Analysis on serum dataset

# 1 Abstract

   Untargeted mass spectrometry is a robust tool for biological research, but researchers universally time consumed by dataset parsing. We developed MCnebula, a novel visualization strategy proposed with multidimensional view, termed multi-chemical nebulae, involving in scope of abundant classes, classification, structures, sub-structural characteristics and fragmentation similarity. Many state-of-the-art technologies and popular methods were incorporated in MCnebula workflow to boost chemical discovery. Notably, MCnebula can be applied to explore classification and structural characteristics of unknown compounds that beyond the limitation of spectral library. MCnebula was integrated in R package and public available for custom R statistical pipeline analysis. Now, MCnebula2 (R object-oriented programming with S4 system) is further available for more friendly applications.

# 2 Introduction

   We know that the analysis of untargeted LC-MS/MS dataset generally begin with feature detection. It detects ‘peaks’ as features in MS1 data. Each feature may represents a compound, and assigned with MS2 spectra. The MS2 spectra was used to find out the compound identity. The difficulty lies in annotating these features to discover their compound identity, mining out meaningful information, so as to serve further biological research. Herein, a classified visualization method, called MCnebula, was used for addressing this difficulty. MCnebula utilizes the state-of-the-art computer prediction technology, SIRIUS workflow (SIRIUS, ZODIAC, CSI:fingerID, CANOPUS)1–5, for compound formula prediction, structure retrieve and classification prediction. MCnebula integrates an abundance-based classes (ABC) selection algorithm into features annotation: depending on the user, MCnebula focuses chemical classes with more or less features in the dataset (the abundance of classes), visualizes them, and displays the features they involved; these classes can be dominant structural classes or sub-structural classes. With MCnebula, we can switch from untargeted to targeted analysis, focusing precisely on the compound or chemical class of interest to the researcher.

# 3 Set-up

   Load the R package used for analysis. In the following analysis process, to illustrate the source of the function, we use the symbol :: to mark the functions, e.g., dplyr::filter. The functions that were not marked may source from MCnebula2 or the packages that R (version 4.2) loaded by default.

library(MCnebula2)  
library(exMCnebula2)

# 4 Integrate data and Create Nebulae

## 4.1 Initialize analysis

Set SIRIUS project path and its version to initialize mcnebula object.

mcn <- mcnebula()  
mcn <- initialize\_mcnebula(mcn, "sirius.v4", ".")  
ion\_mode(mcn) <- "pos"

Create a temporary folder to store the output data.

tmp <- paste0(tempdir(), "/temp\_data")  
dir.create(tmp, F)  
export\_path(mcn) <- tmp

In order to demonstrate the process of analyzing data with MCnebula2, we provide a ‘mcnebula’ object that was extracted in advance using the collate\_used function, which means that all the data used in the subsequent analysis has already stored in this ‘mcnebula’ object, without the need to obtain it from the original Project directory. This avoids the hassle of downloading and storing a dozen GB of raw files. The following, we use the collated dataset containing 6501 features with chemical formula identification. This dataset was origin and processed from the research in article: <https://doi.org/10.1016/j.cell.2020.07.040>

exfiles <- system.file("extdata", package = "exMCnebula2")

Load the ‘.rdata’ file.

load(paste0(exfiles, "/mcn\_serum6501.rdata"))  
mcn <- mcn\_serum6501  
rm(mcn\_serum6501)

## 4.2 Filter candidates

   Suppose we predicted a potential compound represented by LC-MS/MS spectrum, and obtained the candidates of chemical molecular formula, structure and chemical class. These candidates include both positive and negative results: for chemical molecular formula and chemical structure, the positive prediction was unique; for chemical class, multiple positive predictions that belong to various classification were involved. We did not know the exact negative and positive. Normally, we ranked and filtered these according to the scores. There were numerious scores, for isotopes, for mass error, for structural similarity, for chemical classes… Which score selected to rank candidates depends on the purpose of research. Such as:

* To find out the chemical structure mostly be positive, ranking the candidates by structural score.
* To determine whether the potential compound may be of a certain chemical classes, ranking the candidates by the classified score.

Ether by filter\_formula(), filter\_structure() or filter\_ppcp(), the candidate with top score can be obtained. However, for the three module (formula, structure, classes), sometimes thier top score candidates were not in line with each other. That is, thier top score towards different chemical molecular formulas. To find out the corresponding data in other modules, create\_reference() should be performed to establish the ‘specific\_candidate’ for subsequent filtering.

mcn <- filter\_structure(mcn)  
mcn <- create\_reference(mcn)  
mcn <- filter\_formula(mcn, by\_reference = T)

## 4.3 Filter chemical classes

   The PPCP (Posterior Probability of Classification Prediction) data for each ‘feature’ contains the prediction of thousands of classes for the potential compound (even if the chemical structure was unknown). See <http://www.nature.com/articles/s41587-020-0740-8> for details about the prediction. The data contains attributes of:

* class.name: name of classes.
* pp.value: value of posterior probability. hierarchy: hierarchy of classes in the taxonomy. See <https://jcheminf.biomedcentral.com/articles/10.1186/s13321-016-0174-y> for details about hierarchy and taxonomy of chemical classification.
* …

The method create\_stardust\_classes() use these inner attributes to filter classes candidates for each ‘feature’.

Compared to the chemical class filtering within PPCP data by create\_stardust\_classes(), the filtering within ‘stardust\_classes’ data by cross\_filter\_stardust() is fundamentally different.

* For create\_stardust\_classes(), the PPCP data belongs to each ‘feature’. When performing the filtering, only simple threshold conditions or absolute conditions are set to filter the chemical classes; there is no crossover between the different attributes and no crossover between the ‘features’. Therefore, we consider this as ‘inner’ filtering.
* For cross\_filter\_stardust(), the data of the chemical classes and their classified ‘features’, i.e. ‘stardust\_classes’ data, were combined and then grouped upon the chemical classes. After grouping, each chemical class has a certain quantity of “features”. When filtering, statistics may be performed on ‘features’ data within a group; statistics may be performed on these data in conjunction with ‘features\_annotation’ data; and statistics may be performed to compare groups with each other. As its crossover, we consider this as ‘cross’ filtering.

Use help(cross\_filter\_stardust) to get more details about the algorithm.

mcn <- create\_stardust\_classes(mcn)  
mcn <- create\_features\_annotation(mcn)  
mcn <- cross\_filter\_stardust(mcn,  
 max\_ratio = 0.05, cutoff = 0.4,  
 identical\_factor = 0.6  
)  
classes <- unique(stardust\_classes(mcn)$class.name)  
table.filtered.classes <- backtrack\_stardust(mcn)

Manually filter some repetitive classes or sub-structural classes. By means of Regex matching, we obtained a number of recurring name of chemical classes that would contain manay identical compounds as their sub-structure.

classes

## [1] "Pyrans" "Pyridines and derivatives"   
## [3] "Pyrroles" "Ketones"   
## [5] "Benzopyrans" "Glycerolipids"   
## [7] "Indoles and derivatives" "Sugar acids and derivatives"   
## [9] "Glycerophospholipids" "Steroids and steroid derivatives"   
## [11] "Prenol lipids" "Branched fatty acids"   
## [13] "Unsaturated fatty acids" "Hydroxy fatty acids"   
## [15] "Hydroxy acids and derivatives" "Pyranones and derivatives"   
## [17] "Lineolic acids and derivatives" "Steroidal glycosides"   
## [19] "Acyl carnitines" "Glycinated bile acids and derivatives"   
## [21] "Diacylglycerols" "Carboxylic acid salts"   
## [23] "Oxosteroids" "Hydroxysteroids"   
## [25] "Phosphatidylcholines" "Lysophosphatidylcholines"   
## [27] "Bile acids, alcohols and derivatives" "Steroid glucuronide conjugates"   
## [29] "Diterpenoids" "Bilirubins"   
## [31] "Tertiary alcohols" "Beta hydroxy acids and derivatives"   
## [33] "Terpene glycosides" "Hydroxy bile acids, alcohols and derivatives"  
## [35] "Glycerophosphocholines" "Substituted pyrroles"   
## [37] "Indoles" "Long-chain fatty acids"   
## [39] "Organic cations" "Organic salts"   
## [41] "Vinylogous acids"

pattern <- c("stero", "fatty acid", "pyr", "hydroxy", "^orga")  
dis <- unlist(lapply(pattern, grep, x = classes, ignore.case = T))  
dis <- classes[dis]  
dis

## [1] "Steroids and steroid derivatives" "Steroidal glycosides"   
## [3] "Oxosteroids" "Hydroxysteroids"   
## [5] "Steroid glucuronide conjugates" "Branched fatty acids"   
## [7] "Unsaturated fatty acids" "Hydroxy fatty acids"   
## [9] "Long-chain fatty acids" "Pyrans"   
## [11] "Pyridines and derivatives" "Pyrroles"   
## [13] "Benzopyrans" "Pyranones and derivatives"   
## [15] "Substituted pyrroles" "Hydroxy fatty acids"   
## [17] "Hydroxy acids and derivatives" "Hydroxysteroids"   
## [19] "Beta hydroxy acids and derivatives" "Hydroxy bile acids, alcohols and derivatives"  
## [21] "Organic cations" "Organic salts"

dis <- dis[-1]

## 4.4 Create Nebulae

Create Nebula-Index data. This data created based on ‘stardust\_classes’ data.

mcn <- backtrack\_stardust(mcn, dis, remove = T)  
mcn <- create\_nebula\_index(mcn)

   Whether it is all filtered by the algorithm provided by MCnebula2’s function or custom filtered for some chemical classes, we now have a data called ‘nebula\_index’. This data records a number of chemical classes and the ‘features’ attributed to them. The subsequent analysis process or visualization will be based on it. Each chemical class is considered as a ‘nebula’ and its classified ‘features’ are the components of these ‘nebulae’. In the visualization, these ‘nebulae’ will be visualized as networks. Formally, we call these ‘nebulae’ formed on the basis of ‘nebula\_index’ data as Child-Nebulae. In comparison, when we put all the ‘features’ together to form a large network, then this ‘nebula’ is called Parent-Nebulae.

mcn <- compute\_spectral\_similarity(mcn)  
mcn <- create\_parent\_nebula(mcn)  
mcn <- create\_child\_nebulae(mcn)

## 4.5 Visualize Nebulae

Create layouts for Parent-Nebula or Child-Nebulae visualizations.

mcn <- create\_parent\_layout(mcn)  
mcn <- create\_child\_layouts(mcn)  
mcn <- activate\_nebulae(mcn)

The available chemical classes for visualization and its sequence in storage.

table.nebulae <- visualize(mcn)

## [INFO] MCnebula2: visualize

## Specify item as following to visualize:

table.nebulae

## # A tibble: 22 × 3  
## seq hierarchy class.name   
## <int> <dbl> <chr>   
## 1 1 5 Acyl carnitines   
## 2 2 4 Bile acids, alcohols and derivatives   
## 3 3 4 Bilirubins   
## 4 4 5 Carboxylic acid salts   
## 5 5 5 Diacylglycerols   
## 6 6 4 Diterpenoids   
## 7 7 3 Glycerolipids   
## 8 8 4 Glycerophosphocholines   
## 9 9 3 Glycerophospholipids   
## 10 10 5 Glycinated bile acids and derivatives  
## # … with 12 more rows

Draw and save as .png or .pdf image files.

p <- visualize(mcn, "parent")  
ggsave(f4.61 <- paste0(tmp, "/parent\_nebula.png"), p)  
pdf(f4.62 <- paste0(tmp, "/child\_nebula.pdf"), 12, 14)  
visualize\_all(mcn)  
dev.off()

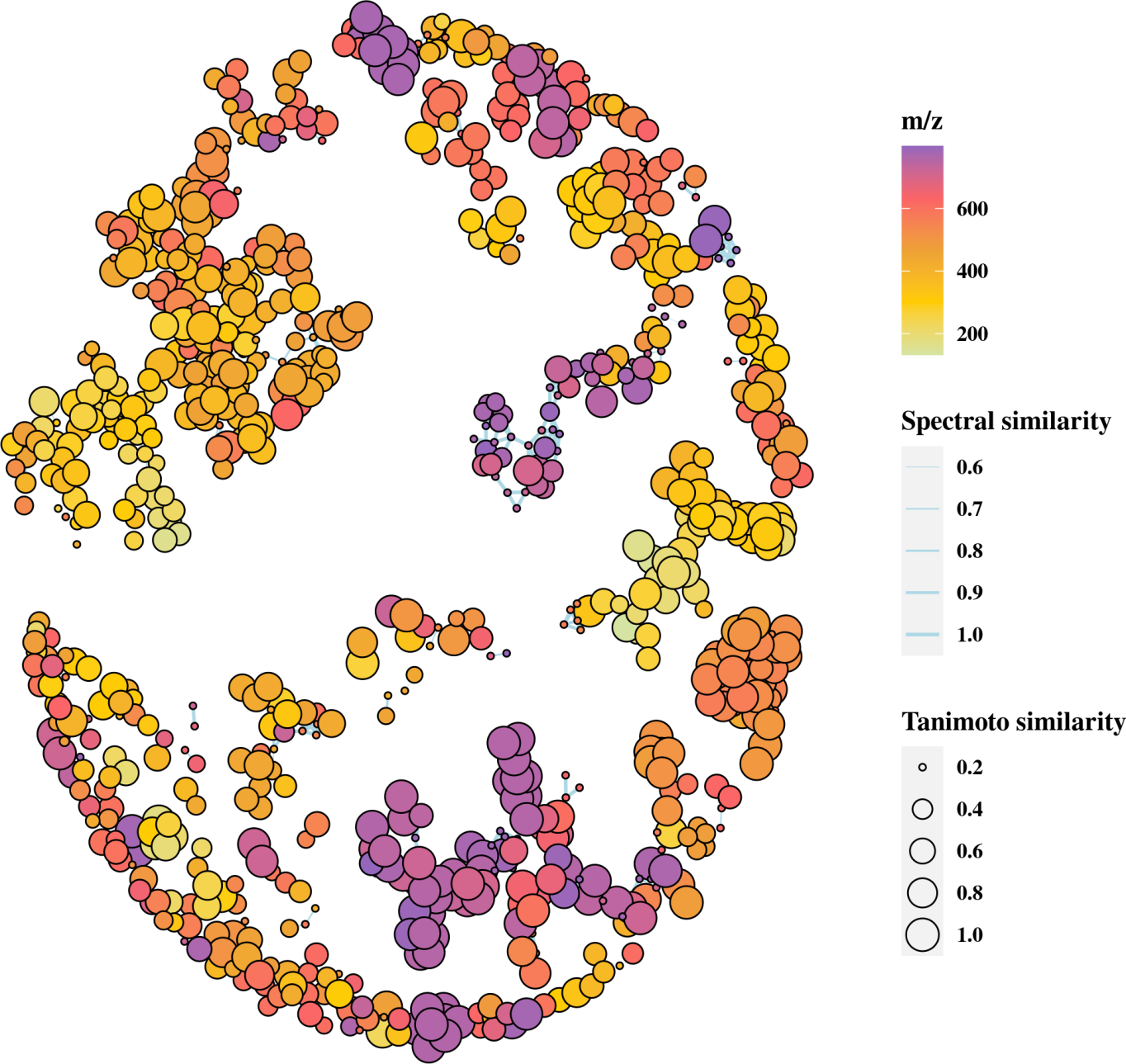


图1 Parent-Nebula

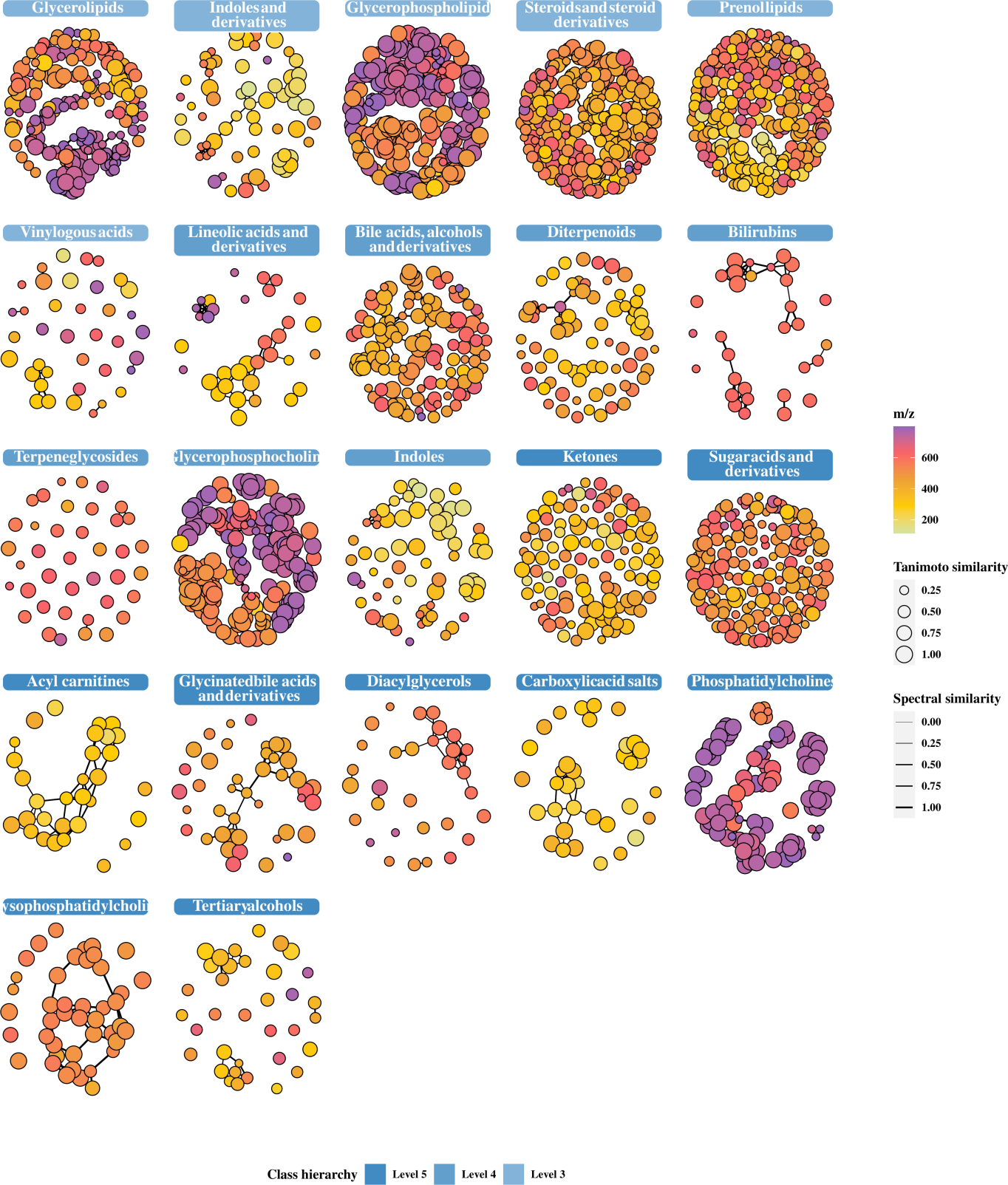


图2 Child-Nebulae

# 5 Nebulae for Downstream analysis

In general, Parent-Nebulae (Fig. [1](#parent)) is too informative to show, so Child-Nebulae (Fig. [2](#child)) was used to dipict the abundant classes of features (metabolites) in a grid panel, intuitively. In a bird’s eye view of Child-Nebulae, we can obtain many characteristics of features, involving classes distribution, structure identified accuracy, as well as spectral similarity within classes.

## 5.1 Statistic analysis

Next we perform a statistical analysis with quantification data of the features. Note that the SIRIUS project does not contain quantification data of features, so our object mcn naturally does not contain that either. We need to get it from elsewhere.

utils::untar(paste0(exfiles, "/serum.tar.gz"), exdir = tmp)  
origin <- data.table::fread(paste0(tmp, "/serum\_origin\_mmc3.tsv"),  
 skip = 1  
)  
origin <- tibble::as\_tibble(origin)

Its original data can obtained from: <https://www.cell.com/cms/10.1016/j.cell.2020.07.040/attachment/f13178d1-d1ee-4179-9d33-227a02e604f1/mmc3.xlsx>. Now, let’s check the columns in the table.

origin

## # A tibble: 5,280 × 225  
## Unique\_ID `m/z` RT Subnetwork Molecular\_Informati… Spectral\_Librar… Superclass Class Subclass  
## <int> <dbl> <dbl> <int> <chr> <chr> <chr> <chr> <chr>   
## 1 349 540. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 2 228 548. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 3 1963 1092. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 4 971 547. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 5 13 546. 6.57 45 <NA> <NA> <NA> <NA> <NA>   
## 6 4146 541. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 7 854 289. 4.12 -1 <NA> <NA> <NA> <NA> <NA>   
## 8 4046 1036. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 9 1374 788. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## 10 2304 789. 6.57 -1 <NA> <NA> <NA> <NA> <NA>   
## # … with 5,270 more rows, and 216 more variables: Direct\_Parent <chr>, Infection\_pvalue <dbl>,  
## # Infection\_FC <dbl>, Infection\_BH\_CV <dbl>, Infection\_BH\_Sig <chr>, Mortality\_pvalue <chr>,  
## # Mortality\_FC <chr>, Mortality\_BH\_CV <dbl>, Mortality\_BH\_Sig <chr>, EFS\_Rank <int>,  
## # MW\_Rank <int>, Avg\_Rank <dbl>, ANOVA\_pvalue <dbl>, `Cluster\_#` <int>, ANOVA\_CV <dbl>,  
## # ANOVA\_BH\_Sig <chr>, NN1 <dbl>, NN10 <dbl>, NN11 <dbl>, NN12 <dbl>, NN13 <dbl>, NN14 <dbl>,  
## # NN15 <dbl>, NN2 <dbl>, NN3 <dbl>, NN4 <dbl>, NN5 <dbl>, NN6 <dbl>, NN7 <dbl>, NN8 <dbl>,  
## # NN9 <dbl>, HN1 <dbl>, HN10 <dbl>, HN2 <dbl>, HN3 <dbl>, HN4 <dbl>, HN5 <dbl>, HN6 <dbl>, …

Remove the rest of the columns and keep only the columns for ID, m/z, retention time, and quantification.

keep <- grep("^[A-Z]{2}[0-9]{1,3}$", colnames(origin))  
quant <- dplyr::select(origin, oid = 1, mz = 2, rt = 3, dplyr::all\_of(keep))

The IDs in the data quant are different from the IDs in the object mcn, so we need to align them first, according to mz and rt (they originate from the same batch of samples). In the following, we have allowed the two sets of data to be merged as accurately as possible in the form of evaluation of score:

* Score = (1 - rt.difference / rt.tolerance) \* rt.weight + (1 - mz.defference / mz.tolerance) \* mz.weight

meta\_col <- dplyr::select(  
 features\_annotation(mcn), .features\_id,  
 mz, rt.secound  
)  
meta\_col$rt.min <- meta\_col$rt.secound / 60  
merged <- align\_merge(meta\_col, quant, ".features\_id",  
 rt.main = "rt.min",  
 rt.sub = "rt"  
)  
merged <- dplyr::select(  
 merged, -mz.main, -mz.sub, -rt.min, -rt,  
 -rt.secound  
)

Due to the differences in feature detection algorithms, some of the features inevitably do not get matched.

merged

## # A tibble: 3,680 × 202  
## .features\_id oid NN1 NN10 NN11 NN12 NN13 NN14 NN15 NN2  
## <chr> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 2435 0.0000298 0.0000176 7.3 e-6 8.34e-6 4.62e-6 NA NA 4.99e-6  
## 2 100 1614 0.000101 0.0000681 6.34e-5 3.94e-5 9.5 e-5 9.34e-5 6.68e-5 1.06e-4  
## 3 1000 372 0.000623 0.000787 3.50e-4 1.85e-4 5.79e-4 4.27e-4 7.19e-4 6.70e-4  
## 4 1001 1001 0.00110 0.000138 1.66e-4 1.57e-4 2.27e-4 1.87e-4 2.40e-4 6.67e-4  
## 5 1002 2364 NA NA NA NA NA NA NA NA   
## 6 1003 452 NA NA NA NA NA 8.49e-5 NA NA   
## 7 1004 2962 0.0000214 NA NA NA NA NA NA 1.5 e-5  
## 8 1005 1313 0.00000545 0.0000462 3.43e-5 1.46e-5 3.36e-5 1.36e-4 2.65e-5 6.82e-5  
## 9 1006 883 0.000119 0.0000681 1.89e-4 5.92e-5 5.06e-5 8.37e-5 3.21e-5 1.47e-4  
## 10 1007 2889 0.0000567 0.000283 2.35e-4 1.91e-4 4.64e-4 2.35e-4 3.43e-4 7.47e-5  
## # … with 3,670 more rows, and 192 more variables: NN3 <dbl>, NN4 <dbl>, NN5 <dbl>, NN6 <dbl>,  
## # NN7 <dbl>, NN8 <dbl>, NN9 <dbl>, HN1 <dbl>, HN10 <dbl>, HN2 <dbl>, HN3 <dbl>, HN4 <dbl>,  
## # HN5 <dbl>, HN6 <dbl>, HN7 <dbl>, HN8 <dbl>, HN9 <dbl>, HS1 <dbl>, HS10 <dbl>, HS11 <dbl>,  
## # HS12 <dbl>, HS13 <dbl>, HS14 <dbl>, HS15 <dbl>, HS16 <dbl>, HS17 <dbl>, HS18 <dbl>,  
## # HS19 <dbl>, HS2 <dbl>, HS20 <dbl>, HS21 <dbl>, HS22 <dbl>, HS23 <dbl>, HS24 <dbl>, HS25 <dbl>,  
## # HS26 <dbl>, HS27 <dbl>, HS28 <dbl>, HS29 <dbl>, HS3 <dbl>, HS30 <dbl>, HS31 <dbl>, HS32 <dbl>,  
## # HS33 <dbl>, HS34 <dbl>, HS35 <dbl>, HS36 <dbl>, HS37 <dbl>, HS38 <dbl>, HS39 <dbl>, …

Create the metadata table and store it in the mcn object along with the quantification data.

gp <- c(NN = "^NN", HN = "^HN", HS = "^HS", HM = "^HM")  
metadata <- MCnebula2:::group\_strings(colnames(merged), gp, "sample")  
metadata$group\_name <- vapply(metadata$group, switch,  
 FUN.VALUE = character(1),  
 NN = "non-hospital & non-infected", HN = "hospital & non-infected",  
 HS = "hospital & survival", HM = "hospital & mortality"  
)  
metadata$supergroup <- vapply(metadata$group, switch,  
 FUN.VALUE = character(1),  
 NN = "control groups", HN = "control groups", HS = "infection groups",  
 HM = "infection groups"  
)  
features\_quantification(mcn) <- dplyr::select(merged, -oid)  
sample\_metadata(mcn) <- metadata

   Variance analysis was used as a way to detect whether there were differences between the experimental and control groups and whether the differences were significant. Linear models are an effective tool for variance analysis, and it permit very general analyses. The ‘limma’ package6 integrates a number of functions for creating linear models and regression analysis. The statistical analysis provided in MCnebula2 is mainly built around the functions in the ‘limma’ package.

In the following we use the binary\_comparison function for variance analysis. Note that the quantification data in origin has been normalized. To accommodate the downstream analysis of gene expression that the limma package was originally used for, we should log2-transform and centralize this data.

mcn <- binary\_comparison(mcn, (HS + HM) - (NN + HN), HM - HS,  
 fun\_norm = function(x) scale(log2(x), scale = F)  
)  
top.list <- top\_table(statistic\_set(mcn))

To verify the validity of the above variance analysis, the data columns were merged to obtain the IDs from the original analysis.

top.list <- lapply(top.list, merge, y = dplyr::select(  
 merged,  
 .features\_id, oid  
), by = ".features\_id", all.x = T, sort = F)  
top.list <- lapply(top.list, tibble::as\_tibble)

Verify with the EFS\_Rank and MW\_Rank column in the origin data. (The original authors used the two methods to rank the features.)

origin\_top50 <- dplyr::filter(origin, EFS\_Rank <= 50 | MW\_Rank <=  
 50)  
inter. <- lapply(top.list, function(df) {  
 match <- head(df, n = 50)$oid %in% origin\_top50$Unique\_ID  
 oid <- head(df, n = 50)$oid[match]  
 list(table.match = table(match), oid = oid)  
})  
lapply(inter., function(x) x$table.match)

## $`(HS + HM) - (NN + HN)`  
## match  
## FALSE TRUE   
## 48 2   
##   
## $`HM - HS`  
## match  
## FALSE TRUE   
## 13 37

Let’s see which compounds were identified that intersected our ranking and the original ranking of features.

inter.compound <- dplyr::filter(origin, Unique\_ID %in% inter.[[2]]$oid)  
table(inter.compound$Spectral\_Library\_Match, useNA = "if")

##   
## Decanoyl-L-carnitine L-THYROXINE <NA>   
## 1 1 27

Interestingly, these two compounds were critical compounds in the original study.

## 5.2 Set tracer in Child-Nebulae

   Tracking top features obtained by Variance analysis in Nebulae provides insight not only into the chemical classes of these top features, but also into other features (may be analogous metabolites). Other features are not among the top ranked features, but they may contain key features that were missed due to algorithmic specificity. By tracking top features, it is possible to revisit all features at the overall data level.

n <- 50  
tops <- unique(unlist(lapply(top.list, function(df) df$.features\_id[1:n])))  
palette\_set(melody(mcn)) <- colorRampPalette(palette\_set(mcn))(length(tops))  
mcn2 <- set\_tracer(mcn, tops)  
mcn2 <- create\_child\_nebulae(mcn2)  
mcn2 <- create\_child\_layouts(mcn2)  
mcn2 <- activate\_nebulae(mcn2)  
mcn2 <- set\_nodes\_color(mcn2, use\_tracer = T)

Draw and save the image.

pdf(f6.2 <- paste0(tmp, "/tracer\_child\_nebula.pdf"), 12, 14)  
visualize\_all(mcn2)  
dev.off()

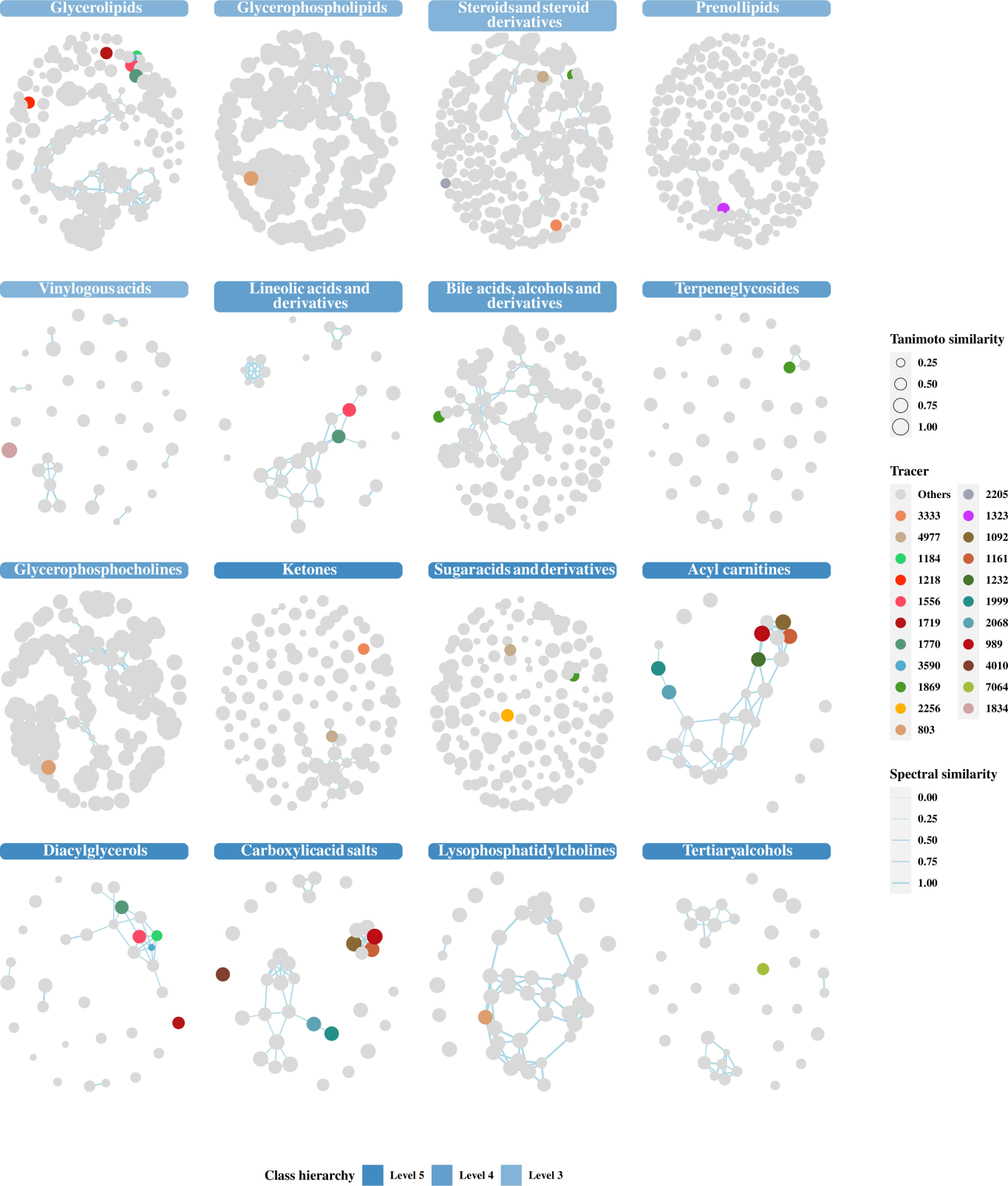


图3 Tracing top features in Child-Nebulae

A part of the top features are marked with colored nodes in Child-Nebulae (Fig. [3](#tracer)) . These features are at least identified with chemical molecular formula. Those that are not identified, or the Nebula-Index data do not contain the chemical class to which these features belong, are absent from the Figure.

## 5.3 Quantification in Child-Nebulae

Show Fold Change (HM versus HS) in Child-Nebulae.

palette\_gradient(melody(mcn2)) <- c("blue", "grey90", "red")  
mcn2 <- set\_nodes\_color(mcn2, "logFC", top.list[[2]])  
pdf(f7.1 <- paste0(tmp, "/logFC\_child\_nebula.pdf"), 12, 14)  
visualize\_all(mcn2, fun\_modify = modify\_stat\_child)  
dev.off()

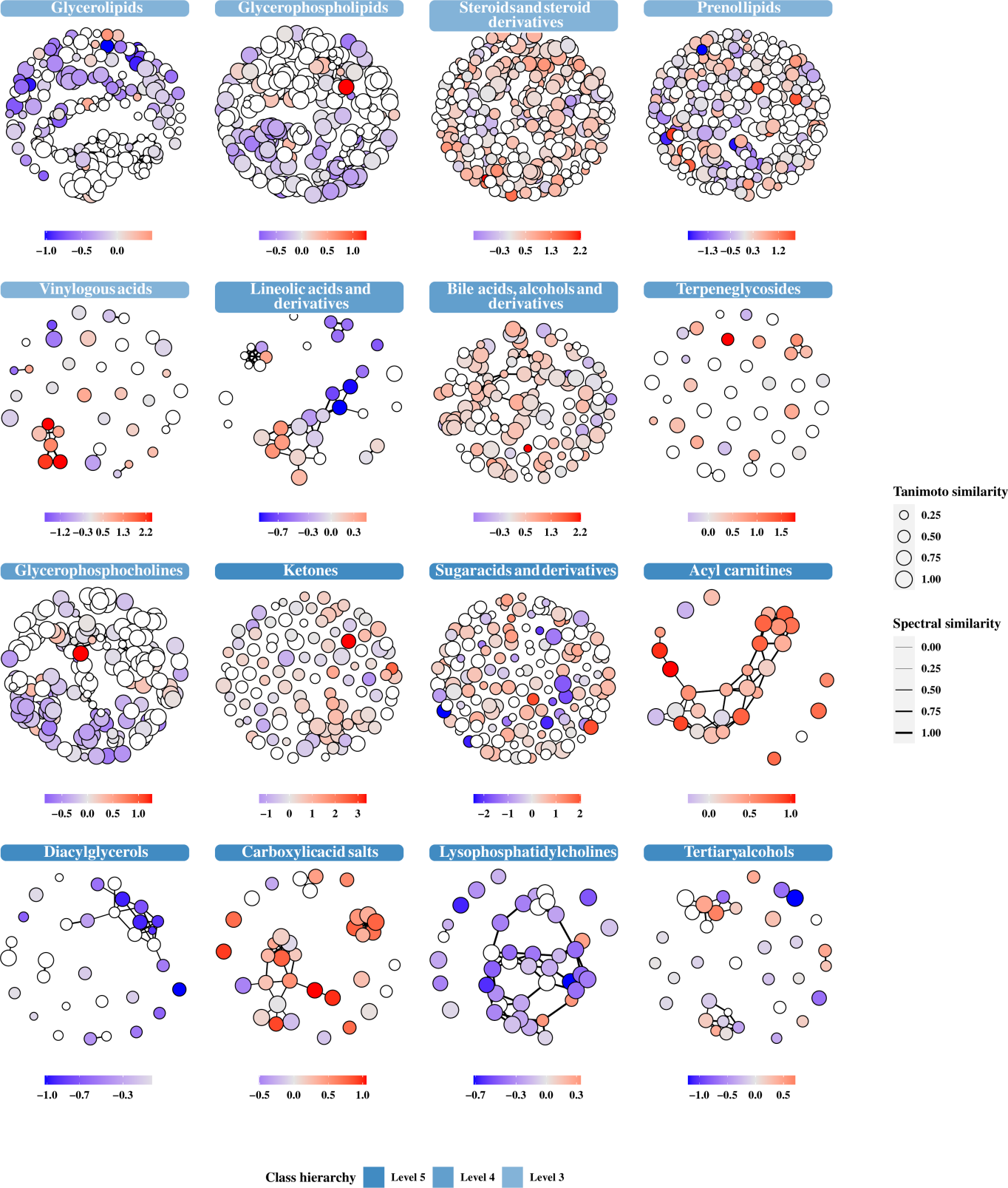


图4 Show log2(FC) in Child-Nebulae

Each Child-Nebula separately shows the overall content variation of the chemical class to which it belongs (Fig. [4](#logFC)) .

## 5.4 Annotate Nebulae

Now, the available Nebulae contains:

table.nebulae2 <- visualize(mcn2)

## [INFO] MCnebula2: visualize

## Specify item as following to visualize:

table.nebulae2

## # A tibble: 16 × 3  
## seq hierarchy class.name   
## <int> <dbl> <chr>   
## 1 1 5 Acyl carnitines   
## 2 2 4 Bile acids, alcohols and derivatives  
## 3 3 5 Carboxylic acid salts   
## 4 4 5 Diacylglycerols   
## 5 5 3 Glycerolipids   
## 6 6 4 Glycerophosphocholines   
## 7 7 3 Glycerophospholipids   
## 8 8 5 Ketones   
## 9 9 4 Lineolic acids and derivatives   
## 10 10 5 Lysophosphatidylcholines   
## 11 11 3 Prenol lipids   
## 12 12 3 Steroids and steroid derivatives   
## 13 13 5 Sugar acids and derivatives   
## 14 14 4 Terpene glycosides   
## 15 15 5 Tertiary alcohols   
## 16 16 3 Vinylogous acids

Next, let us focus on Acyl carnitines, a class that was highlighted in the original research and also appears in Child-Nebulae, marked by plural top features (Likewise, we annotated two other chemical classes of Nebulae).

mcn2 <- set\_nodes\_color(mcn2, use\_tracer = T)  
palette\_stat(melody(mcn2)) <- c(  
 NN = "#B6DFB6", HN = "#ACDFEE",  
 HS = "#EBA9A7", HM = "grey70"  
)  
ac <- "Acyl carnitines"  
lpc <- "Lysophosphatidylcholines"  
ba <- "Bile acids, alcohols and derivatives"  
mcn2 <- annotate\_nebula(mcn2, ac)  
mcn2 <- annotate\_nebula(mcn2, lpc)  
mcn2 <- annotate\_nebula(mcn2, ba)

Draw and save the image.

p <- visualize(mcn2, ac, annotate = T)  
ggsave(f8.2 <- paste0(tmp, "/ac\_child.pdf"), p, height = 5)  
p <- visualize(mcn2, lpc, annotate = T)  
ggsave(f8.2.2 <- paste0(tmp, "/lpc\_child.pdf"), p, height = 5)  
p <- visualize(mcn2, ba, annotate = T)  
ggsave(f8.2.3 <- paste0(tmp, "/ba\_child.pdf"), p, height = 5)

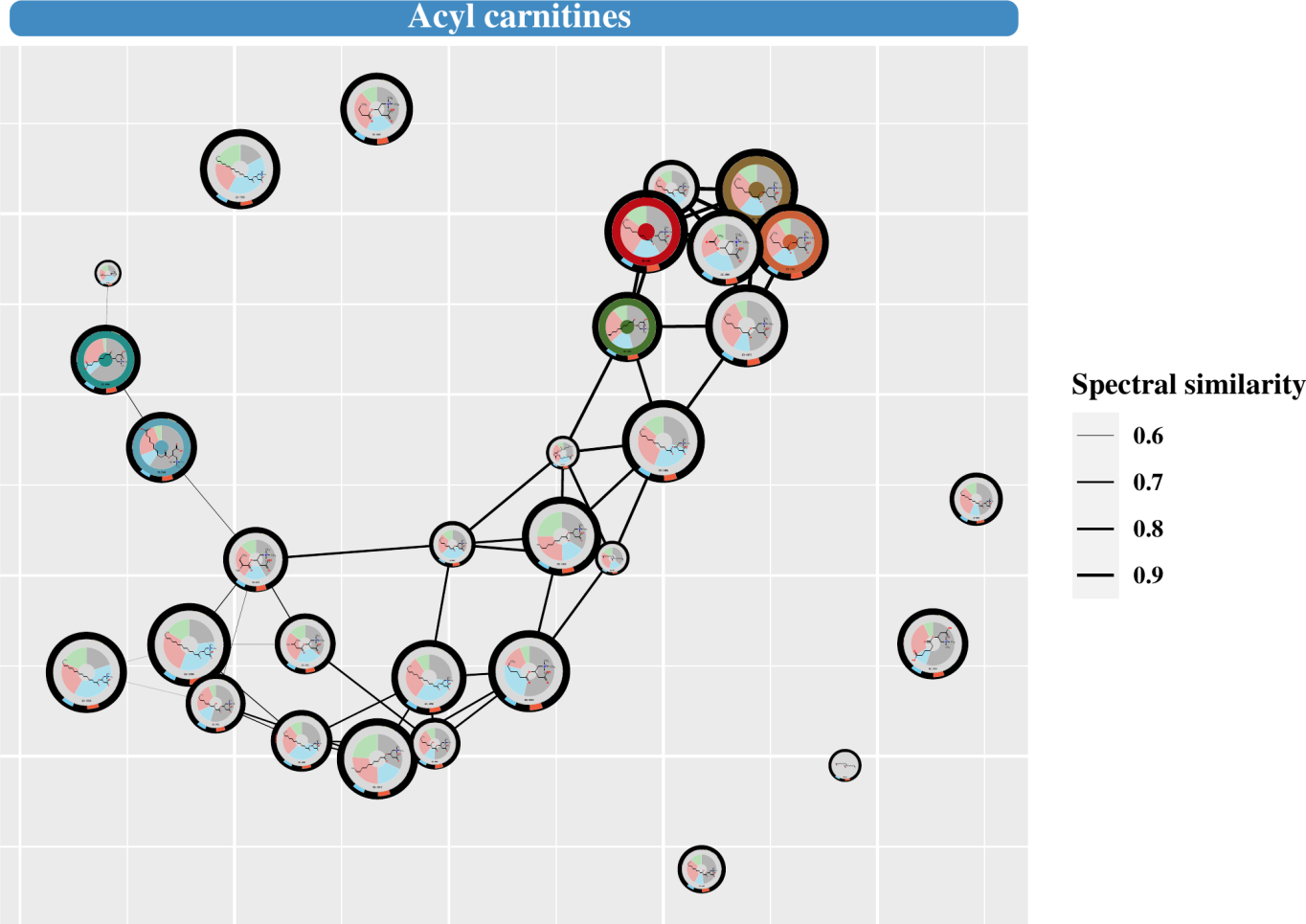


图5 Annotated Nebulae: Acyl carnitines

See results (Fig. [5](#ac)) .    The annotated Nebula presents a multi-dimensional annotation for each FEATURES, involving chemical classes, chemical structures, quantification etc.

Use the show\_node function to get the annotation details for a feature. For example:

ef <- "2068"  
pdf(f8.4 <- paste0(tmp, "/features\_", ef, ".pdf"), 10, 4)  
show\_node(mcn2, ef)  
dev.off()

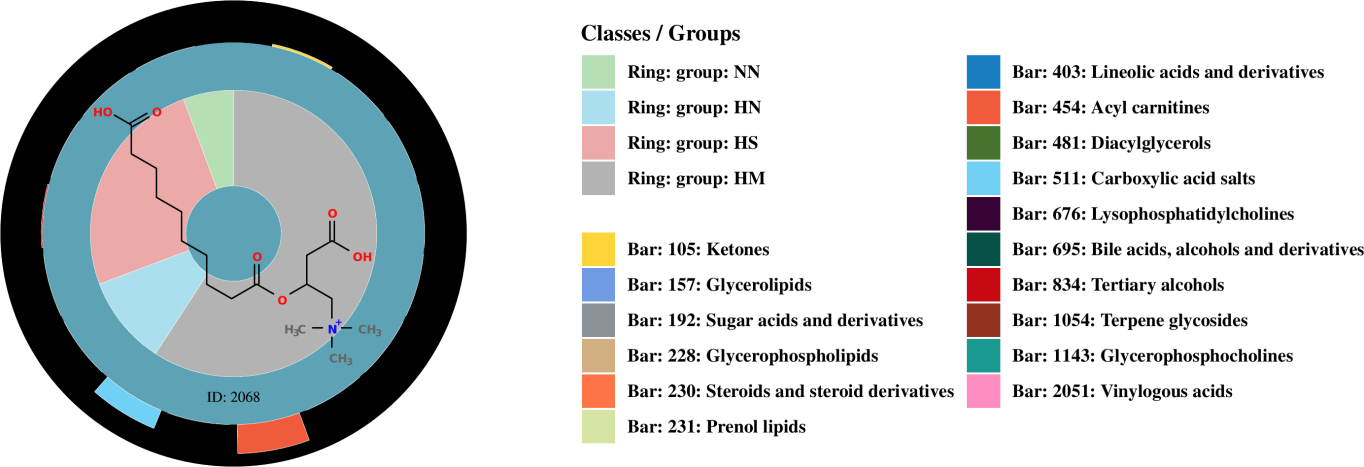


图6 The annotated feature of ID: 2068

See results (Fig. [6](#ef)) .

## 5.5 Query compounds

The features\_annotation(mcn) contains the main annotation information of all the features, i.e., the identity of the compound. Next, we would query the identified compounds based on the ‘inchikey2d’ column therein. Note that the stereoisomerism of the compounds is difficult to be determined due to the limitations of MS/MS spectroscopy. Therefore, we used InChIKey 2D (representing the molecular backbone of the compound) to query the compound instead of InChI.

First we need to format and organize the annotated data of features to get the non-duplicated ‘inchikey2d’. We provide a function with a pre-defined filtering algorithm to quickly organize the table. By default, this function filters the data based on ‘tani.score’ (Tanimoto similarity), and then sorts and de-duplicates it.

feas <- features\_annotation(mcn2)  
feas <- format\_table(feas, export\_name = NULL)  
key2d <- feas$inchikey2d

Create a folder to store the acquired data.

tmp2 <- paste0(tmp, "/query")  
dir.create(tmp2, F)

Query the compound’s InChIKey, chemical class, IUPUA name. If your system is not Linux, the multithreading below may pose some problems, please remove the parameters curl\_cl = 4 and classyfire\_cl = 4.

key.rdata <- query\_inchikey(key2d, tmp2, curl\_cl = 4)  
class.rdata <- query\_classification(key2d, tmp2, classyfire\_cl = 4)  
iupac.rdata <- query\_iupac(key2d, tmp2, curl\_cl = 4)

We will also query for synonyms of compounds, but this is done in ‘CID’ (PubChem’s ID), so some transformation is required.

key.set <- extract\_rdata\_list(key.rdata)  
cid <- lapply(key.set, function(data) data$CID)  
cid <- unlist(cid, use.names = F)  
syno.rdata <- query\_synonyms(cid, tmp2, curl\_cl = 4)

Screen for unique synonyms and chemical classes for all compounds.

syno <- pick\_synonym(key2d, key.rdata, syno.rdata, iupac.rdata)  
feas$synonym <- syno  
class <- pick\_class(key2d, class.rdata)  
feas$class <- class  
feas.table <- rename\_table(feas)  
write\_tsv(feas.table, paste0(tmp, "/compounds\_format.tsv"))

The formatted table as following:

feas.table

## # A tibble: 1,086 × 13  
## Synonym ID `Precursor m/z` `Mass error (p…` `RT (min)` Formula Adduct `Tanimoto simi…`  
## <chr> <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 Stearoyllysop… 803 546. 8 6.6 C26H54… [M + … 0.84  
## 2 3-beta,5-beta… 4977 509. -2.3 4.2 C27H40… [M + … 0.54  
## 3 Etiocholanedi… 3333 289. 8.5 4.1 C19H28… [M + … 0.51  
## 4 Isoleucylprol… 885 229. -6.1 0.4 C11H20… [M + … 0.54  
## 5 sebacoylcarni… 2068 346. 5.7 3.4 C17H31… [M + … 0.84  
## 6 Dimethylguano… 2231 312. -0.8 0.7 C12H17… [M + … 0.99  
## 7 N6-Threonylca… 2199 413. -1.4 2.1 C15H20… [M + … 0.7   
## 8 Dextrothyroxi… 1960 778. 1 4.7 C15H11… [M + … 0.99  
## 9 Hexanoylcarni… 1161 260. -6.6 3 C13H25… [M + … 0.91  
## 10 octanoylcarni… 1092 288. -5.2 4.1 C15H29… [M + … 0.98  
## # … with 1,076 more rows, and 5 more variables: `InChIKey planar` <chr>, `log2(FC)` <dbl>,  
## # `P-value` <dbl>, `Q-value` <dbl>, Class <chr>

Filtering our results based on the ranking of ‘features’ (top 25 of EFS and MWU) and identification by Wozniak et al.

origin\_top50 <- dplyr::select(origin\_top50,  
 oid = 1, EFS\_Rank,  
 MW\_Rank, Spectral\_Library\_Match  
)  
feas.otop <- dplyr::select(merged, .features\_id, oid)  
feas.otop <- merge(origin\_top50, feas.otop, all.x = T, by = "oid")  
feas.otop <- merge(feas.otop, features\_annotation(mcn2),  
 all.x = T,  
 by = ".features\_id"  
)  
feas.otop.format <- format\_table(feas.otop,  
 filter = NULL, select = c(  
 "oid",  
 "EFS\_Rank", "MW\_Rank", "Spectral\_Library\_Match", .select\_format  
 ),  
 export\_name = c(  
 oid = "# Original ID", EFS\_Rank = "# EFS\_Rank",  
 MW\_Rank = "# MW\_Rank", Spectral\_Library\_Match = "# Spectral\_Library\_Match",  
 .export\_name  
 )  
)  
write\_tsv(feas.otop.format, paste0(tmp, "/oTop50\_compounds\_format.tsv"))

## 5.6 Pathway enrichment

A plural number of chemical classes of interest were selected and the IDs of their features were obtained. These features were filtered with the statistic analysis data (Q value < 0.05).

focus <- c("Acyl carnitines", "Lysophosphatidylcholines", "Bile acids, alcohols and derivatives")  
focus <- select\_features(mcn, focus,  
 q.value = 0.05, logfc = 0.3,  
 coef = 1:2  
)

focus.key2d <- maps(feas, focus, ".features\_id", "inchikey2d")  
focus.cid <- lapply(focus.key2d, function(key2d) {  
 set <- key.set[names(key.set) %in% key2d]  
 unlist(lapply(set, function(df) df$CID), use.names = F)  
})  
keggids <- cid.to.kegg(unlist(focus.cid, use.names = F))  
focus.kegg <- maps(  
 keggids, lapply(focus.cid, as.character),  
 "Query", "KEGG"  
)

Let’s see what we get.

focus.kegg

## $`Acyl carnitines`  
## 11953814 11953821 439829   
## "C02838" "C03299" "C02862"   
##   
## $`Bile acids, alcohols and derivatives`  
## 53477753 42622727 53477907 22833540 23617285 6675   
## "C03033" "C03033" "C05462" "C05466" "C01921" "C05122"   
##   
## $Lysophosphatidylcholines  
## 460602 86555 497299 11757087 24779458 11005824 24779463 24779473 460604 52924053 52924051   
## "C04230" "C04102" "C04230" "C04230" "C04230" "C04230" "C04230" "C04230" "C04230" "C04230" "C04230"

Using package ‘FELLA’ for pathway enrichment analysis. The following step create a ‘database’ for enrichment (It is quite time consuming and can take up to 30 minutes). Subsequently, load the data.

db.dir <- init\_fella(tmp, "hsa")  
db.data <- load\_fella(db.dir)

Perform enrichment.

focus.enrich <- enrich\_fella(focus.kegg, db.data)  
focus.graph <- graph\_fella(focus.enrich, db.data, "pagerank")  
names(focus.graph) <- names(focus.kegg)

Some compounds are not present in the KEGG graph and are only background compounds. Let’s check the enrichment results.

!vapply(focus.graph, is.null, logical(1))

## Acyl carnitines Bile acids, alcohols and derivatives   
## FALSE TRUE   
## Lysophosphatidylcholines   
## TRUE

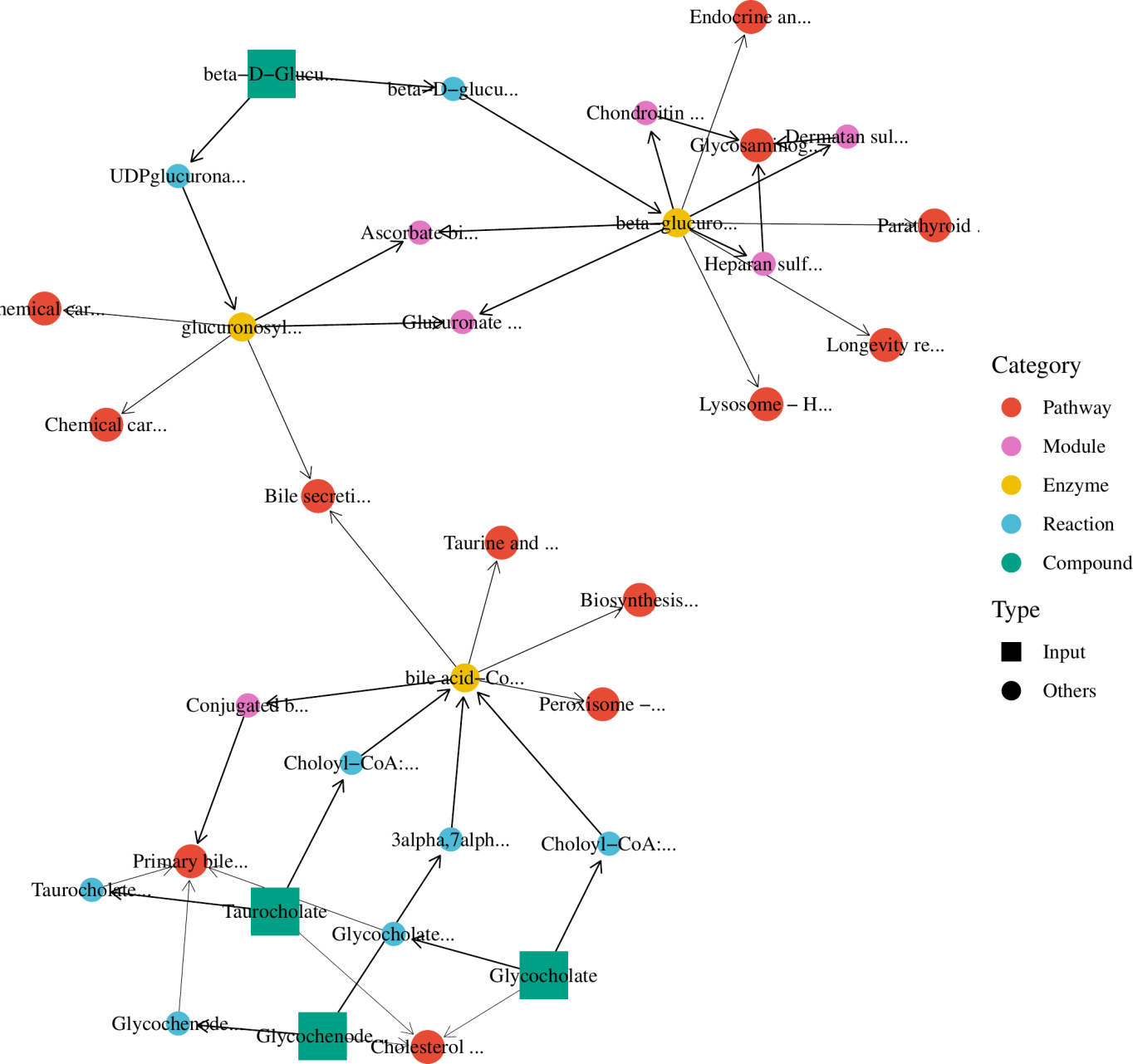


图7 Enrichment of Pagerank of BA compounds

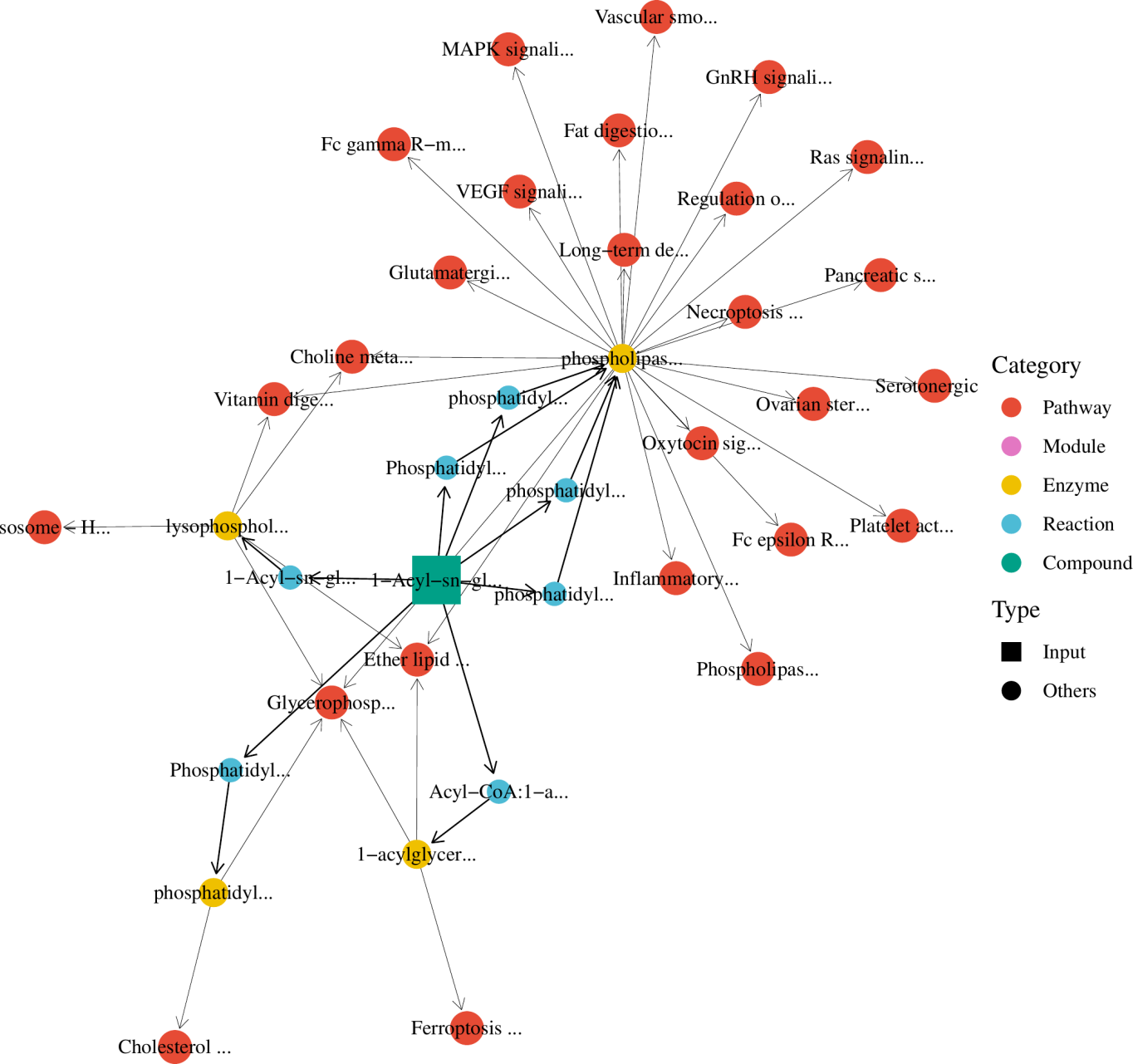


图8 Enrichment of Pagerank of LPC compounds

## 5.7 Heatmap analysis

Since the quantitative data obtained by merging contain many missing values, they need to be processed before plotting the heat map: For each subset of data, the missing values will be filled with the average value; if the set is all missing values, they will be filled with zero.

hp.data <- handling\_na(features\_quantification(mcn2), metadata = sample\_metadata(mcn2))

Convert wide data to long data; log transform the values; if there is a value 0, replace it with 1/10 of the minimum value of the value column.

hp.data <- log\_trans(hp.data)

Draw heat maps for three chemical classes.

hp.lst <- plot\_heatmap(focus, hp.data, metadata, pal\_group = palette\_stat(mcn2))  
ggsave(f12.31 <- paste0(tmp, "/ac\_heatmap.pdf"), hp.lst[[1]],  
 width = 13, height = 4  
)  
ggsave(f12.32 <- paste0(tmp, "/ba\_heatmap.pdf"), hp.lst[[2]],  
 width = 13, height = 7  
)  
ggsave(f12.33 <- paste0(tmp, "/lpc\_heatmap.pdf"), hp.lst[[3]],  
 width = 13, height = 4  
)

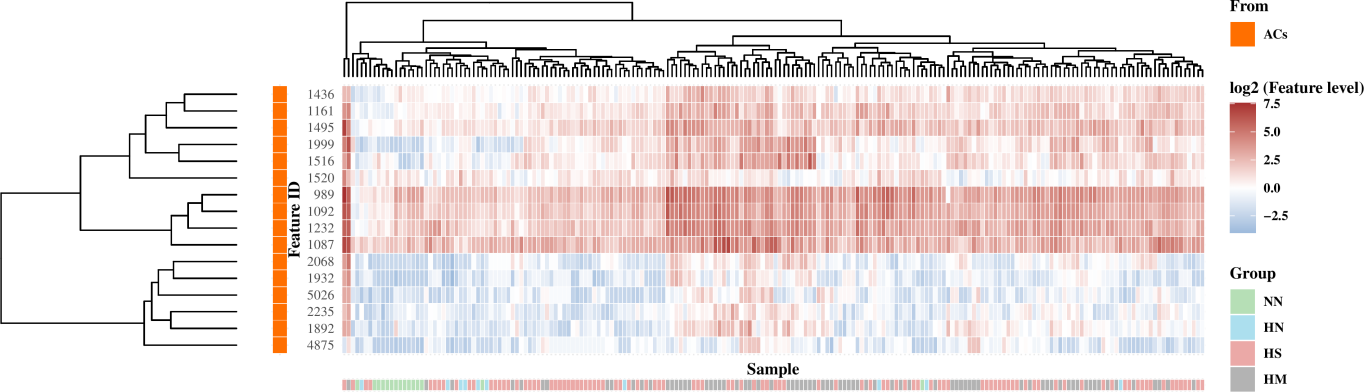


图9 Heatmap of ACs

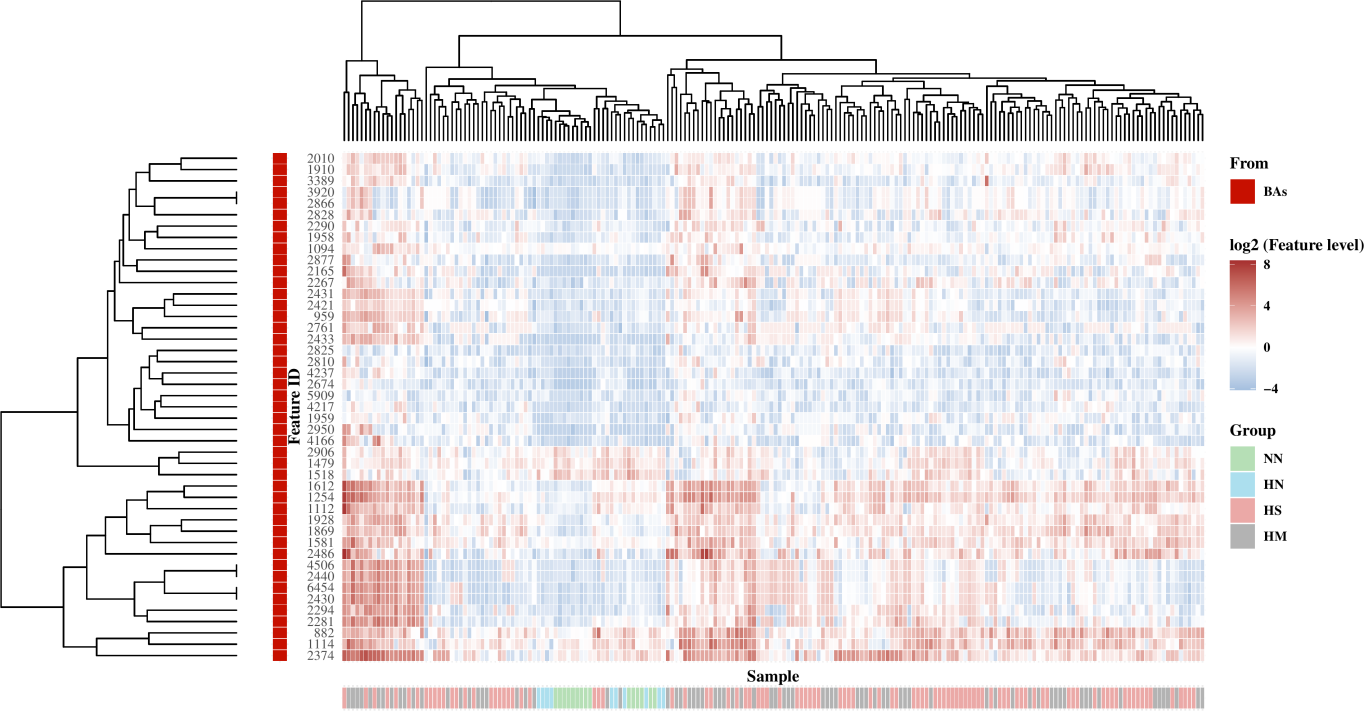


图10 Heatmap of BAs

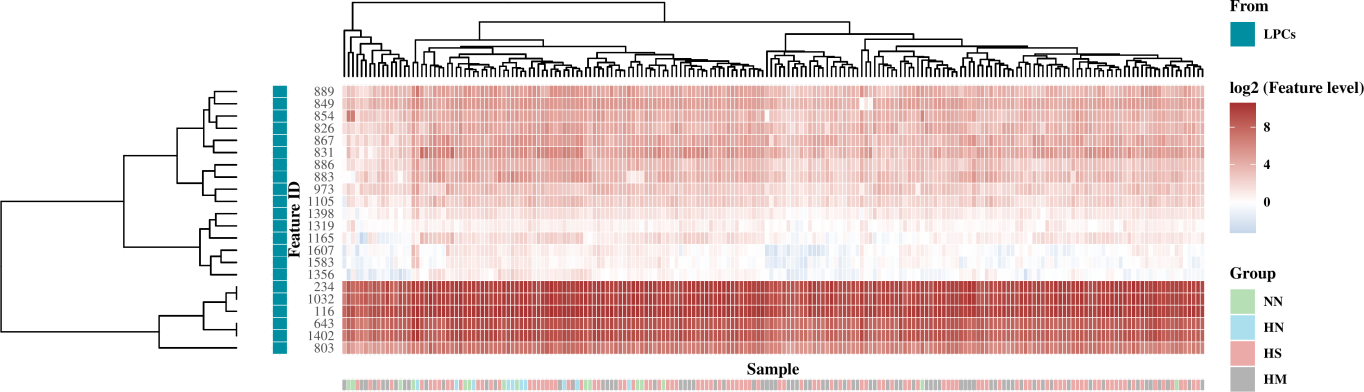


图11 Heatmap of LPCs

As shown in figure, Fig. [9](#acHp), [10](#baHp), and [11](#lpcHp) ACs and BAs implied a high correlation with disease, while LPCs showed a relatively weak correlation.

# 6 Verify Identification

In research of Wozniak et al, a subset of ACs compounds were identified. In addition, four top rank metabolites were identified. However, Most of the top 25 metabolites (EFS rank) were not identified.

ac\_names <- c(  
 "Palmitoyl-carnitine", "Octanoyl-carnitine", "Acetyl-carnitine",  
 "Hexanoyl-carnitine", "Decanoyl-carnitine"  
)  
ac\_inchikey2d <- c(  
 "XOMRRQXKHMYMOC", "CXTATJFJDMJMIY", "RDHQFKQIGNGIED",  
 "VVPRQWTYSNDTEA", "LZOSYCMHQXPBFU"  
)  
names(ac\_inchikey2d) <- ac\_names  
other\_4m <- c(  
 "Hepc", "D-erythro-Sphingosine-1-phosphate", "L-THYROXINE",  
 "Decanoyl-L-carnitine"  
)  
other\_4m\_inchikey2d <- c(  
 "BDPQVGIMLZYZQA", "DUYSYHSSBDVJSM",  
 "XUIIKFGFIJCVMT", "LZOSYCMHQXPBFU"  
)  
other\_4m.seq <- unlist(lapply(other\_4m, grep,  
 x = origin$Spectral\_Library\_Match,  
 ignore.case = T  
))  
other\_4m <- origin[other\_4m.seq, ]  
origin.top25 <- c(  
 349, 746, 854, 228, 320, 971, 2532, 670, 92,  
 1363, 13, 798, 1379, 1947, 4146, 736, 1656, 464, 731, 289,  
 4431, 3865, 476, other\_4m$Unique\_ID  
)  
origin.top25.featureID <- dplyr::filter(merged, oid %in% origin.top25)$.features\_id

Confirm which compounds were identified:

ac\_inchikey2d %in% features\_annotation(mcn2)$inchikey2d

## [1] TRUE TRUE TRUE TRUE TRUE

other\_4m\_inchikey2d %in% features\_annotation(mcn2)$inchikey2d

## [1] FALSE TRUE TRUE TRUE

reIdentify.origin.top25 <- dplyr::filter(  
 features\_annotation(mcn2),  
 .features\_id %in% origin.top25.featureID  
)  
reIdentify.origin.top25 <- dplyr::select(  
 reIdentify.origin.top25,  
 .features\_id, tani.score, inchikey2d  
)  
print(reIdentify.origin.top25, n = Inf)

## # A tibble: 21 × 3  
## .features\_id tani.score inchikey2d   
## <chr> <dbl> <chr>   
## 1 1011 0.93 AEHPOYAOLCAMIU  
## 2 1071 0.44 PRHHYVQTPBEDFE  
## 3 1104 0.85 SATGKQGFUDXGAX  
## 4 1184 0.45 RWXWCAZLEFJOFU  
## 5 1446 NA <NA>   
## 6 1671 0.4 XRDKWFXOXXUQJS  
## 7 1869 0.54 ONLXJASEXIXGRM  
## 8 1960 0.99 XUIIKFGFIJCVMT  
## 9 2045 0.41 OVEVHVURWWTPFC  
## 10 3333 0.51 RAJWOBJTTGJROA  
## 11 3590 NA <NA>   
## 12 4353 NA <NA>   
## 13 5344 NA <NA>   
## 14 6237 NA <NA>   
## 15 630 0.51 KPGYIOMDVKQYDK  
## 16 6646 0.19 RCDRLZYAKDYXMM  
## 17 803 0.84 IHNKQIMGVNPMTC  
## 18 886 1 RJZVWDTYEWCUAR  
## 19 952 0.99 DUYSYHSSBDVJSM  
## 20 981 0.39 OAUYENAPBFTAQT  
## 21 989 0.99 LZOSYCMHQXPBFU

# 7 Session infomation

sessionInfo()

## R version 4.2.1 (2022-06-23)  
## Platform: x86\_64-pc-linux-gnu (64-bit)  
## Running under: Pop!\_OS 22.04 LTS  
##   
## Matrix products: default  
## BLAS: /usr/lib/x86\_64-linux-gnu/blas/libblas.so.3.10.0  
## LAPACK: /usr/lib/x86\_64-linux-gnu/lapack/liblapack.so.3.10.0  
##   
## locale:  
## [1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C LC\_TIME=en\_US.UTF-8   
## [4] LC\_COLLATE=en\_US.UTF-8 LC\_MONETARY=en\_US.UTF-8 LC\_MESSAGES=en\_US.UTF-8   
## [7] LC\_PAPER=en\_US.UTF-8 LC\_NAME=C LC\_ADDRESS=C   
## [10] LC\_TELEPHONE=C LC\_MEASUREMENT=en\_US.UTF-8 LC\_IDENTIFICATION=C   
##   
## attached base packages:  
## [1] grid stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] utils.tool\_0.0.0.9000 FELLA\_1.16.0 MCnebula2\_0.0.9000 ggplot2\_3.3.6   
## [5] nvimcom\_0.9-142   
##   
## loaded via a namespace (and not attached):  
## [1] uuid\_1.1-0 systemfonts\_1.0.4 plyr\_1.8.7   
## [4] igraph\_1.3.2 lazyeval\_0.2.2 ChemmineOB\_1.34.0   
## [7] usethis\_2.1.6 GenomeInfoDb\_1.32.2 digest\_0.6.29   
## [10] yulab.utils\_0.0.4 htmltools\_0.5.2 viridis\_0.6.2   
## [13] magick\_2.7.3 fansi\_1.0.3 magrittr\_2.0.3   
## [16] memoise\_2.0.1 grImport2\_0.2-0 remotes\_2.4.2   
## [19] Biostrings\_2.64.0 graphlayouts\_0.8.0 officer\_0.6.0   
## [22] svglite\_2.1.0 askpass\_1.1 gfonts\_0.2.0   
## [25] prettyunits\_1.1.1 jpeg\_0.1-9 colorspace\_2.0-3   
## [28] ggrepel\_0.9.1 xfun\_0.31 dplyr\_1.0.9   
## [31] callr\_3.7.0 crayon\_1.5.2 RCurl\_1.98-1.7   
## [34] jsonlite\_1.8.0 officedown\_0.3.0 ape\_5.6-2   
## [37] glue\_1.6.2 polyclip\_1.10-0 rvg\_0.3.2   
## [40] gtable\_0.3.0 zlibbioc\_1.42.0 XVector\_0.36.0   
## [43] pkgbuild\_1.3.1 BiocGenerics\_0.42.0 fontquiver\_0.2.1   
## [46] scales\_1.2.0 DBI\_1.1.2 Rcpp\_1.0.8.3   
## [49] xtable\_1.8-4 viridisLite\_0.4.0 gridtext\_0.1.4   
## [52] gridGraphics\_0.5-1 tidytree\_0.3.9 fontLiberation\_0.1.0   
## [55] stats4\_4.2.1 rsvg\_2.3.1 httr\_1.4.3   
## [58] ellipsis\_0.3.2 pkgconfig\_2.0.3 XML\_3.99-0.10   
## [61] farver\_2.1.0 crul\_1.3 utf8\_1.2.2   
## [64] later\_1.3.0 ggplotify\_0.1.0 tidyselect\_1.2.0   
## [67] rlang\_1.0.6 munsell\_0.5.0 tools\_4.2.1   
## [70] cachem\_1.0.6 cli\_3.5.0 generics\_0.1.2   
## [73] devtools\_2.4.3 evaluate\_0.15 stringr\_1.4.0   
## [76] fastmap\_1.1.0 yaml\_2.3.5 ggtree\_3.4.0   
## [79] processx\_3.6.0 knitr\_1.39 fs\_1.5.2   
## [82] tidygraph\_1.2.1 zip\_2.2.0 purrr\_0.3.4   
## [85] KEGGREST\_1.36.2 ggraph\_2.0.5 pbapply\_1.5-0   
## [88] nlme\_3.1-159 mime\_0.12 aplot\_0.1.6   
## [91] xml2\_1.3.3 BiocStyle\_2.24.0 brio\_1.1.3   
## [94] compiler\_4.2.1 rstudioapi\_0.13 curl\_4.3.2   
## [97] png\_0.1-7 testthat\_3.1.4 gt\_0.6.0   
## [100] treeio\_1.20.0 tibble\_3.1.7 tweenr\_1.0.2   
## [103] stringi\_1.7.6 ggimage\_0.3.1 highr\_0.9   
## [106] ps\_1.7.0 desc\_1.4.1 gdtools\_0.3.1   
## [109] lattice\_0.20-45 Matrix\_1.5-1 fontBitstreamVera\_0.1.1  
## [112] ggsci\_2.9 vctrs\_0.5.1 pillar\_1.7.0   
## [115] lifecycle\_1.0.3 BiocManager\_1.30.18 data.table\_1.14.2   
## [118] bitops\_1.0-7 httpuv\_1.6.5 patchwork\_1.1.1   
## [121] R6\_2.5.1 promises\_1.2.0.1 bookdown\_0.27   
## [124] gridExtra\_2.3 IRanges\_2.30.0 sessioninfo\_1.2.2   
## [127] MASS\_7.3-58 assertthat\_0.2.1 pkgload\_1.2.4   
## [130] openssl\_2.0.2 rprojroot\_2.0.3 httpcode\_0.3.0   
## [133] withr\_2.5.0 S4Vectors\_0.34.0 GenomeInfoDbData\_1.2.8   
## [136] parallel\_4.2.1 ggtext\_0.1.1 ggfun\_0.0.6   
## [139] tidyr\_1.2.0 rmarkdown\_2.14 ggforce\_0.3.3   
## [142] shiny\_1.7.1 base64enc\_0.1-3

# Reference

1. Dührkop, K. *et al.* [SIRIUS 4: A rapid tool for turning tandem mass spectra into metabolite structure information](https://doi.org/10.1038/s41592-019-0344-8). *Nature Methods* **16**, 299–302 (2019).

2. Böcker, S., Letzel, M. C., Lipták, Z. & Pervukhin, A. [SIRIUS: Decomposing isotope patterns for metabolite identification](https://doi.org/10.1093/bioinformatics/btn603). *Bioinformatics* **25**, 218–224 (2009).

3. Dührkop, K., Shen, H., Meusel, M., Rousu, J. & Böcker, S. [Searching molecular structure databases with tandem mass spectra using CSI:FingerID](https://doi.org/10.1073/pnas.1509788112). *Proceedings of the National Academy of Sciences* **112**, 12580–12585 (2015).

4. Ludwig, M. *et al.* [Database-independent molecular formula annotation using Gibbs sampling through ZODIAC](https://doi.org/10.1038/s42256-020-00234-6). *Nature Machine Intelligence* **2**, 629–641 (2020).

5. Dührkop, K. *et al.* [Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra](https://doi.org/10.1038/s41587-020-0740-8). *Nature Biotechnology* **39**, 462–471 (2021).

6. Smyth, G. K. Limma: Linear Models for Microarray Data. in *Bioinformatics and Computational Biology Solutions Using R and Bioconductor* (eds. Gentleman, R., Carey, V. J., Huber, W., Irizarry, R. A. & Dudoit, S.) 397–420 (Springer-Verlag, 2005). doi:[10.1007/0-387-29362-0\_23](https://doi.org/10.1007/0-387-29362-0_23).