Basic workflow

About [script of generating report of Basic workflow](/docs/workflow/basic_workflow/#option-workflow-mode)

### 1. Initialize object

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

### The `path` is where your SIRIUS project saved.  
path <- "."  
mcn <- mcnebula()  
mcn <- initialize\_mcnebula(mcn, "sirius.v5", path)  
### "pos" or "neg"  
ion\_mode(mcn) <- "pos"

### 2. Filter candicates

#### 2.1 Filter molecular formula and chemical structure

According to the top chemical structure to build specific\_candidate of reference. [Learn more](/docs/prologue/introduction/#chemical-structure-and-formula)

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

* Tips:
  1. Use dplyr::\* in filter\_structure to customize the filtering, such as: filter\_structure(mcn, dplyr::filter, tani.score >= .5). This is similar in filter\_formula or filter\_ppcp.
* Tips (after above):
  1. Use latest(mcn, subscript = '.f3\_fingerid' ) to get the filtered candidates of the chemical structure (Top candidate for each feature)
  2. Use latest(mcn, 'project\_dataset' , subscript = '.f3\_fingerid' ) to get all candidates of all features
  3. Use reference(mcn)$specific\_candidate to get the selected specific\_candidate
  4. Use latest(mcn, subscript = '.f2\_formula' ) to get the filtered candidates of the molecular formula (Top candidate for each feature)
  5. Use latest(mcn, 'project\_dataset' , subscript = '.f2\_formula' ) to get all candidates of all features

#### 2.2 Filter chemical classification

[Learn more](/docs/prologue/introduction/#chemical-classification) about chemical classification.

mcn <- create\_stardust\_classes(mcn)  
mcn <- create\_features\_annotation(mcn)  
mcn <- cross\_filter\_stardust(  
 mcn, max\_ratio = .05,  
 cutoff = .4, identical\_factor = .6  
)

* Tips (after above):
  1. Use stardust\_classes(mcn) to get the data about’ stardust\_classes’
  2. Use features\_annotation(mcn) to get the data about …
  3. Use cross\_filter\_stardust() to get the default parameters about this methods.

#### 2.3 (Option) Manually check the filtered chemical classes

Get the kept classes or the filtered out classes.

### The kept classes  
classes <- unique(stardust\_classes(mcn)$class.name)  
### The filtered out classes  
table.filtered.classes <- backtrack\_stardust(mcn)

#### 2.4 (Option) Manually filter the chemical classes

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

### Regex match  
pattern <- c("stero", "fatty acid", "pyr", "hydroxy", "^orga")  
dis <- unlist(lapply(pattern, grep, x = classes, ignore.case = T))  
dis <- classes[dis][-1]  
### Remove these classes in `stardust\_classes(mcn)`  
mcn <- backtrack\_stardust(mcn, dis, remove = T)

* Tips:
  1. Use backtrack\_stardust(mcn, dis) without remove to recover these classes.

### 3. Create Nebula-Index

After following, 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.

mcn <- create\_nebula\_index(mcn)

* Tips (after about):
  1. Use nebula\_index(mcn) to get the data …

### 4. Create Nebulae

#### 4.1 Compute spectral similarity

See help(compute\_spectral\_similarity) about the algorithm.

mcn <- compute\_spectral\_similarity(mcn)

#### 4.2 Create Parent-Nebula

Create network data for Parent-Nebula.

mcn <- create\_parent\_nebula(mcn)  
mcn <- create\_parent\_layout(mcn)

* Tips (after above):
  1. Use function of igraph::write\_graph to output .graphml (e.g., use for Cytoscape), such as: igraph::write\_graph(igraph(parent\_nebula(mcn)), 'test.graphml' , 'graphml' )
  2. Use parent\_nebula(mcn) to get the data …
  3. Use igraph(parent\_nebula(mcn)) to get the data …
  4. …

#### 4.3 Create Child-Nebulae

Create network data for Child-Nebulae.

mcn <- create\_child\_nebulae(mcn)  
mcn <- create\_child\_layouts(mcn)

* Tips (after above):
  1. See help(create\_child\_nebulae) for example about how to export ‘.graphml’ of all Child-Nebulae.

#### 4.4 Activate the Nebulae for visualization

This would create’ ggset’ about functions and parameters for ggplot2 to visualize the Nebulae.

mcn <- activate\_nebulae(mcn)

* Tips (after above):
  1. Use function of get\_ggset to get the’ ggset’ of Nebulae. Such as: get\_ggset(mcn, 'parent' , modify\_set\_labs).
  2. Use method of call\_command to visualize’ ggset’ , such as: call\_command(get\_ggset(mcn, 'parent' , modify\_set\_labs)).

### 5. Visualize Nebulae

#### 5.1 Visualize Parent-Nebula

visualize(mcn, "parent")

#### 5.2 Visualize Child-Nebulae

### Visualize the specific item (number or classes name)  
visualize(mcn, 1)  
### Visualize all into one plot  
visualize\_all(mcn)

### 6. (Option) Workflow Mode

Use the following command to get the code that outputs the full report containing the above analysis. You can modify the parameters on the basis of the output code to fit your data or your needs.

workflow(mode = "print")

* Tips: Using sink() to catch the text output in the command lines, such as:

file <- tempfile(fileext = ".R")  
sink(file)  
workflow(mode = "print")  
sink()