Analysis on *E. ulmoides* dataset

Contents

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 MS^1 data. Each feature may represents a compound, and assigned with MS^2 spectra. The MS^2 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.

Eucommia ulmoides Oliv. (*E. ulmoides*), as a traditional Chinese medicine (TCM), after being processed with saline water, was applied to the treatment of renal diseases for a long time in China. Due to its complex composition, discovering chemical changes during processing (such as processed with saline water) is challenging. We would next demonstrate the addressing of this challenge with MCnebula, which may be enlightening for the study of phytopharmaceuticals.

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) <- "neg"
```

Create a temporary folder to store the output data.

```
tmp <- paste0(tempdir(), "/temp_data")
dir.create(tmp, F)
```

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 1612 features with chemical formula identification.

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

Load the '.rdata' file.

```
load(paste0(exfiles, "/mcn_herbal1612.rdata"))
mcn <- mcn_herbal1612
rm(mcn_herbal1612)
export_path(mcn) <- tmp
```

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 = 0.1, 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] "Lactones"
##  [2] "Ketones"
##  [3] "Benzoic acids and derivatives"
##  [4] "Methoxyphenols"
##  [5] "Sugar acids and derivatives"
##  [6] "Phenylpropanoids and polyketides"
##  [7] "Benzoyl derivatives"
##  [8] "Fatty acid esters"
##  [9] "Unsaturated fatty acids"
## [10] "Hydroxy fatty acids"
## [11] "Hydroxy acids and derivatives"
## [12] "Lineolic acids and derivatives"
## [13] "Dialkyl ethers"
## [14] "Cyclic alcohols and derivatives"
## [15] "Benzoic acid esters"
## [16] "Alpha hydroxy acids and derivatives"
## [17] "Lignans, neolignans and related compounds"
## [18] "Lignan glycosides"
## [19] "Disaccharides"
## [20] "Monoterpenoids"
## [21] "Bicyclic monoterpenoids"
## [22] "Iridoids and derivatives"
## [23] "Tertiary alcohols"
## [24] "Beta hydroxy acids and derivatives"
## [25] "Fatty acyl glycosides"
## [26] "Furofurans"
## [27] "Furofuran lignans"
## [28] "Terpene glycosides"
## [29] "Alkyl glycosides"
## [30] "Methoxybenzoic acids and derivatives"
## [31] "Long-chain fatty acids"
## [32] "Medium-chain fatty acids"
## [33] "Furanoid lignans"
## [34] "Fatty acyl glycosides of mono- and disaccharides"
## [35] "Vinylogous esters"
## [36] "Iridoid O-glycosides"
## [37] "Dimethoxybenzenes"
```

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

```
## [1] "Fatty acid esters"                  "Unsaturated fatty acids"
```

```
## [3] "Hydroxy fatty acids"                  "Long-chain fatty acids"
## [5] "Medium-chain fatty acids"             "Hydroxy fatty acids"
## [7] "Hydroxy acids and derivatives"        "Alpha hydroxy acids and derivatives"
## [9] "Beta hydroxy acids and derivatives"
```

Remove these classes.

```
mcn <- backtrack_stardust(mcn, dis, remove = T)
```

4.4 Create Nebulae

Create Nebula-Index data. This data created based on 'stardust_classes' data.

```
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: 29 x 3
##      seq hierarchy class.name
##    <int>     <dbl> <chr>
##  1     1         5 Alkyl glycosides
##  2     2         5 Benzoic acid esters
##  3     3         4 Benzoic acids and derivatives
##  4     4         4 Benzoyl derivatives
##  5     5         5 Bicyclic monoterpenoids
##  6     6         5 Cyclic alcohols and derivatives
##  7     7         5 Dialkyl ethers
##  8     8         5 Dimethoxybenzenes
##  9     9         5 Disaccharides
## 10    10         4 Fatty acyl glycosides
## # ... with 19 more rows
```

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

```
p <- visualize(mcn, "parent")
ggsave(f5.61 <- paste0(tmp, "/parent_nebula.png"), p)
pdf(f5.62 <- paste0(tmp, "/child_nebula.pdf"), 12, 14)
visualize_all(mcn)
dev.off()
```

In general, Parent-Nebulae (Fig. 1) is too informative to show, so Child-Nebulae (Fig. 2) 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 Nebulae for Downstream analysis

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, "/herbal.tar.gz"), exdir = tmp)
origin <- data.table::fread(paste0(tmp, "/features.csv"))
origin <- tibble::as_tibble(origin)
```

Now, let's check the columns in the table.

```
origin
```

```
## # A tibble: 2,579 x 25
##    `row ID` `row m/z` `row retention time` `EU-BlANK.mzML Peak ~` `EU-BlANK.mzML~` `EU-BlANK.mzML~`
##       <int>     <dbl>                <dbl>                  <dbl>            <dbl>            <dbl>
##  1        1      591.                 15.7                   15.6             15.8       175173005.
##  2        2      194.                 22.4                   22.2             22.5       142181248.
##  3        3      635.                 16.1                   16.1             16.3       133667589.
##  4        4      547.                 15.0                   14.9             15.1       127952206.
##  5        5      250.                 24.8                   24.8             24.9        93773479.
##  6        6      299.                 24.9                   24.7             25.1       184652820.
##  7        7      293.                 22.1                   22.0             22.2        98564860.
##  8        8      242.                 20.5                   20.6             20.7         1671972.
##  9        9      293.                 22.8                   22.6             22.8        74335303.
## 10       10      191.                 22.2                   22.0             22.3        93057814.
## # ... with 2,569 more rows, and 19 more variables: `EU-Raw3.mzML Peak RT start` <dbl>,
## #   `EU-Raw3.mzML Peak RT end` <dbl>, `EU-Raw3.mzML Peak area` <dbl>,
## #   `EU-Pro2.mzML Peak RT start` <dbl>, `EU-Pro2.mzML Peak RT end` <dbl>,
## #   `EU-Pro2.mzML Peak area` <dbl>, `EU-Pro3.mzML Peak RT start` <dbl>,
## #   `EU-Pro3.mzML Peak RT end` <dbl>, `EU-Pro3.mzML Peak area` <dbl>,
## #   `EU-Raw1.mzML Peak RT start` <dbl>, `EU-Raw1.mzML Peak RT end` <dbl>,
## #   `EU-Raw1.mzML Peak area` <dbl>, `EU-Raw2.mzML Peak RT start` <dbl>, ...
```

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

```
quant <- dplyr::select(origin, id = 1, dplyr::contains("Peak area"))
colnames(quant) <- gsub("\\.mzML Peak area", "", colnames(quant))
quant <- dplyr::mutate(quant, .features_id = as.character(id))
```

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

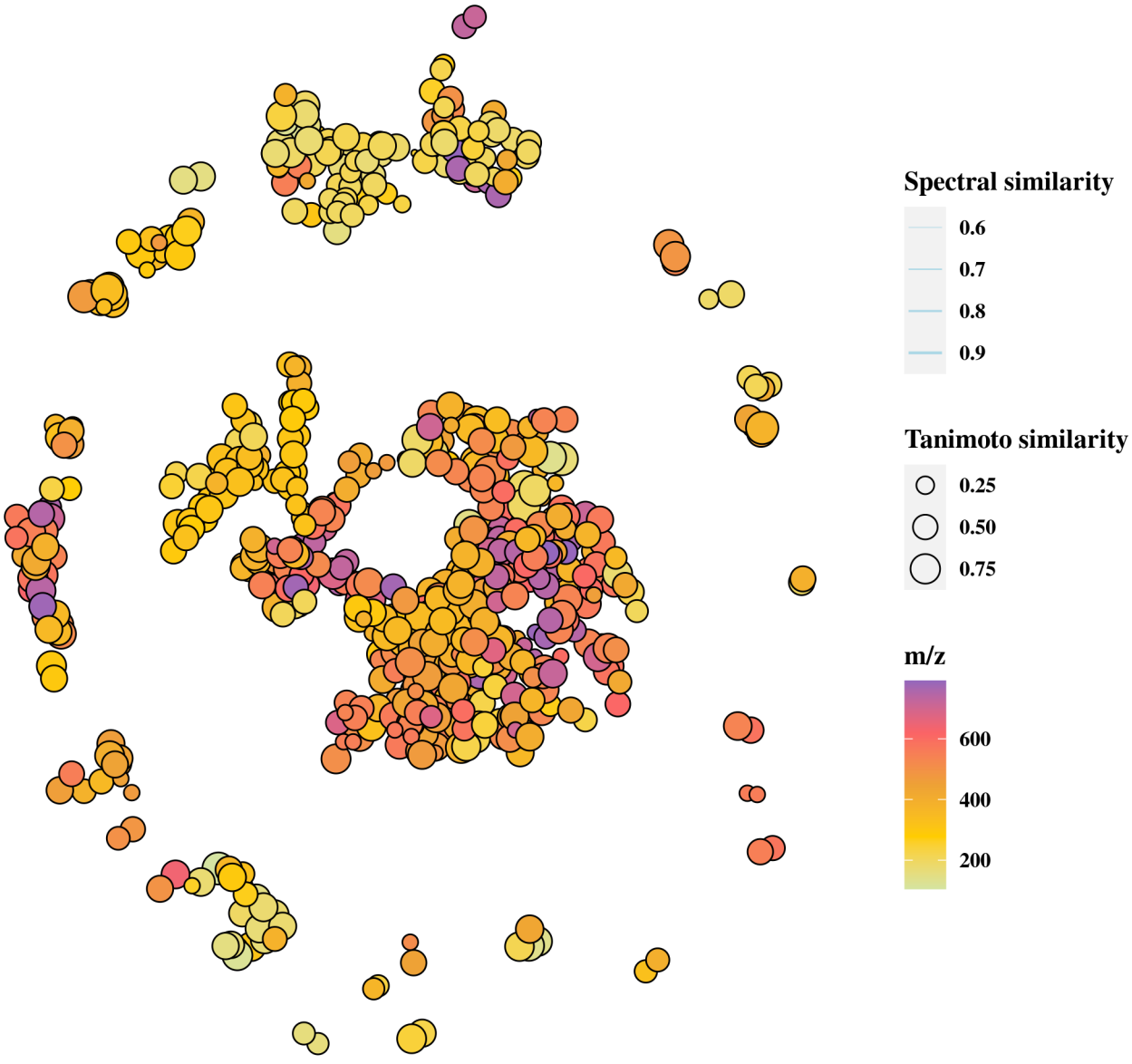

Figure 1: Parent-Nebula

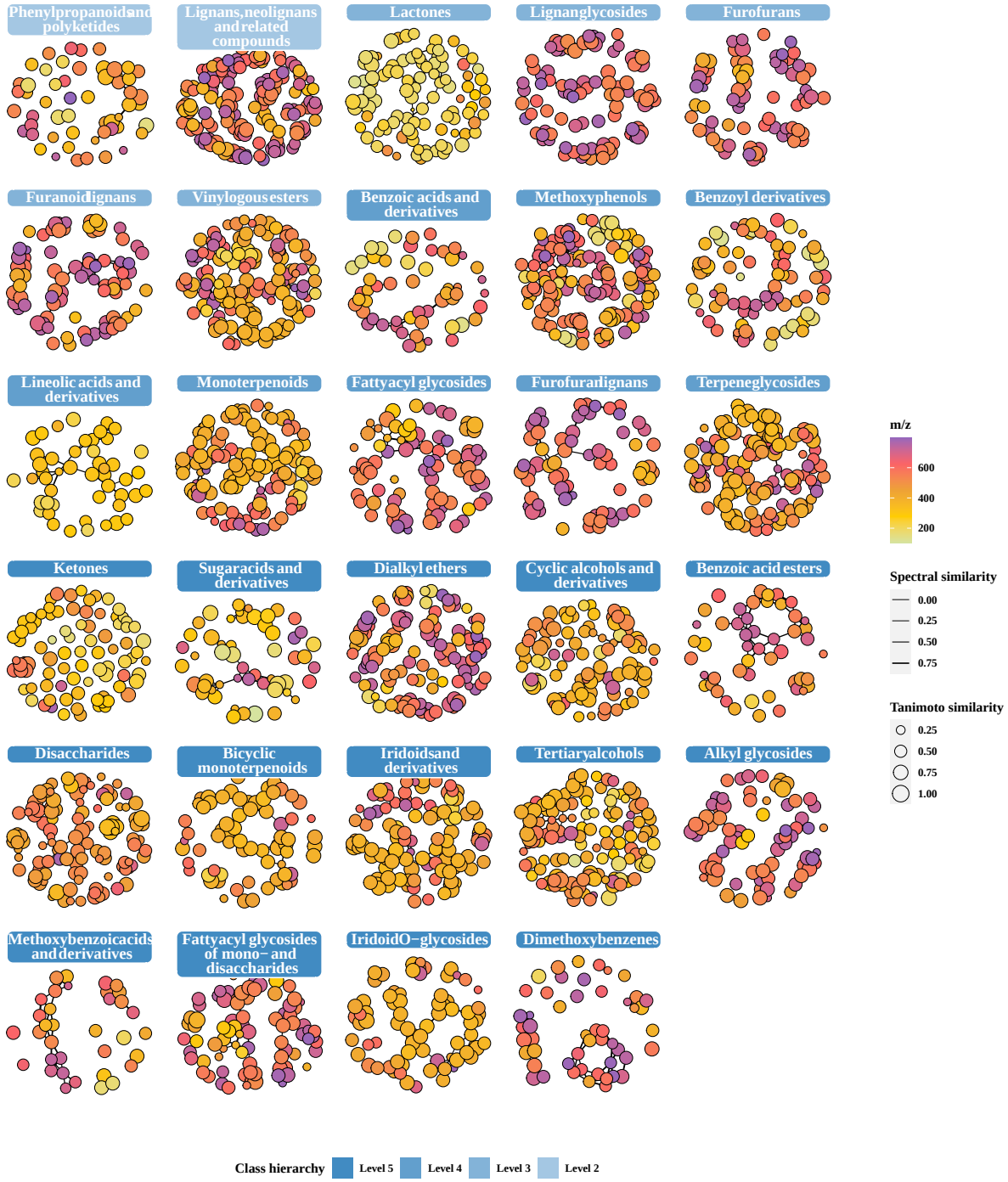

Figure 2: Child-Nebulae

```
gp <- c(Blank = "EU-BlANK", Raw = "EU-Raw", Pro = "EU-Pro")
metadata <- MCnebula2:::group_strings(colnames(quant), gp, "sample")
metadata$annotation <- vapply(metadata$group, switch,
  FUN.VALUE = character(1),
  Blank = "methanol/water (1:1, v/v)", Raw = "Raw bark of Eucommia ulmoides Oliv.",
  Pro = "Precessed (with saline water) bark of Eucommia ulmoides Oliv."
)
features_quantification(mcn) <- dplyr::select(quant, -id)
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' package[6] 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. To accommodate the downstream analysis of gene expression that the `limma` package was originally used for, we should log2-transform and centralize this data (use the default parameter 'fun_norm' of `binary_comparison()`).

```
mcn <- binary_comparison(mcn, Pro - Raw)
top.list <- top_table(statistic_set(mcn))
```

Check the results.

```
top.list[[1]]
```

```
## # A tibble: 2,579 x 7
##    .features_id logFC AveExpr     t  P.Value     adj.P.Val     B
##    <chr>        <dbl>   <dbl> <dbl>    <dbl>         <dbl> <dbl>
##  1 1623          24.9   -9.42  196. 1.96e-11 0.00000000708  10.6
##  2 1980          24.1   -9.75  192. 2.19e-11 0.00000000708  10.5
##  3 353          -25.8   -9.47 -191. 2.21e-11 0.00000000708  10.5
##  4 2318          23.8   -9.91  190. 2.30e-11 0.00000000708  10.5
##  5 1977          24.1   -9.74  189. 2.37e-11 0.00000000708  10.5
##  6 1629          24.2   -9.71  187. 2.53e-11 0.00000000708  10.5
##  7 1644          23.9   -9.86  182. 2.88e-11 0.00000000708  10.5
##  8 1641          24.0   -9.82  181. 2.94e-11 0.00000000708  10.5
##  9 2323          23.2  -10.2   181. 2.96e-11 0.00000000708  10.5
## 10 2339          23.2  -10.1   180. 3.03e-11 0.00000000708  10.5
## # ... with 2,569 more rows
```

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 <- 20
tops <- select_features(mcn2,
  tani.score_cutoff = 0.5, order_by_coef = 1,
  togather = T
)
top20 <- tops[1:n]
palette_set(melody(mcn)) <- colorRampPalette(palette_set(mcn))(n)
```

```
mcn2 <- set_tracer(mcn, top20)
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(f8.2 <- paste0(tmp, "/tracer_child_nebula.pdf"), 12, 14)
visualize_all(mcn2)
dev.off()
```

A part of the top features are marked with colored nodes in Child-Nebulae (Fig. 3).

5.3 Quantification in Child-Nebulae

Show Fold Change (Pro versus Raw) in Child-Nebulae.

```
palette_gradient(melody(mcn2)) <- c("blue", "grey90", "red")
mcn2 <- set_nodes_color(mcn2, "logFC", top.list[[1]])
pdf(f9.1 <- paste0(tmp, "/logFC_child_nebula.pdf"), 12, 14)
visualize_all(mcn2, fun_modify = modify_stat_child)
dev.off()
```

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

5.4 Annotate Nebulae

Now, the available Nebulae contains:

```
table.nebulae2 <- visualize(mcn2)
```

```
## [INFO] MCnebula2: visualize
```

```
##  Specify item as following to visualize:
```

```
print(table.nebulae2, n = Inf)
```

```
## # A tibble: 22 x 3
##      seq hierarchy class.name
##    <int>     <dbl> <chr>
##  1     1         5 Alkyl glycosides
##  2     2         5 Bicyclic monoterpenoids
##  3     3         5 Cyclic alcohols and derivatives
##  4     4         5 Dialkyl ethers
##  5     5         5 Disaccharides
##  6     6         4 Fatty acyl glycosides
##  7     7         5 Fatty acyl glycosides of mono- and disaccharides
##  8     8         3 Furanoid lignans
##  9     9         4 Furofuran lignans
## 10    10         3 Furofurans
## 11    11         5 Iridoid O-glycosides
## 12    12         5 Iridoids and derivatives
## 13    13         3 Lactones
## 14    14         3 Lignan glycosides
## 15    15         2 Lignans, neolignans and related compounds
## 16    16         4 Methoxyphenols
```

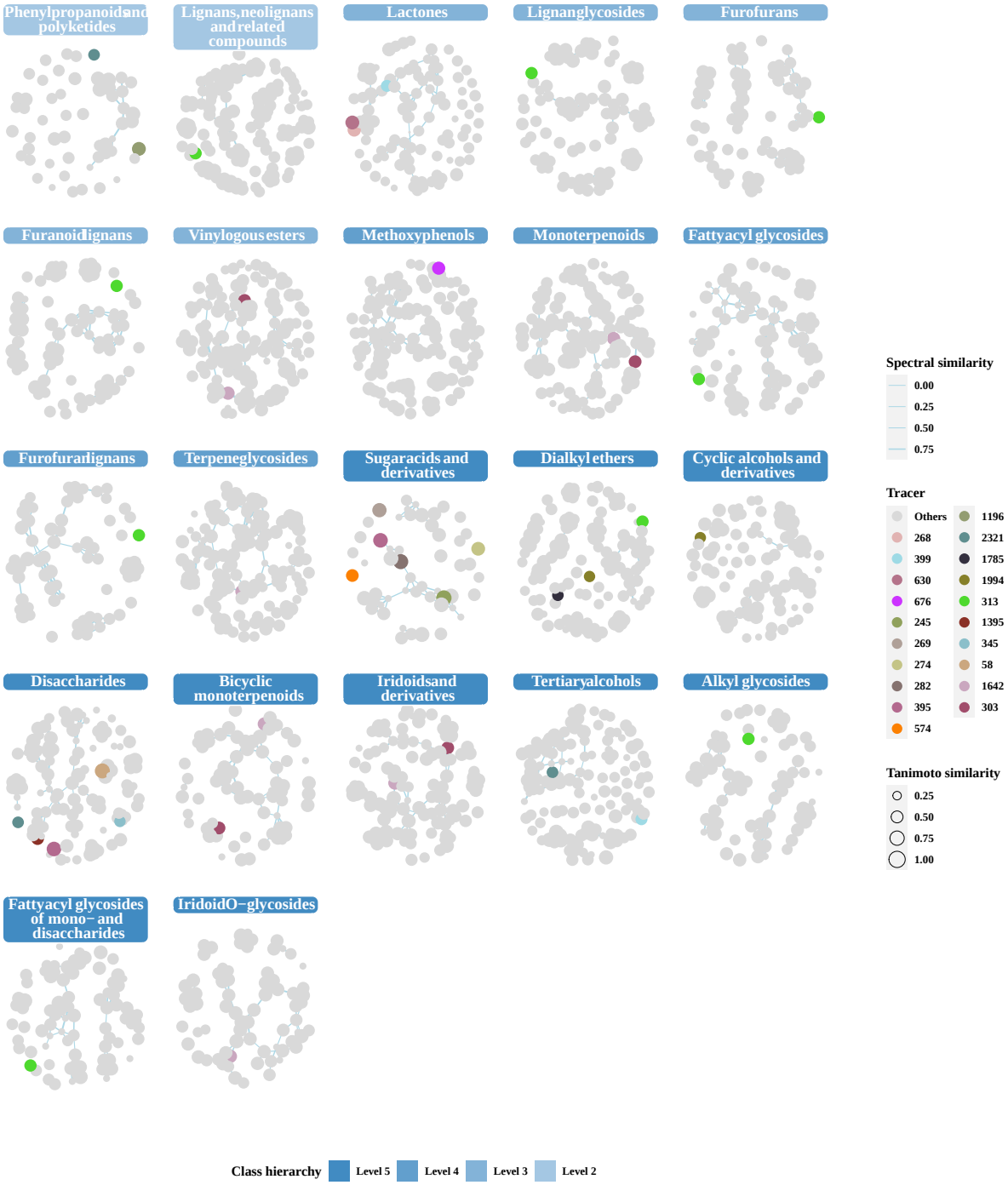

Figure 3: Tracing top features in Child-Nebulae

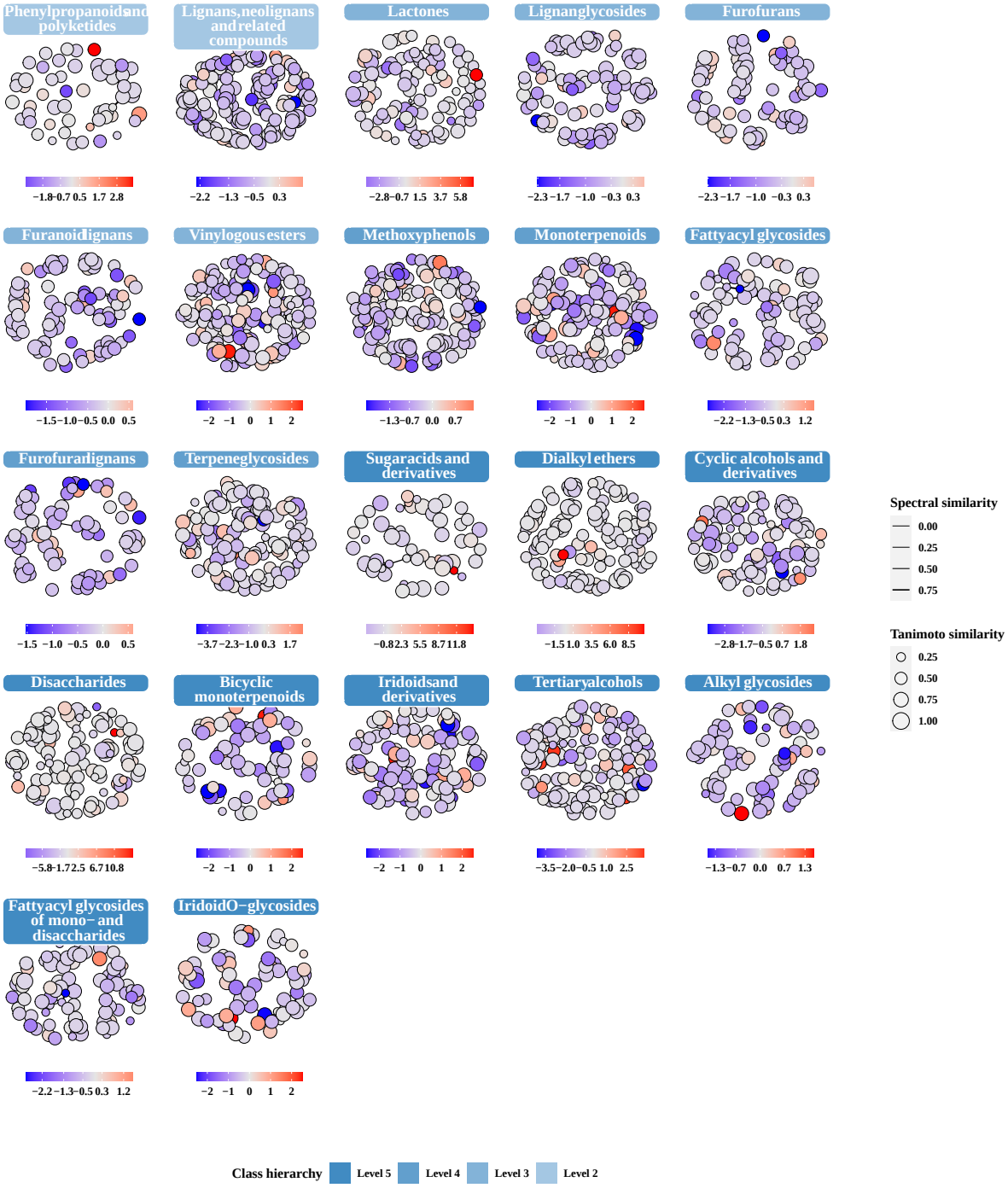

Figure 4: Show log2(FC) in Child-Nebulae

```
## 17    17         4 Monoterpenoids
## 18    18         2 Phenylpropanoids and polyketides
## 19    19         5 Sugar acids and derivatives
## 20    20         4 Terpene glycosides
## 21    21         5 Tertiary alcohols
## 22    22         3 Vinylogous esters
```

Next, let us focus on 'Lignans, neolignans and related compounds' and 'Iridoids and derivatives'. They were representative chemical classes in E. ulmoides.

```
mcn2 <- set_nodes_color(mcn2, use_tracer = T)
palette_stat(melody(mcn2)) <- c(
  Pro = "#EBA9A7", Raw = "#ACDFEE",
  Blank = "grey80"
)
lig <- "Lignans, neolignans and related compounds"
iri <- "Iridoids and derivatives"
mcn2 <- annotate_nebula(mcn2, lig)
mcn2 <- annotate_nebula(mcn2, iri)
```

Draw and save the image.

```
p <- visualize(mcn2, lig, annotate = T)
ggsave(f10.2.1 <- paste0(tmp, "/lig_child.pdf"), p,
  width = 6,
  height = 4
)
p <- visualize(mcn2, iri, annotate = T)
ggsave(f10.2.2 <- paste0(tmp, "/iri_child.pdf"), p,
  width = 6,
  height = 4
)
```

See results (Fig. 5 and 6).

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 <- "1642"
pdf(f10.4 <- paste0(tmp, "/features_", ef, ".pdf"), 10, 6)
show_node(mcn2, ef)
dev.off()
```

See results (Fig. 7) (feature in 6) .

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 spectra. 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.

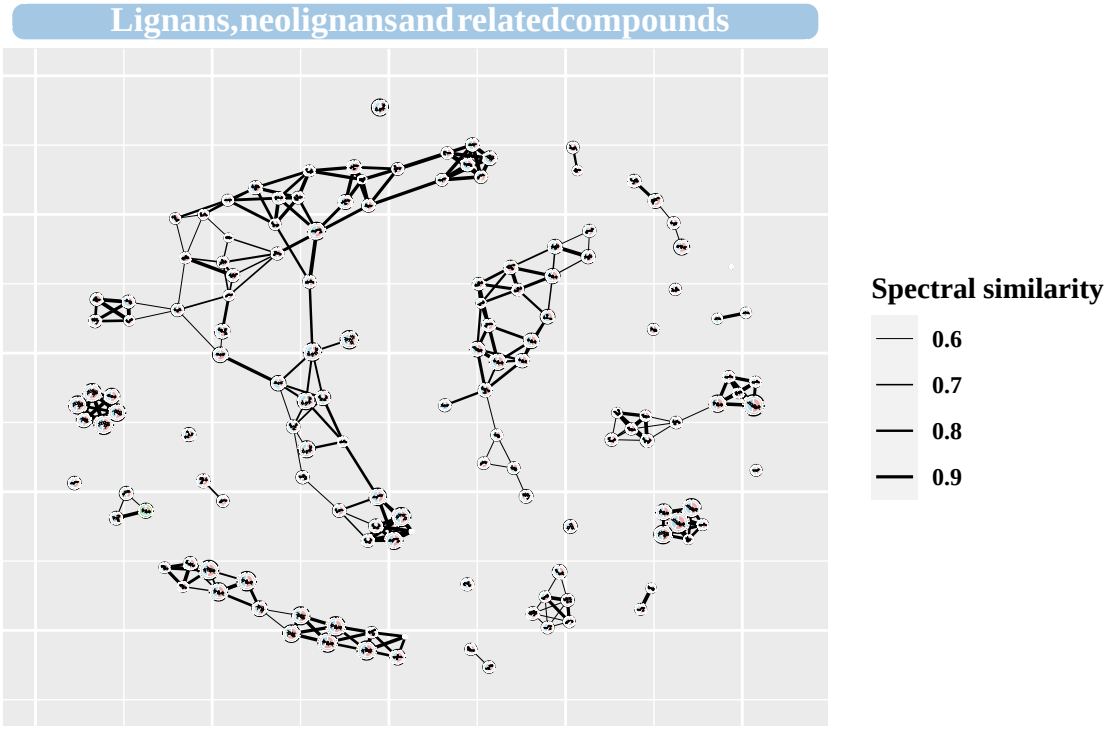

Figure 5: Annotated Nebulae: Lignans, neolignans and related compounds

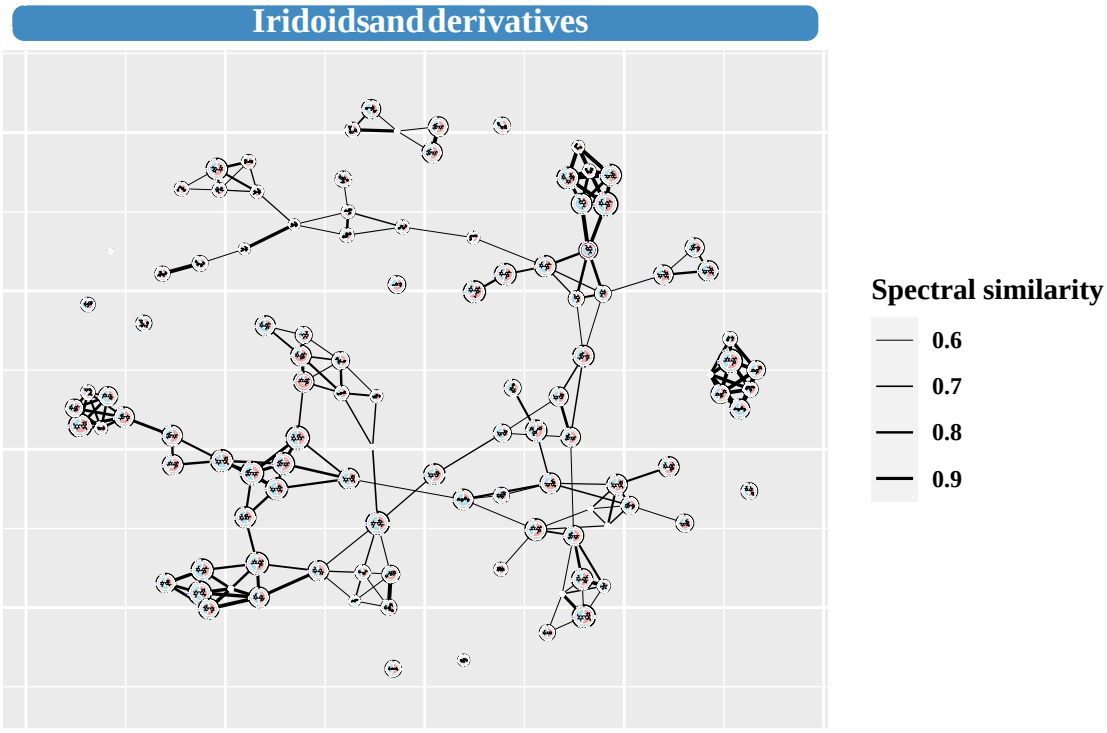

Figure 6: Annotated Nebulae: Iridoids and derivatives

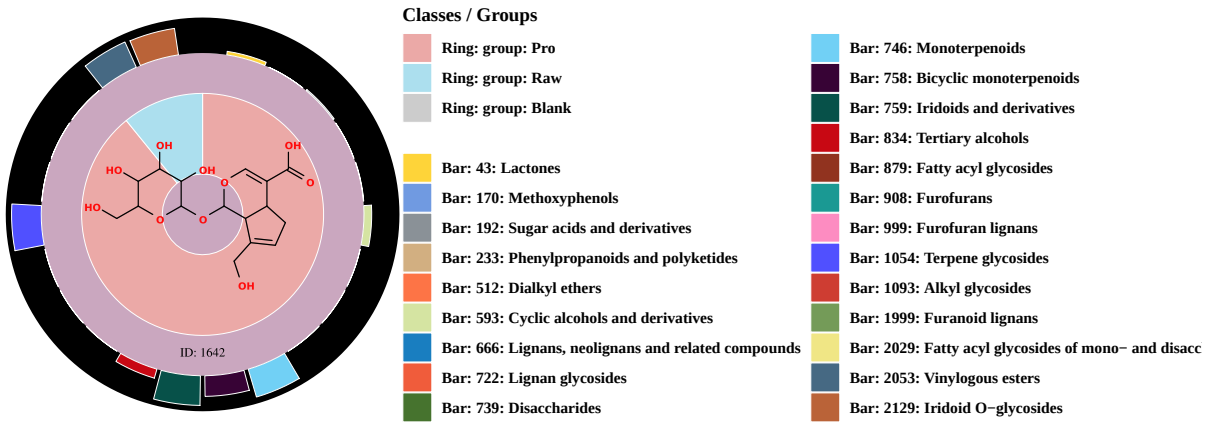

Figure 7: The annotated feature of ID: 1642

```
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 multi-threading 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: 564 x 13
##    Synonym        ID    `Precursor m/z` `Mass error (p~` `RT (min)` Formula Adduct `Tanimoto simi~`
##    <chr>          <chr>           <dbl>            <dbl>      <dbl> <chr>   <chr>              <dbl>
##  1 ethyl-dUTP     1644             495.              2.6        1   C11H19~ [M - ~              0.67
##  2 8-Aminoadenos~ 2339             555.              9.8        1.1 C15H22~ [M - ~              0.57
##  3 D-sulfolactate 288              169.             -7.2        1   C3H6O6S [M - ~              0.81
##  4 (1beta,4beta)~ 399              217.             -3.1        1.5 C8H10O7 [M - ~              0.56
##  5 3-[4-(beta-D-~ 1655             327.             -1.3        9.2 C15H20~ [M - ~              0.71
##  6 3,4-Dihydroxy~ 1639             131.             -8.8        1.7 C5H8O4  [M - ~              0.6
##  7 AltA           268              193.             -5.1        1   C6H10O7 [M - ~              0.7
##  8 dec-6-enoate   482              169.             -6.8       21.2 C10H18~ [M - ~              0.53
##  9 Geniposidinsa~ 1642             373.             -1.2        9.2 C16H22~ [M - ~              0.69
## 10 4-O-methyl-d-~ 274              223.             -4.2        1   C7H12O8 [M - ~              0.75
## # ... with 554 more rows, and 5 more variables: `InChIKey planar` <chr>, `log2(FC)` <dbl>,
## #   `P-value` <dbl>, `Q-value` <dbl>, Class <chr>
```

5.6 Plot MS/MS spectra of top 'features'

Drawing of MS/MS spectra of top 'features'.

```
mcn2 <- draw_structures(mcn2, .features_id = top20)
pdf(f12.1 <- paste0(tmp, "/msms_tops_identified.pdf"), 12, 10)
plot_msms_mirrors(mcn2, top20)
dev.off()
```

See results (Fig. 8).

Plot EIC spectra of top 'features'. (Since the code below requires .mzML mass spectrometry data, these files are too large to be stored in a package, so the code below will not be run. But we have saved the result data.)

```
metadata$file <- paste0(metadata$sample, ".mzML")
data <- plot_EIC_stack(top20, metadata,
  quant.path = paste0(
    tmp,
    "/features.csv"
  ), mzml.path = "/media/echo/back/thermo_mzML_0518/",
  palette = palette_stat(mcn2)
)
save(data, file = paste0(tmp, "/eic_data.rdata"))
```

Load the saved data and draw the figure.

```
load(paste0(tmp, "/eic_data.rdata"))
pdf(f12.6 <- paste0(tmp, "/eic_tops_identified.pdf"), 12, 10)
print(data$p)
dev.off()
```

Or use following to re-plot the 'ggplot' object.

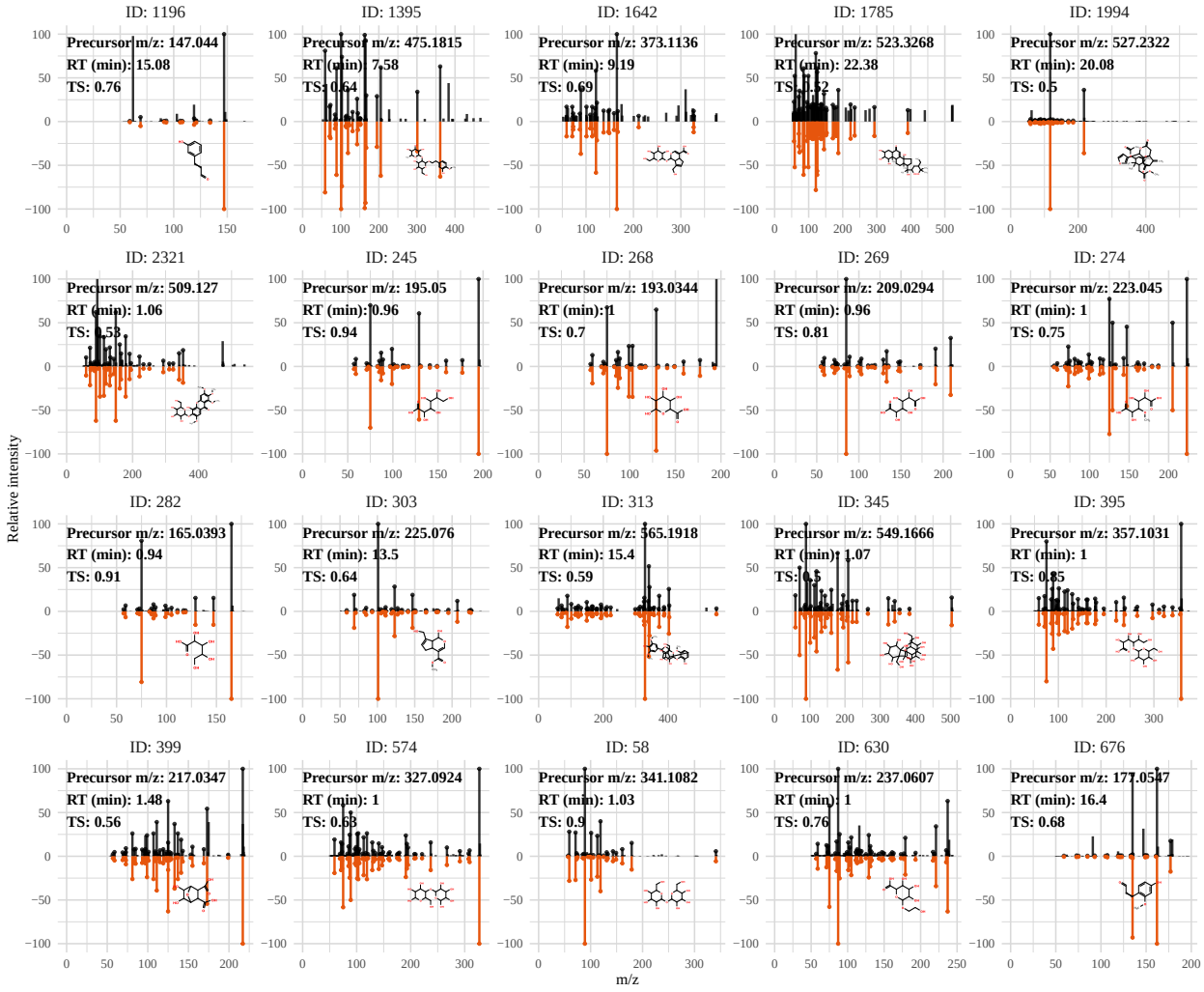

Figure 8: MS/MS spectra of top features (identified)

```
data <- plot_EIC_stack(data = data, palette = palette_stat(mcn2))
```

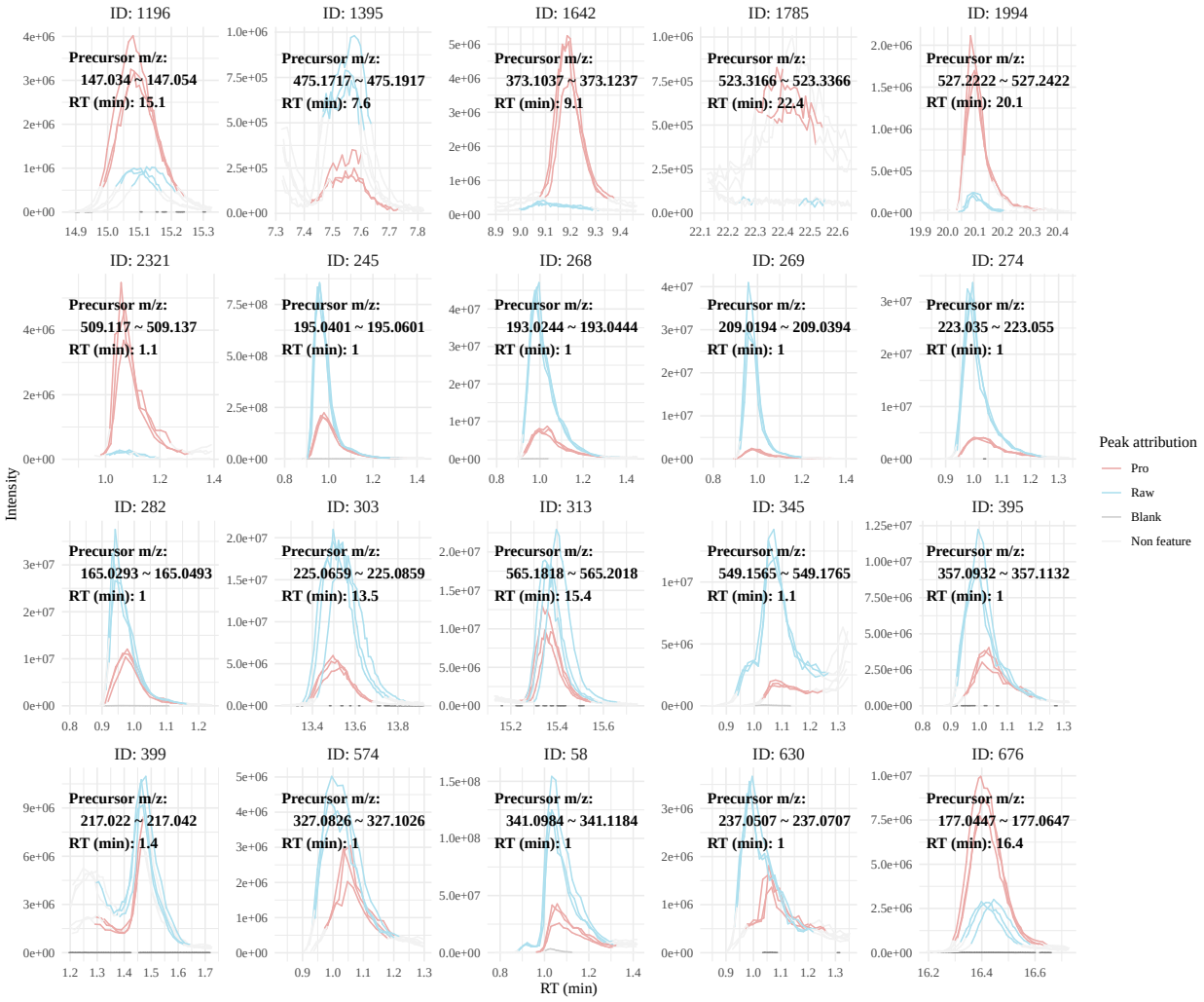

Figure 9: Extracted Ions Chromatograph (EIC) of top features (identified)

See results (Fig. 9). It can be assumed that 1642, 1785, and 2321 are newly generated compounds after the processing. According to Fig. 3, they were belong to chemical classes of 'Iridoids and derivatives', 'Dialkyl ethers' and 'Phenylpropanoids and polyketides'.

5.7 Discover more around top 'features' in Child-Nebulae

Plot the Child-Nebulae of the two chemical classes that we have not annotated yet.

```
dia <- "Dialkyl ethers"
phe <- "Phenylpropanoids and polyketides"
mcn2 <- annotate_nebula(mcn2, dia)
mcn2 <- annotate_nebula(mcn2, phe)
p <- visualize(mcn2, dia, annotate = T)
ggsave(f13.1.1 <- paste0(tmp, "/dia_child.pdf"), p,
  width = 6,
  height = 4
```

```
)
p <- visualize(mcn2, phe, annotate = T)
ggsave(f13.1.2 <- paste0(tmp, "/phe_child.pdf"), p,
  width = 6,
  height = 4
)
```

Draw their partial views and put them together.

```
grob_iri <- zoom_pdf(f10.2.2, c(0.28, 0.515), c(0.1, 0.1), dpi = 3000)
grob_dia <- zoom_pdf(f13.1.1, c(0.25, 0.3), c(0.1, 0.1), dpi = 3000)
grob_phe <- zoom_pdf(f13.1.2, c(0.465, 0.825), c(0.06, 0.06),
  dpi = 3000
)
local_1642 <- into(grecta("a"), grob_iri)
local_1785 <- into(grecta("b"), grob_dia)
local_2321 <- into(grecta("c"), grob_phe)
locals <- frame_row(
  c(local_1642 = 1, local_1785 = 1, local_2321 = 1),
  namel(local_1642, local_1785, local_2321)
)
```

We found interesting adjacent compounds (ID:2110 and ID:854) in `local_1642`, which has a similar chemical structure to 'feature' 1642.

```
grob_struc2110 <- grid.grabExpr(show_structure(mcn2, "2110"))
grob_struc2110 <- into(grecta("d"), grob_struc2110)
grob_struc854 <- grid.grabExpr(show_structure(mcn2, "854"))
grob_struc854 <- into(grecta("e"), grob_struc854)
grob_msms2110 <- as_grob(plot_msms_mirrors(mcn2, c("2110", "854"),
  structure_vp = NULL
))
grob_msms2110 <- into(grecta("f"), grob_msms2110)
```

Again, we don't run the following code, but we save the results.

```
data <- plot_EIC_stack(c("2110", "854"), metadata,
  quant.path = paste0(
    tmp,
    "/features.csv"
  ), mzml.path = "/media/echo/back/thermo_mzML_0518/",
  palette = palette_stat(mcn2)
)
save(data, file = paste0(tmp, "/eic_data2110.rdata"))
```

Load the data and convert picture.

```
load(paste0(tmp, "/eic_data2110.rdata"))
grob_eic2110 <- as_grob(data$p)
grob_eic2110 <- into(grecta("g"), grob_eic2110)
```

Draw the final figure.

```
frame <- frame_col(
  c(grob_struc2110 = 1, grob_struc854 = 1),
  namel(grob_struc2110, grob_struc854)
)
```

```
frame <- frame_row(
  c(frame = 1, grob_msms2110 = 1, grob_eic2110 = 1),
  namel(frame, grob_msms2110, grob_eic2110)
)
frame <- frame_col(c(locals = 1, frame = 1.5), namel(
  locals,
  frame
))
frame <- ggather(frame, vp = viewport(, , 0.95, 0.95))
pdf(f13.6 <- paste0(tmp, "/complex.pdf"), 14, 10)
draw(frame)
dev.off()
```

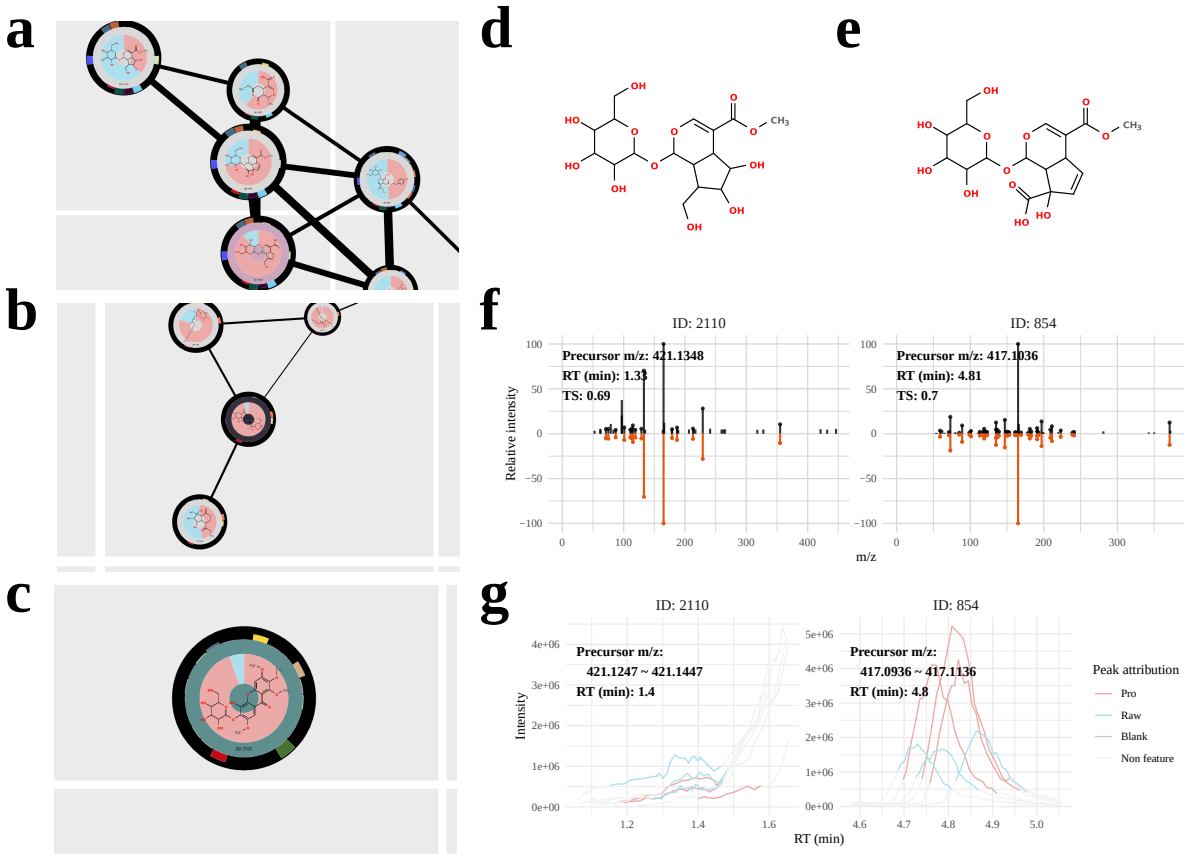

Figure 10: Discover chemical changes using MCnebula

It can be speculated that the changes in the levels of ID 1642 and ID 845 were caused by structural changes of ID 2110 during the processing, which may have involved reactions such as dehydration and rearrangement (Fig. 10).

6 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] exMCnebula2_0.0.0.9000 MCnebula2_0.0.9000     testthat_3.1.4         ggplot2_3.3.6
## [5] nvimcom_0.9-142
##
## loaded via a namespace (and not attached):
##   [1] ModelMetrics_1.2.2.2   R.methodsS3_1.8.2      tidyr_1.2.0            bit64_4.0.5
##   [5] knitr_1.39             R.utils_2.11.0         styler_1.7.0           data.table_1.14.2
##   [9] rpart_4.1.16           doParallel_1.0.17      KEGGREST_1.36.2        hardhat_1.2.0
##  [13] RCurl_1.98-1.7         generics_0.1.2         preprocessCore_1.58.0  BiocGenerics_0.42.0
##  [17] callr_3.7.0            usethis_2.1.6          RSQLite_2.2.14         RApiSerialize_0.1.2
##  [21] future_1.26.1          bit_4.0.4              BiocStyle_2.24.0       xml2_1.3.3
##  [25] lubridate_1.8.0        ggsci_2.9              assertthat_0.2.1       viridis_0.6.2
##  [29] gower_1.0.1            xfun_0.31              evaluate_0.15          fansi_1.0.3
##  [33] scrime_1.3.5           caTools_1.18.2         igraph_1.3.2           DBI_1.1.2
##  [37] htmlwidgets_1.5.4      ChemmineOB_1.34.0      stats4_4.2.1           purrr_0.3.4
##  [41] ellipsis_0.3.2         dplyr_1.0.9            backports_1.4.1        bookdown_0.27
##  [45] grImport2_0.2-0        RcppParallel_5.1.5     vctrs_0.5.1            Biobase_2.56.0
##  [49] remotes_2.4.2          Cairo_1.6-0            caret_6.0-93           cachem_1.0.6
##  [53] withr_2.5.0            ggforce_0.3.3          checkmate_2.1.0        treeio_1.20.0
##  [57] prettyunits_1.1.1      svglite_2.1.0          cluster_2.1.3          FELLA_1.16.0
##  [61] ape_5.6-2              lazyeval_0.2.2         crayon_1.5.2           edgeR_3.38.1
##  [65] recipes_1.0.3          pkgconfig_2.0.3        tweenr_1.0.2           GenomeInfoDb_1.32.2
##  [69] ProtGenerics_1.28.0    nlme_3.1-159           pkgload_1.2.4          nnet_7.3-17
##  [73] devtools_2.4.3         rlang_1.0.6            globals_0.15.1         lifecycle_1.0.3
##  [77] affyio_1.66.0          rsvg_2.3.1             rprojroot_2.0.3        polyclip_1.10-0
##  [81] MetaboAnalystR_3.3.0   Matrix_1.5-1           aplot_0.1.6            base64enc_0.1-3
##  [85] processx_3.6.0         mzR_2.30.0             png_0.1-7              viridisLite_0.4.0
##  [89] stringfish_0.15.7      bitops_1.0-7           R.oo_1.25.0            KernSmooth_2.23-20
##  [93] pROC_1.18.0            Biostrings_2.64.0      blob_1.2.3             pdftools_3.2.1
##  [97] stringr_1.4.0          parallelly_1.32.0      qpdf_1.2.0             R.cache_0.15.0
## [101] jpeg_0.1-9             gridGraphics_0.5-1     S4Vectors_0.34.0       scales_1.2.0
## [105] memoise_2.0.1          magrittr_2.0.3         plyr_1.8.7             gplots_3.1.3
## [109] zlibbioc_1.42.0        compiler_4.2.1         RColorBrewer_1.1-3     clue_0.3-61
## [113] pcaMethods_1.88.0      affy_1.74.0            cli_3.5.0              XVector_0.36.0
## [117] listenv_0.8.0          patchwork_1.1.1        pbapply_1.5-0          ps_1.7.0
## [121] htmlTable_2.4.0        Formula_1.2-4          MASS_7.3-58            tidyselect_1.2.0
## [125] vsn_3.64.0             stringi_1.7.6          highr_0.9              yaml_2.3.5
## [129] askpass_1.1            locfit_1.5-9.5         MALDIquant_1.21        latticeExtra_0.6-29
```

```
## [133] ggrepel_0.9.1          fastmatch_1.1-3       tools_4.2.1            future.apply_1.9.0
## [137] parallel_4.2.1         rstudioapi_0.13       MsCoreUtils_1.8.0      qs_0.25.4
## [141] foreach_1.5.2          foreign_0.8-82        crmn_0.0.21            classyfireR_0.3.8
## [145] gridExtra_2.3          prodlim_2019.11.13    mzID_1.34.0            farver_2.1.0
## [149] ggraph_2.0.5           digest_0.6.29         BiocManager_1.30.18    ggtext_0.1.1
## [153] lava_1.7.0             Rcpp_1.0.8.3          gridtext_0.1.4         siggenes_1.70.0
## [157] ncdf4_1.19             MSnbase_2.22.0        httr_1.4.3             colorspace_2.0-3
## [161] brio_1.1.3             XML_3.99-0.10         fs_1.5.2               IRanges_2.30.0
## [165] splines_4.2.1          yulab.utils_0.0.4     tidytree_0.3.9         graphlayouts_0.8.0
## [169] multtest_2.52.0        ggplotify_0.1.0       plotly_4.10.0          sessioninfo_1.2.2
## [173] systemfonts_1.0.4      jsonlite_1.8.0        ggtree_3.4.0           tidygraph_1.2.1
## [177] timeDate_4021.107      glasso_1.11           ggfun_0.0.6            ipred_0.9-13
## [181] ggimage_0.3.1          R6_2.5.1              Hmisc_4.7-0            pillar_1.7.0
## [185] htmltools_0.5.2        glue_1.6.2            fastmap_1.1.0          BiocParallel_1.30.3
## [189] class_7.3-20           codetools_0.2-18      fgsea_1.22.0           pkgbuild_1.3.1
## [193] utf8_1.2.2             lattice_0.20-45       tibble_3.1.7           gtools_3.9.2.2
## [197] magick_2.7.3           survival_3.4-0        limma_3.52.1           rmarkdown_2.14
## [201] desc_1.4.1             munsell_0.5.0         GenomeInfoDbData_1.2.8 iterators_1.0.14
## [205] impute_1.70.0          reshape2_1.4.4        gtable_0.3.0
```

Reference

1. Dührkop K, Fleischauer M, Ludwig M, Aksenov AA, Melnik AV, Meusel M, et al. SIRIUS 4: A rapid tool for turning tandem mass spectra into metabolite structure information. Nature Methods. 2019 Apr;16(4):299–302.
2. Böcker S, Letzel MC, Lipták Z, Pervukhin A. SIRIUS: Decomposing isotope patterns for metabolite identification†. Bioinformatics. 2009 Jan;25(2):218–24.
3. Dührkop K, Shen H, Meusel M, Rousu J, Böcker S. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. Proceedings of the National Academy of Sciences. 2015 Oct;112(41):12580–5.
4. Ludwig M, Nothias L-F, Dührkop K, Koester I, Fleischauer M, Hoffmann MA, et al. Database-independent molecular formula annotation using Gibbs sampling through ZODIAC. Nature Machine Intelligence. 2020 Oct;2(10):629–41.
5. Dührkop K, Nothias L-F, Fleischauer M, Reher R, Ludwig M, Hoffmann MA, et al. Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. Nature Biotechnology. 2021 Apr;39(4):462–71.
6. Smyth GK. Limma: Linear Models for Microarray Data. In: Gentleman R, Carey VJ, Huber W, Irizarry RA, Dudoit S, editors. Bioinformatics and Computational Biology Solutions Using R and Bioconductor. New York: Springer-Verlag; 2005. p. 397–420.