MCnebula workflow for LC-MS/MS dataset analysis

Contents

Introduction . . . 1
R and other softs Setup . . . 1
R setup . . . 1
Others setup . . . 2
Data preprocessing . . . 2
Raw data processing . . . 2
SIRIUS computation workflow . . . 2
MCnebula processing . . . 2
Information . . . 2
Reference . . . 6

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.

R and other softs Setup

R setup

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

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.

Data preprocessing

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.

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"
    )
)
```

MCnebula processing

???

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] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] ggraph_2.0.5          utils.tool_0.0.0.9000 edgeR_3.38.1
##  [4] limma_3.52.1          MCnebula_0.0.0.9000   testthat_3.1.4
##  [7] gt_0.6.0              stringr_1.4.0         ggsci_2.9
## [10] ggplot2_3.3.6         dplyr_1.0.9           data.table_1.14.2
## [13] devtools_2.4.3        usethis_2.1.6
##
## loaded via a namespace (and not attached):
##   [1] BiocFileCache_2.4.0
##   [2] systemfonts_1.0.4
##   [3] plyr_1.8.7
##   [4] igraph_1.3.2
##   [5] BiocParallel_1.30.3
##   [6] GenomeInfoDb_1.32.2
##   [7] digest_0.6.29
##   [8] foreach_1.5.2
##   [9] htmltools_0.5.2
##  [10] GO.db_3.15.0
##  [11] viridis_0.6.2
##  [12] fansi_1.0.3
##  [13] magrittr_2.0.3
##  [14] memoise_2.0.1
##  [15] cluster_2.1.3
##  [16] doParallel_1.0.17
##  [17] remotes_2.4.2
##  [18] Biostrings_2.64.0
##  [19] graphlayouts_0.8.0
##  [20] matrixStats_0.62.0
##  [21] svglite_2.1.0
##  [22] prettyunits_1.1.1
##  [23] colorspace_2.0-3
##  [24] blob_1.2.3
##  [25] rappdirs_0.3.3
##  [26] ggrepel_0.9.1
##  [27] xfun_0.31
##  [28] jsonlite_1.8.0
##  [29] callr_3.7.0
##  [30] crayon_1.5.1
##  [31] RCurl_1.98-1.7
##  [32] org.Mm.eg.db_3.15.0
##  [33] graph_1.74.0
##  [34] Mus.musculus_1.3.1
##  [35] impute_1.70.0
##  [36] iterators_1.0.14
##  [37] glue_1.6.2
##  [38] polyclip_1.10-0
##  [39] gtable_0.3.0
##  [40] zlibbioc_1.42.0
##  [41] XVector_0.36.0
```

```
##  [42] DelayedArray_0.22.0
##  [43] pkgbuild_1.3.1
##  [44] BiocGenerics_0.42.0
##  [45] scales_1.2.0
##  [46] vsn_3.64.0
##  [47] DBI_1.1.2
##  [48] Rcpp_1.0.8.3
##  [49] mzR_2.30.0
##  [50] viridisLite_0.4.0
##  [51] progress_1.2.2
##  [52] clue_0.3-61
##  [53] bit_4.0.4
##  [54] OrganismDbi_1.38.1
##  [55] preprocessCore_1.58.0
##  [56] stats4_4.2.0
##  [57] MsCoreUtils_1.8.0
##  [58] httr_1.4.3
##  [59] ellipsis_0.3.2
##  [60] pkgconfig_2.0.3
##  [61] XML_3.99-0.10
##  [62] farver_2.1.0
##  [63] sass_0.4.1
##  [64] dbplyr_2.2.0
##  [65] locfit_1.5-9.5
##  [66] utf8_1.2.2
##  [67] tidyselect_1.1.2
##  [68] rlang_1.0.2
##  [69] reshape2_1.4.4
##  [70] AnnotationDbi_1.58.0
##  [71] munsell_0.5.0
##  [72] tools_4.2.0
##  [73] cachem_1.0.6
##  [74] stevetemplates_0.7.0
##  [75] cli_3.3.0
##  [76] generics_0.1.2
##  [77] RSQLite_2.2.14
##  [78] evaluate_0.15
##  [79] fastmap_1.1.0
##  [80] yaml_2.3.5
##  [81] mzID_1.34.0
##  [82] knitr_1.39
##  [83] processx_3.6.0
##  [84] bit64_4.0.5
##  [85] fs_1.5.2
##  [86] tidygraph_1.2.1
##  [87] purrr_0.3.4
##  [88] KEGGREST_1.36.2
##  [89] ncdf4_1.19
##  [90] RBGL_1.72.0
##  [91] pbapply_1.5-0
##  [92] xml2_1.3.3
##  [93] biomaRt_2.52.0
##  [94] brio_1.1.3
##  [95] compiler_4.2.0
```

```
##  [96] rstudioapi_0.13
##  [97] filelock_1.0.2
##  [98] curl_4.3.2
##  [99] png_0.1-7
## [100] affyio_1.66.0
## [101] tibble_3.1.7
## [102] tweenr_1.0.2
## [103] bslib_0.3.1
## [104] stringi_1.7.6
## [105] ps_1.7.0
## [106] GenomicFeatures_1.48.3
## [107] desc_1.4.1
## [108] MSnbase_2.22.0
## [109] lattice_0.20-45
## [110] Matrix_1.4-1
## [111] ProtGenerics_1.28.0
## [112] vctrs_0.4.1
## [113] pillar_1.7.0
## [114] lifecycle_1.0.1
## [115] BiocManager_1.30.18
## [116] jquerylib_0.1.4
## [117] MALDIquant_1.21
## [118] TxDb.Mmusculus.UCSC.mm10.knownGene_3.10.0
## [119] bitops_1.0-7
## [120] rtracklayer_1.56.0
## [121] GenomicRanges_1.48.0
## [122] BiocIO_1.6.0
## [123] R6_2.5.1
## [124] pcaMethods_1.88.0
## [125] affy_1.74.0
## [126] gridExtra_2.3
## [127] IRanges_2.30.0
## [128] sessioninfo_1.2.2
## [129] codetools_0.2-18
## [130] MASS_7.3-57
## [131] assertthat_0.2.1
## [132] pkgload_1.2.4
## [133] SummarizedExperiment_1.26.1
## [134] rjson_0.2.21
## [135] rprojroot_2.0.3
## [136] withr_2.5.0
## [137] GenomicAlignments_1.32.0
## [138] Rsamtools_2.12.0
## [139] S4Vectors_0.34.0
## [140] GenomeInfoDbData_1.2.8
## [141] parallel_4.2.0
## [142] hms_1.1.1
## [143] tidyr_1.2.0
## [144] rmarkdown_2.14
## [145] MatrixGenerics_1.8.0
## [146] ggforce_0.3.3
## [147] Biobase_2.56.0
## [148] tinytex_0.40
## [149] restfulr_0.0.15
```

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 Computational 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.