Analytical Pipeline for LC-MS/MS Metabolomic Studies

Packages dependencies

MultiABLER is dependent on the following packages:

- xcms
- CAMERA
- metid
- ProteoMM
- limma

We would also use tidyverse will be used to manipulate the dataframe and cowplot will be used for plotting the figures.

Guides to install these packages can be found in the package's page. However, MultiABLER is also able to install the above packages automatically.

```
library(cowplot)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v forcats 0.5.1
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::annotate() masks CAMERA::annotate()
## x xcms::collect()     masks dplyr::collect()
## x MSnbase::combine()  masks Biobase::combine(), BiocGenerics::combine(), dplyr::combine()
## x S4Vectors::expand() masks tidyr::expand()
## x metid::filter()     masks dplyr::filter(), stats::filter()
## x S4Vectors::first()  masks dplyr::first()
## x xcms::groups()      masks dplyr::groups()
## x dplyr::lag()        masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()     masks MSnbase::reduce()
## x S4Vectors::rename() masks dplyr::rename()
```

```
library(xcms)
library(MultiABLER)
```

Data

Data used in this tutorial is collected from Talib et al.[1]. The data can be find at the EMBL-EBI Biostudies repository S-BSST1039. Analysed data can also be found in the demo data folder.

Data Preprocessing

TO process the data, raw LC-MS file is first converted into mzML using MSconvert. LC-MS data preprocessing was performed with XCMS and CAMERA. To start analysis, we first read in the location, sample information, and mzML file location. Here, we use the lipidomic (Lx) positive mode as an example.

```
data_dir = file.path("AnalChem_data")
sample.info = read_csv(file.path(data_dir, "sample.info.csv"))
mzML_dir = file.path(data_dir, "mzML")

mzMLs = list.files(mzML_dir, full.names = TRUE, pattern = ".mzML")
```

XCMS requires three parameters to perform alignment: CentWaveParam, ObiwarpParam, and PeakDensityParam. User may define their own XCMS parameters, or the following parameters is build-in with the pipeline.

```
cwp <- CentWaveParam(ppm = 10,
                             snthr = 6,
                             peakwidth = c(10, 60),
                             mzdiff = 0.01,
                             noise = 0,
                             prefilter = c(3, 500))

owp <- ObiwarpParam(binSize = 0.5)

pdp <- PeakDensityParam(sampleGroups = sample.info$group,
                           bw = 5,
                           binSize = 0.025,
                           minFraction = 0.5,
                           minSamples = 1)
```

To run XCMS, we use the function `runXCMS`. User may choose to save the XCMS xdata with the parameter `save.xdata` (default is `FALSE`), with a provided save location. As mentioned above, the XCMS parameters can also be omited and the above parameters will be used for XCMS alignment.

This function will also runs CAMERA annotation, providing a full XCMS and CAMERA annotated table as output. The following parameters is our default settings, but user may change the parameters to adjust for their data. `camera.sigma`, `camera.perfwhm` and `camera.intval` is used in the function `CAMERA::groupFWHM`. `camera.intval`,`camera.ppm`,`camera.mzabs`,`camera.maxcharge`, and `camera.maxiso` are used in the function `CAMERA::findIsotopes`.

```
xcms.table <- runXCMS(mzMLs,
                         sample.info = sample.info,
                         polarity = "positive",
                         pdp = pdp,
                         cwp = cwp,
                         owp = owp,
                         save.location = file.path(mzML_dir, "xdata"),
                         save.xdata = TRUE,
                         camera.sigma = 6,
                         camera.perfwhm = 0.6,
                         camera.intval = "into",
                         camera.ppm = 10,
```

```
                        camera.mzabs = 0.015,
                        camera.maxcharge = 3,
                        camera.maxiso = 5
                        )
```

After xcms alignment, we can use the function `write_lipidfinder_csv` to save a csv for LipidFinder.

```
write_lipidfinder_csv(xcms.table, file.path(data_dir, "Lx_pos.csv"))
```

Feature Annotation with metID

Feature annotation is performed using metID with the function `metid_annotate`. Feature table is annotated using the function `metid::identify_metabolites`. Two database is included in our package for metabolites and lipids. Both are molecules identified in the Human Metabolome Database (HMDB). User may use the parameters `ms1.match.ppm` and `rt.match.tol` to modifiy the annotation parameters.

```
annotated <- xcms.table %>% dplyr::rename(mz = mzmed, rt = rtmed) %>%
  metid_annotate(polarity = "positive", omic = "lx")
```

The above process is repeated for the lipidomic negative mode, as well as the metabolomic data collected from positive and negative mode.

LipidFinder filtering

After data preprocessing with XCMS, LipidFinder was used to normalise the data based on the blank solvent control. Lipidfinder is available both online and as python code. Instruction on installing the python code can be found in here. The `peakfilter` filter module was used to filter the XCMS peaks. Parameters for the module is setup using the `config_params.py` as described in the instruction of LipidFinder. The parameter json file can also be found in `demo_data`.

```
> run_peakfilter.py -i AnalChem_data/Lx_pos.csv -o AnalChem_data/LipidFinder_lx_results
-p demo_data/param.filter.demo.pos.json

> run_peakfilter.py -i AnalChem_data/Lx_neg.csv -o AnalChem_data/LipidFinder_lx_results
-p demo_data/param.filter.demo.neg.json

> run_peakfilter.py -i AnalChem_data/Mx_pos.csv -o AnalChem_data/LipidFinder_mx_results
-p demo_data/param.filter.demo.pos.json

> run_peakfilter.py -i AnalChem_data/Mx_neg.csv -o AnalChem_data/LipidFinder_mx_results
-p demo_data/param.filter.demo.neg.json
```

After the clean up using LipidFinder, the data is imported to MultiABLER for further processing. We use the `read_LipidFinder_csv` function to read the LipidFinder filtered result into MultiABLER. The omic argument is used to tag whether the feature table is lipidomic (Lx) or metabolomic (Mx). The file should be the output from LipidFinder, including a column for "Polarity". The `read_LipidFinder_csv` would read and format the feature table. We can then use the `bind_rows` function from `dplyr` (part of `tidyverse`) to join the four feature tables.

```
lx_neg <- read_LipidFinder_csv("AnalChem_data/LipidFinder_lx_results/peakfilter_negative_summary.csv",
lx_pos <- read_LipidFinder_csv("AnalChem_data/LipidFinder_lx_results/peakfilter_positive_summary.csv",
mx_neg <- read_LipidFinder_csv("AnalChem_data/LipidFinder_mx_results/peakfilter_negative_summary.csv",
mx_pos <- read_LipidFinder_csv("AnalChem_data/LipidFinder_mx_results/peakfilter_positive_summary.csv",
```

```
lipidFinder.data <- bind_rows(lx_neg, lx_pos, mx_neg, mx_pos)
head(lipidFinder.data)
```

```
## # A tibble: 6 x 25
##   Omic  Polarity Feature~1 mzmed rtmed     L1     L2     L3     L4     L5     L6
##   <chr> <chr>    <chr>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Lx    Negative LN00001    60.0  0.46  18065  17504  17552 1.75e4  18732  17310
## 2 Lx    Negative LN00003    61.0  0.46      0   3910   3909 3.89e3   4058   3983
## 3 Lx    Negative LN00007    62.0  7.25 367353 291493      0 0           0 142533
## 4 Lx    Negative LN00006    62.0 18.2  264383 268703 272369 0           0 280192
## 5 Lx    Negative LN00004    62.0 20.8  322881 312454 339718 1.08e6 328364 325937
## 6 Lx    Negative LN00009    69.0  0.45  18582      0  25420 1.91e4  19135      0
## # ... with 14 more variables: L7 <dbl>, L8 <dbl>, L9 <dbl>, NL1 <dbl>,
## #   NL10 <dbl>, NL11 <dbl>, NL2 <dbl>, NL3 <dbl>, NL4 <dbl>, NL5 <dbl>,
## #   NL6 <dbl>, NL7 <dbl>, NL8 <dbl>, NL9 <dbl>, and abbreviated variable name
## #   1: Feature.ID
```

Data processing with MultiABLER

After the data is imported, we can first extract the feature table from the additional information. We will then use the `tbManipulate` function of `MultiABLER` to filter, impute and transform the feature table. We will also read in the sample information provided in `demo_data`.

```
lipidFinder.data <- lipidFinder.data %>% mutate(across(where(is.numeric), ~na_if(.,0)))
feature.info <- lipidFinder.data %>% select(Omic, Polarity, Feature.ID, mzmed, rtmed)
feature.df <- lipidFinder.data %>% select(-mzmed, -rtmed)
sample.only.info <- sample.info %>% filter(group == "sample") %>% select(-group)

head(feature.df)
```

```
## # A tibble: 6 x 23
##   Omic  Polar~1 Featu~2     L1     L2     L3      L4     L5     L6     L7     L8
##   <chr> <chr>   <chr>    <dbl>  <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Lx    Negati~ LN00001  18065  17504  17552   17508  18732  17310  16730  15953
## 2 Lx    Negati~ LN00003     NA   3910   3909    3892   4058   3983   3563   3497
## 3 Lx    Negati~ LN00007 367353 291493     NA      NA     NA 142533  31817     NA
## 4 Lx    Negati~ LN00006 264383 268703 272369      NA     NA 280192 264281 253273
## 5 Lx    Negati~ LN00004 322881 312454 339718 1083642 328364 325937 975363 959286
## 6 Lx    Negati~ LN00009  18582     NA  25420   19093  19135     NA  22799  23078
## # ... with 12 more variables: L9 <dbl>, NL1 <dbl>, NL10 <dbl>, NL11 <dbl>,
## #   NL2 <dbl>, NL3 <dbl>, NL4 <dbl>, NL5 <dbl>, NL6 <dbl>, NL7 <dbl>,
## #   NL8 <dbl>, NL9 <dbl>, and abbreviated variable names 1: Polarity,
## #   2: Feature.ID
```

Data processing in the `tbManipulate` function includes filtering, missing value imputation, and logarithmic transformation.

Filtering The first step in the data preprocessing is feature filtering. There are multiple method for data filtering that are commonly used, such as filtering the features by missing values and by the mean abundance of the features. Here, we use a missing value threshold of 3 to filter the data. We would remove any features

that has less than 3 values. While the threshold is an arbitually selected number, we use this threshold to maximise the number of features found in the dataset without affecting the normalisation algorithm. The filter threshold can be set in the argument `filter_threshold`.

Missing Value Imputation For missing value imputation, we use the global minimum method. All missing values are imputed with half of the global minimum. This is to provide a small value to the missing values so that the normalisation algorithm may run without drastically affecting the data shape. Imputation implemented includes using half of the global minimum ("gm2"), and a fifth of the global minimum("gm5"). The argument can be set in the field `impute`.

Logarithmic transformation Finally, we transformed the data with log base 2. Base 2 is used here so that in subsequent differential analysis, a fold change of 1 represents a 2-fold change. The log base can be changed using the argument `log_base`.

We use the `group_by` function and the `group_modify` function in `dplyr` to pre-process all four table at once.

```
mani.df <- feature.df %>% group_by(Omic, Polarity) %>% group_modify(~tbManipulate(.x, 3))
head(mani.df)
```

```
## # A tibble: 6 x 23
## # Groups:   Omic, Polarity [1]
##   Omic  Polarity Feature~1    L1    L2    L3    L4    L5    L6    L7    L8    L9
##   <chr> <chr>    <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Lx    Negative LN00001   14.1  14.1  14.1  14.1  14.2  14.1   14.0 14.0  14.1
## 2 Lx    Negative LN00003    9.97 11.9  11.9  11.9  12.0  12.0   11.8 11.8  11.9
## 3 Lx    Negative LN00007   18.5  18.2   9.97  9.97  9.97 17.1   15.0  9.97  9.97
## 4 Lx    Negative LN00006   18.0  18.0  18.1   9.97  9.97 18.1   18.0 18.0   9.97
## 5 Lx    Negative LN00004   18.3  18.3  18.4  20.0  18.3  18.3   19.9 19.9  18.2
## 6 Lx    Negative LN00009   14.2   9.97 14.6  14.2  14.2   9.97  14.5 14.5  14.3
## # ... with 11 more variables: NL1 <dbl>, NL10 <dbl>, NL11 <dbl>, NL2 <dbl>,
## #   NL3 <dbl>, NL4 <dbl>, NL5 <dbl>, NL6 <dbl>, NL7 <dbl>, NL8 <dbl>,
## #   NL9 <dbl>, and abbreviated variable name 1: Feature.ID
```

Data Normalisation

For data normalisation, we used EigenMS, which is implemented by the R package `ProteoMM`. EigenMS is a normalisation algorithm that uses single value decomposition (SVD) to remove systematic bias from LC-MS experimetal runs based on the experimental group of the samples. It is useful for experimental design that includes multiple sample groups, as it normalises the sample across the dataset, while retaining the differences between treatment groups.

To run EigenMS, we would use the `eig_norm1` and `eig_norm2` function in the `ProteoMM` package.

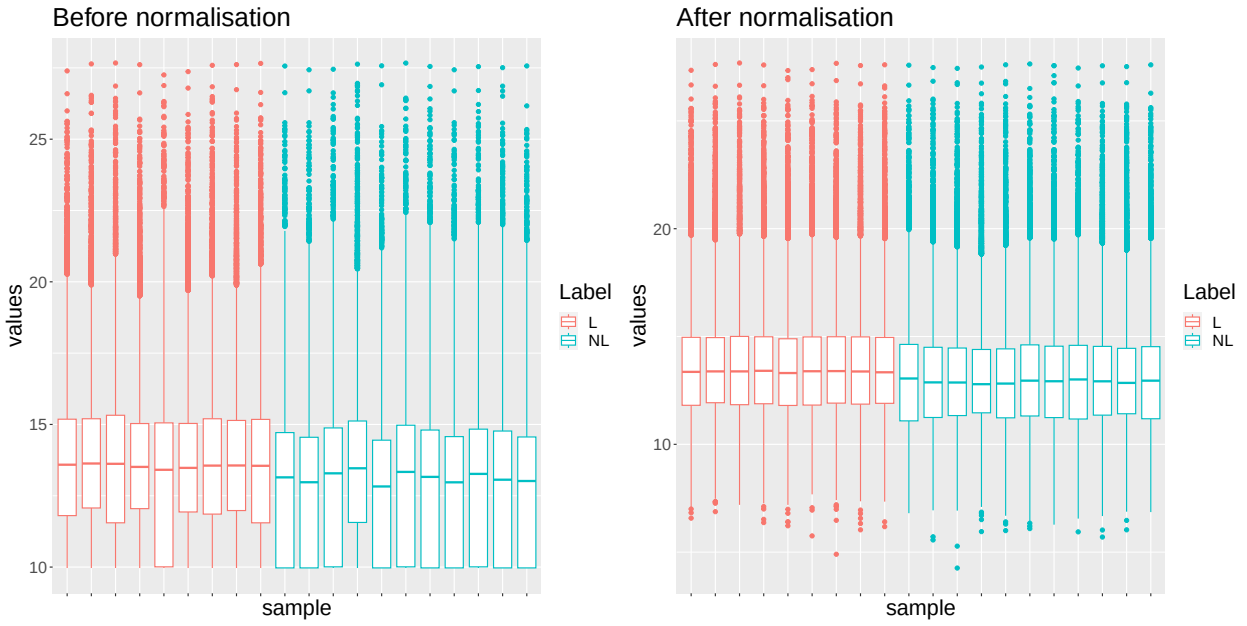
```
eigenMS.df <- mani.df %>% group_by(Omic, Polarity) %>% group_modify(~normEigenMS(.x, dplyr::rename(sampl
head(eigenMS.df)
```

```
## # A tibble: 6 x 23
## # Groups:   Omic, Polarity [1]
##   Omic  Polarity Feature~1    L1    L2    L3    L4    L5    L6    L7    L8    L9
##   <chr> <chr>    <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Lx    Negative LN00001    14.1  14.8  14.3 14.1   13.3  14.8  14.0  13.9  13.4
## 2 Lx    Negative LN00003    10.6  11.7  12.0 12.4   11.8  11.9  11.1  11.8  11.9
```

```
## 3 Lx    Negative LN00007    16.3  16.0  12.7  9.51  10.9  12.4  13.7  10.9  16.0
## 4 Lx    Negative LN00006    18.4  15.8  17.9 10.3   10.4  16.7  16.8  19.1  12.6
## 5 Lx    Negative LN00004    18.3  18.7  20.6 19.0   18.9  17.0  20.0  18.9  18.3
## 6 Lx    Negative LN00009    13.9  13.1  13.3 14.4   12.4  12.3  15.1  14.2  11.7
## # ... with 11 more variables: NL1 <dbl>, NL2 <dbl>, NL3 <dbl>, NL4 <dbl>,
## #   NL5 <dbl>, NL6 <dbl>, NL7 <dbl>, NL8 <dbl>, NL9 <dbl>, NL10 <dbl>,
## #   NL11 <dbl>, and abbreviated variable name 1: Feature.ID
```

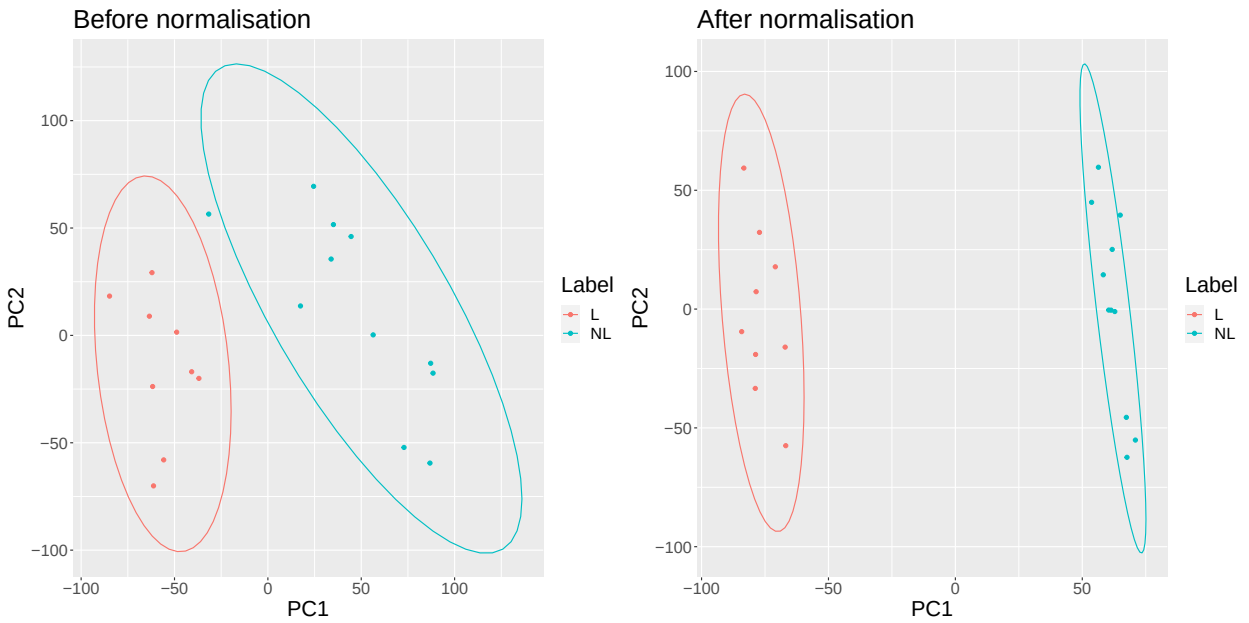
We can use the box plot and PCA plot to evaluate the normalisation of the data. After EigenMS normalisation, the sample is normalised across the sample within each tissue group, and PCA result shows that the difference between different tissue group is retained. Noticeably, while EigenMS takes the tissue group into account during normalisation, no extra information regarding the treatment is provided. However, the normalised result shows the plaque data (stable plaque and unstable plaque) are closer to each other than the leison-free data, suggesting the difference between the grouping is real.

```
p1 <- stack(mani.df %>% ungroup() %>% select(-Omic, -Polarity, -Feature.ID)) %>% inner_join(sample.info
ggplot(aes(x = ind, y = values, color = Label)) + geom_boxplot() +
ggtitle("Before normalisation") + theme(axis.text.x = element_blank()) + xlab("sample")
p2 <- stack(eigenMS.df %>% ungroup() %>% select(-Omic, -Polarity, -Feature.ID)) %>% inner_join(sample.i
ggplot(aes(x = ind, y = values, color = Label)) + geom_boxplot() +
ggtitle("After normalisation") + theme(axis.text.x = element_blank()) + xlab("sample")

cowplot::plot_grid(p1, p2, nrow = 1)
```




```
before.norm <- prcomp(t(mani.df %>% ungroup() %>% select(-Omic, -Polarity) %>% column_to_rownames("Featu
after.norm <- prcomp(t(eigenMS.df %>% ungroup() %>% select(-Omic, -Polarity) %>% column_to_rownames("Fea

before.norm.df <- data.frame(before.norm$x)
before.norm.df$sample <- rownames(before.norm.df)

after.norm.df <- data.frame(after.norm$x)
after.norm.df$sample <- rownames(after.norm.df)
```

```
p1 <- before.norm.df %>% inner_join(sample.info, by = "sample") %>% ggplot(aes(PC1, PC2, color = Label))
geom_point() + stat_ellipse() + ggtitle("Before normalisation")
p2 <- after.norm.df %>% inner_join(sample.info, by = "sample") %>% ggplot(aes(PC1, PC2, color = Label))
geom_point() + stat_ellipse() + ggtitle("After normalisation")

cowplot::plot_grid(p1, p2, nrow = 1)
```



Although EigenMS normalisation normalise the sample within the tissue group, it does not standardise the overall feature table to the same median. Therefore, we further standardise the range of the sample to the same median so that we can compare the value across different samples.

```
eigenMS.median.df <- eigenMS.df %>% group_modify(~normMedian(.x))
head(eigenMS.median.df)
```

```
## # A tibble: 6 x 23
## # Groups:   Omic, Polarity [1]
##   Omic  Polarity Feature~1    L1    L2    L3    L4    L5    L6    L7    L8    L9
##   <chr> <chr>    <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Lx    Negative LN00001    13.9  14.6  14.1 13.7   13.1  14.5  13.8  13.6  13.1
## 2 Lx    Negative LN00003    10.4  11.5  11.8 12.1   11.5  11.6  10.9  11.6  11.6
## 3 Lx    Negative LN00007    16.1  15.8  12.6  9.21  10.7  12.1  13.5  10.6  15.7
## 4 Lx    Negative LN00006    18.2  15.6  17.8 10.0   10.2  16.4  16.5  18.9  12.3
## 5 Lx    Negative LN00004    18.1  18.5  20.4 18.7   18.7  16.7  19.7  18.6  18.1
## 6 Lx    Negative LN00009    13.7  12.9  13.1 14.1   12.2  12.0  14.9  13.9  11.4
## # ... with 11 more variables: NL1 <dbl>, NL2 <dbl>, NL3 <dbl>, NL4 <dbl>,
## #   NL5 <dbl>, NL6 <dbl>, NL7 <dbl>, NL8 <dbl>, NL9 <dbl>, NL10 <dbl>,
## #   NL11 <dbl>, and abbreviated variable name 1: Feature.ID
```

Finally, we use the annotation from the previous step to identify unique metabolites in our data using `group_annotation`. For each metabolite, the function will use the features that contain the largest median intensity as the representative of that feature. User can change the `match.id` argument to specify which column in the annotation table should the function use as unique feature ID.

```
group.feature.table <- group_annotation(eigenMS.median.df,
                                        annotated,
                                        match.id = "HMDB.ID")
head(group.feature.table)
```

```
## # A tibble: 6 x 22
##   Omic  HMDB.ID        L1    L2    L3    L4    L5    L6    L7    L8    L9   NL1
##   <chr> <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mx    HMDB0000021  15.0  16.2 16.1  15.6   15.4 15.7   15.3 15.7  15.9  15.9
## 2 Mx    HMDB0000034  10.8  12.5  8.76 15.0   11.4 11.9   12.3 12.0   9.72  8.79
## 3 Lx    HMDB0000037  13.7  13.9 12.3  10.8   11.9 10.3   12.5 12.8  12.1  10.7
## 4 Mx    HMDB0000051  11.7  11.8  9.97 13.0   11.5 12.8   12.8  9.27 12.8  13.8
## 5 Lx    HMDB0000053  11.9  11.6 12.2   9.39  10.5  9.74  10.2 11.0  11.2   9.77
## 6 Lx    HMDB0000063  14.0  14.2 13.5  13.1   13.5 13.3   13.7 13.8  13.8  13.2
## # ... with 10 more variables: NL2 <dbl>, NL3 <dbl>, NL4 <dbl>, NL5 <dbl>,
## #   NL6 <dbl>, NL7 <dbl>, NL8 <dbl>, NL9 <dbl>, NL10 <dbl>, NL11 <dbl>
```

Demonstration on Statistical analysis

Statistical analysis is performed on the normalised feature table. Here, we presented differential analysis using the `limma` package. However, there are other statistical analysis could be performed, such as network analysis and marker identification, with the annotated normalised feature table, or dimension reduction method using sparse partial least squares projection (sPLS), and regularised canonical correlation analysis (rCCA) using mixOmics[2].

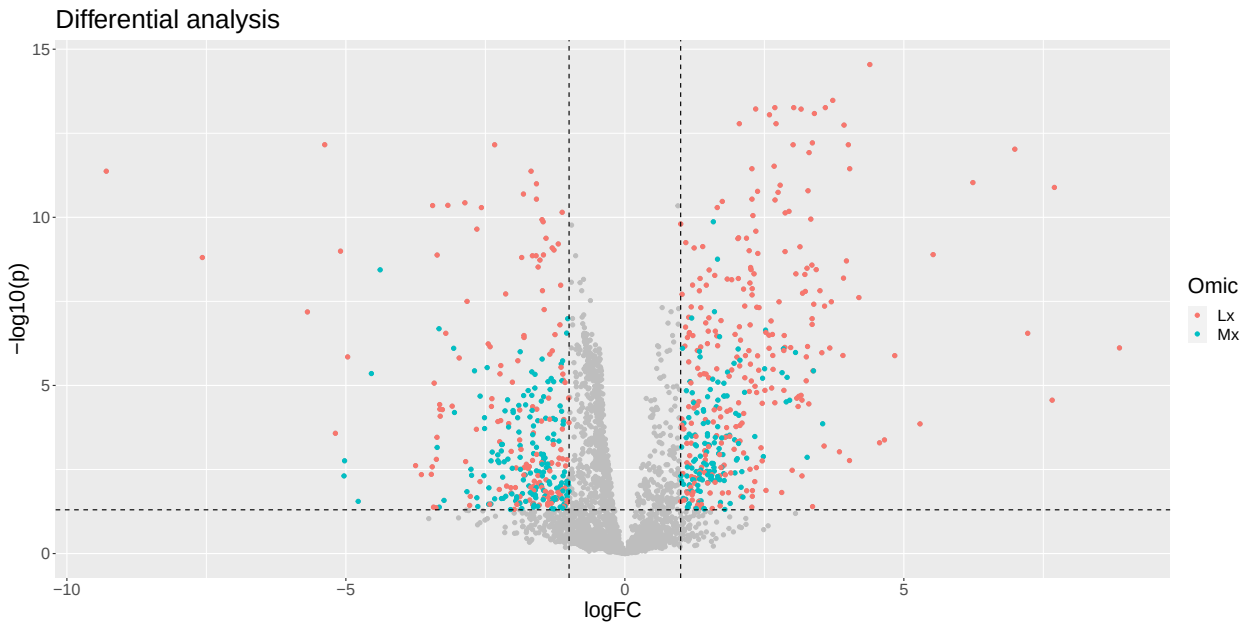
```
label <- as.factor(sample.only.info$Label)
design <- model.matrix(~ label)
colnames(design)[1] <- "Intercept"

cont.matrix <- limma::makeContrasts(NLvL = -labelNL, levels = design)

annotated.feature.info <- group.feature.table %>% select(Omic, Feature.ID = HMDB.ID)

limma.result <- group.feature.table %>% select(-Omic) %>% dplyr::rename(Feature.ID = HMDB.ID) %>% runLi
head(limma.result)
```

```
## # A tibble: 6 x 3
##   Feature.ID  logFC.NLvL adj.P.Val.NLvL
##   <chr>            <dbl>          <dbl>
## 1 HMDB0062458       4.39       2.85e-15
## 2 HMDB0115218       3.73       3.33e-14
## 3 HMDB0007057       3.03       5.46e-14
## 4 HMDB0240629       3.59       5.46e-14
## 5 HMDB0009240       2.69       5.46e-14
## 6 HMDB0009276       2.34       6.00e-14
```

Finally, we can plot the differential analysis result using the output from Limma.

```
limma.result %>% inner_join(annotated.feature.info, by = "Feature.ID") %>% ggplot(aes(logFC.NLvL, -log1
  geom_point(data = . %>% filter(abs(logFC.NLvL) > 1, adj.P.Val.NLvL < 0.05), aes(color = Omic)) +
xlab("logFC") + ylab("-log10(p)") + geom_hline(yintercept = -log10(0.05), lty = "dashed") +
```

```
geom_vline(xintercept = c(-1,1), lty = "dashed") +
ggtitle("Differential analysis")
```



Reference

[1] Talib, J., Hains, P. G., Tumanov, S., Hodson, M. P., Robinson, P. J., & Stocker, R. (2019). Barocycler-Based Concurrent Multiomics Method To Assess Molecular Changes Associated with Atherosclerosis Using Small Amounts of Arterial Tissue from a Single Mouse. *Analytical Chemistry*, *91*(20), 12670–9.

[2] Lê Cao, K.-A., González, I., & Déjean, S. (2009). integrOmics: An R package to unravel relationships between two omics datasets. *Bioinformatics*, *25*(21), 2855–6.

Session Info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS Linux 8
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib64/libopenblasp-r0.3.12.so
##
## locale:
##  [1] LC_CTYPE=en_HK.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_HK.UTF-8        LC_COLLATE=en_HK.UTF-8
##  [5] LC_MONETARY=en_HK.UTF-8    LC_MESSAGES=en_HK.UTF-8
##  [7] LC_PAPER=en_HK.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_HK.UTF-8 LC_IDENTIFICATION=C
```

```
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] forcats_0.5.1       purrr_1.0.1         ggplot2_3.4.1
##  [4] tidyverse_1.3.2     cowplot_1.1.1       MultiABLER_1.0.1
##  [7] metid_1.2.19        CAMERA_1.48.0       xcms_3.14.1
## [10] MSnbase_2.18.0      ProtGenerics_1.26.0 S4Vectors_0.32.4
## [13] mzR_2.26.1          Rcpp_1.0.10         Biobase_2.54.0
## [16] BiocGenerics_0.40.0 BiocParallel_1.28.3 ProteoMM_1.10.0
## [19] limma_3.48.3        tibble_3.1.8        tidyr_1.3.0
## [22] stringr_1.5.0       readr_2.1.4         dplyr_1.1.0
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.3                  tidyselect_1.2.0
##   [3] htmlwidgets_1.6.1           grid_4.1.0
##   [5] devtools_2.4.5              munsell_0.5.0
##   [7] codetools_0.2-18            preprocessCore_1.54.0
##   [9] interp_1.1-3                future_1.31.0
##  [11] miniUI_0.1.1.1              withr_2.5.0
##  [13] colorspace_2.1-0            knitr_1.42
##  [15] rstudioapi_0.14             robustbase_0.95-0
##  [17] mzID_1.32.0                 listenv_0.8.0
##  [19] labeling_0.4.2              MatrixGenerics_1.6.0
##  [21] GenomeInfoDbData_1.2.7      farver_2.1.1
##  [23] bit64_4.0.5                 rprojroot_2.0.3
##  [25] parallelly_1.34.0           vctrs_0.5.2
##  [27] generics_0.1.3              xfun_0.37
##  [29] timechange_0.2.0            R6_2.5.1
##  [31] doParallel_1.0.17           GenomeInfoDb_1.30.1
##  [33] clue_0.3-59                 MsCoreUtils_1.6.2
##  [35] bitops_1.0-7                cachem_1.0.7
##  [37] gridGraphics_0.5-1          DelayedArray_0.20.0
##  [39] assertthat_0.2.1            vroom_1.6.1
##  [41] promises_1.2.0.1            scales_1.2.1
##  [43] googlesheets4_1.0.0         nnet_7.3-16
##  [45] gtable_0.3.1                Cairo_1.5-12.2
##  [47] affy_1.72.0                 globals_0.16.2
##  [49] processx_3.8.0              rlang_1.0.6
##  [51] GlobalOptions_0.1.2         splines_4.1.0
##  [53] lazyeval_0.2.2              Rdisop_1.54.0
##  [55] gargle_1.2.0                impute_1.68.0
##  [57] broom_1.0.3                 checkmate_2.1.0
##  [59] modelr_0.1.8                BiocManager_1.30.19
##  [61] yaml_2.3.7                  backports_1.4.1
##  [63] httpuv_1.6.9                Hmisc_5.0-0
##  [65] MassSpecWavelet_1.60.1      RBGL_1.68.0
##  [67] tools_4.1.0                 usethis_2.1.6
##  [69] ggplotify_0.1.0             affyio_1.64.0
##  [71] ellipsis_0.3.2              RColorBrewer_1.1-3
##  [73] sessioninfo_1.2.2           plyr_1.8.8
##  [75] progress_1.2.2              base64enc_0.1-3
```

```
##  [77] zlibbioc_1.40.0             RCurl_1.98-1.10
##  [79] ps_1.7.2                    prettyunits_1.1.1
##  [81] rpart_4.1-15                deldir_1.0-6
##  [83] pbapply_1.4-3               GetoptLong_1.0.5
##  [85] urlchecker_1.0.1            haven_2.4.3
##  [87] SummarizedExperiment_1.24.0 ggrepel_0.9.3
##  [89] cluster_2.1.2               fs_1.6.1
##  [91] furrr_0.3.1                 magrittr_2.0.3
##  [93] masstools_1.0.8             data.table_1.14.8
##  [95] openxlsx_4.2.5.2            circlize_0.4.13
##  [97] reprex_2.0.1                RANN_2.6.1
##  [99] googledrive_2.0.0           pcaMethods_1.84.0
## [101] matrixStats_0.63.0          pkgload_1.3.2
## [103] hms_1.1.2                   mime_0.12
## [105] evaluate_0.20               xtable_1.8-4
## [107] XML_3.99-0.13               jpeg_0.1-10
## [109] readxl_1.4.1                IRanges_2.28.0
## [111] gridExtra_2.3               shape_1.4.6
## [113] compiler_4.1.0              ncdf4_1.21
## [115] crayon_1.5.2                htmltools_0.5.4
## [117] later_1.3.0                 tzdb_0.3.0
## [119] Formula_1.2-5               lubridate_1.9.2
## [121] DBI_1.1.3                   dbplyr_2.1.1
## [123] ComplexHeatmap_2.8.0        MASS_7.3-54
## [125] Matrix_1.5-3                cli_3.6.0
## [127] vsn_3.62.0                  parallel_4.1.0
## [129] igraph_1.3.5                GenomicRanges_1.46.1
## [131] pkgconfig_2.0.3             foreign_0.8-81
## [133] plotly_4.10.1               xml2_1.3.3
## [135] MALDIquant_1.22             foreach_1.5.2
## [137] XVector_0.34.0              massdataset_1.0.12
## [139] rvest_1.0.3                 yulab.utils_0.0.6
## [141] callr_3.7.3                 digest_0.6.31
## [143] graph_1.70.0                cellranger_1.1.0
## [145] rmarkdown_2.20              htmlTable_2.4.1
## [147] shiny_1.7.4                 rjson_0.2.21
## [149] jsonlite_1.8.4              lifecycle_1.0.3
## [151] viridisLite_0.4.1           desc_1.4.2
## [153] fansi_1.0.4                 pillar_1.8.1
## [155] ggsci_2.9                   lattice_0.20-44
## [157] httr_1.4.5                  fastmap_1.1.1
## [159] DEoptimR_1.0-11             pkgbuild_1.4.0
## [161] survival_3.2-12             glue_1.6.2
## [163] remotes_2.4.2               zip_2.2.2
## [165] png_0.1-8                   iterators_1.0.14
## [167] bit_4.0.5                   stringi_1.7.12
## [169] profvis_0.3.7               latticeExtra_0.6-30
## [171] memoise_2.0.1
```