Cannot patch longtable

LipidSigR tutorial - Profiling

Wei-Chung Cheng[1]

[1]China Medical University

1 April 2022

Contents

[]/home/mhtsai/R/x86$_6$4$-pc-linux-gnu-library/4.0/BiocStyle/resources/tex/Bioconductor$

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 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 **??**.

```
# 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
```

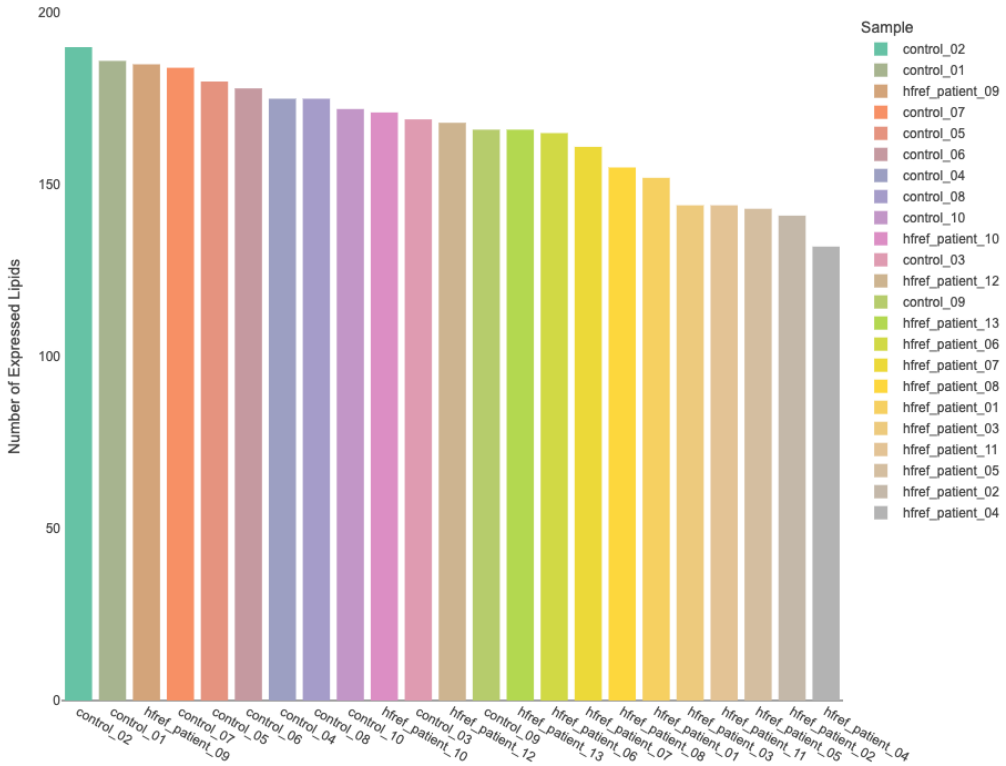
2 Cross-sample variability

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).

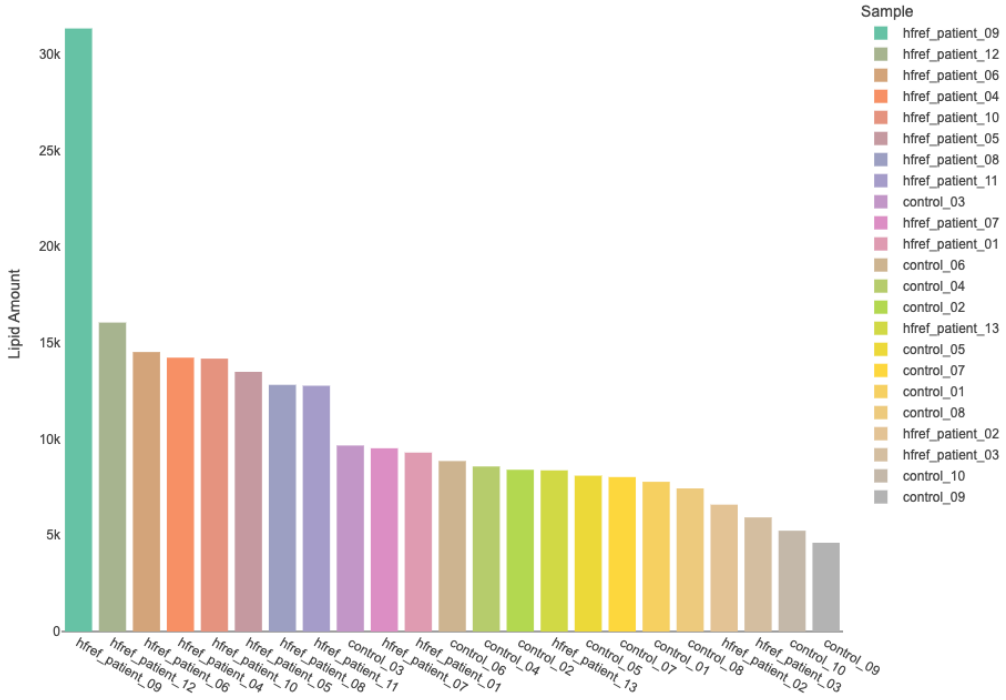
```
# conduct profiling
profiling_result <- exp_profiling(exp_data)
```

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

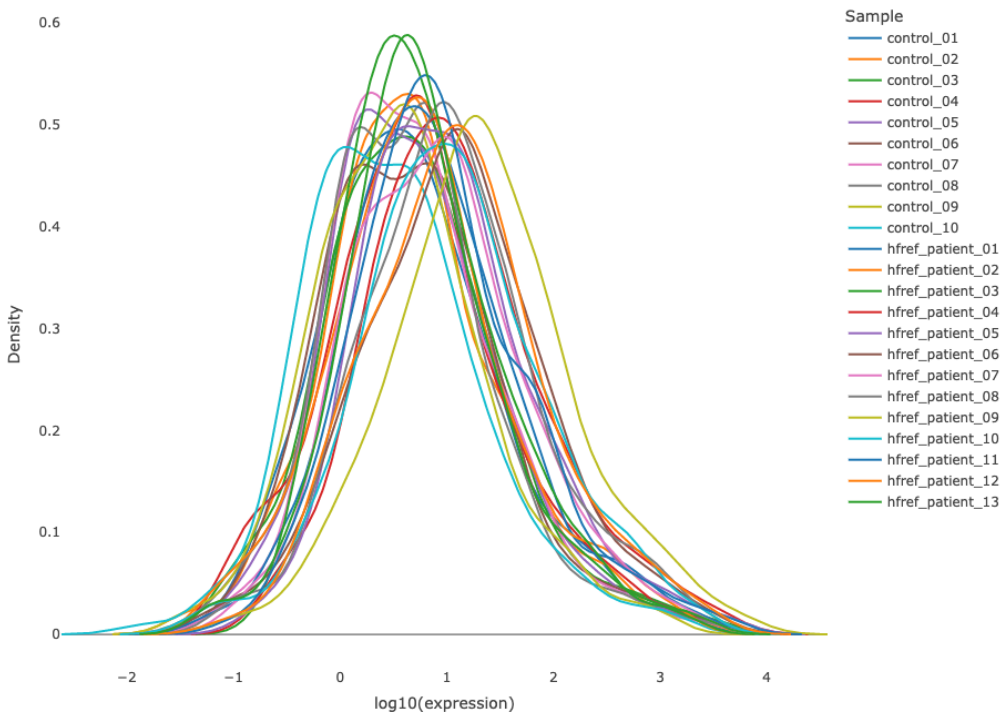
```
# view result: histograms (number of expressed lipids)
profiling_result$i.expr.lip
```




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



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



3 Dimensionality reduction

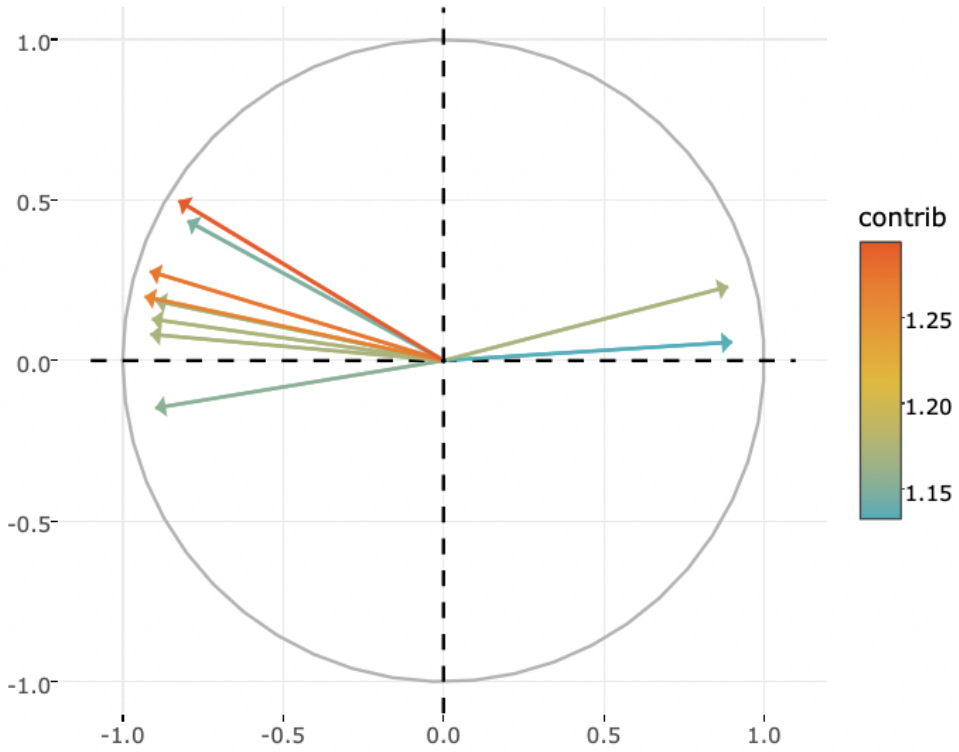
Dimensionality reduction is commonly used when dealing with large numbers of observations and/or large numbers of variables in lipids analysis. It transforms data from a high-dimensional space into a low-dimensional space so that it retains vital properties of the original data and is close to its intrinsic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

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

3.1 PCA

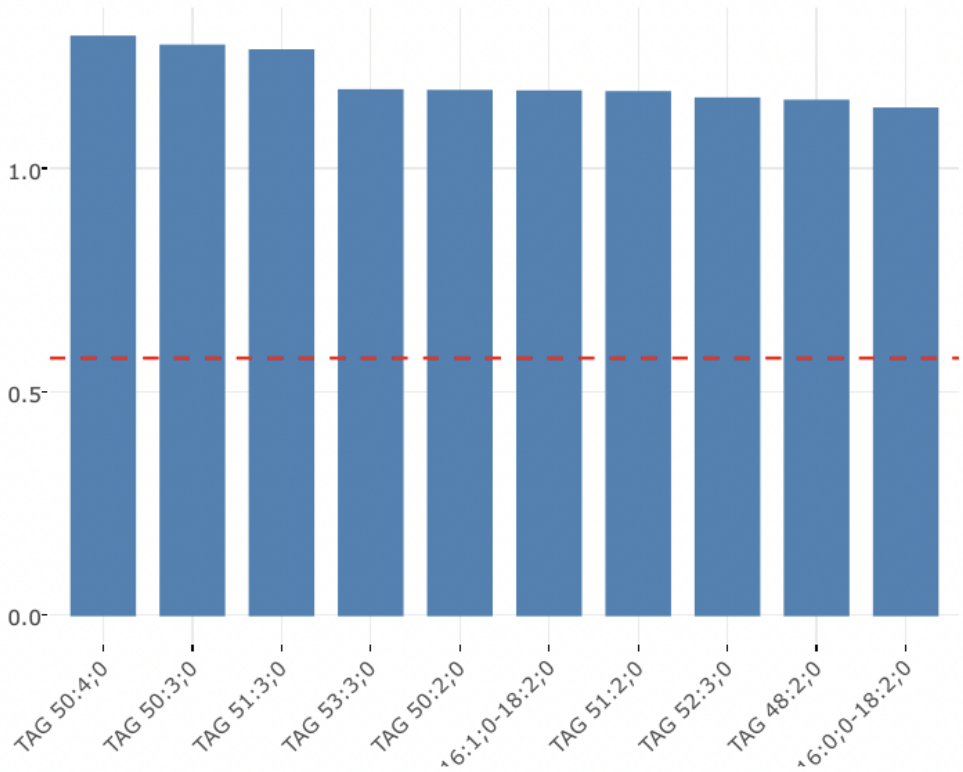
PCA (Principal component analysis) is an unsupervised linear dimensionality reduction and data visualization technique for high dimensional data, which tries to preserve the global structure of the data. Scaling (by default) indicates that the variables should be scaled to have unit variance before the 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



\begin{kframe}

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]]
```



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.[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)
# 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)
```

3.3 UMAP

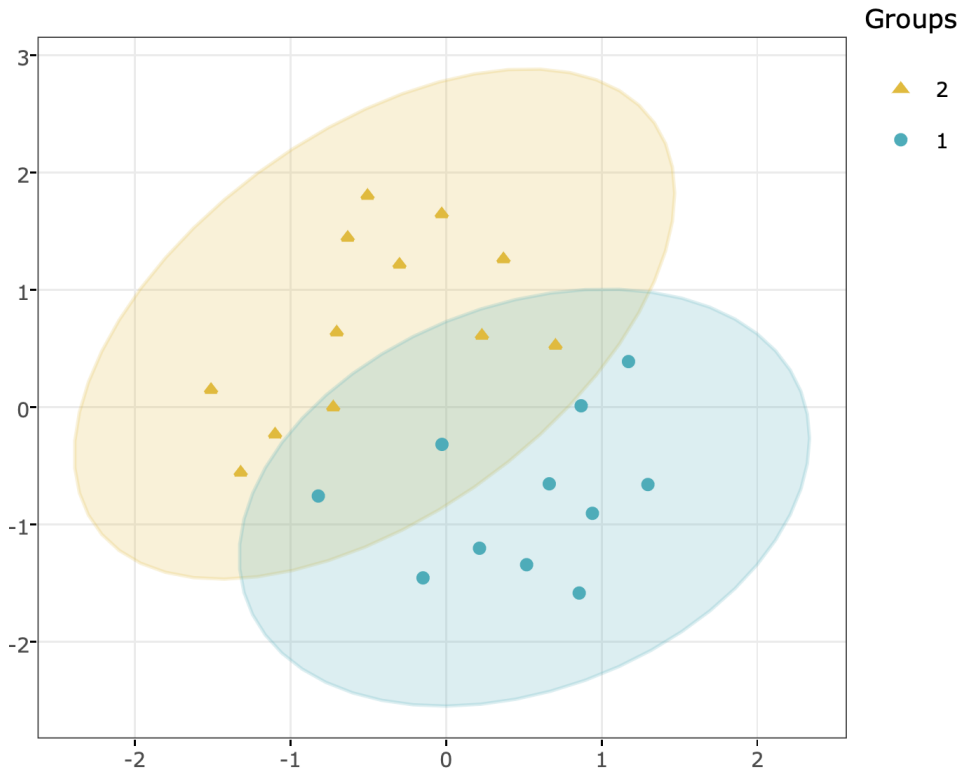
UMAP (Uniform Manifold Approximation and Projection) using a nonlinear dimensionality reduction method, Manifold learning, which effectively visualizing clusters or groups of data points and their relative proximities. Both tSNE and UMAP are intended to predominantly preserve the local structure that 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.[2]

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
## 1  control_01     2  0.2104180 -0.8313750
## 2  control_02     1  0.2321617 -0.1729150
## 3  control_03     2 -1.7448065 -0.6621704
## 4  control_04     1  0.7294066 -0.8516733
## 5  control_05     2 -0.7390144 -0.4466308
```

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



4 Correlation heatmap

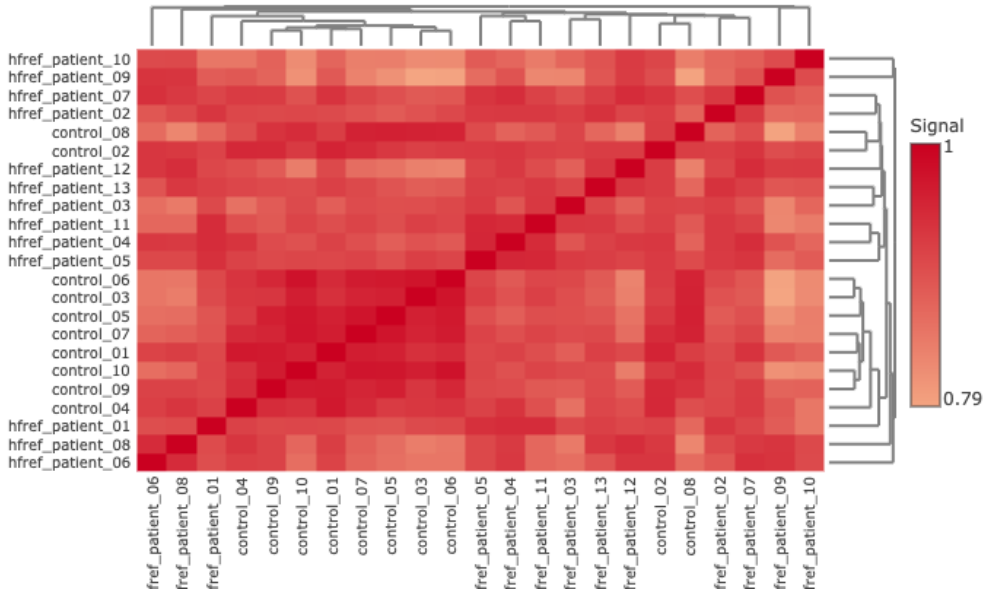
The correlation heatmap illustrates the correlation between samples or lipid species and also depicts the patterns in each group. The correlation is calculated by the method defined by parameter `corr_method`, and the correlation coefficient is then clustered depending on method defined by parameter `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
```

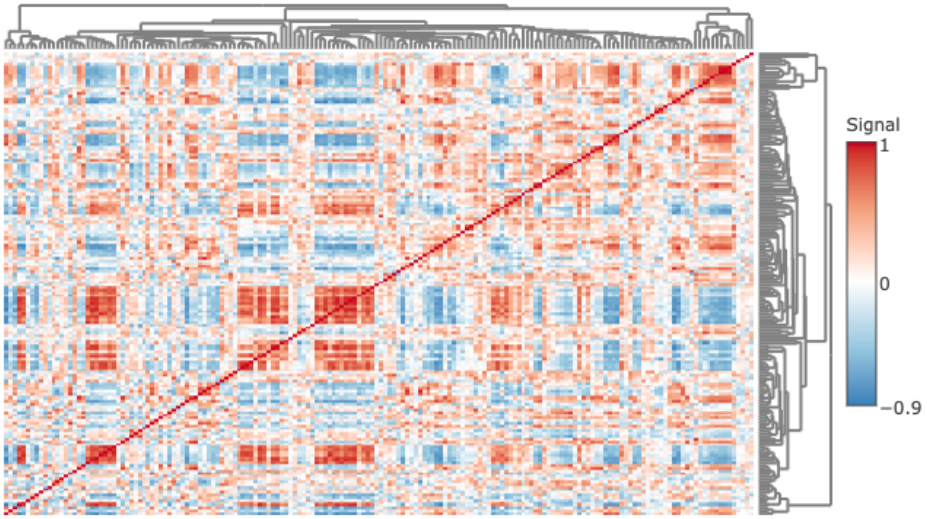
```
## control_02  0.8436678  1.0000000  0.7457617  0.8255156  0.8040823
## control_03  0.8369328  0.7457617  1.0000000  0.7739019  0.7849288
## control_04  0.8540800  0.8255156  0.7739019  1.0000000  0.7201342
## control_05  0.8734609  0.8040823  0.7849288  0.7201342  1.0000000
# view result: matrix 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 0.22211240 1.460114e-01
## Cer 40:1;2  2.169008e-01 0.000000e+00 0.01095933 8.635027e-06
## Cer 40:2;2  2.221124e-01 1.095933e-02 0.00000000 2.271903e-02
## Cer 42:1;2  1.460114e-01 8.635027e-06 0.02271903 0.000000e+00
## Cer 42:2;2  9.384919e-01 3.621739e-02 0.04556576 3.897876e-02
```

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



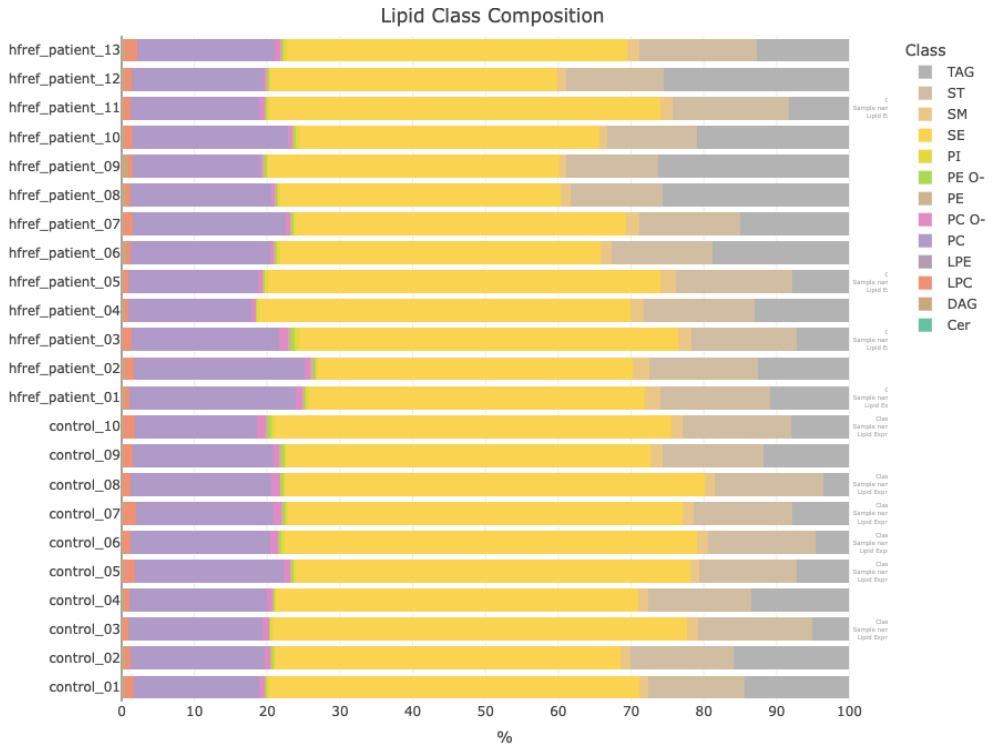
The above heatmap reveals sample to sample correlations. Correlations between lipid species are colored from strong positive correlations (red) to no correlation (white).

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



The above 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).

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



Session info

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS Linux 7 (Core)
##
## Matrix products: default
## BLAS:   /usr/local/lib64/R/lib/libRblas.so
## LAPACK: /usr/local/lib64/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] plotly_4.9.4.1   ggplot2_3.3.5    LipidSigR_0.1.0  BiocStyle_2.16.1
##
## loaded via a namespace (and not attached):
```

```
##  [1] httr_1.4.2          tidyr_1.1.4         jsonlite_1.7.2
##  [4] viridisLite_0.4.0   carData_3.0-4       assertthat_0.2.1
##  [7] BiocManager_1.30.16 iheatmapr_0.5.1     cellranger_1.1.0
## [10] yaml_2.2.1          ggrepel_0.9.1       factoextra_1.0.7
## [13] pillar_1.6.3        backports_1.2.1     lattice_0.20-45
## [16] glue_1.4.2          digest_0.6.28       RColorBrewer_1.1-2
## [19] ggsignif_0.6.3      colorspace_2.0-2    htmltools_0.5.2
## [22] Matrix_1.3-4        pkgconfig_2.0.3     broom_0.7.9
## [25] haven_2.4.3         bookdown_0.24       purrr_0.3.4
## [28] scales_1.1.1        webshot_0.5.2       processx_3.5.2
## [31] RSpectra_0.16-0     openxlsx_4.2.4      Rtsne_0.15
## [34] rio_0.5.27          tibble_3.1.4        generics_0.1.0
## [37] farver_2.1.0        car_3.0-11          ellipsis_0.3.2
## [40] ggpubr_0.4.0        withr_2.4.2         fastcluster_1.2.3
## [43] lazyeval_0.2.2      magrittr_2.0.1      crayon_1.4.1
## [46] readxl_1.3.1        evaluate_0.14       ps_1.6.0
## [49] fansi_0.5.0         rstatix_0.7.0       forcats_0.5.1
## [52] foreign_0.8-81      FNN_1.1.3           tools_4.0.0
## [55] data.table_1.14.2   hms_1.1.1           lifecycle_1.0.1
## [58] stringr_1.4.0       munsell_0.5.0       zip_2.2.0
## [61] callr_3.7.0         compiler_4.0.0      rlang_0.4.11
## [64] grid_4.0.0          htmlwidgets_1.5.4   crosstalk_1.1.1
## [67] labeling_0.4.2      rmarkdown_2.11      gtable_0.3.0
## [70] abind_1.4-5         DBI_1.1.1           curl_4.3.2
## [73] R6_2.5.1            knitr_1.34          dplyr_1.0.7
## [76] fastmap_1.1.0       uwot_0.1.10         utf8_1.2.2
## [79] stringi_1.7.4       Rcpp_1.0.7          vctrs_0.3.8
## [82] tidyselect_1.1.1    xfun_0.26
```

References

1. Van der Maaten, L. & Hinton, G. Visualizing data using t-sne. *Journal of machine learning research* **9,** (2008).

2. McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).