clinical features and the columns are the lipid characteristics.

Session info

- R version 4.1.3 (2022-03-10), `x86_64-apple-darwin17.0`
- Locale: `C/zh_TW.UTF-8/zh_TW.UTF-8/C/zh_TW.UTF-8/zh_TW.UTF-8`
- Running under: `macOS Big Sur/Monterey 10.16`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: LipidSigR 0.99.0, SHAPforxgboost 0.1.1, ggplot2 3.3.5, pathview 1.34.0, plotly 4.10.0, visNetwork 2.1.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.56.2, BBmisc 1.12, Biobase 2.54.0, BiocGenerics 0.40.0, BiocManager 1.30.16, BiocParallel 1.28.3, BiocStyle 2.22.0, Biostrings 2.62.0, DBI 1.1.2, FNN 1.1.3, Formula 1.2-4, GenomeInfoDb 1.30.1, GenomeInfoDbData 1.2.7, Hmisc 4.6-0, IRanges 2.28.0, KEGGREST 1.34.0, KEGGgraph 1.54.0, MASS 7.3-56, Matrix 1.4-1, ModelMetrics 1.2.2.2, R6 2.5.1, RColorBrewer 1.1-2, RCurl 1.98-1.6, RSQLite 2.2.11, RSpectra 0.16-0, Rcpp 1.0.8.3, Rgraphviz 2.38.0, Rtsne 0.15, S4Vectors 0.32.4, TSP 1.2-0, XML 3.99-0.9, XVector 0.34.0, abind 1.4-5, assertthat 0.2.1, backports 1.4.1, base64enc 0.1-3, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.2, broom 0.7.12, cachem 1.0.6, car 3.0-12, carData 3.0-5, caret 6.0-91, checkmate 2.0.0, class 7.3-20, cli 3.2.0, cluster 2.1.3, codetools 0.2-18, colorspace 2.0-3, compiler 4.1.3, corpcor 1.6.10, crayon 1.5.1, crosstalk 1.2.0, data.table 1.14.2, dendextend 1.15.2, digest 0.6.29, dplyr 1.0.8, e1071 1.7-9, ellipse 0.4.2, ellipsis 0.3.2, evaluate 0.15, factoextra 1.0.7, fansi 1.0.3, farver 2.1.0, fastcluster 1.2.3, fastmap 1.1.0, fastshap 0.0.7, forcats 0.5.1, foreach 1.5.2, foreign 0.8-82, furrr 0.2.3, future 1.24.0, future.apply 1.8.1, generics 0.1.2, ggdendro 0.1.23, ggforce 0.3.3, ggpubr 0.4.0, ggrepel 0.9.1, ggsignif 0.6.3, ggthemes 4.2.4, globals 0.14.0, glue 1.6.2, gower 1.0.0, graph 1.72.0, grid 4.1.3, gridExtra 2.3, gtable 0.3.0, hardhat 0.2.0, heatmaply 1.3.0, htmlTable 2.4.0, htmltools 0.5.2, htmlwidgets 1.5.4, httr 1.4.2, hwordcloud 0.1.0, igraph 1.2.11, iheatmapr 0.5.1, ipred 0.9-12, iterators 1.0.14, jpeg 0.1-9, jsonlite 1.8.0, knitr 1.38, labeling 0.4.2, lattice 0.20-45, latticeExtra 0.6-29, lava 1.6.10, lazyeval 0.2.2, lifecycle 1.0.1, listenv 0.8.0, lubridate 1.8.0, magrittr 2.0.3, matrixStats 0.61.0, memoise 2.0.1, mixOmics 6.18.1, munsell 0.5.0, nlme 3.1-157, nnet 7.3-17, org.Hs.eg.db 3.14.0, pROC 1.18.0, parallel 4.1.3, parallelly 1.30.0, pillar 1.7.0, pkgconfig 2.0.3, plyr 1.8.7, png 0.1-7, polyclip 1.10-0, prodlim 2019.11.13, proxy 0.4-26, purrr 0.3.4, rARPACK 0.11-0, ranger 0.13.1, recipes 0.2.0, registry 0.5-1, reshape2 1.4.4, rlang 1.0.2, rmarkdown 2.13, rpart 4.1.16, rsample 0.1.1, rstatix 0.7.0, rstudioapi 0.13, scales 1.1.1, seriation 1.3.5, showimage 1.0.0, splines 4.1.3, stats4 4.1.3, stringi 1.7.6, stringr 1.4.0, survival 3.3-1, tibble 3.1.6, tidyr 1.2.0, tidyselect 1.1.2, timeDate 3043.102, tools 4.1.3, tweenr 1.0.2, utf8 1.2.2, uwot 0.1.11, vctrs 0.4.0, viridis 0.6.2, viridisLite 0.4.0, webshot 0.5.2, withr 2.5.0, xfun 0.30, xgboost 1.5.2.1, yaml 2.3.5, yardstick 0.0.9, zlibbioc 1.40.0

References

[1] Lipidsig: a web-based tool for lipidomic data analysis. URL: http://www.chenglab.cmu.edu.tw/lipidsig/.

[2] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[3] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[4] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[5] Qing-Song Xu and Yi-Zeng Liang. Monte carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, 56(1):1–11, 2001.

[6] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[7] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[8] Russell P Bowler, Sean Jacobson, Charmion Cruickshank, Grant J Hughes, Charlotte Siska, Daniel S Ory, Irina Petrache, Jean E Schaffer, Nichole Reisdorph, and Katerina Kechris. Plasma sphingolipids associated with chronic obstructive pulmonary disease phenotypes. *American journal of respiratory and critical care medicine*, 191(3):275–284, 2015.

[9] Mavuto M Mukaka. A guide to appropriate use of correlation coefficient in medical research. *Malawi medical journal*, 24(3):69–71, 2012.