MCnebula workflow for LC-MS/MS dataset analysis

Contents

1 MCnebula workflow

1.1 Introduction

A classified visualization method, called MCnebula, was used for the analysis of untargeted LC-MS/MS datasets. MCnebula utilizes the state-of-the-art computer prediction technology, SIRIUS workflow (SIRIUS, ZODIAC, CSI:fingerID, CANOPUS)[1–6], for compound formula prediction, structure retrieve and classification prediction. MCnebula integrates an abundance-based class selection algorithm into compound annotation. The benefits of molecular networking, i.e. intuitive visualization and a large amount of integratable information, were incorporated into MCnebula visualization. With MCnebula, we can switch from untargeted to targeted analysis, focusing precisely on the compound or chemical class of interest to the researcher.

MCnebula primarily performs an abundance-based class selection algorithm before visualization. MCnebula tend to filter out those classes with too large scope (e.g., possibly be 'Lipids and lipid-like molecules' but data dependent) or too sparse compounds (data depend). We termed these summarised classes as nebula-index. To begin with, like classical feature-based molecular networking (FBMN) pattern,[7] features make up the initial network, which we termed parent-nebula. Subsequently, according to nebula-index and the posterior probability of classes prediction (PPCP) of features,[6] nodes or edges from parent-nebula are divided into sub-networks. We termed these sub-networks as child-nebulae and their names, termed nebula-name, are in line with the classes name within nebulae-index. The nebula-names contained the sub-structural or dominant-structural characteristics for features within child-nebulae. Collectively, all the network and sub-networks termed multi-chemical nebulae. In general, parent-nebula is too informative to show, so child-nebulae was used to dipict the abundant classes of 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.

1.2 R and other softs Setup

1.2.1 R setup

```
library(MCnebula)
library(dplyr)
library(ggplot2)
library(ggraph)
library(grid)
```

1.2.2 Others setup

The prerequisite soft for MCnebula is SIRIUS 4.

MZmine 2 (version 2.53) (https://github.com/mzmine/mzmine2) were utilized for LC-MS/MS data processing.

Before mass spectrometry data been processed, the raw data should be converted to .mzML or .mzXML file in most cases. ProteoWizard (https://proteowizard.sourceforge.io/) were implemented for the conversion.

1.3 Data preprocessing

1.3.1 Raw data processing

For MZmine2 processing, an XML batch file outlined the example parameters for waters Qtof could be find in https://github.com/Cao-lab-zcmu/research-supplementary.

1.3.2 SIRIUS computation workflow

The computational parameters of SIRIUS CLI tools were set as following:

```
mgf.path <- "my.mgf"
system(
  paste0(
    "sirius -i ",
    mgf.path,
    " -o test --maxmz 800 formula -c 50 zodiac structure canopus"
    )
)
```

1.3.3 MCnebula processing

Run MCnebual basic workflow.

```
initialize_mcnebula(".")
collate_structure()
build_classes_tree_list()
collate_ppcp(min_possess = 20, max_possess_pct = 0.1)
generate_parent_nebula(rm_parent_isolate_nodes = T)
generate_child_nebulae()
visualize_parent_nebula(layout = "kk", width = 20, height = 17)
```

The parent-nebula shows the superclass of compounds in dataset (Fig. 1). However, it was too informatics to illustration.

```
visualize_child_nebulae(width = 15, height = 20, nodes_size_range = c(2, 4))
```

The child-nebulae shows the overall abundant classes and compound distribution for dataset (Fig. 2).

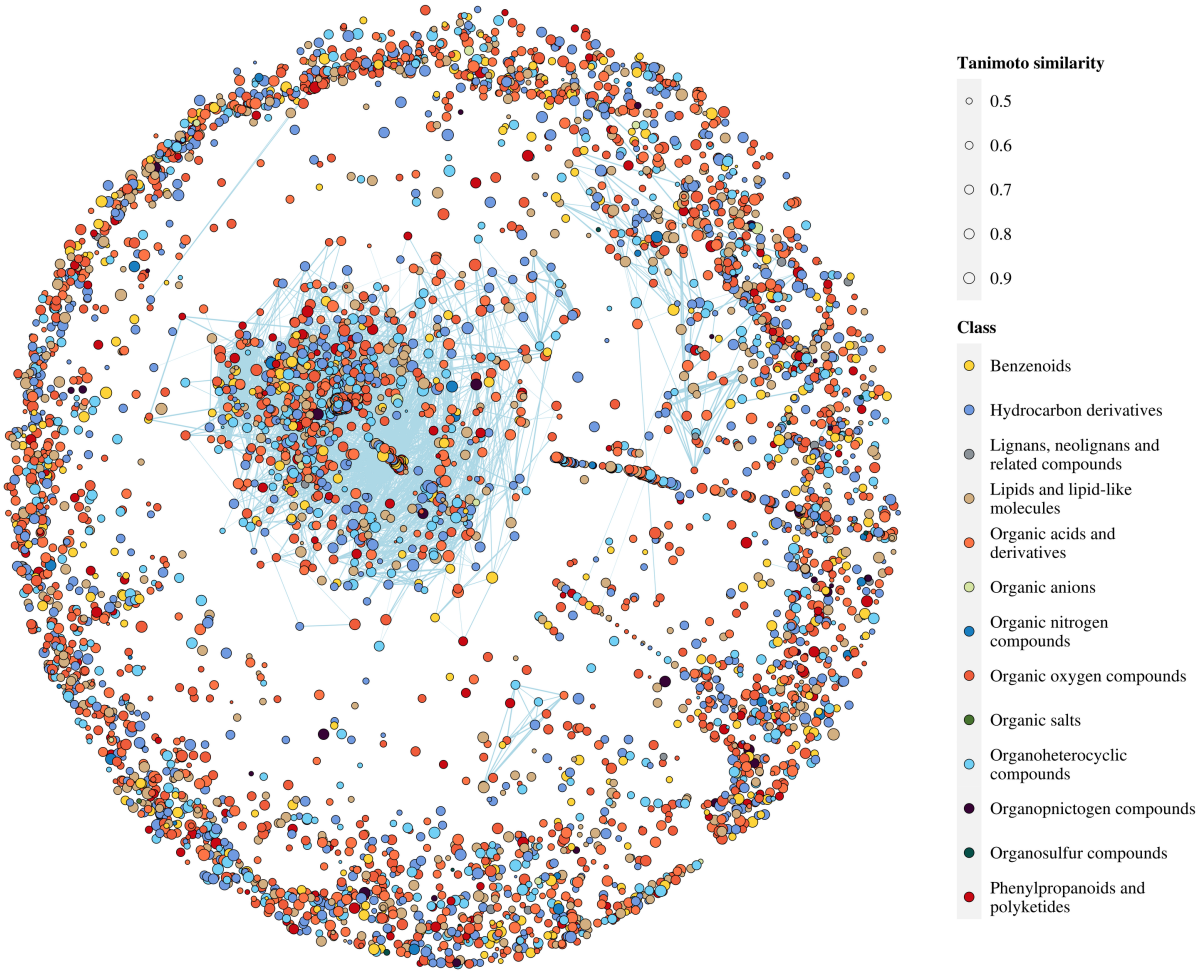

Figure 1: parent-nebula

Figure 2: child-nebulae

Format quantification table and summarise mean value for each group.

```
stat <- format_quant_table("../earlist.neg.csv",
                           meta.group = c(blank = "BLANK",
                                          raw = "^S", pro = "^J"))
dplyr::as_tibble(stat)
```

Set color palette for each group.

```
palette_stat <- c(blank = "grey",
                  raw = "lightblue", pro = "pink")
```

Herein, Taxifolin and SARRACENIN are marked in child-nebulae. Simply, compounds with identical formulae of above were marked with specific color.

```
formula.tax <- "C15H12O7"
formula.sar <- "C11H14O5"
id.tax <- dplyr::filter(.MCn.formula_set, molecularFormula == formula.tax)$.id
id.sar <- dplyr::filter(.MCn.formula_set, molecularFormula == formula.sar)$.id
nodes_mark <- data.frame(
  .id = c(id.tax, id.sar, "Others"),
  mark = c(rep(c("tax", "sar"), c(length(id.tax), length(id.sar))), "Others")
)
palette <- c(Others = "#D9D9D9", tax = "#DDFF77", sar = "#CCCCFF")
```

A function of visualization of child-nebula was defined for reproducibly use.

```
tmp_anno <- function(nebula_name, nebula_index = .MCn.nebula_index){
  annotate_child_nebulae(
    ## string, i.e. class name in nebula-index
    nebula_name = nebula_name,
    nebula_index = nebula_index,
    layout = "fr",
    ## a table to mark color of nodes
    nodes_mark = nodes_mark,
    plot_nodes_id = T,
    plot_structure = T,
    plot_ppcp = T,
    ## manually define the color of nodes
    palette = palette,
    ## feature quantification table
    ratio_df = stat,
    ## A vector of the hex color with names or not
    palette_stat = palette_stat,
    ## control nodes size in child-nebula, zoom in or zoom out globally.
    global.node.size = 0.8,
    ## the args of `ggplot::theme`
    theme_args = list(
      panel.background = element_rect(),
      panel.grid = element_line()
    ),
    return_plot = F
  )
}
```

'Flavonoids' and 'Iridoids and derivatives' are classes of interest. They are classes of Taxifolin and SARRACENIN belong to. Due to 'Flavonoids' not exists in nebula-index (Fig. 2), MCnebula was used to target

clustering the 'Flavonoids'.

```
target_class <- "Flavonoids"
target_index <- method_summarize_target_index(target_class)
hq.structure <- dplyr::filter(.MCn.structure_set, tanimotoSimilarity >= 0.1)
hq.target_index <- dplyr::filter(target_index, .id %in% hq.structure$.id)
```

Re-compute the spectral similarity within child-nebula.

```
spec.path <- method_target_spec_compare(target_class,
                                        hq.target_index,
                                        edge_filter = 0.5)
```

Use the re-computated spectral similarity file to generate parent-nebula.

```
call_fun_mc.space("generate_parent_nebula",
                  list(write_output = F, edges_file = spec.path),
                  clear_start = T,
                  clear_end = F)
```

Generate child-nebula.

```
test <- call_fun_mc.space("generate_child_nebulae",
                          list(nebula_index = hq.target_index),
                          clear_start = F,
                          clear_end = F)
```

Use `molconvert` to visualize chemical structure.

```
hq.amino <- dplyr::filter(hq.structure, .id %in% hq.target_index$.id)
vis_via_molconvert(hq.amino$smiles, hq.amino$.id)
```

Visualized the child-nebula with annotation of structure and quantification.

```
call_fun_mc.space("tmp_anno",
                  list(nebula_name = target_class,
                       nebula_index = hq.target_index),
                  clear_start = F,
                  clear_end = F)
```

Figure 3 show 'Flavonoids'...

Subsequently, the child-nebula of 'Iridoids and derivatives' was visualized.

```
iri.index <- dplyr::filter(.MCn.nebula_index, name == 'Iridoids and derivatives')
hq.iri <- dplyr::filter(hq.structure, .id %in% iri.index$.id)
vis_via_molconvert(hq.amino$smiles, hq.amino$.id)
tmp_anno('Iridoids and derivatives')
```

Figure 4 show 'Iridoids and derivatives'...

The extracted ion chromatogram of specific ID (Taxifolin and SARRACENIN)...

```
## The package could found at: https://github.com/Cao-lab-zcmu/utils_tool
devtools::load_all("~/utils_tool")
metadata <- format_quant_table("../earlist.neg.csv", get_metadata = T,
                               meta.group = c(blank = "BLANK", raw = "^S", pro = "^J")) %>%
  dplyr::slice(-1)
stack_ms1(idset = c(id.tax, id.sar),
          metadata = metadata,
          quant.path = "../earlist.neg.csv",
```

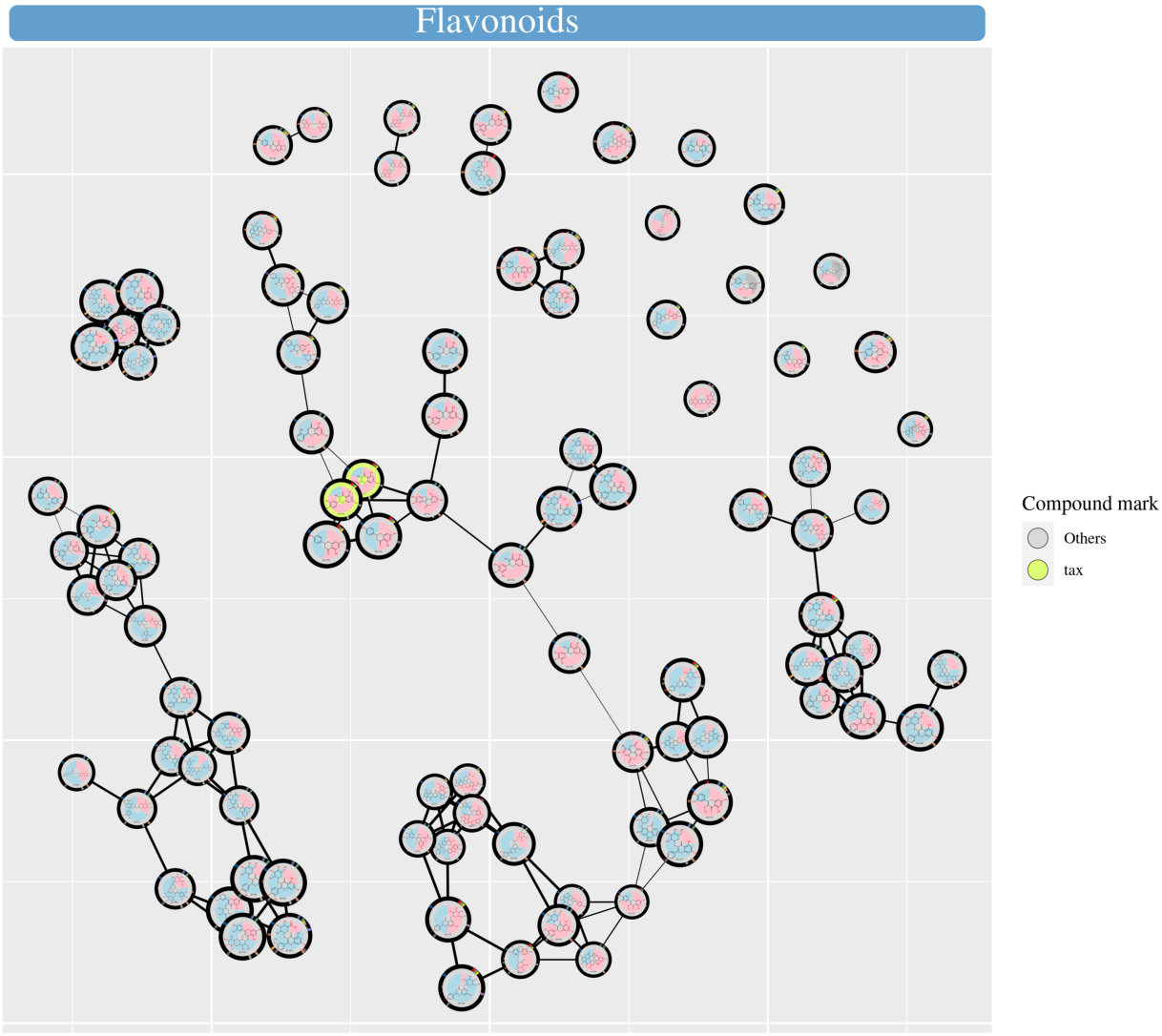

Figure 3: child-nebula of Flavonoids

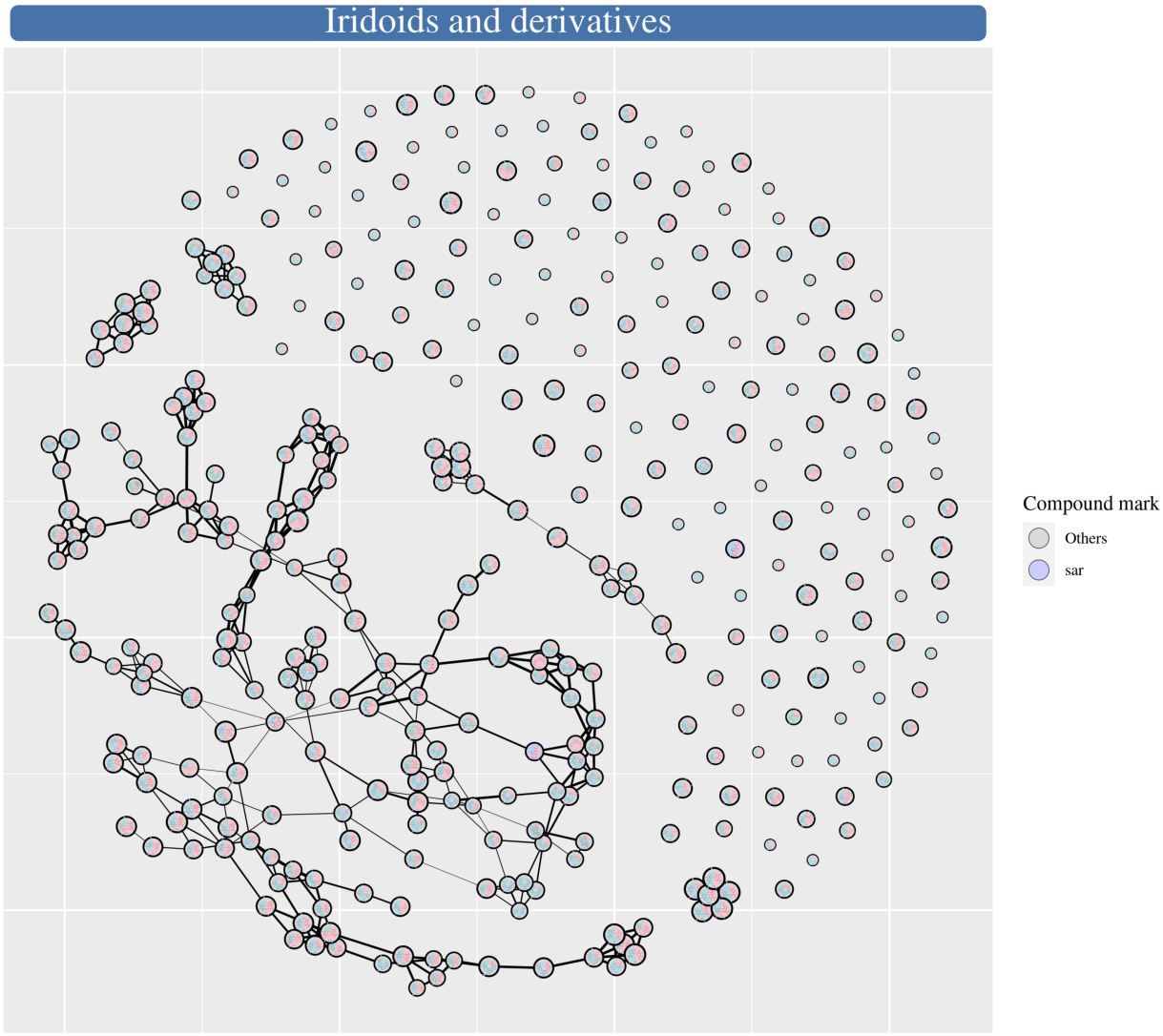

Figure 4: child-nebula of Iridoids.

```
        mzml.path = "../Thermo_raw",
        palette = palette_stat
)
```

Figure 5...

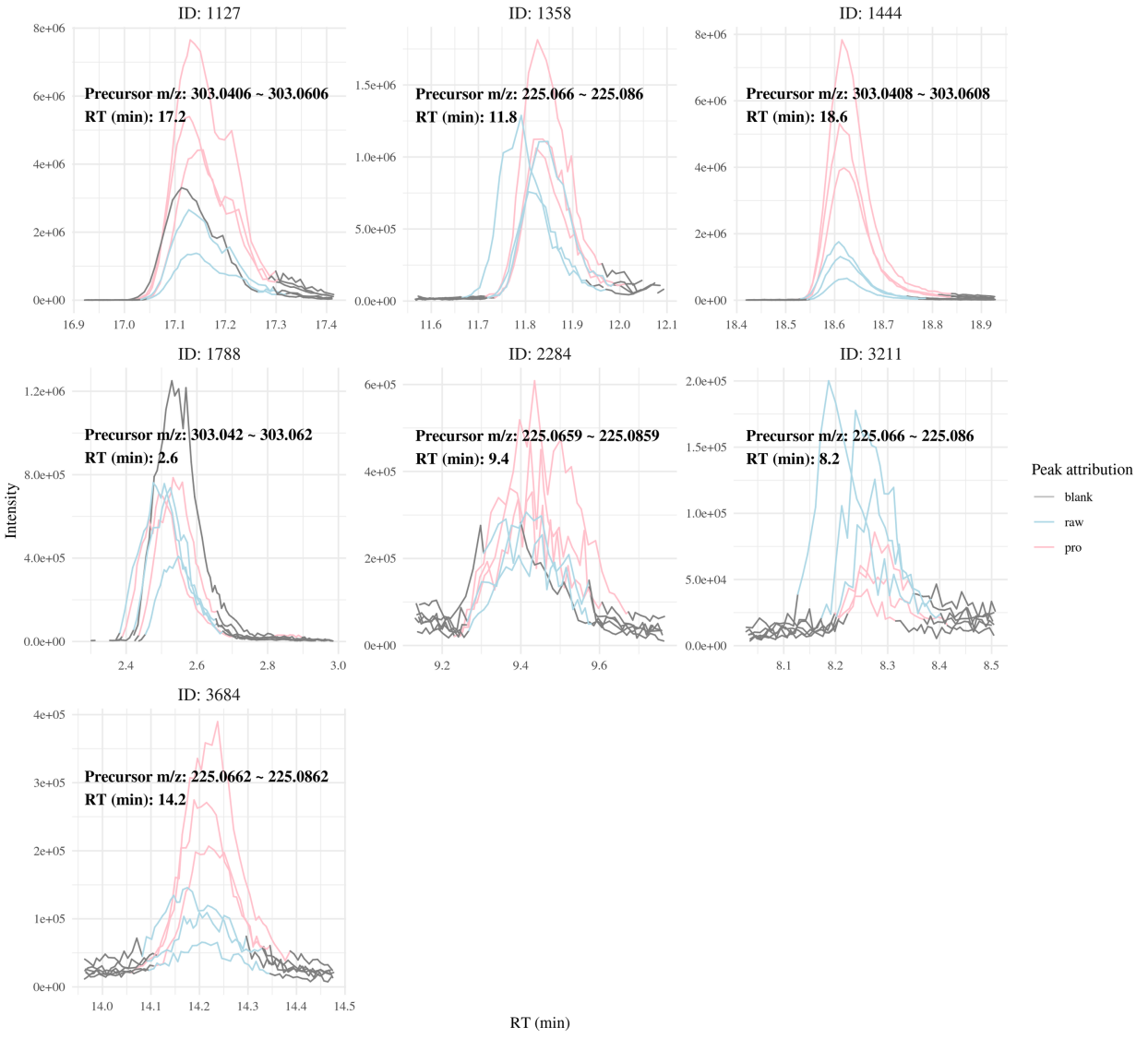

Figure 5: EIC of candidates feature of Taxifolin and SARRACENIN

The MS/MS spectrum of specific ID...

```
stack_ms2(c(id.tax, id.sar))
```

Figure 6

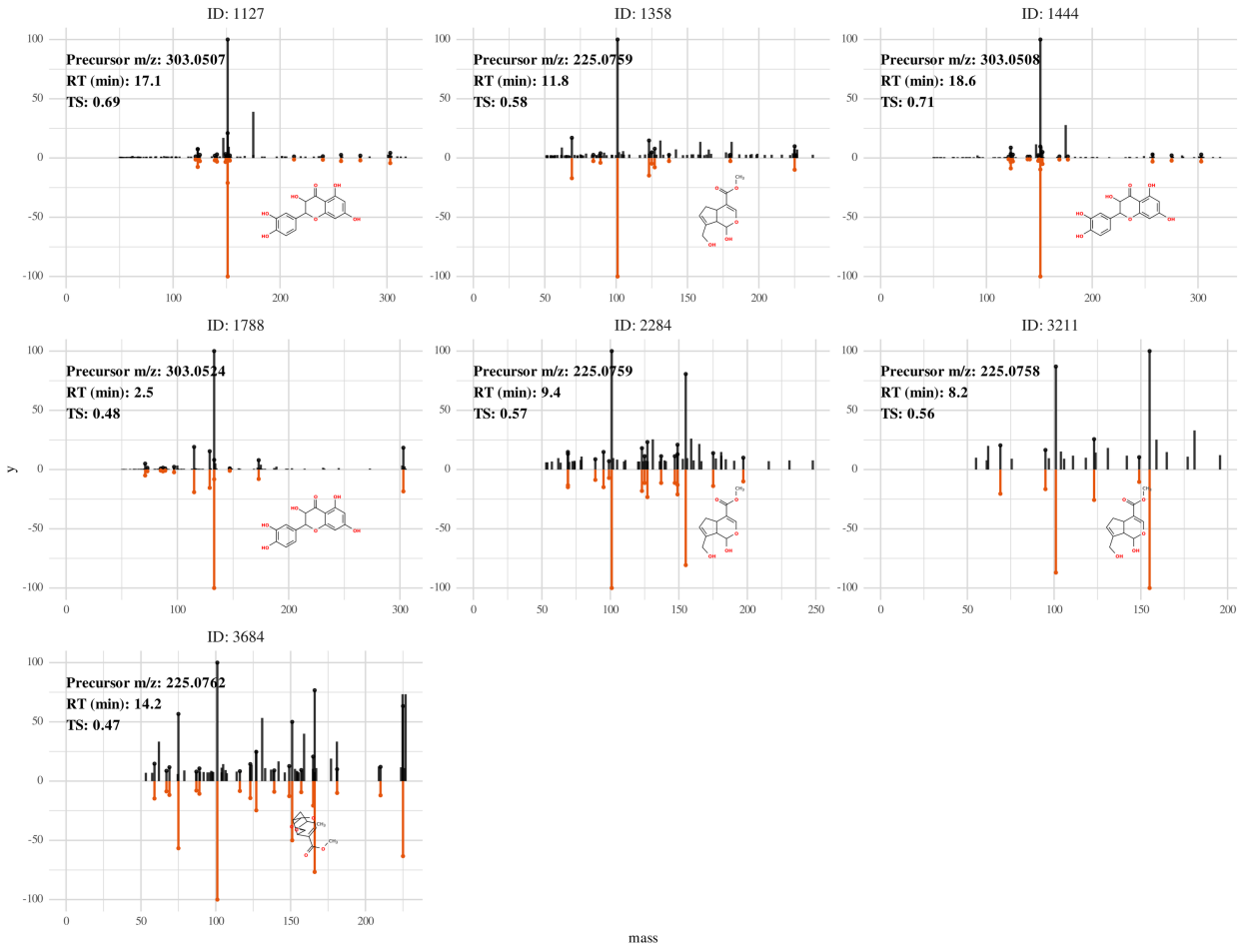

Figure 6: MS/MS spectrum of candidates feature of Taxifolin and SARRACENIN

1.4 Differential analysis

Limma is generally used for gene differential expression analysis. Here, we use several statistical functions in limma to analyze the difference of compound level between raw. and pro.. Check the changes of compounds level during processing.

```
mutate_stat <- dplyr::summarise_at(stat, 2:ncol(stat), log2) %>%
  dplyr::mutate(.id = stat$.id) %>%
  dplyr::mutate(log2fc = pro - raw,
                change = ifelse(log2fc > 1, "up",
                                ifelse(log2fc < -1, "down", "-"))) %>%
  dplyr::relocate(.id, log2fc, change)
```

1.5 Guidance

1.5.1 Details of workflow

An end-to-end analysis of MCnebula workflow from sample to multi-chemical-nebulae is show in figure 7

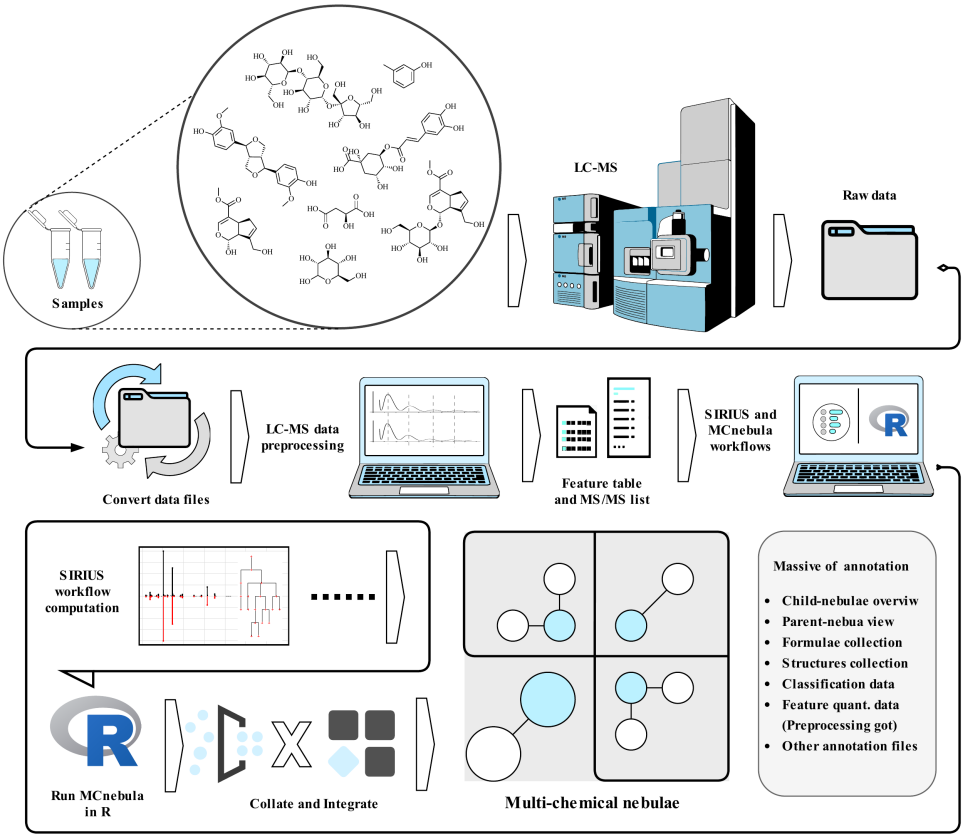

Figure 7: workflow

1.6 Information

All output files:

```
list.files(paste0(.MCn.output, "/", .MCn.results), recursive = T)
```

Other information:

```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22)
## 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
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    grid      stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] utils.tool_0.0.0.9000 MCnebula_0.0.0.9000   xcms_3.18.0
##  [4] MSnbase_2.22.0        ProtGenerics_1.28.0   S4Vectors_0.34.0
##  [7] mzR_2.30.0            Rcpp_1.0.8.3          Biobase_2.56.0
## [10] BiocGenerics_0.42.0   BiocParallel_1.30.3   dplyr_1.0.9
## [13] ggraph_2.0.5          ggplot2_3.3.6         devtools_2.4.3
## [16] usethis_2.1.6
##
## loaded via a namespace (and not attached):
##   [1] systemfonts_1.0.4           plyr_1.8.7
##   [3] igraph_1.3.1                ChemmineOB_1.34.0
##   [5] GenomeInfoDb_1.32.2         digest_0.6.29
##   [7] foreach_1.5.2               yulab.utils_0.0.4
##   [9] htmltools_0.5.2             viridis_0.6.2
##  [11] magick_2.7.3                fansi_1.0.3
##  [13] magrittr_2.0.3              memoise_2.0.1
##  [15] grImport2_0.2-0             cluster_2.1.3
##  [17] doParallel_1.0.17           limma_3.52.1
##  [19] remotes_2.4.2               graphlayouts_0.8.0
##  [21] matrixStats_0.62.0          MsFeatures_1.4.0
##  [23] svglite_2.1.0               prettyunits_1.1.1
##  [25] jpeg_0.1-9                  colorspace_2.0-3
##  [27] ggrepel_0.9.1               xfun_0.31
##  [29] RCurl_1.98-1.7              callr_3.7.0
##  [31] crayon_1.5.1                jsonlite_1.8.0
##  [33] impute_1.70.0               iterators_1.0.14
##  [35] glue_1.6.2                  polyclip_1.10-0
##  [37] gtable_0.3.0                XVector_0.36.0
##  [39] zlibbioc_1.42.0             DelayedArray_0.22.0
##  [41] pkgbuild_1.3.1              DEoptimR_1.0-11
##  [43] scales_1.2.0                vsn_3.64.0
##  [45] DBI_1.1.3                   viridisLite_0.4.0
```

```
##  [47] gridtext_0.1.4              clue_0.3-61
##  [49] gridGraphics_0.5-1          preprocessCore_1.58.0
##  [51] MsCoreUtils_1.8.0           rsvg_2.3.1
##  [53] RColorBrewer_1.1-3          ellipsis_0.3.2
##  [55] pkgconfig_2.0.3             XML_3.99-0.10
##  [57] farver_2.1.0                utf8_1.2.2
##  [59] ggplotify_0.1.0             tidyselect_1.1.2
##  [61] labeling_0.4.2              rlang_1.0.2
##  [63] reshape2_1.4.4              munsell_0.5.0
##  [65] tools_4.2.0                 cachem_1.0.6
##  [67] cli_3.3.0                   generics_0.1.2
##  [69] evaluate_0.15               stringr_1.4.0
##  [71] fastmap_1.1.0               mzID_1.34.0
##  [73] yaml_2.3.5                  processx_3.6.0
##  [75] knitr_1.39                  fs_1.5.2
##  [77] tidygraph_1.2.1             robustbase_0.95-0
##  [79] RANN_2.6.1                  purrr_0.3.4
##  [81] ncdf4_1.19                  pbapply_1.5-0
##  [83] xml2_1.3.3                  brio_1.1.3
##  [85] compiler_4.2.0              rstudioapi_0.13
##  [87] png_0.1-7                   testthat_3.1.4
##  [89] affyio_1.66.0               MassSpecWavelet_1.62.0
##  [91] tibble_3.1.7                tweenr_1.0.2
##  [93] stringi_1.7.6               ggimage_0.3.1
##  [95] highr_0.9                   ps_1.7.0
##  [97] desc_1.4.1                  lattice_0.20-45
##  [99] Matrix_1.4-1                markdown_1.1
## [101] ggsci_2.9                   vctrs_0.4.1
## [103] pillar_1.7.0                lifecycle_1.0.1
## [105] BiocManager_1.30.18         MALDIquant_1.21
## [107] bitops_1.0-7                data.table_1.14.2
## [109] GenomicRanges_1.48.0        R6_2.5.1
## [111] pcaMethods_1.88.0           affy_1.74.0
## [113] bookdown_0.27               gridExtra_2.3
## [115] IRanges_2.30.0              sessioninfo_1.2.2
## [117] codetools_0.2-18            MASS_7.3-57
## [119] assertthat_0.2.1            pkgload_1.2.4
## [121] SummarizedExperiment_1.26.1 rprojroot_2.0.3
## [123] withr_2.5.0                 GenomeInfoDbData_1.2.8
## [125] parallel_4.2.0              ggtext_0.1.1
## [127] ggfun_0.0.6                 tidyr_1.2.0
## [129] rmarkdown_2.14              MatrixGenerics_1.8.0
## [131] ggforce_0.3.3               base64enc_0.1-3
## [133] tinytex_0.40
```

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, Böcker S. Fragmentation Trees Reloaded. In: Przytycka TM, editor. Research in Computa-

tional Molecular Biology. Cham: Springer International Publishing; 2015. pp. 65–79.

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, 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.

6. 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.

7. Nothias L-F, Petras D, Schmid R, Dührkop K, Rainer J, Sarvepalli A, et al. Feature-based molecular networking in the GNPS analysis environment. Nature Methods. 2020 Sep;17(9):905–8.