Analysis...

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.

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)
```

4 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)
```

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 dowload 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

```
url <- paste0(
  "https://raw.githubusercontent.com/Cao-lab-zcmu/utils_tool/master/",
  "inst/extdata/mcn_serum6501.rdata"
)
rdata <- RCurl::getURLContent(url)
file <- paste0(tmp, "/mcn.rdata")
target <- file(file, "wb")
writeBin(rdata, target)
close(target)
rm(rdata)
```

Load the downloaded '.rdata' file.

```
load(file)
mcn <- mcn_serum6501
```

5 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)
```

6 Filter chemical classes

The PPCP 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, 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"                              "Fatty acid esters"
## [13] "Branched fatty acids"                       "Unsaturated fatty acids"
## [15] "Hydroxy fatty acids"                        "Dicarboxylic acids and derivatives"
## [17] "Hydroxy acids and derivatives"              "Pyranones and derivatives"
## [19] "Lineolic acids and derivatives"             "Steroidal glycosides"
## [21] "Acyl carnitines"                            "Glycinated bile acids and derivatives"
## [23] "Diacylglycerols"                            "Carboxylic acid salts"
## [25] "Oxosteroids"                                "Cyclic alcohols and derivatives"
## [27] "Hydroxysteroids"                            "Phosphatidylcholines"
## [29] "Lysophosphatidylcholines"                   "Bile acids, alcohols and derivatives"
## [31] "Steroid glucuronide conjugates"             "Diterpenoids"
## [33] "Bilirubins"                                 "Acetals"
## [35] "Tertiary alcohols"                          "Beta hydroxy acids and derivatives"
## [37] "Terpene glycosides"                         "Hydroxy bile acids, alcohols and derivatives"
## [39] "Glycerophosphocholines"                     "Substituted pyrroles"
## [41] "Indoles"                                    "Long-chain fatty acids"
## [43] "Organic cations"                            "Organic salts"
## [45] "Vinylogous acids"
```

```
pattern <- c("stero", "fatty acid", "pyr", "hydroxy")
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"             "Fatty acid esters"
##  [7] "Branched fatty acids"                       "Unsaturated fatty acids"
##  [9] "Hydroxy fatty acids"                        "Long-chain fatty acids"
## [11] "Pyrans"                                     "Pyridines and derivatives"
## [13] "Pyrroles"                                   "Benzopyrans"
## [15] "Pyranones and derivatives"                  "Substituted pyrroles"
## [17] "Hydroxy fatty acids"                        "Hydroxy acids and derivatives"
## [19] "Hydroxysteroids"                            "Beta hydroxy acids and derivatives"
## [21] "Hydroxy bile acids, alcohols and derivatives"
```

```
dis <- dis[-1]
```

7 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)
mcn <- clear_dataset(mcn)
```

8 Create Nebulae

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)
```

9 Visualize Nebulae

```
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
```

seq	hierarchy	class.name
1	5	Acetals
2	5	Acyl carnitines
3	4	Bile acids, alcohols and derivatives
4	4	Bilirubins
5	5	Carboxylic acid salts
6	5	Cyclic alcohols and derivatives
7	5	Diacylglycerols
8	4	Dicarboxylic acids and derivatives
9	4	Diterpenoids
10	3	Glycerolipids
11	4	Glycerophosphocholines
12	3	Glycerophospholipids
13	5	Glycinated bile acids and derivatives
14	4	Indoles
15	3	Indoles and derivatives
16	5	Ketones
17	4	Lineolic acids and derivatives
18	5	Lysophosphatidylcholines
19	2	Organic cations
20	2	Organic salts
21	5	Phosphatidylcholines
22	3	Prenol lipids
23	3	Steroids and steroid derivatives
24	5	Sugar acids and derivatives
25	4	Terpene glycosides
26	5	Tertiary alcohols
27	3	Vinylogous acids

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

```
p <- visualize(mcn, "parent")
ggsave(s5.fig1 <- paste0(tmp, "/parent_nebula.png"), p)
pdf(s5.fig2 <- paste0(tmp, "/child_nebula.pdf"), 12, 14)
visualize_all(mcn)
dev.off()
p <- visualize(mcn, 3)
ggsave(s5.fig3 <- paste0(tmp, "/bile_acids.png"), p)
```


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.

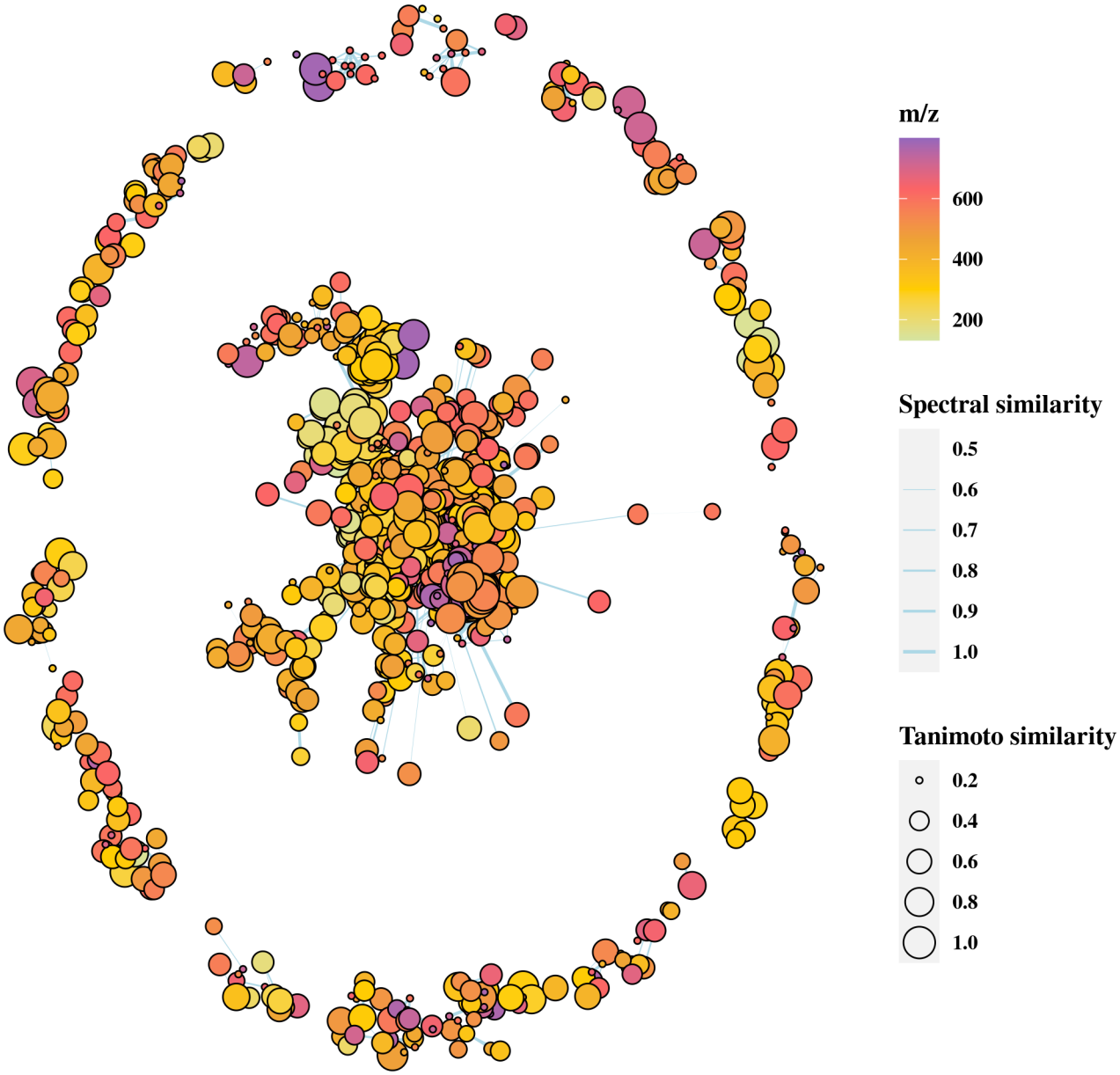

Figure 1: Parent-Nebula

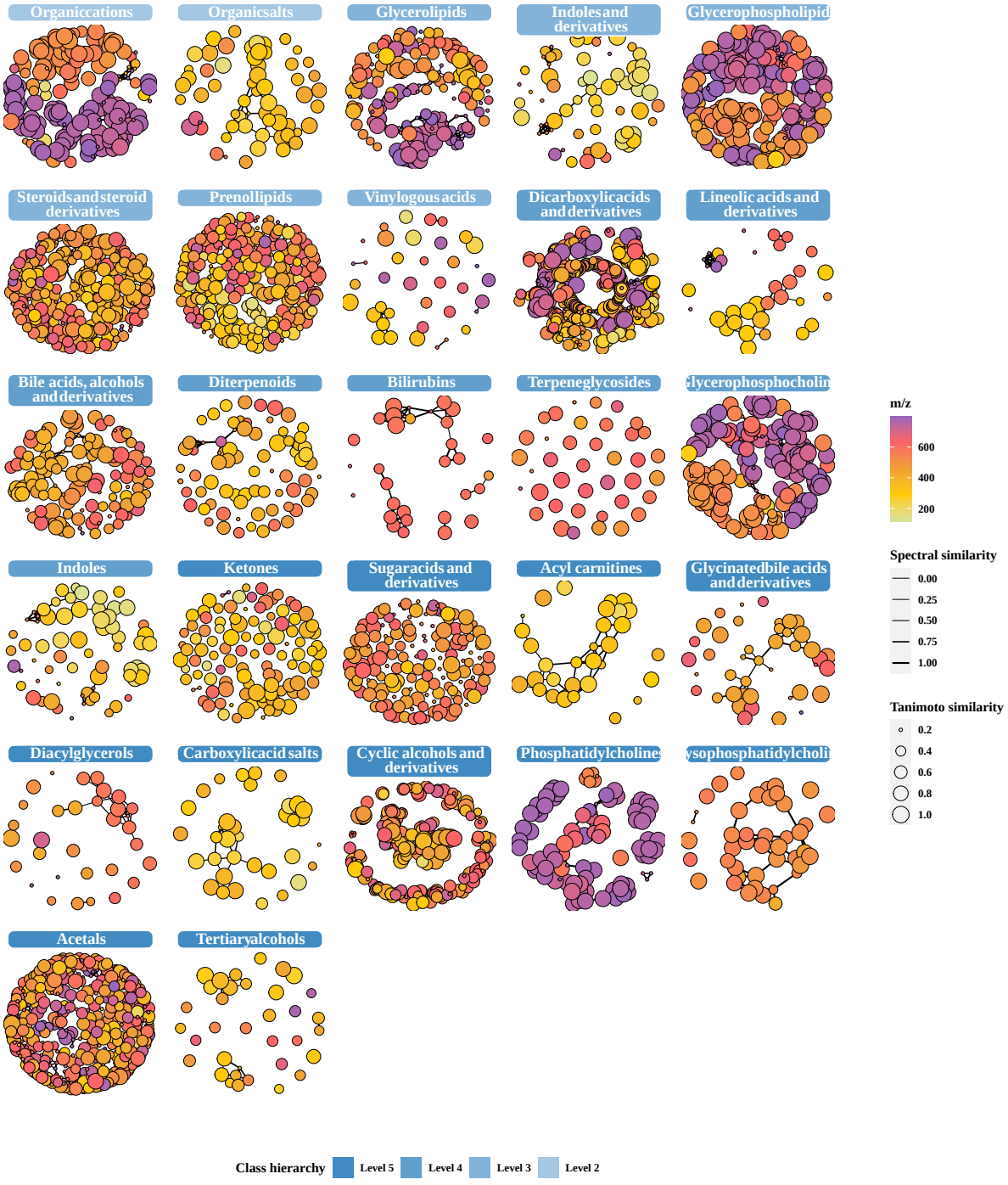

Figure 2: Child-Nebulae

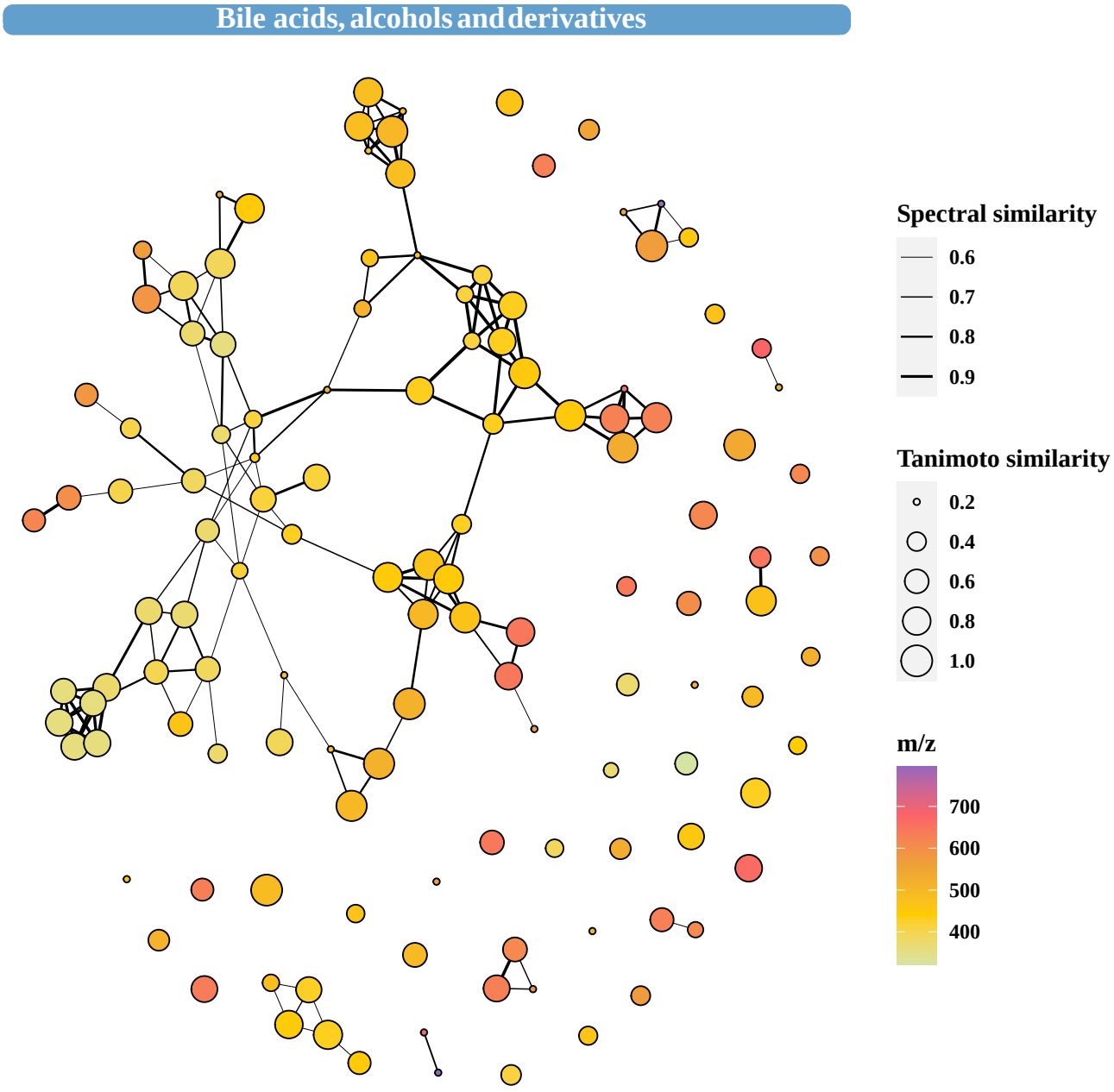

Figure 3: Bile acids, alcohols and derivatives