R codes of build and test MCnebula2

Contents

1 File: combine_fig.pathway.R

```r
# ==========================================================================
# combine 3 figures of png of pathway enrichment
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
pngs <- list.files(pattern = "pathway.*\\.png")
pngs <- lapply(pngs, png::readPNG)
names(pngs) <- c("ACs", "BAs", "LPCs")

pngs <- lapply(pngs,
  function(png) {
    grid.grabExpr(grid.raster(png))
  })

grobs <- lapply(1:3,
  function(n, x = c("a", "b", "c")) {
    into(grecta(x[n], cex = 3), pngs[[ n ]])
  })
```

```r
names(grobs) <- names(pngs)

grobs.path <- frame_row(c(ACs = 1, BAs = 1, LPCs = 1), grobs)
grobs.path <- ggather(grobs.path, vp = viewport(, , .95, .95))

pdf("pathway.pdf", 7, 7)
draw(grobs.path)
dev.off()
```

2 File: create_exMCnebula2.R

```r
# ==========================================================================
# An attached package that packages the analysis tools used for the study.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

new_package.fromFiles("~/exMCnebula2",
  c("aaa.R", "grid_draw.R", "dot_heatmap.R", "query_synonyms.R",
    "query_inchikey.R", "query_classification.R", "query_others.R",
    "output_identification.R", "pick_annotation.R", "alignment_merge.R",
    "pathway_enrichment.R", "cross_select.R", "exReport.R", "plot_EIC_stack.R"),
  path = "~/utils.tool/R/", exclude = c("MSnbase", "MetaboAnalystR"))

# list.files("~/utils.tool/inst/extdata", full.names = T)
files <- c(
  "/home/echo/utils.tool/inst/extdata/mcn_serum6501.rdata",
  "/home/echo/utils.tool/inst/extdata/mcn_herbal1612.rdata",
  "/home/echo/utils.tool/inst/extdata/serum.tar.gz",
  "/home/echo/utils.tool/inst/extdata/herbal.tar.gz",
  "/home/echo/utils.tool/inst/extdata/svg",
  "/home/echo/utils.tool/inst/extdata/toActiv30.rdata",
  "/home/echo/utils.tool/inst/extdata/toBinary5.rdata",
  "/home/echo/utils.tool/inst/extdata/toAnno5.rdata",
  "/home/echo/utils.tool/inst/extdata/evaluation.tar.gz"
)

lapply(files, file.copy, to = "~/exMCnebula2/inst/extdata",
       recursive = T)

files2 <- list(
  evaluation_workflow = c("/home/echo/outline/mc.test/evaluation_workflow/evaluation_workflow.R",
    "~/tmp_backup/R/evaluation/report.pdf"),
  eucommia_workflow = c("/home/echo/outline/mc.test/eucommia_workflow/eucommia_workflow.R",
    "~/tmp_backup/R/eucommia/temp_data/report.pdf"),
  serum_workflow = c("/home/echo/outline/mc.test/serum_workflow/serum_workflow.R",
    "~/tmp_backup/R/serum/temp_data/report.pdf"),
  mcn_principle = c("/home/echo/outline/mc.test/mcn_principle/a_elements.R",
    "/home/echo/outline/mc.test/mcn_principle/b_gather.R",
    "/home/echo/outline/mc.test/mcn_principle/figure_mech.pdf"),
  mcn_structure = c("/home/echo/outline/mc.test/mcn_structure/a_project.R",
    "/home/echo/outline/mc.test/mcn_structure/b_mcn_dataset.R",
    "/home/echo/outline/mc.test/mcn_structure/c_nebulae.R",
    "/home/echo/outline/mc.test/mcn_structure/d_across.R",
    "/mnt/data/wizard/Documents/article/MCnebula2/data_stream.pdf")
)

path <- "~/exMCnebula2/inst/extdata/scripts_evaluation"
dir.create(path)
lapply(names(files2),
  function(name) {
    dir <- paste0(path, "/", name)
    if (!file.exists(dir))
      dir.create(dir)
    lapply(files2[[ name ]], file.copy,
      to = dir, overwrite = T, recursive = T)
  })

## update simulate_and_evaluate.R
```

3 File: delete_examples.R

```r
setwd("~/MCnebula2/R")
library(magrittr)

script <- list.files(".", pattern = "\\.R$") %>%
  sapply(readLines)

target <-
  lapply(script,
         function(text){
           sig <- 0
           if_exam <- rep(F, length(text))
```

```r
          for (i in 1:length(text)) {
            if (!grepl("^#'", text[i])) {
              next
            }
            if (grepl("^#' @examples", text[i])) {
              sig <- 2
              if_exam[i] <- T
              next
            }
            if (sig == 2) {
              if (grepl("^#' @", text[i])) {
                sig <- 1
              } else {
                if_exam[i] <- T
              }
            }
          }
          text[!if_exam]
        })

lapply(target, writeLines)

path <- "~/code_backup/mcnebula2"
dir.create(path, recursive = T)

lapply(names(target),
       function(name){
         writeLines(target[[name]], paste0(path, "/", name))
       })

system("cp ~/code_backup/mcnebula2/* -t ~/MCnebula2/R")
```

4 File: eucommia_workflow.R

```r
# ==========================================================================
# workflow to process data and output report (Eucomma)
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# workflow(mode = "print")

s0.1 <- new_section("Abstract", 1, reportDoc$abstract, NULL)
```

```r
s0.2 <- new_section("Introduction", 1,
  c(reportDoc$introduction, "",
    "*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."), NULL)

s0.3 <- new_section("Set-up", 1, reportDoc$setup,
  rblock({
    library(MCnebula2)
    library(exMCnebula2)
  }, F)
)

s0.9 <- new_heading("Integrate data and Create Nebulae", 1)

s1 <- new_heading("Initialize analysis", 2)

s1.1 <- new_section2(
  c("Set SIRIUS project path and its version to initialize mcnebula object."),
  rblock({
    mcn <- mcnebula()
    mcn <- initialize_mcnebula(mcn, "sirius.v4", ".")
    ion_mode(mcn) <- "neg"
  }, eval = F)
)

s1.5 <- new_section2(
  c("Create a temporary folder to store the output data."),
  rblock({
    tmp <- paste0(tempdir(), "/temp_data")
    dir.create(tmp, F)
  })
)

s1.6 <- new_section2(
  c("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."),
  rblock({
    exfiles <- system.file("extdata", package = "exMCnebula2")
  }))

s1.7 <- new_section2(
  c("Load the '.rdata' file."),
  rblock({
    load(paste0(exfiles, "/mcn_herbal1612.rdata"))
    mcn <- mcn_herbal1612
    rm(mcn_herbal1612)
    export_path(mcn) <- tmp
  })
)

s2 <- new_heading("Filter candidates", 2)

s2.1 <- new_section2(
  reportDoc$filter,
  rblock({
    mcn <- filter_structure(mcn)
    mcn <- create_reference(mcn)
    mcn <- filter_formula(mcn, by_reference = T)
  })
)

s3 <- new_heading("Filter chemical classes", 2)

s3.1 <- new_section2(
  reportDoc$stardust,
  rblock({
    mcn <- create_stardust_classes(mcn)
    mcn <- create_features_annotation(mcn)
    mcn <- cross_filter_stardust(mcn, max_ratio = .1, cutoff = .4, identical_factor = .6)
    classes <- unique(stardust_classes(mcn)$class.name)
    table.filtered.classes <- backtrack_stardust(mcn)
  })
)

s3.5 <- new_section2(
  c("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."),
  rblock({
    classes
    pattern <- c("fatty acid", "hydroxy")
    dis <- unlist(lapply(pattern, grep, x = classes, ignore.case = T))
    dis <- classes[dis]
    dis
  }, args = list(eval = T))
)

s3.6 <- new_section2(
  c("Remove these classes."),
  rblock({
    mcn <- backtrack_stardust(mcn, dis, remove = T)
  })
)

s4 <- new_heading("Create Nebulae", 2)

s4.1 <- new_section2(
  c("Create Nebula-Index data. This data created based on 'stardust_classes' data."),
  rblock({
    mcn <- create_nebula_index(mcn)
  })
)

s4.5 <- new_section2(
  reportDoc$nebulae,
  rblock({
    mcn <- compute_spectral_similarity(mcn)
    mcn <- create_parent_nebula(mcn)
    mcn <- create_child_nebulae(mcn)
  })
```

```r
)

s5 <- new_heading("Visualize Nebulae", 2)

s5.1 <- new_section2(
  c("Create layouts for Parent-Nebula or Child-Nebulae visualizations."),
  rblock({
    mcn <- create_parent_layout(mcn)
    mcn <- create_child_layouts(mcn)
    mcn <- activate_nebulae(mcn)
  })
)

s5.3 <- new_section2(
  c("The available chemical classes for visualization and its",
    "sequence in storage."),
  rblock({
    table.nebulae <- visualize(mcn)
    table.nebulae
  }, args = list(eval = T))
)

s5.6 <- new_section2(
  c("Draw and save as .png or .pdf image files."),
  rblock({
    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()
  })
)

s5.6.fig1 <- include_figure(f5.61, "parent", "Parent-Nebula")
s5.6.fig2 <- include_figure(f5.62, "child", "Child-Nebulae")

ref <- function(x) {
  paste0("(Fig. ", get_ref(x), ")")
}

s5.8 <- c(
  "In general, Parent-Nebulae", ref(s5.6.fig1),
  "is too informative to show, so Child-Nebulae", ref(s5.6.fig2),
  "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."
)

s6 <- new_heading("Nebulae for Downstream analysis", 1)

## Statistic analysis
s7 <- new_heading("Statistic analysis", 2)

s7.1 <- new_section2(
  c("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."),
  rblock({
    utils::untar(paste0(exfiles, "/herbal.tar.gz"), exdir = tmp)
    origin <- data.table::fread(paste0(tmp, "/features.csv"))
    origin <- tibble::as_tibble(origin)
  })
)

s7.2 <- new_section2(
  c("Now, let's check the columns in the table."),
  rblock({
    origin
  }, args = list(eval = T))
)

s7.3 <- new_section2(
  c("Remove the rest of the columns and keep only the columns for ID,",
    "m/z, retention time, and quantification."),
  rblock({
    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))
  })
)

s7.6 <- new_section2(
  c("Create the metadata table and store it in the `mcn` object",
    "along with the quantification data."),
  rblock({
    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
  })
)

s7.7 <- new_section2(
  c(reportDoc$statistic, "", "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()`)."),
  rblock({
    mcn <- binary_comparison(mcn, Pro - Raw)
    top.list <- top_table(statistic_set(mcn))
  })
)

s7.8 <- new_section2(
  c("Check the results."),
  rblock({
    top.list[[1]]
  }, args = list(eval = T, echo = T))
)
```

```r
## Set tracer in Child-Nebulae
s8 <- new_heading("Set tracer in Child-Nebulae", 2)

s8.1 <- new_section2(
  reportDoc$tracer,
  rblock({
    n <- 20
    tops <- select_features(
      mcn, tani.score_cutoff = .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)
  })
)

s8.2 <- new_section2(
  c("Draw and save the image."),
  rblock({
    pdf(f8.2 <- paste0(tmp, "/tracer_child_nebula.pdf"), 12, 14)
    visualize_all(mcn2)
    dev.off()
  })
)

s8.2.fig1 <- include_figure(f8.2, "tracer", "Tracing top features in Child-Nebulae")

s8.3 <- c("A part of the top features are marked with colored nodes in",
          "Child-Nebulae", paste0(ref(s8.2.fig1), "."))

## Quantification in Child-Nebulae
s9 <- new_heading("Quantification in Child-Nebulae", 2)

s9.1 <- new_section2(
  c("Show Fold Change (Pro versus Raw) in Child-Nebulae."),
  rblock({
    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()
  })
)

s9.1.fig1 <- include_figure(f9.1, "logFC", "Show log2(FC) in Child-Nebulae")

s9.2 <- c("Each Child-Nebula separately shows the overall content variation of",
          "the chemical class to which it belongs", ref(s9.1.fig1), ".")

## Annotate Nebulae
s10 <- new_heading("Annotate Nebulae", 2)

s10.0 <- new_section2(
  c("Now, the available Nebulae contains:"),
  rblock({
    table.nebulae2 <- visualize(mcn2)
    print(table.nebulae2, n = Inf)
  }, args = list(eval = T))
)

s10.1 <- new_section2(
  c("Next, let us focus on 'Lignans, neolignans and related compounds' and",
    "'Iridoids and derivatives'. They were representative chemical classes in",
    "E. ulmoides."),
  rblock({
    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)
  })
)

s10.2 <- new_section2(
  c("Draw and save the image."),
  rblock({
    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)
  })
)

# pngGrob <- zoom_pdf(f10.2.2, dpi = 1000, position = c(.3, .5))
# grid.draw(pngGrob)

s10.2.fig1 <- include_figure(f10.2.1, "lig", paste0("Annotated Nebulae: ", lig))
s10.2.fig2 <- include_figure(f10.2.2, "iri", paste0("Annotated Nebulae: ", iri))

s10.3 <- c(
  "See results (Fig.", paste0(get_ref(s10.2.fig1), " and ", get_ref(s10.2.fig2)),
  ").", "", reportDoc$annotate
)

s10.4 <- new_section2(
  c("Use the `show_node` function to get the annotation details",
    "for a feature. For example:"),
  rblock({
    ef <- "1642"
    pdf(f10.4 <- paste0(tmp, "/features_", ef, ".pdf"), 11, 4)
    show_node(mcn2, ef)
    dev.off()
  })
)

s10.4.fig1 <- include_figure(f10.4, "ef", "The annotated feature of ID: 1642")

s10.5 <- c(
  "See results", ref(s10.4.fig1),
  paste0("(feature in ", get_ref(s10.2.fig2), ")"), "."
)

## Output identification table
s11 <- new_heading("Query compounds", 2)

s11.1 <- c("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.")

s11.2 <- new_section2(
  c("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."),
  rblock({
    feas <- features_annotation(mcn2)
    feas <- merge(feas, top.list[[1]], by = ".features_id", all.x = T)
    feas <- dplyr::mutate(feas, arrange.rank = adj.P.Val)
    feas <- format_table(feas, export_name = NULL)
    key2d <- feas$inchikey2d
  })
)

s11.3 <- new_section2(
  c("Create a folder to store the acquired data."),
  rblock({
    tmp2 <- paste0(tmp, "/query")
    dir.create(tmp2, F)
  })
)

s11.4 <- new_section2(
  c("Query the compound's InChIKey, chemical class, IUPUA name.",
    "If your system is not Linux, the multithreading below may pose some problems,",
    "please remove the parameters `curl_cl = 4` and `classyfire_cl = 4`."),
  rblock({
    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)
```

```r
  })
)

s11.5 <- new_section2(
  c("We will also query for synonyms of compounds, but this is done in",
    "'CID' (PubChem's ID), so some transformation is required."),
  rblock({
    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)
  })
)

s11.6 <- new_section2(
  c("Screen for unique synonyms and chemical classes for all compounds."),
  rblock({
    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"))
  })
)

s11.7 <- new_section2(
  c("The formatted table as following:"),
  rblock({
    feas.table
  }, args = list(eval = T))
)

## Plot spectra of top 'features'
s12 <- new_heading("Plot spectra of top 'features'", 2)

s12.1 <- new_section2(
  c("Drawing of MS/MS spectra of top 'features'."),
  rblock({
    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()
  })
)

s12.2 <- include_figure(f12.1, "msmsTops", "MS/MS spectra of top features (identified)")

s12.3 <- c("See results", paste0(ref(s12.2), "."))

s12.5 <- new_section2(
  c("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.)"),
  rblock({
    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"))
  }, F)
)

s12.6 <- new_section2(
  c("Load the saved data and draw the figure."),
  rblock({
    load(paste0(tmp, "/eic_data.rdata"))
    pdf(f12.6 <- paste0(tmp, "/eic_tops_identified.pdf"), 12, 10)
    print(data$p)
    dev.off()
  })
)

s12.7 <- new_section2(
  c("Or use following to re-plot the 'ggplot' object."),
  rblock({
    data <- plot_EIC_stack(data = data, palette = palette_stat(mcn2))
  }, F, args = list(eval = F))
)
```

```
s12.8 <- include_figure(f12.6, "eic",
  "Extracted Ions Chromatograph (EIC) of top features (identified)")

s12.9 <- c("See results", paste0(ref(s12.8), "."),
  "It can be assumed that 1642, 1785, and 2321 are newly generated compounds after",
  "the processing. According to Fig. \\??, they were belong to chemical",
  "classes of 'Iridoids and derivatives', 'Dialkyl ethers' and",
  "'Phenylpropanoids and polyketides'."
)

s13 <- new_heading("Discover more around top 'features' in Child-Nebulae", 2)

s13.1 <- new_section2(
  c("Plot the Child-Nebulae of the two chemical classes that we have not annotated",
    "yet."),
  rblock({
    dia <- "Dialkyl ethers"
    phe <- "Phenylpropanoids and polyketides"
    mcn2 <- annotate_nebula(mcn2, dia)
    mcn2 <- annotate_nebula(mcn2, phe)
    p <- visualize(mcn2, dia, annotate = T)
    ## c(.3, .4)
    ggsave(f13.1.1 <- paste0(tmp, "/dia_child.pdf"), p, width = 6, height = 4)
    p <- visualize(mcn2, phe, annotate = T)
    ## c(.5, .8)
    ggsave(f13.1.2 <- paste0(tmp, "/phe_child.pdf"), p, width = 6, height = 4)
  })
)

s13.2 <- new_section2(
  c("Draw their partial views and put them together."),
  rblock({
    grob_iri <- zoom_pdf(f10.2.2, c(.28, .515), c(.1, .1), dpi = 3000)
    grob_dia <- zoom_pdf(f13.1.1, c(.25, .3), c(.1, .1), dpi = 3000)
    grob_phe <- zoom_pdf(f13.1.2, c(.465, .825), c(.06, .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))
  })
```

```r
)

s13.3 <- new_section2(
  c("We found interesting adjacent compounds (ID:2110 and ID:854) in `local_1642`,",
    "which has a similar chemical structure to 'feature' 1642."),
  rblock({
    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 <- grid.grabExpr(plot_msms_mirrors(mcn2, c("2110", "854"), structure_vp = NULL))
    grob_msms2110 <- into(grecta("f"), grob_msms2110)
  })
)

s13.4 <- new_section2(
  c("Again, we don't run the following code, but we save the results."),
  rblock({
    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"))
  }, F)
)

s13.5 <- new_section2(
  c("Load the data and convert picture."),
  rblock({
    load(paste0(tmp, "/eic_data2110.rdata"))
    grob_eic2110 <- as_grob(data$p)
    grob_eic2110 <- into(grecta("g"), grob_eic2110)
  })
)

s13.6 <- new_section2(
  c("Draw the final figure."),
  rblock({
    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(, , .95, .95))
    pdf(f13.6 <- paste0(tmp, "/complex.pdf"), 14, 10)
    draw(frame)
    dev.off()
  })
)

s13.7 <- include_figure(f13.6, "complex",
  "Discover chemical changes using MCnebula")

s13.8 <- c("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. \\??).")

s100 <- new_heading("Session infomation", 1)

s100.1 <- rblock({
  sessionInfo()
}, args = list(eval = T))
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

sections <- gather_sections()
report <- do.call(new_report, sections)
yaml(report)[1] <- c("title: Analysis on *E. ulmoides* dataset")
render_report(report, file <- paste0(tmp, "/report.Rmd"))
rmarkdown::render(file)
```

```r
# ==========================================================================
# as biocStyle
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# library(MCnebula2)

# write_biocStyle(report, file2 <- paste0(tmp, "/report_biocStyle_nofloat.Rmd"),
#   title <- paste0(yaml(report)[1], "\nauthor: 'LiChuang Huang'")
# )

# rmarkdown::render(file2)
```

```r
# ==========================================================================
# draw extra...
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# p <- visualize(mcn, "parent")

pgg <- modify_set_labs(mcn@parent_nebula@ggset, mcn)
pgg <- mutate_layer(pgg, 3, stroke = 0, color = "transparent")
# ggsave("herbal_parent.pdf", call_command(pgg), width = 7, height = 7)

mcn@child_nebulae@ggset %<>%
  lapply(function(ggset) {
    mutate_layer(ggset, 3, stroke = 0, color = "transparent")
})

pc <- nebulae_as_grob(mcn)
frame <- frame_col(c(p = 1.2, pc = 1), namel(p = as_grob(call_command(pgg)), pc))
# dev.new(width = 23, height = 14)

pdf("parent_and_child.pdf", 23 * .8, 14 * .8)
draw(frame)
dev.off()

# 2110 -> 854
# 2110 -> 1642

# library(MCnebula2)

# grob_struc2110 <- grid.grabExpr(show_structure(mcn2, "2110"))
# grob_struc1642 <- grid.grabExpr(show_structure(mcn2, "1642"))
# grob_struc854 <- grid.grabExpr(show_structure(mcn2, "854"))
# lst <- namel(grob_struc2110, grob_struc1642, grob_struc854)
# setwd(tmp)

# ids <- c("2110", "1642", "854")
# data <- dplyr::filter(feas, .features_id %in% ids)
# data.table::fwrite(data, "compounds_1642s.csv")
# for (i in names(lst)) {
```

```r
#   pdf(paste0(i, ".pdf"))
#   draw(lst[[ i ]])
#   dev.off()
# }

# pdf("compounds_1642_surounds.pdf", 8, 6)
# draw(grob_iri)
# dev.off()

# mcn2 <- activate_nebulae(mcn2)
# p <- visualize(mcn2, "Iridoids and derivatives",
#   function(ggset, x = mcn2) {
#     modify_default_child(modify_tracer_node(ggset), x)
#   })
# ggsave("iri.pdf", p, width = 4, height = 4)
```

5 File: evaluation_workflow.R

```r
# ==========================================================================
# script for evaluation
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

s0.1 <- new_heading("Introduction", 1)

s0.2 <- c(
  "This document provides the code to evaluate MCnebula2. Most of the data used",
  "in this is already included in the package 'exMCnebula2'. By downloading and",
  "installing the 'exMCnebula2' package, the following code can be run without",
  "trouble. The code in this document can be divided into two parts:",
  "",
  "- The first part is the code used to generate the predicate dataset (for",
  "evaluation), which is time consuming to run, but we have already run it in",
  "advance and included the outcome data in the package 'exMCnebula2', so users",
  "do not need to rerun them, unless they are tested for feasibility.",
  "",
  "- The second part of the code is the code for evaluating the results, including",
  "the data processing and data visualization modules, which is lightweight and",
  "can be run quickly to get the results."
)

s0.3 <- new_heading("Set-up", 1)
```

```
s0.4 <- new_section2(
  c("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."),
  rblock({
    library(MCnebula2)
    library(exMCnebula2)
  }, F)
)

s0.5 <- new_heading("Initialisation", 1)

s0.9 <- new_section2(
  c("Create a temporary folder to store the output data."),
  rblock({
    tmp <- paste0(tempdir(), "/temp_data")
    dir.create(tmp, F)
    exfiles <- system.file("extdata", package = "exMCnebula2")
    utils::untar(paste0(exfiles, "/evaluation.tar.gz"), exdir = tmp)
    source(paste0(tmp, "/evaluation.R"))
  })
)

s1 <- new_heading("Use simulated dataset", 1)

s1.1 <- new_heading("Convert .msp as .mgf", 2)

s1.2 <- new_section2(
  c("We downloaded a collated spectral data (.msp file) originating from GNPS from a",
    "third-party website.", "(<http://prime.psc.riken.jp/compms/msdial/main.html#MSP>)",
    "The .msp was converted as .mgf file and, at the same time, the simulated",
    "isotope peaks were added to MS1. Then the .mgf file could be used for",
    "SIRIUS computation. The following would not be evaluated, as the converted",
    ".mgf file has been included in .tar file."),
  rblock({
    if (!requireNamespace("rcdk", quietly = T))
      install.packages("rcdk")
    msp_to_mgf(
      name = "origin_gnps_pos.msp",
      id_prefix = "gnps",
      path = tmp,
      mass_level = "all",
      fun = "deal_with_msp_record")
  }, F)
)

s1.3 <- new_heading("Query classification for compounds", 2)

s1.4 <- new_section2(
  c("In order for this data to be used for evaluation, we need to first obtain the",
    "classes of these compounds. (The results has been included in .tar files.",
    "So the following need not to be run again, as its time-consuming.).",
    "(Please note that the following cl (`curl_cl`, `classyfire_cl`) sets the number of threads,",
    "if it is not a Unix like system, there may be an error, please set this parameter to",
    "`NULL`, or refer to <https://github.com/psolymos/pbapply>.)",
    ),
  rblock({
    mgf_metadata <- data.table::fread(
      paste0(tmp, "/", "origin_gnps_pos.meta.tsv")
    )
    # mgf_metadata <- data.table::fread(
    # paste0("/media/echo/back/test_mcnebula/mgf/used", "/", "origin_gnps_pos.meta.tsv")
    # )
    dir.create(tmp1 <- paste0(tmp, "/query"))
    key2d <- unique(stringr::str_extract(mgf_metadata$INCHIKEY, "^[A-Z]*"))
    key.rdata <- query_inchikey(key2d, tmp1, curl_cl = 10)
    class.rdata <- query_classification(key2d, tmp1, classyfire_cl = 20)
  }, F)
)

s1.5 <- new_heading("Add noise peak", 2)

s1.6 <- new_section2(
  c("The 'noise' include mass shift, peak intensity shift, and external",
    "noise peaks for original spectra. (refer to <https://doi.org/10.1038/s41587-021-01045-9>).",
    "First, we collected a subset of the noise peaks used to insert the spectra:"),
  rblock({
    load(files[1])
    non_noise <- dplyr::select(
      latest(mcn, "project_dataset", ".f3_spectra"),
      .features_id, mz, rel.int.
    )
    non_noise <- split(non_noise, ~ .features_id)
    non_noise <- lapply(non_noise, dplyr::select, -.features_id)
    rm(mcn)
    origin_lst <- filter_mgf(NULL, paste0(tmp, "/origin_gnps_pos.mgf"))
    # origin_lst <- filter_mgf(NULL, "/media/echo/back/test_mcnebula/mgf/used/origin_gnps_pos.mgf")
    noise_pool <- collate_as_noise_pool(origin_lst, non_noise)
  }, F)
)

s1.7 <- new_section2(
  c("Two noise models were simulated:"),
  rblock({
    medium_noise_lst <- spectrum_add_noise(
      origin_lst,
      int.sigma = 1,
      global.sigma = 10/3 * 1e-6,
      indivi.sigma = 10/3 * 1e-6,
      sub.factor = 0.03,
      alpha = 0.2,
      .noise_pool = noise_pool
    )
    high_noise_lst <- spectrum_add_noise(
      origin_lst,
      int.sigma = 2^(1/2),
      global.sigma = 15/3 * 1e-6,
      indivi.sigma = 15/3 * 1e-6,
      sub.factor = 0.03,
      alpha = 0.4,
      .noise_pool = noise_pool
    )
  }, F)
)

s1.8 <- new_heading("Output .mgf for SIRIUS", 2)

s1.85 <- new_section2(
  c("Export the above three list data to the .mgf format required by SIRIUS."),
  rblock({
    lst <- list(
      origin = origin_lst,
      medium_noise = medium_noise_lst,
      high_noise = high_noise_lst
    )
    dir.create(tmp2 <- paste0(tmp, "/simutate"))
    lapply(names(lst),
      function(name) {
        data <- data.table::rbindlist(lst[[ name ]])
        write.table(
          data, paste0(tmp2, "/", name, "_gnps_pos.sirius.mgf"),
          quote = F, col.names = F, row.names = F
        )
      })
  }, F)
)

s1.9 <- new_heading("Output .mgf for GNPS", 2)

s1.95 <- new_section2(
  c("Export the above three list data to the .mgf format required by GNPS (FBMN).",
    "(<https://doi.org/10.1038/s41592-020-0933-6>).",
    "FBMN worklow required two types of data for upload, .mgf file and",
    ".csv (quantification table)"),
  rblock({
    quant_table <- simulate_gnps_quant(
      mgf_metadata, tmp2, return_df = T
    )
    lapply(names(lst),
      function(name) {
        lst <- discard_level1(lst[[ name ]])
        lst <- pbapply::pblapply(lst, mgf_add_anno.gnps)
        data <- data.table::rbindlist(lst)
        data <- dplyr::mutate(
          data, V1 = gsub("RTINSECONDS=", "RTINSECONDS=1000", V1),
          V1 = gsub("CHARGE=+1", "CHARGE=1+", V1),
          V1 = gsub("FEATURE_ID=gnps", "FEATURE_ID=", V1)
        )
        write.table(
          data, paste0(tmp2, "/", name, "_gnps_pos.gnps.mgf"),
          quote = F, col.names = F, row.names = F
        )
        quant <- dplyr::filter(
```

```r
          quant_table, `row m/z` <= 800, `row ID` %in% names(lst)
        )
        quant$`row ID` <- stringr::str_extract(quant$`row ID`, "[0-9]{1,}$")
        write.table(
          quant, paste0(tmp2, "/", name, "_gnps_pos.gnps.meta.csv"),
          sep = ",", row.names = F, col.names = T, quote = F)
      })
  }, F)
)

s2 <- new_heading("Evaluate MCnebula2", 1)

s2.05 <- new_heading("Use pre-computed data", 2)

s2.05 <- new_section2(
  c("Extract the required data from the pre-computed SIRIUS projects, using the",
    "following code:"),
  rblock({
    dirs <- c("origin", "medium_noise", "high_noise")
    lst <- lapply(dirs,
      function(dir){
        mcn <- collated_used()
        mcn <- mcnebula()
        mcn <- initialize_mcnebula(mcn, "sirius.v4", dir)
        mcn <- collate_used(mcn)
      })
  }, F)
)

s2.1 <- new_heading("Integrate data", 2)

s2.11 <- new_section2(
  c("Integrate data to classifiy the 'features' via MCnebula2.",
    "(The following .rdata files were not provided in packages of 'exMCnebula2'.",
    "But you can downloaded them via URL such as:",
    "<https://raw.githubusercontent.com/Cao-lab-zcmu/utils_tool/master/inst/extdata/evaluationLarge/orig
    ),
  rblock({
    files <- paste0(
      system.file("extdata/evaluationLarge", package = "utils.tool"), "/",
      c("origin_gnps_pos.rdata", "medium_noise_gnps_pos.rdata", "high_noise_gnps_pos.rdata")
    )
    lst <- lapply(files,
      function(file) {
        load(file)
        mcn <- filter_structure(mcn)
        mcn <- create_reference(mcn)
        mcn <- filter_formula(mcn, by_reference = T)
        mcn <- create_stardust_classes(mcn)
        mcn <- create_features_annotation(mcn)
        mcn <- cross_filter_stardust(
          mcn, min_number = 50, max_ratio = .1, cutoff = .4, identical_factor = .6
        )
        mcn <- create_nebula_index(mcn, force = T)
        if (file == files[1]) {
          mcn <- create_hierarchy(mcn)
          mcnHier. <- mcnebula()
          reference(mcn_dataset(mcnHier.))$hierarchy <- hierarchy(mcn)
          save(mcnHier., file = paste0(system.file("extdata/evaluation",
                package = "utils.tool"), "/mcnHier.rdata"))
        }
        list(features_annotation = features_annotation(mcn),
          nebula_index = nebula_index(mcn))
      })
    names(lst) <- dirs
    save(lst, file = paste0(system.file("extdata/evaluation",
          package = "utils.tool"),
        "/integrated.rdata"))
  }, F)
)

s2.2 <- new_heading("Use pre-integrate data", 2)

s2.21 <- new_section2(
  c("Load the integrated data in the 'exMCnebula2' package. This data contains",
    "the results of the three levels, as well as the corresponding classified",
    "results and identification results."),
  rblock({
    load(paste0(tmp, "/integrated.rdata"))
    # load(paste0(exfiles, "/evaluation/integrated.rdata"))
  })
)
```

```r
s2.3 <- new_heading("Download results of finished jobs from GNPS service", 2)

s2.31 <- new_section2(
  c("In order to compare MCnebula2 with MolNetEnhancer in GNPS, we have",
    "pre-uploaded and completed jobs of FBMN and MolNetEnhancer.",
    "The URL of finished jobs in GNPS service were as following.",
    "(MolNetEnhancer: <https://doi.org/10.1101/654459>)"),
  rblock({
    fbmn_lst <- list(
      origin =
        "https://gnps.ucsd.edu/proteosafe/status.jsp?task=05f492249df5413ba72a1def76ca973d",
      medium_noise =
        "https://gnps.ucsd.edu/proteosafe/status.jsp?task=c65abe76cd9846c99f1ae47ddbd34927",
      high_noise =
        "https://gnps.ucsd.edu/proteosafe/status.jsp?task=62b25cf2dcf041d3a8b5593fdbf5ac5e"
    )
    molnet_lst <- list(
      origin =
        "https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=9d9c7f83fa2046c2bf615a3dbe35ca62",
      medium_noise =
        "https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=7cc8b5a2476f4d4e90256ec0a0f94ca7",
      high_noise =
        "https://gnps.ucsd.edu/ProteoSAFe/status.jsp?task=f6d08a335e814c5eac7c97598b26fb80"
    )
  }, F)
)

s2.32 <- new_section2(
  c("The data that would be used has been extracted."),
  rblock({
    molnet_files <- paste0(tmp, "/", "gnps_results")
    # molnet_files <- paste0(exfiles, "/evaluation/gnps_results")
    molnet_lst <- sapply(c("origin", "medium_noise", "high_noise"), simplify = F,
      function(dir) {
        data.table::fread(paste0(molnet_files, "/", dir, "/ClassyFireResults_Network.txt"))
      })
  })
)

s2.4 <- new_heading("Evaluate accuracy of classify", 2)
```

```r
s2.5 <- c("Evaluate the 'features' of each chemical class within the Nebula-Index:",
  "whether the compounds relative to the chemical class.")

s2.51 <- new_heading("Load assessment data and reference data.", 3)

s2.511 <- new_section2(
  c("The evaluation and reference data have already been saved in the previous",
    "steps and only need to be loaded:"),
  rblock({
    mgf_metadata <- data.table::fread(
      paste0(tmp, "/", "origin_gnps_pos.meta.tsv")
    )
    id2key <- stringr::str_extract(mgf_metadata$INCHIKEY, "^[A-Z]*")
    names(id2key) <- mgf_metadata$.id
    id2key <- as.list(id2key)
    load(paste0(tmp, "/mcnHier.rdata"))
    class.db <- extract_rdata_list(paste0(tmp, "/classification.rdata"))
  })
)

s2.52 <- new_heading("Filter assessment data.", 3)

s2.521 <- new_section2(
  c("Filtered according to: the common 'features' (at least identified to the",
    "chemical formula) in the three levels; whether included in reference data (`class.db`)."),
  rblock({
    common <- lapply(lst, function(l)  l$features_annotation$.features_id )
    common <- common[[1]][
      (common[[1]] %in% common[[2]]) &
      (common[[1]] %in% common[[3]])
    ]
    keep <- id2key %in% names(class.db)
    common <- common[ common %in% names(id2key)[keep] ]
    ## length(common) # [1] 7524
    lst <- lapply(lst,
      function(l) {
        l$nebula_index <- dplyr::filter(
          l$nebula_index, .features_id %in% common
        )
        return(l)
      })
  })
)

s2.53 <- new_heading("Count the results.", 3)

s2.531 <- new_section2(
  c("Assess the accuracy and derive ratios for each type of result."),
  rblock({
    res <- lapply(lst,
      function(l) {
        l <- lapply(split(l$nebula_index, ~ class.name),
          function(df) df[[ ".features_id" ]])
        stat_list <- sapply(names(l), simplify = F,
          function(class.name) {
            stat_classify(
              l[[ class.name ]], class.name,
              id2key, mcnHier., class.db
            )
          })
        stat_table <- data.table::rbindlist(
          lapply(stat_list, table_app, prop = T), fill = T, idcol = T
        )
        stat_table <- dplyr::rename(stat_table, class.name = .id)
        stat_table <- dplyr::summarise_all(
          stat_table, function(x) ifelse(is.na(x), 0, x)
        )
      })
  })
)

s2.54 <- new_heading("Post-filtering.", 3)

s2.541 <- new_section2(
  c("Filter out chemical classes with insufficient number of features; keep only",
    "those contained in 'origin'."),
  rblock({
    res[[1]] <- dplyr::filter(res[[1]], sum >= 50)
    keep <- res[[1]]$class.name
    res[2:3] <- lapply(res[2:3], dplyr::filter, class.name %in% keep)
  })
)
```

```r
s2.55 <- new_heading("Distinguish the chemical class of the dominant structure.", 3)

s2.551 <- new_section2(
  c("Our reference data (`class.db`) contains only chemical classes for the",
    "dominant structure of a compound, but not for the sub-structure of a",
    "compound.  Our evaluation data (`lst`) contain both chemical classes of",
    "substructures and classes of dominant structures. This makes the assessment",
    "biased. But they are easily distinguishable:"),
  rblock({
    dominant_res <- lapply(res, dplyr::filter, false < .4)
    sub_res <- lapply(res, dplyr::filter, false >= .4)
    summarise <- function(df) {
      false <- round(mean(df$false) * 100, 1)
      sum <- round(mean(df$sum), 1)
      list(false = false, sum = sum)
    }
    summary <- lapply(dominant_res, summarise)
  })
)

s2.56 <- new_section2(
  c("These chemical classes represent mostly local structures in compounds (they",
    "represent very small structures that are present in many other classes of",
    "compounds):"),
  rblock({
    sub_res[[1]]$class.name
  }, args = list(eval = T))
)

s2.57 <- new_heading("Visualizaion", 3)

s2.571 <- new_section2(
  c("Visualize the results of evaluation:"),
  rblock({
    vis_lst <- lapply(dominant_res, visualize_stat, mcn = mcnHier.)
    pdfs <- list()
    for (i in names(vis_lst)) {
      pdfs[[ i ]] <- paste0(tmp, "/", i, "_classified_accuracy.pdf")
      pdf(pdfs[[ i ]], 13, 11)
      draw(vis_lst[[ i ]])
      dev.off()
    }
  })
)

s2.572 <- include_figure(pdfs[[ 1 ]], "origin",
  "Classified accuracy (MCnebula2) of origin dataset")
s2.573 <- include_figure(pdfs[[ 2 ]], "medium",
  "Classified accuracy (MCnebula2) of medium noise dataset")
s2.574 <- include_figure(pdfs[[ 3 ]], "high",
  "Classified accuracy (MCnebula2) of high noise dataset")

ref <- function(x) {
  paste0("(Fig. ", get_ref(x), ")")
}
s2.575 <- c(
  paste0("Totally ", length(common), " compounds used for evaluation."),
  "For the origin dataset, the average false rate of MCnebula2 classifying is",
  paste0(summary$origin$false, "% ", ref(s2.572), ";"),
  "the average classified number of 'features' is",
  paste0(summary$origin$sum, "."),
  "For the medium noise dataset, the average false rate of MCnebula2 classifying is",
  paste0(summary$medium_noise$false, "% ", ref(s2.573), ";"),
  "the average classified number of 'features' is",
  paste0(summary$medium_noise$sum, "."),
  "For the high noise dataset, the average false rate of MCnebula2 classifying is",
  paste0(summary$high_noise$false, "% ", ref(s2.574), "."),
  "the average classified number of 'features' is",
  paste0(summary$high_noise$sum, ".")
)

s2.576 <- new_section2(
  c("See following:"),
  rblock({
    summary
  }, args = list(eval = T, echo = T))
)

s2.577 <- new_section2(
  c("Gather three levels (origin, medium_noise, high_noise) of results:"),
  rblock({
    vis <- visualize_statComplex(
```

```r
      dominant_res, mcnHier., weight = c(pl = .7, pm = 1.1, pr = .8)
    )
    pdf(f2.577 <- paste0(tmp, "/gather_classified_accuracy.pdf"), 15.5, 13)
    draw(vis)
    dev.off()
  })
)

s2.578 <- include_figure(f2.577, "gather",
  "Classified accuracy (MCnebula2) of three levels dataset")

s2.579 <- c("See results", paste0(ref(s2.578), "."))

s2.58 <- new_heading("Compare with MolNetEnhancer", 3)

s2.581 <- new_section2(
  c("Pre-process the data."),
  rblock({
    molnet_lst <- lapply(molnet_lst,
      function(df) {
        df <- dplyr::select(df, .id = `cluster index`, ends_with("class"))
        df <- dplyr::mutate(df, .id = paste0("gnps", .id))
        df <- dplyr::filter(df, .id %in% !!common)
        lst <- lapply(2:4,
          function(n) {
            lst <- split(df, df[[ n ]])
            lst <- lst[!names(lst) %in% c("", "no matches")]
            lst <- lapply(lst,
              function(df) {
                if (length(df[[ ".id" ]]) >= 50) df[[ ".id" ]]
                else NULL
              })
          })
        lst <- unlist(lst, recursive = F)
        lst[!vapply(lst, is.null, logical(1))]
      })
  })
)

s2.582 <- new_section2(
  c("Count the results."),
  rblock({
    res_molnet <- lapply(molnet_lst,
      function(l) {
        stat_list <- sapply(names(l), simplify = F,
          function(class.name) {
            stat_classify(
              l[[ class.name ]], class.name,
              id2key, mcnHier., class.db
            )
          })
        stat_table <- data.table::rbindlist(
          lapply(stat_list, table_app, prop = T), fill = T, idcol = T
        )
        stat_table <- dplyr::rename(stat_table, class.name = .id)
        stat_table <- dplyr::summarise_all(
          stat_table, function(x) ifelse(is.na(x), 0, x)
        )
      })
    res_molnet[[1]] <- dplyr::filter(res_molnet[[1]], sum >= 50)
    keep <- res_molnet[[1]]$class.name
    res_molnet[2:3] <- lapply(res_molnet[2:3], dplyr::filter, class.name %in% keep)
    summary_molnet <- lapply(res_molnet, summarise)
  })
)

s2.583 <- new_section2(
  c("Visualize the results."),
  rblock({
    vis_lst <- lapply(
      res_molnet, visualize_stat, mcn = mcnHier.,
      weight = c(pl = 1, pm = .8, pr = .7)
    )
    pdfs2 <- list()
    for (i in names(vis_lst)) {
      pdfs2[[ i ]] <- paste0(tmp, "/", i, "_classified_accuracy_Molnet.pdf")
      w <- if (i == "origin") 15 else 11
      pdf(pdfs2[[ i ]], w, 11)
      draw(vis_lst[[ i ]])
      dev.off()
    }
    vis <- visualize_statComplex(
      res_molnet, mcnHier.,
      y_cut_left = c(50, 700),
      y_cut_right = c(800, 2000),
      y_cut_left_breaks = c(50, seq(100, 700, by = 200)),
      y_cut_right_breaks = c(1200, 1600),
    )
    pdf(f2.583 <- paste0(tmp, "/gather_classified_accuracy_Molnet.pdf"), 18, 13)
    draw(vis)
    dev.off()
  })
)

s2.584 <- include_figure(pdfs2[[ 1 ]], "originMolnet",
  "Classified accuracy (MolnetEnhancer) of origin dataset")
s2.585 <- include_figure(pdfs2[[ 2 ]], "mediumMolnet",
  "Classified accuracy (MolNetEnhancer) of medium noise dataset")
s2.586 <- include_figure(pdfs2[[ 3 ]], "highMolnet",
  "Classified accuracy (MolNetEnhancer) of high noise dataset")

s2.587 <- c(
  "For the origin dataset, the average false rate of MolNetEnhancer classifying is",
  paste0(summary_molnet$origin$false, "% ", ref(s2.584), ";"),
  "the average classified number of 'features' is",
  paste0(summary_molnet$origin$sum, "."),
  "For the medium noise dataset, the average false rate of MolNetEnhancer classifying is",
  paste0(summary_molnet$medium_noise$false, "% ", ref(s2.585), ";"),
  "the average classified number of 'features' is",
  paste0(summary_molnet$medium_noise$sum, "."),
  "For the high noise dataset, the average false rate of MolNetEnhancer classifying is",
  paste0(summary_molnet$high_noise$false, "% ", ref(s2.586), ";"),
  "the average classified number of 'features' is",
  paste0(summary_molnet$high_noise$sum, ".")
)

s2.588 <- new_section2(
  c("See following:"),
  rblock({
    summary_molnet
  }, args = list(eval = T, echo = T))
)
```

```r
s2.589 <- include_figure(f2.583, "gatherMol",
  "Classified accuracy (MolNetEnhancer) of three levels dataset")

s2.5891 <- c("See results", paste0(ref(s2.589), "."))

s2.59 <- new_section2(
  c("Compare classified number of features for two methods in three levels:"),
  rblock({
    vis <- visualize_comparison(dominant_res, res_molnet)
    pdf(f2.8 <- paste0(tmp, "/classified_number_comparison.pdf"))
    draw(vis)
    dev.off()
  })
)

s2.591 <- include_figure(f2.8, "numberComp",
  "Comparison of classified number for MCnebula2 and MolNetEnhancer"
)

s2.592 <- c("As shown", paste0(ref(s2.591), ","),
  "MCnebula2 has a higher noise tolerance than MolNetEnhancer.",
)

s2.59 <- new_heading("Summary", 3)

s2.591 <- new_section2(
  c("The following data is available."),
  rblock({
    res_parallel <- attr(vis, "data")
    comman_class <- unique(res_parallel[[1]]$class.name)
    res_mcnebula <- dominant_res
    res_molnet
    summary_mcnebula <- summary
    summary_molnet
  })
)

s2.592 <- new_section2(
  c("Also required:"),
  rblock({
    res_mcnebula_common <- lapply(
      res_mcnebula, dplyr::filter, class.name %in% comman_class
    )
    res_molnet_common <- lapply(
      res_molnet, dplyr::filter, class.name %in% comman_class
    )
    summary_mcnebula_common <- lapply(res_mcnebula_common, summarise)
    summary_molnet_common <- lapply(res_molnet_common, summarise)
  })
)

s2.593 <- new_section2(
  c("Draw the figure:"),
  rblock({
    lst_molnet <- visualize_summary(summary_molnet_common)
    lst_mcnebula <- visualize_summary(summary_mcnebula_common)
    vis <- lapply(namel(lst_mcnebula, lst_molnet),
      function(lst) {
        bar <- as_grob(lst$p.num)
        ring <- list(
          p.precision = as_grob(lst$p.precision$p.m),
          p.ratioSt = as_grob(lst$p.ratioSt$p.m),
          p.recall = as_grob(lst$p.recall$p.m)
        )
        frame <- frame_col(c(p.ratioSt = 2, p.precision = 3, p.recall = 3), ring)
        frame_col(c(bar = 2, frame = 8), c(namel(bar), namel(frame)))
      })
    vis <- frame_row(c(lst_mcnebula = 1, lst_molnet = 1), vis)
    label_mcnebula <- gtext90("MCnebula", "#4DBBD5FF")
    label_molnet <- gtext90("GNPS", "#E64B35FF")
    group_labels <- frame_row(c(label_mcnebula = 1, null = .1, label_molnet = 1),
      namel(label_mcnebula, label_molnet, null = nullGrob()))
    vis2 <- frame_col(c(group_labels = .1, vis = 5), namel(group_labels, vis))
    legend <- frame_col(c(null = 2, null = 2, null = 3, null = 3),
      list(null = nullGrob(), p.precision = lst_molnet$p.precision$p.l,
        p.ratioSt = lst_molnet$p.ratioSt$p.l))
    vis3 <- frame_row(c(vis2 = 5, legend = .5), namel(vis2, legend))
    label_sum <- gtext90("Sum number", "#709AE1FF", 0)
    label_false <- gtext90("Precision", "#FED439FF", 0)
    label_stab <- gtext90("Stability", "#91D1C2", 0)
    label_recall <- gtext90("Recall", "#D5E4A2FF", 0)
    title_labels <- frame_col(
      c(null = .2, label_sum = 2, null = .1, label_stab = 2, null = .1,
        label_false = 3, null = .1, label_recall = 3),
      namel(label_sum, label_false, label_stab, label_recall, null = nullGrob())
    )
    vis4 <- frame_row(
      c(title_labels = 1, null = .5, vis3 = 15),
      namel(title_labels, vis3, null = nullGrob())
    )
    vis4 <- ggather(vis4, vp = viewport(, , .95, .95))
    pdf(f2.59 <- paste0(tmp, "/evaluation_summary.pdf"), 17, 5)
    draw(vis4)
    dev.off()
  })
)

s2.594 <- include_figure(f2.59, "summary", "Evaluation sumary for MCnebula and benchmark method")

s2.595 <- c("Under the same conditions (common chemical classes), MCnebula outperforms the",
  "benchmark method (Fig. \\??).",
  "The formula for Relative false rate and Recall (TP / (TP + FN)):",
  "- RelativeFalseRate = 1 - (1 - FalseRate) * (1 - AverageLostRate)",
  "- Recall = (1 - RelativeFalseRate) / (1 - RelativeFalseRate + AverageLostRate)"
)

s2.6 <- new_section2(
  c("Combine figure..."),
  rblock({
    vis_summary <- into(grecta("a"), vis4)
    vis_classify <- visualize_comparison(dominant_res, res_molnet,
      from = c("MCnebula", "GNPS"))
    vis_classify <- into(grecta("b"), vis_classify)
    vis_id <- visualize_idRes(list(`No cut-off` = idRes, `0.5 cut-off` = idRes.5))
    vis_id <- into(grecta("c"), vis_id)
    frame1 <- frame_col(
      c(vis_classify = 2, vis_id = 1),
      namel(vis_classify, vis_id)
    )
    frame2 <- frame_row(
      c(vis_summary = 1, frame1 = 2),
      namel(vis_summary, frame1)
    )
    frame2 <- ggather(frame2, vp = viewport(, , .95, .95))
    pdf(fs <- paste0(tmp, "/compare_accuracy.pdf"), 18, 14)
    draw(frame2)
    dev.off()
  })
)

s2.61 <- include_figure(fs, "comP", "Overall summary")

s2.7 <- new_heading("Evaluate accuracy of identification", 2)

s2.8 <- new_section2(
  c("Evaluate the accuracy of identification with origin dataset:"),
  rblock({
    identified <- dplyr::select(
      lst[[1]]$features_annotation, .features_id, inchikey2d, tani.score
    )
    identified <- dplyr::filter(identified, !is.na(tani.score))
    # nrow(identified) # [1] 6610
    index <- dplyr::filter(lst[[1]]$nebula_index, .features_id %in% identified$.features_id)
    index <- split(index, ~ class.name)
    index <- index[names(index) %in% dominant_res[[1]]$class.name]
    ref_inchikey2d <- dplyr::mutate(
      mgf_metadata, inchikey2d = stringr::str_extract(INCHIKEY, "^[A-Z]*")
    )
    ref_inchikey2d <- dplyr::select(ref_inchikey2d, .features_id = .id, inchikey2d)
    idRes <- stat_identification(index, identified, ref_inchikey2d)
    idRes.summary <- summarise(idRes)
  })
)

s2.81 <- new_section2(
  c("Set a cut-off for 'tani.score' (Tanimoto similarity)."),
  rblock({
    identified.5 <- dplyr::filter(identified, tani.score >= .5)
    index.5 <- lapply(index, dplyr::filter, .features_id %in% identified.5$.features_id)
    idRes.5 <- stat_identification(index.5, identified.5, ref_inchikey2d)
    idRes.5.summary <- summarise(idRes.5)
  })
)
```

```r
s2.82 <- c(
  "For all identified compounds, the average false rate was",
  paste0(idRes.summary$false, "%."), "Set 0.5 for 'tani.score' as threshold,
  the average false rate was", paste0(idRes.5.summary$false, "%.")
)

s2.83 <- new_section2(
  c("Visualize the results:"),
  rblock({
    vis <- visualize_idRes(list(`No cut-off` = idRes, `0.5 cut-off` = idRes.5))
    pdf(f2.83 <- paste0(tmp, "/identified_accuracy.pdf"), 6, 9)
    draw(vis)
    dev.off()
  })
)

s2.84 <- include_figure(f2.83, "idres",
  "Identified accuracy of compounds in each classified chemical class")

s2.85 <- c("See results", paste0(ref(s2.84), "."))

s100 <- new_heading("Session infomation", 1)

s100.1 <- rblock({
  sessionInfo()
}, args = list(eval = T))
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

sections <- gather_sections()
report <- do.call(new_report, sections)
yaml(report)[1] <- c("title: Evaluation of MCnebula2")
render_report(report, file <- paste0(tmp, "/report.Rmd"))
rmarkdown::render(file)
```

6 File: ABSTRACT-MCnebula2.R

```r
## codes
```

7 File: activate_nebulae-methods.R

```
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_parent_nebula(test1, 0.01)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)

## default parameters
activate_nebulae()

test1 <- activate_nebulae(test1)
## see results
ggset(parent_nebula(test1))
head(ggset(child_nebulae(test1)))

## visualize
call_command(ggset(parent_nebula(test1)))
## or
visualize(test1, "parent")
## child nebula
call_command(ggset(child_nebulae(test1))[[1]])
## or
visualize(test1, 1)
```

8 File: annotate_nebula-methods.R

```
## codes
test <- mcn_5features
```

```r
## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

## set features quantification data
ids <- features_annotation(test1)$.features_id
quant. <- data.frame(
  .features_id = ids,
  sample_1 = rnorm(length(ids), 1000, 200),
  sample_2 = rnorm(length(ids), 2000, 500)
)
metadata <- data.frame(
  sample = paste0("sample_", 1:2),
  group = c("control", "model")
)
features_quantification(test1) <- quant.
sample_metadata(test1) <- metadata

## optional 'nebula_name'
visualize(test1)
## a class for example
class <- visualize(test1)$class.name[1]
tmp <- export_path(test1)
test1 <- annotate_nebula(test1, class)

## The following can be run before "annotate_nebula()"
## to customize the visualization of nodes.
# test1 <- draw_structures(test1, "Fatty Acyls")
## set parameters for visualization of nodes
# test1 <- draw_nodes(
#   test1, "Fatty Acyls",
#   add_id_text = T,
#   add_structure = T,
#   add_ration = T,
#   add_ppcp = T
# )
# test1 <- annotate_nebula(test1, class)

## see results
ggset <- ggset_annotate(child_nebulae(test1))
ggset[[class]]
## visualize 'ggset'
call_command(ggset[[class]])
```

9 File: backtrack__stardust-methods.R

```
## codes
```

10 File: binary__comparison-methods.R

```
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_features_annotation(test1)

## set up a simulated quantification data.
test1 <- .simulate_quant_set(test1)
## the simulated data
features_quantification(test1)
sample_metadata(test1)

test1 <- binary_comparison(
  test1, control - model,
  model - control, 2 * model - control
)
## see results
top_table(statistic_set(test1))
```

```r
## the default parameters
binary_comparison()
```

11 File: clear.R

```r
## codes
```

12 File: code_block-class.R

```r
## codes
## general
codes <- "df <- data.frame(x = 1:10)
  df<-dplyr::mutate(df,y=x*1.5)%>%
  dplyr::filter(x >= 5)
  p <- ggplot(df)+
  geom_point(aes(x=x,y=y))
  p"
block <- new_code_block("r", codes, list(eval = T, echo = T, message = T))
## see results
block
call_command(block)
writeLines(call_command(block))

## figure
fig_block <- new_code_block_figure(
  "plot1",
  "this is a caption",
  codes = codes
)
## see results
fig_block
writeLines(call_command(fig_block))
command_args(fig_block)
cat(get_ref(fig_block), "\n")

## table
codes <- "df <- data.frame(x = 1:10) %>%
  dplyr::mutate(y = x, z = x * y)
  knitr::kable(df, format = 'markdown', caption = 'this is a caption') "
tab_block <- new_code_block_table("table1", codes = codes)
```

```
## see results
tab_block
cat(get_ref(tab_block), "\n")

## default parameters
new_code_block()
```

13 File: collate_data-methods.R

```
## codes
## The raw data used for the example
tmp <- paste0(tempdir(), "/temp_data")
dir.create(tmp)
eg.path <- system.file("extdata", "raw_instance.tar.gz",
                       package = "MCnebula2")

utils::untar(eg.path, exdir = tmp)

## initialize 'mcnebula' object
test <- mcnebula()
test <- initialize_mcnebula(test, "sirius.v4", tmp)

## extract candidates data in SIRIUS project directory
## chemical structure
test <- collate_data(test, ".f3_fingerid")
latest(project_dataset(test))

## chemical formula
test <- collate_data(test, ".f2_formula")
latest(project_dataset(test))

## chemical classes
test <- collate_data(test, ".f3_canopus")
latest(project_dataset(test))

## mz and rt
test <- collate_data(test, ".f2_info")
latest(project_dataset(test))

## classification description
test <- collate_data(test, ".canopus")
```

```r
## the extracted data in 'mcnebula'
dataset(project_dataset(test))
entity(dataset(project_dataset(test))$.f3_fingerid)

unlink(tmp, T, T)
```

14 File: command-class.R

```r
## codes
## example 1
com <- new_command(plot, x = 1:10)
com
call_command(com)

## example 2
com <- new_command(data.frame, x = 1:10, y = 1:10, z = 1:10)
call_command(com)

## example 3
data <- data.frame(x = 1:10, y = 1:10)
com1 <- new_command(ggplot, data)
com2 <- new_command(geom_point, aes(x = x, y = y))
call_command(com1) + call_command(com2)

## slots
command_name(com)
command_args(com)
command_function(com)
```

15 File: compute_spectral_similarity-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
```

```r
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)

test1 <- compute_spectral_similarity(test1)
## see results
spectral_similarity(test1)
## or
reference(test1)$spectral_similarity
## or
reference(mcn_dataset(test1))$spectral_similarity

## compare the two spectra individually
spectra <- latest(test1, "project_dataset", ".f3_spectra")
data1 <- dplyr::select(
  dplyr::filter(spectra, .features_id == "gnps1537"),
  mz, int.
)
data2 <- dplyr::select(
  dplyr::filter(spectra, .features_id == "gnps1539"),
  mz, int.
)
e1 <- compute_spectral_similarity(sp1 = data1, sp2 = data2)
e1
# [1] 0.7670297

## MSnbase
if (requireNamespace("MSnbase")) {
  MSnbase::compareSpectra
  spec1 <- new("Spectrum2", mz = data1$mz, intensity = data1$int.)
  spec2 <- new("Spectrum2", mz = data2$mz, intensity = data2$int.)
  e2 <- MSnbase::compareSpectra(spec1, spec2, fun = "dotproduct")
  identical(e1, e2)
}
```

16 File: create_child_layouts-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
```

```r
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_child_nebulae(test1, 0.01)

## function to generate default parameters
create_child_layouts()
## default parameters
create_child_layouts()(test1)

test1 <- create_child_layouts(test1)
## see results (a object for 'ggraph' package to visualization)
lapply(
  layout_ggraph(child_nebulae(test1)),
  tibble::as_tibble
)
```

17 File: create_child_nebulae-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)

## default parameters
create_child_nebulae()

test1 <- create_child_nebulae(test1, 0.01)
```

```r
## see results
igraph(child_nebulae(test1))
## write output for 'Cytoscape' or other network software
tmp <- paste0(tempdir(), "/child_nebulae/")
dir.create(tmp)
res <- igraph(child_nebulae(test1))
lapply(
  names(res),
  function(name) {
    igraph::write_graph(
      res[[name]],
      file = paste0(tmp, name, ".graphml"),
      format = "graphml"
    )
  }
)
list.files(tmp)

unlink(tmp, T, T)
```

18 File: create_features_annotation-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference=T)
test1 <- create_stardust_classes(test1)

test1 <- create_features_annotation(test1)
## see results
features_annotation(test1)
## or
reference(test1)$features_annotation
## or
reference(mcn_dataset(test1))$features_annotation

## merge additional data
ids <- features_annotation(test1)$.features_id
```

```r
data <- data.frame(.features_id = ids, quant. = rnorm(length(ids), 1000, 200))
test1 <- create_features_annotation(test1, extra_data = data)
```

19 File: create_hierarchy-methods.R

```r
## codes
```

20 File: create_nebula_index-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)

test1 <- create_nebula_index(test1)
## see results
nebula_index(test1)
## or
reference(test1)$nebula_index
## or
reference(mcn_dataset(test1))$nebula_index
```

21 File: create_parent_layout-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
```

```r
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_parent_nebula(test1, 0.01)

## default parameters
create_parent_layout()

test1 <- create_parent_layout(test1)
## see results (a object for 'ggraph' package to visualization)
tibble::as_tibble(layout_ggraph(parent_nebula(test1)))
```

22 File: create_parent_nebula-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)

## default parameters
create_parent_nebula()

test1 <- create_parent_nebula(test1, 0.01)
## see results
igraph(parent_nebula(test1))
## write output for 'Cytoscape' or other network software
tmp <- tempdir()
igraph::write_graph(
  igraph(parent_nebula(test1)),
  file = paste0(tmp, "/parent_nebula.graphml"),
  format = "graphml"
)

unlink(tmp, T, T)
```

23 File: create_reference-methods.R

```r
## codes
test <- mcn_5features

## set specific candidate
## ---------------------------------------
## from chemical structure
test1 <- filter_structure(test)
test1 <- create_reference(test1)
## see results
specific_candidate(test1)
## or
reference(test1)$specific_candidate
## or
reference(mcn_dataset(test1))$specific_candidate
## 'create_reference(test1)' equals to
test1 <- create_reference(test1, from = "structure", fill = T)
e1 <- specific_candidate(test1)

## the above equals to following:
data <- latest(filter_structure(test1))
test1 <- create_reference(test1, data = data, fill = T)
e2 <- specific_candidate(test1)
identical(e1, e2)

## the 'specific_candidate' data used for filtering
test1 <- filter_formula(test1, by_reference = T)

## ---------------------------------------
## from chemical formula
test1 <- filter_formula(test1)
test1 <- create_reference(test1, from = "formula")

## ---------------------------------------
## from chemical classes
## A complex example:
## suppose there were some classes we were interested in
all_classes <- latest(test1, "project_dataset", ".canopus")$class.name
```

```r
set.seed(1)
classes <- sample(all_classes, 50)
classes
test1 <- filter_ppcp(test1,
  dplyr::filter,
  class.name %in% classes,
  pp.value > 0.5,
  by_reference = F
)
data <- latest(test1)
data
## 'feature' have a plural number of candidates.
ids <- data$.features_id
id <- unique(ids[duplicated(ids)])
## get the candidate of top chemical structural score.
`%>%` <- magrittr::`%>%`
candidates <- filter_structure(test1, dplyr::filter, .features_id %in% id) %>%
  latest() %>%
  dplyr::filter(.candidates_id %in% data$.candidates_id) %>%
  dplyr::arrange(.features_id, dplyr::desc(csi.score)) %>%
  dplyr::distinct(.features_id, .keep_all = T)
## for refecrence
data <- data %>%
  dplyr::filter(
    .features_id != candidates$.features_id |
      (.features_id == candidates$.features_id &
        .candidates_id == candidates$.candidates_id)
  )
test1 <- create_reference(test1, data = data, fill = T)
specific_candidate(test1)
```

24 File: create_stardust_classes-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)

test1 <- create_stardust_classes(test1)
```

```
## see results
stardust_classes(test1)
## or
reference(test1)$stardust_classes
## or
reference(mcn_dataset(test1))$stardust_classes

## the default parameters
create_stardust_classes()
```

25 File: cross_filter_stardust-methods.R

```
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)

## the default parameters
cross_filter_stardust()
# This is a simulated dataset with only 5 'features',
# so the default parameters are meaningless for it.

# Note that real datasets often contain thousands of "features"
# and the following 'min_number' and 'max_ratio' parameter values are not suitable.
test1 <- cross_filter_stardust(
  test1,
  min_number = 2,
  max_ratio = 1
)
## see results
stardust_classes(test1)
## or
reference(test1)$stardust_classes
## or
reference(mcn_dataset(test1))$stardust_classes
```

```r
e1 <- stardust_classes(test1)

## see the filtered classes
backtrack_stardust(test1)

## reset the 'stardust_classes'
test1 <- create_stardust_classes(test1)

## customized filtering
# Note that real datasets often contain thousands of "features"
# and the following 'min_number' and 'max_ratio' parameter values are not suitable.
test1 <- cross_filter_quantity(test1, min_number = 2, max_ratio = 1)
test1 <- cross_filter_score(test1,
  types = "tani.score",
  cutoff = 0.3,
  tolerance = 0.6
)
test1 <- cross_filter_identical(
  test1,
  hierarchy_range = c(3, 11),
  identical_factor = 0.7
)
e2 <- stardust_classes(test1)

identical(e1, e2)

## reset
test1 <- create_stardust_classes(test1)
## targeted plural attributes
test1 <- cross_filter_stardust(
  test1,
  min_number = 2,
  max_ratio = 1,
  types = c("tani.score", "csi.score"),
  cutoff = c(0.3, -150),
  tolerance = c(0.6, 0.3)
)
```

26 File: draw_nodes-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

## set features quantification data
ids <- features_annotation(test1)$.features_id
quant. <- data.frame(
  .features_id = ids,
  sample_1 = rnorm(length(ids), 1000, 200),
  sample_2 = rnorm(length(ids), 2000, 500)
)
metadata <- data.frame(
  sample = paste0("sample_", 1:2),
  group = c("control", "model")
)
features_quantification(test1) <- quant.
sample_metadata(test1) <- metadata

## optional 'nebula_name'
visualize(test1)
## a class for example
class <- visualize(test1)$class.name[1]
tmp <- export_path(test1)
test1 <- draw_structures(test1, class)
test1 <- draw_nodes(test1, class)

## see results
grobs <- nodes_grob(child_nebulae(test1))
```

```r
grobs
grid::grid.draw(grobs[[1]])
## visualize with ID of 'feature' (.features_id)
## with legend
ids <- names(grobs)
x11(width = 9, height = 5)
show_node(test1, ids[1])

## default parameters
draw_nodes()

unlink(tmp, T, T)
```

27 File: draw_structures-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

## optional 'nebula_name'
visualize(test1)
## a class for example
class <- visualize(test1)$class.name[1]
tmp <- export_path(test1)
test1 <- draw_structures(test1, class)

## see results
grobs <- structures_grob(child_nebulae(test1))
grobs
```

```
grid::grid.draw(grobs[[1]])
## visualize with ID of 'feature' (.features_id)
ids <- names(grobs)
show_structure(test1, ids[1])

unlink(tmp, T, T)
```

28 File: filter_formula-methods.R

```
## codes
test <- mcn_5features

## filter chemical formula candidates
## use default parameters
test1 <- filter_formula(test)
latest(test1)

## the default parameters:
filter_formula()

## customized filtering
## according to score
test1 <- filter_formula(test1, dplyr::filter, zodiac.score > 0.5)
latest(test1)

## get top rank
test1 <- filter_formula(test1, dplyr::filter, rank.formula <= 3)
latest(test1)

## complex filtering
test1 <- filter_formula(
  test1, dplyr::filter,
  ## molecular formula
  !grepl("N", mol.formula),
  ## mass error
  abs(error.mass) < 0.001
)
latest(test1)

## select columns
test1 <- filter_formula(test1, dplyr::select, 1:5)
```

```r
latest(test1)
```

29 File: filter_ppcp-methods.R

```r
## codes
test <- mcn_5features

## filter chemical class candidates
## the default parameters:
filter_ppcp()

## if 'by_reference' set with TRUE, 'create_reference' should be
## run previously.
test1 <- filter_ppcp(test, by_reference = F)
latest(test1)

## customized filtering
## according to score
test1 <- filter_ppcp(test1, dplyr::filter, pp.value > 0.5,
                     by_reference = F)
latest(test1)

## complex filtering
test1 <- filter_ppcp(
  test1, dplyr::filter,
  ## PPCP value
  pp.value > 0.5,
  ## speicifid class
  class.name %in% c("Azoles"),
  by_reference = F
)
latest(test1)

## select columns
test1 <- filter_ppcp(test1, dplyr::select, 1:5,
                     by_reference = F)
latest(test1)
```

30 File: filter_structure-methods.R

```r
## codes
test <- mcn_5features

## filter chemical structure candidates
## use default parameters
test1 <- filter_structure(test)
latest(test1)

## the default parameters:
filter_structure()

## customized filtering
## according to score
test1 <- filter_structure(test1, dplyr::filter, tani.score > 0.4)
latest(test1)

## get top rank
test1 <- filter_structure(test1, dplyr::filter, rank.structure <= 3)
latest(test1)

## complex filtering
test1 <- filter_structure(
  test1, dplyr::filter,
  ## molecular formula
  !grepl("N", mol.formula),
  ## Tanimoto similarity
  tani.score > 0.4
)
latest(test1)

## select columns
test1 <- filter_structure(test1, dplyr::select, 1:5)
latest(test1)
```

31 File: fun_modify.R

```r
## codes
```

32 File: gather_sections.R

```r
## codes
```

33 File: ggset-class.R

```r
## codes
data <- data.frame(x = 1:10, y = 1:10)
layer1 <- new_command(ggplot, data)
layer2 <- new_command(geom_point, aes(x = x, y = y))
layer3 <- new_command(labs, x = "x label", y = "y label")
layer4 <- new_command(theme, text = element_text(family = "Times"))

## gather
ggset <- new_ggset(layer1, layer2, layer3, layer4)
ggset
## visualize
p <- call_command(ggset)
p

## add layers
layer5 <- new_command(
  geom_text,
  aes(x = x, y = y, label = paste0("label_", x))
)
layer6 <- new_command(ggtitle, "this is title")
ggset <- add_layers(ggset, layer5, layer6)
call_command(ggset)

## delete layers
ggset <- delete_layers(ggset, 5:6)
call_command(ggset)

## mutate layer
ggset <- mutate_layer(ggset, "theme",
  legend.position = "none",
  plot.background = element_rect(fill = "red")
)
ggset <- mutate_layer(ggset, "geom_point",
  mapping = aes(x = x, y = y, color = x)
)
```

```
call_command(ggset)
```

34 File: history__rblock-methods.R

```
## codes
test1 <- 1
test2 <- 2
test3 <- 3

block <- history_rblock(, "^test1", "^test3")
block
```

35 File: include__figure-methods.R

```
## codes
tmp <- paste0(tempdir(), "/test.pdf")
pdf(tmp)
plot(1:10)
dev.off()

fig_block <- include_figure(
  tmp, "plot", "This is caption"
)
fig_block
```

36 File: include__table-methods.R

```
## codes
data <- data.frame(x = 1:10, y = 1:10)
tab_block <- include_table(
  data, "table1",
  "This is caption"
)
tab_block
```

37 File: initialize__mcnebula-methods.R

```
## codes
## The raw data used for the example
```

```r
tmp <- paste0(tempdir(), "/temp_data")
dir.create(tmp)
eg.path <- system.file("extdata", "raw_instance.tar.gz",
                       package = "MCnebula2")

utils::untar(eg.path, exdir = tmp)

## initialize 'mcnebula' object
test <- mcnebula()
test <- initialize_mcnebula(test, "sirius.v4", tmp)
## check the setting
export_path(test)
palette_set(test)
ion_mode(test)
project_version(test)

## initialize 'melody' object
test <- new("melody")
test <- initialize_mcnebula(test)
## check...
palette_stat(test)

## initialize 'project_conformation' object
test <- new("project_conformation")
test <- initialize_mcnebula(test, "sirius.v4")
## check
file_name(test)

## initialize 'project_api' object
test <- new("project_api")
test <- initialize_mcnebula(test, "sirius.v4")
## check
methods_format(test)

unlink(tmp, T, T)
```

38 File: mcn_dataset-class.R

```r
## codes
```

39 File: mcnebula-class.R

```r
## codes
test <- mcnebula()
class(test)

test <- mcn_5features
## slots
ion_mode(test)
project_version(test)
melody(test)
export_name(test)
## ...

## 'fast channel'
palette_label(test)
palette_stat(test)
sample_metadata(test)
## ...
```

40 File: melody-class.R

```r
## codes
```

41 File: msframe-class.R

```r
## codes
```

42 File: nebula-class.R

```r
## codes
```

43 File: plot_msms_mirrors.R

```r
## codes
```

44 File: project_api-class.R

```
## codes
```

45 File: project__conformation-class.R

```
## codes
```

46 File: project__dataset-class.R

```
## codes
```

47 File: project__metadata-class.R

```
## codes
```

48 File: project-class.R

```
## codes
```

49 File: rblock.R

```
## codes
rblock({
  test1 <- 1
  test2 <- 2
  test3 <- 3
})

rblock({
  test <- mcn_5features
  ## this annotation line would be ignored
  test1 <- filter_structure(test)
  test1 <- create_reference(test1)
  test1 <- filter_formula(test1, by_reference=T)
  test1 <- create_stardust_classes(test1)
})
```

50 File: render_report.R

```
## codes
```

51 File: report-class.R

```
## codes
s1 <- new_heading("Title 1", 1)

s1.5 <- new_section2(
  c("..."), NULL
)

s2 <- new_section2(
  c("This is sentence 1.",
    "This is sentence 2.",
    "This is sentence 3."),
  rblock({
    df <- data.frame(x = 1:10, y = 1:10)
    df <- dplyr::mutate(df,
      group = rep(paste0("p", 1:3), c(3, 3, 4))
    )
  })
)

s3 <- new_heading("Title 2", 1)

s4 <- new_section2(
  c("This is a paragraph..."),
  rblock({
    p <- ggplot(df) +
      geom_point(aes(x = x, y = y, fill = group))
    ggsave(f <- paste0(tempdir(), "/test.pdf"), p)
  })
)

s5 <- include_figure(f, "name", "Caption: ...")

s6 <- rblock({
  print(1:100)
}, args = list(echo = F, eval = F))
```

```r
s7 <- c("... paragraph ...")

sections <- gather_sections()
report <- do.call(new_report, sections)
render_report(report, paste0(tempdir(), "/report.Rmd"), T)

####################
##### Another example
####################

h1 <- new_heading("heading, level 1", 1)

sec1 <- new_section(
  "sub-heading", 2,
  "This is a description.",
  new_code_block(codes = "seq <- lapply(1:10, cat)")
)

h2 <- new_heading("heading 2, level 1", 1)

fig_block <- new_code_block_figure("plot", "this is a caption",
  codes = "df <- data.frame(x = 1:10, y = 1:10)
    p <- ggplot(df) +
      geom_point(aes(x = x, y = y))
    p"
)
sec2 <- new_section(
  "sub-heading2", 2,
  paste0(
    "This is a description. ",
    "See Figure ", get_ref(fig_block), "."
  ),
  fig_block
)

a_data <- dplyr::storms[1:15, 1:10]
table_block <- include_table(a_data, "table1", "This is a caption")

sec3 <- new_section(
  NULL, ,
  paste0("See Table ", get_ref(table_block, "tab"), "."),
  NULL
)

tmp_p <- paste0(tempdir(), "/test.pdf")
pdf(tmp_p)
plot(1:10)
dev.off()
fig_block_2 <- include_figure(tmp_p, "plot2", "this is a caption")
sec4 <- history_rblock(, "^tmp_p <- ", "^fig_block_2")
sec4

## gather
yaml <- "title: 'title'\noutput:\n  bookdown::pdf_document2"
report <- new_report(
  h1, sec1, h2, sec2,
  table_block, sec3,
  fig_block_2, sec4,
  yaml = yaml
)
report

## output
tmp <- paste0(tempdir(), "/tmp_output.Rmd")
render_report(report, tmp)
rmarkdown::render(tmp)
file.exists(sub("Rmd$", "pdf", tmp))
```

52 File: section-class.R

```
## codes
## -------------------------------------
## heading
new_heading("this is a heading", 2)

## -------------------------------------
## section
## example 1
para <- "This is a paragraph stating"
section <- new_section("this is a heading", 2, para)
## see results
section
```

```r
call_command(section)
writeLines(call_command(section))

## example 2
para <- "This is a paragraph stating"
section <- new_section(NULL, , para, NULL)

## example 3
para <- "This is a paragraph stating"
block <- new_code_block(codes = "df <- data.frame(x = 1:10)")
section <- new_section("heading", 2, para, block)
section

## example 4
codes <- "df <- data.frame(x = 1:10, y = 1:10)
  p <- ggplot(df) +
    geom_point(aes(x = x, y = y))
  p"
fig_block <- new_code_block_figure("plot", "this is caption", codes = codes)
para <- paste0("This is a paragraph describing the picture. ",
               "See Figure ", get_ref(fig_block), ".")
section <- new_section("heading", 2, para, fig_block)
section
## output
tmp <- paste0(tempdir(), "/tmp_output.Rmd")
writeLines(call_command(section), tmp)
rmarkdown::render(tmp, output_format = "bookdown::pdf_document2")
file.exists(sub("Rmd$", "pdf", tmp))
## see [report-class] object:
## A complete output report, including multiple 'section'.

## defalt parameters
new_section()
```

53 File: set_nodes_color-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
```

```r
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_parent_nebula(test1, 0.01)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

ids <- features_annotation(test1)$.features_id
extra_data <- data.frame(
  .features_id = ids,
  attr_1 = rnorm(length(ids), 100, 50),
  attr_2 = sample(c("special", "normal"), 5, replace = T)
)

test1 <- set_nodes_color(test1, "attr_1", extra_data)
visualize(test1, 1)
visualize_all(test1)
## set labal of the legend
export_name(test1) <- c(
  export_name(test1),
  attr_1 = "Continuous attribute",
  attr_2 = "Discrete attribute"
)
visualize_all(test1)

test1 <- set_nodes_color(test1, "attr_2", extra_data)
visualize(test1, 1)
visualize_all(test1)

## set colors for 'tracer'
test1 <- set_tracer(test1, ids[1:2])
## re-build Child-Nebulae
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)
```

```r
## set color
test1 <- set_nodes_color(test1, use_tracer = T)
visualize_all(test1)
```

54 File: set_ppcp_data-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

## optional 'nebula_name'
visualize(test1)
## a class for example
class <- visualize(test1)$class.name[1]
tmp <- export_path(test1)
## customize the chemical classes displayed
## in the radial bar plot in node.
classes <- classification(test1)
## get some random classes
set.seed(10)
classes <- sample(classes$class.name, 50)
classes
test1 <- set_ppcp_data(test1, classes)
test1 <- draw_nodes(test1, class,
  add_structure = F,
  add_ration = F
)

## visualize with ID of 'feature' (.features_id)
## with legend
ids <- names(nodes_grob(child_nebulae(test1)))
x11(width = 15, height = 5)
show_node(test1, ids[1])

## get a function to generate default parameters
set_ppcp_data()
## the default parameters
set_ppcp_data()(test1)

unlink(tmp, T, T)
```

55 File: set_ration_data-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

## set features quantification data
ids <- features_annotation(test1)$.features_id
quant. <- data.frame(
  .features_id = ids,
  sample_1 = rnorm(length(ids), 1000, 200),
  sample_2 = rnorm(length(ids), 2000, 500)
)
quant. <- dplyr::mutate(quant.,
  sample_3 = sample_1 * 1.5,
  sample_4 = sample_2 * 5
)
```

```r
metadata <- data.frame(
  sample = paste0("sample_", 1:4),
  group = rep(c("control", "model"), c(2, 2))
)
features_quantification(test1) <- quant.
sample_metadata(test1) <- metadata

## a more convenient way to obtain simulation data
# test1 <- MCnebula2:::.simulate_quant_set(test1)

## optional 'nebula_name'
visualize(test1)
## a class for example
class <- visualize(test1)$class.name[1]
tmp <- export_path(test1)

test1 <- set_ration_data(test1, mean = F)
test1 <- draw_nodes(test1, class,
  add_structure = F,
  add_ppcp = F
)

## visualize with ID of 'feature' (.features_id)
## with legend
ids <- names(nodes_grob(child_nebulae(test1)))
x11(width = 15, height = 5)
show_node(test1, ids[1])

## the default parameters
set_ration_data()

unlink(tmp, T, T)
```

56 File: set_tracer-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
```

```r
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)

ids <- features_annotation(test1)$.features_id
test1 <- set_tracer(test1, ids[1:2])
## see results
nebula_index(test1)

## see examples in 'set_nodes_color()'
```

57 File: statistic__set-class.R

```r
## codes
```

58 File: VIRTUAL__backtrack-class.R

```r
## codes
```

59 File: VIRTUAL__dataset-class.R

```r
## codes
```

60 File: VIRTUAL__export-class.R

```r
## codes
```

61 File: VIRTUAL__layerSet-class.R

```r
## codes
```

62 File: VIRTUAL__reference-class.R

```r
## codes
```

63 File: VIRTUAL_subscript-class.R

```r
## codes
```

64 File: visualize-methods.R

```r
## codes
test <- mcn_5features

## the previous steps
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference = T)
test1 <- create_stardust_classes(test1)
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 2, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_parent_nebula(test1, 0.01)
test1 <- create_child_nebulae(test1, 0.01)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

## optional Child-Nebulae
visualize(test1)

visualize(test1, "parent")
visualize(test1, 1)
visualize_all(test1)
## ...

## use 'fun_modify'
visualize(test1, 1, modify_default_child)
visualize(test1, 1, modify_unify_scale_limits)
visualize(test1, 1, modify_set_labs)
## ...
```

65 File: workflow-methods.R

```
## codes
```

66 File: export_data.R

```
# ==========================================================================
# get the data
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
tmp <- paste0(tempdir(), "/temp_data")
dir.create(tmp, F)
eg.path <- system.file("extdata", "raw_instance.tar.gz",
                       package = "MCnebula2")

utils::untar(eg.path, exdir = tmp)
```

```
# ==========================================================================
# decrease object.size
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

test <- initialize_mcnebula(mcnebula(), , tmp)
test1 <- filter_structure(test)
test1 <- filter_formula(test1)
test1 <- filter_ppcp(test1, by_reference = F)

dataset(project_dataset(test1))

## ----------------------------------------------------------------------
## chemical structure

df <- entity(dataset(project_dataset(test1))[[ ".f3_fingerid" ]])
df <- dplyr::mutate(df, links = character(1), pubmed.ids = character(1))
df <- dplyr::mutate_if(df, is.numeric, function(x) round(x, 2))

format(object.size(df), units = "MB")
dplyr::as_tibble(df)
df

entity(dataset(project_dataset(test1))[[ ".f3_fingerid" ]]) <- df

## ----------------------------------------------------------------------
## chemical formula
```

```r
df2 <- entity(dataset(project_dataset(test1))[[ ".f2_formula" ]])
df2 <- dplyr::mutate_if(df2, is.numeric, function(x) round(x, 2))

dplyr::as_tibble(df2)
df2

entity(dataset(project_dataset(test1))[[ ".f2_formula" ]]) <- df2

## ---------------------------------------------------------------------
## chmical classes

df3 <- entity(dataset(project_dataset(test1))[[ ".f3_canopus" ]])
df3 <- dplyr::mutate_if(df3, is.numeric, function(x) round(x, 2))
df3 <- dplyr::mutate(df3, description = character(1),
                     rel.index = as.integer(rel.index))

format(object.size(df3), units = "MB")
dplyr::as_tibble(df3)
df3

entity(dataset(project_dataset(test1))[[ ".f3_canopus" ]]) <- df3

## ---------------------------------------------------------------------
## others

reference <- specific_candidate(create_reference(test1, from = "structure"))
reference
test1 <- collate_data(test1, ".f3_spectra", reference = reference)
test1 <- collate_data(test1, subscript = ".f2_info")

## ---------------------------------------------------------------------
## delete tmpfile

mcn_dataset(test1) <- MCnebula2:::.mcn_dataset()

dataset(project_dataset(test1))

format(object.size(test1), units = "MB")

list.files(tmp)
```

```r
# ==========================================================================
# test function
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

project_dataset(test1)
mcn_dataset(test1)
latest(test1, subscript = ".f3_canopus")

test3 <- test1

test3 <- filter_structure(test3)
test3 <- create_reference(test3)
test3 <- filter_formula(test3, by_reference=T)

test3 <- create_stardust_classes(test3)
test3 <- create_features_annotation(test3)
test3 <- cross_filter_stardust(test3, 2, 1)
stardust_classes(test3)

test3 <- create_nebula_index(test3)
test3 <- compute_spectral_similarity(test3)
test3 <- create_parent_nebula(test3, 0.01, F)
test3 <- create_child_nebulae(test3, 0.01, 5)
spectral_similarity(test3)

test3 <- create_parent_layout(test3)
test3 <- create_child_layouts(test3)
test3 <- activate_nebulae(test3)

test3 <- .simulate_quant_set(test3)
test3 <- set_ppcp_data(test3)
test3 <- set_ration_data(test3)
test3 <- binary_comparison(test3, control - model,
                           model - control, 2 * model - control)

top_table(statistic_set(test3))

visualize(test3)
visualize(test3, 3)

call_command(ggset(child_nebulae(test3))[[2]])
```

```r
names(ggset(child_nebulae(test3))[2])
layout_ggraph(child_nebulae(test3))[2]
call_command(ggset(child_nebulae(test3))[[3]])

visualize(test3, "parent")

visualize_all(test3)
visualize()
```

```r
# ========================================================================
# save object
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

mcn_5features <- test1
setwd("~/MCnebula2/")
usethis::use_data(mcn_5features)
```

```r
# ========================================================================
# post modify
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
library(magrittr)
backup <- mcn_5features
test <- mcn_5features

## non-asc
entity(dataset(project_dataset(test))[[".canopus"]])$description %<>%
  enc2utf8()

mcn_5features <- test

## msms
test <- collate_data(test, subscript = ".f2_msms")
format(object.size(test), units = "MB")
project_dataset(test)

mcn_5features <- test

usethis::use_data(mcn_5features, overwrite = T)
# save(mcn_5features, file = "~/mcn_5features.rda")

unlink(tmp, T, T)
```

67 File: grep_names.R

```r
setwd("~/MCnebula2/R")
library(magrittr)

script <- list.files(".", pattern = "\\.R$") %>%
  sapply(readLines)

target <-
  lapply(script,
         function(text){
           lo_1 <- !grepl("@aliases|@exportClass|@family|new\\(", text)
           lo_2 <- grepl("^#'.*(?<![_|a-z|A-Z])nebula(?![-|_])", text, perl = T)
           lo <- lo_1 & lo_2
           text[lo] <- gsub("child(?!_)", "Child", text[lo], perl = T)
           text[lo] <- gsub("parent(?!_)", "Parent", text[lo], perl = T)
           text[lo] <- gsub("(?<![_|a-z|A-Z])nebula(?![-|_])", "Nebula",
                            text[lo], perl = T)
           text[lo] <- gsub("'Nebula'", "Nebula", text[lo])
           text[lo] <- gsub("'Nebulae'", "Nebulae", text[lo])
           text
         })

path <- "~/code_backup/mcnebula2"
dir.create(path, recursive = T)

lapply(names(target),
       function(name){
         writeLines(target[[name]], paste0(path, "/", name))
       })
```

68 File: grid_draw.R

```r
tmp <- paste0(tempdir(), "/test.png")

png(tmp, 480, 490)
pushViewport(viewport(, , 0.5, 0.5))
test <- grecto("test", cex = 3)
draw(test, gtext("shiny", list(cex = 5)))
dev.off()
```

```r
cv <- import("cv2")
# np <- import("numpy")

img <- cv$imread(tmp)

dim <- dim(img)[1:2]
w <- dim[1]
h <- dim[2]

src <- rbind(c(0, 0), c(0, h), c(w, 0), c(w, h))
target <- rbind(c(100, 100), c(0, h), c(300, 100), c(w, h))

trans <- cv$findHomography(src, target)[[1]]

new_img <- cv$warpPerspective(img, trans, dim)
```

69 File: internal_data.R

```r
# ==========================================================================
# MCnebula2 used
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

setwd("~/MCnebula2/")

path <- "~/outline/mc.test/templates/"
names <- eval(.workflow_name)
objs <-
  sapply(names(names), simplify = F,
    function(name) {
      file <- paste0(path, name, ".R")
      if (file.exists(file))
        readLines(file)
    })
.workflow_templ <- objs[!vapply(objs, is.null, logical(1))]

usethis::use_data(.workflow_templ, internal = TRUE, overwrite = T)
```

70 File: abs_2.R

```r
# ==========================================================================
# draw the graphic abstract
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

library(ggplot2)

title_pals <- colorRampPalette(c('white', '#1B1919'))(10)[ 5:8 ]

# alignment
shift <- rnorm(3, 2, 1)
all_range <- list(1:30, 31:60, 61:100)
set.seed(1)
lst <- mapply(shift, 1:length(shift), SIMPLIFY = F, FUN = function(shift, id){
  peak <- mapply(all_range, SIMPLIFY = F,
    FUN = function(range){
      peak <- dnorm(range, median(range) + shift, rnorm(1, 5, 1.2)) *
        rnorm(1, 0.7, 0.15)
    })
  feature <- mapply(1:length(all_range), lengths(all_range),
    FUN = function(seq, rep){
      rep(paste0("peak", seq), rep)
    })
  tibble::tibble(x = unlist(all_range), y = unlist(peak),
    sample = paste0("sample", id),
    peak = unlist(feature)
  )
})
data <- data.table::rbindlist(lst)
data <- dplyr::mutate(data,
  is = ifelse(y >= 0.003, T, F),
  peak = ifelse(is, peak, "non-feature"))
palette <- c(ggsci::pal_npg()(length(all_range)), "transparent")
names(palette) <- c(paste0("peak", 1:length(all_range)), "non-feature")
# detection
all_time <- vapply(all_range, median, 1)
anpi <- 0.05
len <- 5
set.seed(1)
ms1_set <- lapply(all_time,
  function(n){
    ms1 <- c(rnorm(3, 3, 2), rnorm(3, 8, 3), rnorm(3, 3, 2))
    ms1 <- ms1[ ms1 > 0 ]
    ms1 <- data.frame(x = 1:length(ms1), xend = 1:length(ms1),
      y = 0, yend = ms1)
    ms1 <- dplyr::mutate(ms1,
      x = x * len, xend = xend * len,
      y = sinpi(anpi) * x + y, yend = sinpi(anpi) * xend + yend,
      x = cospi(anpi) * x + n, xend = x)
    ms1 <- dplyr::bind_rows(ms1,
      c(x = n, xend = max(ms1$xend),
        y = 0, yend = max(ms1$y)))
    dplyr::mutate(ms1, time = n)
  })
ms1_set <- data.table::rbindlist(ms1_set)
p <- ggplot(dplyr::filter(data, sample == "sample1"), aes(x = x, y = y * 50)) +
  geom_segment(data = ms1_set,
    aes(x = x, xend = xend, y = y, yend = yend),
    color = "grey30", size = 0.8) +
  geom_area(aes(fill = peak)) +
  geom_line() +
  labs(x = "retention time", y = "intensity") +
  scale_fill_manual(values = palette) +
  theme_void() +
  theme(text = element_text(family = "Times"),
    axis.text.y = element_blank(),
    legend.position = "none"
  )
grob.detect <- as_grob(p)
g.detect <- into(grectn(bgp_args = list(lty = "solid")), grob.detect)
## gear
data <- data.frame(x = 1:20, y = rep(c(0, 1), 10))
p.step <- ggplot(dplyr::slice(data, 1:19)) +
  geom_step(aes(x = x, y = y), size = .2) +
  coord_polar() +
  ylim(c(-8, 1)) +
  theme_void() +
  theme(plot.margin = rep(u(-.1, npc), 4))
gear <- as_grob(p.step)
g.gear <- ggather(gear,
  circleGrob(r = c(.2, .4)),
  circleGrob(r = .3, gp = gpar(lwd = 5, col = 'grey70')),
```

```r
  vp = viewport(gp = gpar(alpha = .2)))
MCnebula2:::.smiles_to_cairosvg(
  'C1=C(C=C(C(=C1I)OC2=CC(=C(C(=C2)I)O)I)I)CC(C(=O)O)N',
  'chemEg.svg')
struc <- .rm_background(.cairosvg_to_grob('chemEg.svg'))
struc <- ggather(struc, vp = viewport(, .5, 1.8, 1.8))
mglayer <- function(from = 1:5, n = 5, seed = 100) {
  set.seed(seed)
  ns <- sample(from, n, T)
  grobs <- lapply(ns,
    function(n) {
      ggather(glayer(n), vp = viewport(, , .7, .7))
    })
  sig <- paste0(paste0("glayer", seed), 1:n)
  names(grobs) <- sig
  frame_col(fill_list(sig, 1), grobs)
}
g.predict <- ggather(struc, g.gear)
g.predict <- frame_row(c(g.predict = 4, glayers = 1),
  namel(g.predict, glayers = mglayer(1:4, 4)))
g.before <- frame_row(c(g.detect = 1.5, g.predict = 1),
  namel(g.detect, g.predict))
g.step0 <- grecti2('Detection and prediction', tfill = title_pals[1])
g.step0 <- into(g.step0, ggather(g.before, vp = viewport(, , .95, .8)))

## step2
## rank bar
set.seed(100)
data <- data.frame(x = 1:(n <- 12), y = sort(abs(rnorm(n, 5, 5))))
p.rank <- ggplot(data) +
  geom_col(aes(x = x, y = y), fill = ifelse(data$y > 5, '#D5E4A2', '#FD7446')) +
  geom_hline(yintercept = 5, linetype = "dashed", color = "red") +
  coord_flip() +
  theme_void()
g.rank <- as_grob(p.rank)
## bar
p.bar <- ggplot(data.frame(x = 1, y = c(0.1, 0.9), group = c("false", "true"))) +
  geom_col(aes(x = x, y = y, fill = reorder(group, desc(y))), width = 0.5) +
  annotate("segment", x = 1.7, xend = 1.7, y = 0.12, yend = 0.98,
    arrow = arrow(angle = 10, type = "closed", length = unit(0.05, "npc"))) +
  annotate("segment", x = 1.6, xend = 1.8, y = 0.1, yend = 0.1) +
  scale_fill_manual(values = c('#91D1C2FF', '#DC0000FF')) +
  xlim(c(.5, 2)) +
  theme_void() +
  theme(legend.position = 'none')
p.bar <- as_grob(p.bar)
p.bar <- ggather(p.bar, vp = viewport(, , .95, .9))
## box
simu_score <- dplyr::mutate(
  data.frame(.features_id = 1:50, score = rnorm(50, 0.4, 0.15)),
  group = ifelse(score >= 0.3, "true", "false")
)
p.box <- ggplot(simu_score, aes(x = "", y = score)) +
  geom_boxplot() +
  geom_hline(yintercept = 0.3, linetype = "dashed", color = "red") +
  geom_jitter(width = 0.1, aes(color = group)) +
  theme_void() +
  theme(legend.position = "none")
p.box <- as_grob(p.box)
## upset
p.upbar <- ggplot(data.frame(x = 1, y = 5)) +
  geom_col(aes(x = x, y = y), width = .5) +
  ylim(0, 7) +
  scale_x_continuous(breaks = c(0, 1, 2), limits = c(.5, 1.5)) +
  theme(text = element_blank(),
    axis.ticks.y = element_blank()) +
  geom_blank()
p.upbar <- as_grob(p.upbar)
cirs <- circleGrob(y = (cirs.y <- seq(.1, .9, length.out = 5))[2:4], r = .08,
  gp = gpar(col = 'transparent', fill = 'grey90'))
cirs.sol <- circleGrob(y = cirs.y[c(1, 5)], r = .08,
  gp = gpar(col = 'transparent', fill = 'black'))
line <- linesGrob(x = c(.5, .5), y = c(.1, .9), gp = gpar(lwd = 3))
p.upcir <- ggather(cirs, line, cirs.sol)
g.upset <- frame_row(c(p.upbar = 2, p.upcir = 1),
  namel(p.upbar = ggather(p.upbar, vp = viewport(.48)), p.upcir))
g.upset <- ggather(g.upset, vp = viewport(, , .90, .90))
## gather
g.step1.1t3 <- mapply(list(p.bar, p.box, g.upset), n = 1:3, SIMPLIFY = F,
  FUN = function(g, n) {
    into(grecti2(n, tfill = '#CC0C00'), g)
  })
```

```r
names(g.step1.1t3) <- n(g, 3)
g.step1.1t3 <- frame_col(c(g1 = 1, null = .1, g2 = 1, null = .1, g3 = 1),
  c(g.step1.1t3, list(null = nullGrob())))
g.step1.0t3 <- frame_col(c(g.rank = .5, null = .1, g.step1.1t3 = 3),
  namel(g.rank, g.step1.1t3, null = nullGrob()))
g.step1.0t3 <- into(
  grecti2('Select classes', tfill = title_pals[2]),
  ggather(g.step1.0t3, vp = viewport(, , .95, .8))
)

## step2
# network
load(paste0(.expath, "/toActiv30.rdata"))
test1 <- toActiv30
set.seed(17)
reference(mcn_dataset(test1))$nebula_index %<>%
  dplyr::filter(class.name %in% sample(unique(class.name), 2))
test1 <- set_tracer(test1, c("2027", "2020"), colors = c("#C80813", "#FD7446"))
test1 <- create_child_nebulae(test1, 0.01, 5)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)
test1 <- set_nodes_color(test1, use_tracer = T)
chAsGrob <- function(ch, x) {
  ggset <- modify_default_child(ch)
  ggset@layers$ggtitle@command_args$label %<>%
    gsub("[a-zA-Z]", ".", .) %>%
    gsub("^....", "Class", .) %>%
    gsub("\\.\\.", ".", .)
  as_grob(call_command(ggset))
}
sets <- lapply(ggset(child_nebulae(test1)), chAsGrob, x = test1)
sets <- lapply(names(sets),
  function(name) {
    ggather(sets[[name]],
      vp = viewports(child_nebulae(test1))[[name]])
  })
sets_vp <- viewport(layout = grid_layout(child_nebulae(test1)))
sets <- do.call(ggather, c(sets, list(vp = sets_vp)))

set.seed(120)
data <- data.frame(x = 1:12, y = sort(abs(rnorm(12, 5, 5))))
```

```r
p.fRank <- ggplot(data) +
  geom_col(aes(x = x, y = y, fill = y)) +
  scale_fill_gradientn(colors = c('grey90', 'grey90', 'red')) +
  coord_flip() +
  theme_void() +
  theme(axis.ticks = element_blank(),
    legend.position = 'none') +
  geom_blank()
g.fRank <- as_grob(p.fRank)

omit <- ggplot() +
  ggtitle("...") +
  theme_void() +
  theme(legend.position = 'none',
    plot.title = MCnebula2:::.element_textbox(
      fill = MCnebula2:::.get_label_color()[sample(2:6, 1)])) +
  geom_blank()
omit <- ggather(as_grob(omit), vp = viewport(, , .9))
g.sets <- frame_row(c(sets = 4, omit = .5, omit = .5), namel(sets, omit))
g.step2 <- frame_col(
  c(g.fRank = 1, g.sets = 5, null = .2),
  namel(g.fRank, g.sets, null = nullGrob())
)
g.step2 <- into(grecti2('Tracing top features', tfill = title_pals[3]),
  ggather(g.step2, vp = viewport(, , .95, .95)))

g.ab <- frame_col(c(g.step0 = 1, null = .02, g.step1.0t3 = 2, null = .02, g.step2 = 1),
  namel(g.step0, g.step1.0t3, g.step2, null = nullGrob()))

pdf('tocg.pdf', 9, 4)
draw(g.ab)
dev.off()
# dev.new(height = 4, width = 9)
```

71 File: abs.R

```r
# ==========================================================================
# draw the graphic abstract
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

library(ggplot2)
```

```r
## msms
set.seed(50)
data <- data.frame(x = sample(1:50, 10), y = rnorm(10, mean = 40, 30))
p.msms <- ggplot(data) +
  geom_segment(aes(x = x, xend = x, y = 0, yend = abs(y)), size = .7) +
  theme(text = element_blank(),
    axis.ticks = element_blank()) +
  geom_blank()
g.msms <- into(glayer(3, .1), as_grob(p.msms))
g.msms <- frame_row(
  c(title = 1, g.msms = 5),
  namel(title = gtext("MS/MS", list(cex = 1.5)), g.msms)
)

## ms1
set.seed(100)
data <- data.frame(x = -25:25, y = dnorm(-25:25, 0, sqrt(50)))
data <- dplyr::mutate(data, fill = ifelse(abs(x) < sqrt(50) * 3, 'peak', 'zero'))
p.ms1 <- ggplot(data) +
  geom_line(aes(x = x, y = y)) +
  geom_area(aes(x = x, y = y, fill = fill)) +
  scale_fill_manual(values = c(ggsci::pal_npg()(2)[2], 'grey90')) +
  theme(text = element_blank(),
    axis.ticks = element_blank(),
    legend.position = "none") +
  geom_blank()
g.ms1 <- into(glayer(3, .1), as_grob(p.ms1))
g.ms1 <- frame_row(
  c(title = 1, g.ms1 = 5),
  namel(title = gtext("Features", list(cex = 1.5)), g.ms1)
)

## rank bar
set.seed(100)
data <- data.frame(x = 1:7, y = sort(abs(rnorm(7, 5, 5))))
p.rank <- ggplot(data) +
  geom_col(aes(x = x, y = y, fill = y)) +
  scale_fill_gradientn(colors = c('grey90', 'orange', 'red')) +
  coord_flip() +
  theme_void() +
  theme(axis.ticks = element_blank(),
    legend.position = 'none',
    text = element_blank()) +
  geom_blank()
g.rank <- as_grob(p.rank)
g.rank <- frame_row(
  c(title = 1, g.rank = 5),
  namel(title = gtext("Ranking", list(cex = 1.5)), g.rank)
)

## identify
MCnebula2:::.smiles_to_cairosvg(
  'C1=C(C=C(C(=C1I)OC2=CC(=C(C(=C2)I)O)I)I)CC(C(=O)O)N',
  'chemEg.svg')
struc <- .cairosvg_to_grob('chemEg.svg')
data <- data.frame(x = 1:20, y = rep(c(0, 1), 10))
p.step <- ggplot(dplyr::slice(data, 1:19)) +
  geom_step(aes(x = x, y = y)) +
  coord_polar() +
  ylim(c(-8, 1)) +
  theme_void() +
  theme(plot.margin = rep(u(-.1, npc), 4))
gear <- as_grob(p.step)
g.struc <- ggather(struc, gear)
g.struc <- frame_row(
  c(title = 1, g.struc = 5),
  namel(title = gtext("Prediction", list(cex = 1.5)), g.struc)
)

## network
p.nets <- lapply(list(1:12, 13:20, 20:30),
  function(ids) {
    set.seed(1501)
    data <- random_graph(ids, layout = 'kk')
    p <- ggraph(data) +
      geom_edge_fan(color = 'grey85') +
      geom_node_point(
        aes(x = x, y = y, size = size,
          fill = ifelse(name == '11', 'target', 'non-target')),
        shape = 21, color = 'transparent') +
      scale_fill_manual(values = c('grey85', 'red')) +
      ggtitle('Class ...') +
```

```r
      theme_void() +
      theme(legend.position = 'none',
        text = element_text(family = "Times"),
        plot.title = MCnebula2:::.element_textbox(
          fill = MCnebula2:::.get_label_color()[sample(4:7, 1)])) +
      geom_blank()
    p <- ggather(as_grob(p), vp = viewport(, , .9, .9))
  })
names(p.nets) <- n(net, 3)
omits <- sapply(n(omit, 3), simplify = F,
  function(n) {
    omit <- ggplot() +
      ggtitle("...") +
      theme_void() +
      theme(legend.position = 'none',
        text = element_text(family = "Times"),
        plot.title = MCnebula2:::.element_textbox(
          fill = MCnebula2:::.get_label_color()[sample(2:6, 1)])) +
      geom_blank()
    ggather(as_grob(omit), vp = viewport(, , .9))
  })
g.nets <- frame_row(
  c(net1 = 1, null = .1, net2 = 1, null = .1,
    omit1 = .2, omit2 = .2, omit3 = .2),
  c(p.nets, omits, list(null = nullGrob()))
)
g.nets <- frame_row(
  c(title = 1, g.nets = 12),
  namel(title = gtext("Tracing", list(cex = 1.5)), g.nets)
)
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

null <- nullGrob()
spArr <- polygonGrob(
  c(0, .6, 1, .6, 0), c(0, 0, .5, 1, 1),
  gp = gpar(lwd = u(2, line), col = "grey30")
)
spArr <- ggather(spArr, vp = viewport(, , .4, .5))
frameAr <- frame_row(c(spArr = 1, null = .1, spArr = 1),
  namel(null, spArr))
frame1 <- frame_row(c(g.msms = 1, null = .1, g.ms1 = 1),
  namel(g.msms, g.ms1, null))
frame2 <- frame_row(c(g.struc = 1, null = .1, g.rank = 1),
  namel(g.struc, g.rank, null))
parrow <- frame_row(
  c(null = 1, arrow = 2, null = 4),
  namel(arrow = parrow(col = 'black'), null))
frame1a2a3 <- frame_col(
  c(frame1 = .8, frameAr = .15,
    frame2 = .6, spArr = .15, g.nets = 1, parrow = .15),
  namel(frame1, frame2, g.nets, frameAr, spArr, parrow))

pdf('tocg.pdf', 10, 6)
draw(ggather(frame1a2a3, vp = viewport(, , .95, .95)))
dev.off()
# dev.new(height = 1, width = 2)
# draw(g.nets)
```

72 File: a_elements.R

```
# ==========================================================================
# grobs
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## object
load(paste0(.expath, "/toActiv30.rdata"))
test1 <- toActiv30
ids <- features_annotation(test1)$.features_id
export_path(test1) <- paste0(tempdir(), "/mcnebula_results")
export_path(test1)
the_id <- "2095"
test1 <- draw_structures(test1, .features_id = the_id)

## get classes dataset
class <- latest(filter_ppcp(test1, pp.threshold = 0))
class <- dplyr::mutate(class, class.name = reorder(class.name, dplyr::desc(pp.value)))
class_2095 <- dplyr::filter(class, .features_id == !!the_id)
class_2095 <- dplyr::arrange(class_2095, class.name)
## modified the data
class <- dplyr::select(class, .features_id, class.name, pp.value)
```

```r
class <- lapply(split(data.frame(class), ~.features_id),
  function(df){
    df <- dplyr::mutate(df, class.name = reorder(class.name, dplyr::desc(pp.value)))
    head(dplyr::arrange(df, class.name), n = 30)
  })
class <- data.table::rbindlist(class)

## for inner filter
class_in <- dplyr::filter(class, .features_id %in% as.character(c(2093, 2095, 2097)),
  pp.value <= 0.9)
class_in <- dplyr::bind_rows(class_in,
  data.frame(class.name = paste0("class", 1:3),
    pp.value = 0.5,
    .features_id = "Others"
    ))

## draw horizontal bar plot for inner filter
y.arrow <- seq(-0.4, 0.4, length.out = 3)
check_pal <- c(true = "#91D1C2FF", false = "#E64B35FF")
df <- dplyr::filter(class_in, .features_id != "Others")
df <- dplyr::mutate(df, .features_id = paste0("ID: ", .features_id))
p1 <- ggplot(df) +
  geom_col(aes(x = class.name, y = pp.value,
      fill = ifelse(pp.value >= 0.5, "true", "false"),
      )) +
  geom_segment(data = dplyr::filter(df, !grepl("[0-9]", class.name), pp.value >= 0.5),
    aes(x = class.name, xend = class.name, y = 1, yend = 1.15),
    color = check_pal[["true"]],
    arrow = arrow(8, unit(0.1, "inches"), "last", "closed")) +
  geom_point(data = dplyr::filter(df, grepl("[0-9]", class.name) | pp.value < 0.5),
    aes(x = class.name, y = 1),
    shape = 4, color = check_pal[["false"]]) +
  geom_text(data = dplyr::filter(df, grepl("[0-9]", class.name)),
    aes(x = class.name, y = -0.05, label = stringr::str_wrap(class.name, 30)),
    color = "red",
    family = "Times", hjust = 1, size = 1.2) +
  geom_text(data = dplyr::filter(df, !grepl("[0-9]", class.name)),
    aes(x = class.name, y = -0.05, label = stringr::str_wrap(class.name, 30)),
    color = "black",
    family = "Times", hjust = 1, size = 1.2) +
  geom_hline(aes(yintercept = 0.5),
    linetype = "dashed", color = "red") +
  coord_flip() +
  ylim(c(-1.3,1.3)) +
  labs(x = "Classification") +
  scale_fill_manual(values = check_pal) +
  theme_minimal() +
  facet_grid(.features_id ~ ., scales = "free") +
  theme(text = element_text(family = "Times", face = "bold"),
    axis.title.x = element_blank(),
    axis.text = element_blank(),
    legend.position = "none") +
  geom_blank()
## arrow
p1.ex <- ggplot() +
  geom_segment(data = dplyr::filter(class_in, .features_id == "Others"),
    aes(x = 5, xend = 1, y = y.arrow, yend = y.arrow),
    linetype = "dashed",
    arrow = arrow(10, unit(0.15, "inches"), "last", "closed")) +
  coord_flip() +
  ylim(c(-1.3,1.3)) +
  labs(y = "Posterior probability") +
  scale_fill_manual(values = check_pal) +
  theme_minimal() +
  facet_grid(.features_id ~ ., scales = "free") +
  theme(text = element_text(family = "Times", face = "bold"),
    axis.title.y = element_blank(),
    axis.text = element_blank(),
    legend.position = "none") +
  geom_blank()
## as grob
p1 <- as_grob(p1)
p1.ex <- as_grob(p1.ex)
p1 <- frame_row(c(p1 = 3, p1.ex = .4), namel(p1, p1.ex))

## for stardust
stardust <- dplyr::filter(class, pp.value >= 0.5)
stardust <- dplyr::filter(class, !grepl("[0-9]", class.name))
stardust <- split(data.frame(stardust), ~ as.character(class.name))
hierarchy <- hierarchy(test1) %>%
  dplyr::select(hierarchy, class.name)
index <- as.list(hierarchy$hierarchy)
```

```r
names(index) <- hierarchy$class.name
label_color <- palette_label(test1)
## draw network plots
library(ggraph)
n <- 8
lst <- mapply(head(stardust, n = n), names(stardust)[1:n],
  SIMPLIFY = F,
  FUN = function(df, name){
    edges <- data.frame(.features_id1 = df[1, ]$.features_id,
      .features_id2 = df[1. ]$.features_id)
    graph <- igraph::graph_from_data_frame(edges, directed = T, vertices = df)
    graph <- tidygraph::as_tbl_graph(graph)
    layout_n <- create_layout(graph, layout = "kk")
    ggraph(layout_n) +
      geom_node_point(shape = 21, stroke = 0.2, fill = "#D9D9D9", size = 5) +
      theme_void() +
      ggtitle(stringr::str_wrap(name, width = 30)) +
      theme(text = element_text(family = "Times", face = "bold"),
        plot.title = ggtext::element_textbox(
          size = 10, color = "white",
          fill = label_color[ index[[name]] ],
          box.color = "white",
          halign = 0.5, linetype = 1, r = unit(5, "pt"), width = unit(1, "npc"),
          padding = margin(2, 0, 1, 0), margin = margin(3, 3, 3, 3))
        ) +
      geom_blank()
  })
## as grob
lst <- lapply(lst, as_grob)

## ring diagrame
text_gp <- gpar(color = "black", cex = 1, fontfamily = "Times", fontface = "bold",
  fontsize = 20)
p.ring <- ggplot(data.frame(x = 1, y = c(0.1, 0.9), group = c("true", "false"))) +
  geom_col(aes(x = x, y = y, fill = group)) +
  annotate("segment", x = 1.7, xend = 1.7, y = 0.01, yend = 0.08,
    arrow = arrow(angle = 10, type = "closed", length = unit(0.05, "npc"))) +
  annotate("segment", x = 1.6, xend = 1.8, y = 0, yend = 0) +
  annotate("segment", x = 1.6, xend = 1.8, y = 0.1, yend = 0.1) +
  scale_fill_manual(values = check_pal) +
  xlim(c(-1, 2)) +
  coord_polar(theta = "y") +
  ggtitle("Ratio") +
  theme_void() +
  theme(legend.position = "none",
    plot.title = element_text(family = .font, size = 6, hjust = .5)) +
  geom_blank()
## as grob
p.ring <- as_grob(p.ring)
## another ring...
p.ring2 <- ggplot(data.frame(x = 1, y = c(0.7, 0.3), group = c("true", "false"))) +
  geom_col(aes(x = x, y = y, fill = group)) +
  annotate("segment", x = 1.7, xend = 1.7, y = 0.01, yend = 0.68,
    arrow = arrow(angle = 10, type = "closed", length = unit(0.05, "npc"))) +
  annotate("segment", x = 1.6, xend = 1.8, y = 0, yend = 0) +
  annotate("segment", x = 1.6, xend = 1.8, y = 0.7, yend = 0.7) +
  scale_fill_manual(values = check_pal) +
  xlim(c(-1, 2)) +
  coord_polar(theta = "y") +
  ggtitle("Score distribution") +
  theme_void() +
  theme(legend.position = "none",
    plot.title = element_text(family = .font, size = 6, hjust = .5)) +
  geom_blank()
## as grob
p.ring2 <- as_grob(p.ring2)

## bar diagrame
p.bar <- ggplot(data.frame(x = 1, y = c(0.1, 0.9), group = c("false", "true"))) +
  geom_col(aes(x = x, y = y, fill = reorder(group, desc(y))), width = 0.5) +
  annotate("segment", x = 1.7, xend = 1.7, y = 0.12, yend = 0.98,
    arrow = arrow(angle = 10, type = "closed", length = unit(0.05, "npc"))) +
  annotate("segment", x = 1.6, xend = 1.8, y = 0.1, yend = 0.1) +
  scale_fill_manual(values = check_pal) +
  xlim(c(0, 2)) +
  ggtitle("Number") +
  theme_void() +
  theme(legend.position = "none",
    plot.title = element_text(family = .font, size = 6, hjust = .5)) +
  geom_blank()
## as grob
p.bar <- as_grob(p.bar)
```

```r
## peak simbolized for feature
p.peak <- ggplot(data.frame(x = 1:20, y = dnorm(1:20, 10, 3)), aes(x = x, y = y)) +
  geom_line() +
  theme_minimal() +
  labs(x = "RT", y = "Intensity") +
  theme(text = element_blank())
p.peak.yellow <- p.peak +
  geom_area(fill = "lightyellow")
p.peak.grey <- p.peak +
  geom_area(fill = "#D9D9D9")
## as grob
p.peak <- as_grob(p.peak)
p.peak.yellow <- as_grob(p.peak.yellow)
p.peak.grey <- as_grob(p.peak.grey)

## boxplot
struc_2095 <- structures_grob(child_nebulae(test1))[[ the_id ]]
set.seed(100)
simu_score <- data.frame(.features_id = 1:50, score = rnorm(50, 0.4, 0.15)) %>%
  dplyr::mutate(group = ifelse(score >= 0.3, "true", "false"))
p.box <- ggplot(simu_score, aes(x = "", y = score)) +
  geom_boxplot() +
  geom_hline(yintercept = 0.3, linetype = "dashed", color = "red") +
  geom_jitter(width = 0.1, aes(color = group)) +
  labs(x = "", y = "") +
  # ggtitle("Identified score") +
  scale_fill_manual(values = check_pal) +
  theme_void() +
  theme(legend.position = "none",
    plot.title = element_text(hjust = .5),
    text = element_text(family = .font, size = 5),
    # panel.grid = element_blank(),
    # plot.background = element_blank(),
    # panel.background = element_blank(),
    axis.ticks = element_blank()) +
  geom_blank()
## as grob
p.box <- as_grob(p.box)

## ratio bar plot
ratio <- table(simu_score$group) %>%
  prop.table() %>%
  as.data.frame()
p.ratio <- ggplot(ratio, aes(x = 1, y = Freq, fill = reorder(Var1, desc(Freq)))) +
  ggchicklet::geom_chicklet(radius = unit(0.1, "npc")) +
  geom_hline(yintercept = 0.6, linetype = "dashed", color = "red") +
  scale_fill_manual(values = check_pal) +
  xlim(c(0, 2)) +
  theme_minimal() +
  theme(text = element_blank(),
    legend.position = "none") +
  geom_blank()
  ## as_grob
p.ratio <- as_grob(p.ratio)

## assessment
df <- dplyr::filter(stardust_classes(test1), hierarchy >= 3)
parents <- get_parent_classes(unique(df$class.name), test1)
set <- parents[!vapply(parents, is.null, logical(1))]
set <- set[vapply(set, function(cl) if (any(cl %in% unique(df$class.name) )) T else F, T)]
## draw
ec <- c("Carboxylic acids")
ec. <- "Child..."
ep <- c("Carboxylic acids and derivatives")
ep. <- "Parent..."
labels <- lapply(c(ec., ep., "Class..."),
  function(lab) {
    zo(into(grectn(bgp_args = list(lty = "solid")),
        gtextp(stringr::str_wrap(lab, 15), list(col = "white", cex = .6))))
  })
names(labels) <- c("ec", "ep", "omit")
labels$null <- nullGrob()
## frame
scolor <- function(grobs, hier, color = palette_label(test1)[hier]) {
  lapply(grobs,
    function(grob) {
      if (is(grob, "null")) return(grob)
      else {
        grob$children[[1]]$children[[1]]$gp$fill <- color
        grob$children[[1]]$children[[1]]$gp$col <- "transparent"
        return(grob)
      }
```

```r
    })
}
frame1 <- frame_row(c(null = .3, omit = .4, null = .3, omit = .4, null = .3),
  scolor(labels, 2))
frame2 <- frame_row(c(omit = .5, null = .3, ep = 1, null = .3, omit = .5),
  scolor(labels, 3))
frame3 <- frame_row(c(null = .3, ec = 1, null = .3, omit = .5, null = .3),
  scolor(labels, 4))
frame <- frame_col(c(frame1 = 1, null = .6, frame2 = 1, null = .6, frame3 = 1),
  namel(frame1, frame2, frame3, null = nullGrob()))
## arrows
path <- c("frame1::.*::omit.rep.2", "frame2::.*::omit",
  "frame2::.*::ep", "frame2::.*::omit.rep.2", "frame3::.*::ec", "frame3::.*omit")
nulls <- lapply(path, function(p) {
  list(l = setnullvp(p, list(x = 0, y = .5), frame),
    r = setnullvp(p, list(x = 1, y = .5), frame))
  })
names(nulls) <- c("n1.2", "n2.1", "ep", "n2.3", "ec", "n3.2")
arr1_2 <- c("n2.1", "ep", "n2.3")
cur <- c(-.1, .2, .3)
arr_gpar <- gpar(fill = palette_label(test1)[2], col = palette_label(test1)[2])
arr1_2 <- lapply(1:length(arr1_2),
  function(n) {
    no <- arr1_2[n]
    garrow(nulls$n1.2$r, nulls[[no]]$l,
      list(curvature = cur[n], gp = arr_gpar))
  })
arr2_3 <- c("ec", "n3.2")
cur <- c(-.1, .2)
arr2_3 <- lapply(1:length(arr2_3),
  function(n) {
    no <- arr2_3[n]
    garrow(nulls$ep$r, nulls[[no]]$l,
      list(curvature = cur[n], gp = arr_gpar))
  })
## gather
bg <- rectGrob(.3, , .7, 1.1, just = c("left", "centre"),
  gp = gpar(col = "transparent", fill = "lightyellow"))
hier_tree <- do.call(ggather, c(list(bg), list(frame), arr1_2, arr2_3))

## venn plot
```

```r
conts <- list(dplyr::filter(df, class.name == ep)$.features_id,
  dplyr::filter(df, class.name == ec)$.features_id)
feas <- unique(c(conts[[1]], conts[[2]]))
data <- data.frame(a = feas %in% conts[[1]], b = feas %in% conts[[2]])
names <- stringr::str_trunc(c(ep, ec), 15, "cen")
cir <- ggather(circleGrob(.3, gp = gpar(alpha = .5, fill = palette_label(test1)[3])),
  circleGrob(.7, gp = gpar(alpha = .5, fill = palette_label(test1)[4])))
venn.text <- frame_col(c(x = 1, null = 1),
  list(x = gtextp("Overlap Ratio", list(cex = .5), x = .8, y = -1),
    null = nullGrob()))
venn <- frame_row(c(null = .05, tit = .1, cont = 2, null = .05),
  list(tit = venn.text, cont = cir, null = nullGrob()))

## upset plot
set.seed(101)
rand <- sample(unique(df$class.name), 3)
comc <- c(ep, ec, rand)
data <- dplyr::filter(df, class.name %in% comc)
mutate <- .as_dic(c(ep., ec., paste0(n(Class, 2), "..."), "..."), c(ep, ec, rand), , F)
data <- dplyr::mutate(data, class.name =
  vapply(class.name, FUN.VALUE = character(1), USE.NAMES = F,
    function(cl) {
      do.call(switch, c(list(cl), mutate))
    }))
data <- lapply(split(data, ~.features_id), function(df) df$class.name)
data <- lapply(data,
  function(cl) {
    unlist(lapply(cl,
        function(cl) {
          mutate <- mutate[mutate != cl]
          vapply(mutate, FUN.VALUE = character(1),
            function(v) {
              paste0(sort(c(v, cl)), collapse = "_")
            })
        }))
  })
data <- tibble::tibble(Class = unlist(data))
p.upset <- ggplot(data, aes(x = Class)) +
  geom_bar() +
  ggupset::axis_combmatrix(sep = "_", levels = unlist(mutate, use.names = F)) +
  theme(text = element_text(family = .font),
    axis.text.y = element_text(size = 5),
    plot.title = element_text(hjust = .5, size = 6),
    axis.title = element_text(size = 6)) +
  labs(y = "Number") +
  ggtitle("Overlap Number") +
  ggupset::theme_combmatrix(
    combmatrix.label.text = element_text(family = .font, size = 5)) +
  geom_blank()
p.upset <- as_grob(p.upset)
```

73 File: b_gather.R

```r
# ==========================================================================
# gather the grobs as a figure
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

pal <- c(fun = "#BC3C29FF", ex = "#E18727FF")

## inner filter
inner <- grecti3("inner filter", tfill = pal[[ "fun" ]])
inner <- into(inner, p1)

## stardust classes
stardust <- grecti3("stardust classes", tfill = pal[[ "ex" ]])
stardust <- into(stardust, frame_row(fill_list(names(lst), 1), lst))

## quantity
ass <- into(glayer(4), p.peak.yellow)
sum <- into(glayer(6), p.peak.grey)
seg <- segmentsGrob(0, 0, 1, 1)
arr <- parrow(3, "black", "solid")
mp.ring <- ggather(p.ring, vp = viewport(.4, , .8))
c1 <- frame_col(c(ass = 1, seg = .4, sum = 1, arr = .3, mp.ring = 1),
                namel(ass, seg, sum, arr, mp.ring))
mp.bar <- ggather(p.bar, vp = viewport(.2, , .8))
c2 <- frame_col(c(ass = 1, arr = .3, mp.bar = 1), namel(ass, arr, mp.bar))
sep <- segmentsGrob(.5, .1, .5, .9)
## gather
content <- frame_col(c(c1 = 1, c2 = .7), namel(c1, c2))
quantity <- grectn_frame(zo(content, h = .7), gtext0("quantity  (Abundance selection)"), zo = F)
```

```r
## score
struc <- into(glayer(4), struc_2095)
## gather
content <- frame_col(c(struc = 1, arr = .3, p.box = 1, arr = .3, p.ring2 = 1),
                     namel(struc, arr, p.box, p.ring2))
score <- grectn_frame(zo(content, h = .8), gtext0("score  (Goodness assessment)"), zo = F)

## identical
identical <- frame_col(c(hier_tree = .7, arr = .15, p.upset = 1.4, null = .1, venn = .3),
                       namel(hier_tree, arr, venn, null = nullGrob(), p.upset))
identical <- grectn_frame(identical, gtext0("identical  (Identicality assessment)"))

## cross
cross <- grecti3("cross filter", tfill = pal[[ "fun" ]])
obj <- sapply(namel(quantity, score, identical), zo, simplify = F)
obj <- frame_row(fill_list(names(obj), 1), obj)
cross <- into(cross, obj)
# draw(cross)
```

```r
# ==========================================================================
# gather all
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

prin <- frame_col(c(inner = 1, null = .1, stardust = .7, null = .1, cross = 3.5),
                  namel(inner, stardust, cross, null = nullGrob()))
vp <- viewport(, , .95, .95)
prin <- ggather(prin, vp = vp)

pdf("figure_mech.pdf", 11, 7)
draw(prin)
dev.off()
```

74 File: a_project.R

```r
# ==========================================================================
# external grob
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## external element
grob.lcms <- ex_grob("lcms")
grob.sample <- ex_grob("sample")
grob.convert <- ex_grob("convert")
grob.sirius <- ex_grob("sirius")
grob.collate <- ex_grob("collate_data")
```

```r
# ==========================================================================
# sub.frame
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## melody: palette
type <- c("class", "slot", "sub.slot", "function", "custom")
pal <- MCnebula2:::.as_dic(palette_set(mcn_5features)[-3], type, na.rm = T)
pal[[ "report" ]] <- "black"

grobs.project <- lst_grecti("dataset", pal, , grecti2)
```

```r
# ==========================================================================
# content
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## special arrow
spArr <- polygonGrob(
  c(0, 1, 1, .5, 0), c(1, 1, .4, 0, .4),
  gp = gpar(lwd = u(2, line), col = "grey30")
)

# alignment
shift <- rnorm(3, 2, 1)
all_range <- list(1:30, 31:60, 61:100)
set.seed(1)
lst <- mapply(shift, 1:length(shift), SIMPLIFY = F, FUN = function(shift, id){
  peak <- mapply(all_range, SIMPLIFY = F,
    FUN = function(range){
      peak <- dnorm(range, median(range) + shift, rnorm(1, 5, 1.2)) *
        rnorm(1, 0.7, 0.15)
    })
  feature <- mapply(1:length(all_range), lengths(all_range),
    FUN = function(seq, rep){
      rep(paste0("peak", seq), rep)
    })
  tibble::tibble(x = unlist(all_range), y = unlist(peak),
    sample = paste0("sample", id),
    peak = unlist(feature)
  )
})
```

```r
data <- data.table::rbindlist(lst)
data <- dplyr::mutate(data,
  is = ifelse(y >= 0.003, T, F),
  peak = ifelse(is, peak, "non-feature"))
palette <- c(ggsci::pal_npg()(length(all_range)), "transparent")
names(palette) <- c(paste0("peak", 1:length(all_range)), "non-feature")
p <- ggplot(data, aes(x = x, y = y)) +
  geom_area(aes(fill = peak)) +
  geom_line() +
  geom_vline(xintercept = vapply(all_range, median, 1), linetype = "dashed") +
  geom_hline(yintercept = 0.003, linetype = "dashed") +
  facet_grid(sample ~ .) +
  scale_fill_manual(values = palette) +
  labs(x = "retention time", y = "intensity") +
  theme_void() +
  theme(
    text = element_text(family = "Times"), axis.text.y = element_blank(),
    strip.text = element_blank(),
    legend.position = "none"
  )
grob.align <- as_grob(p)
# detection
all_time <- vapply(all_range, median, 1)
anpi <- 0.05
len <- 5
set.seed(1)
ms1_set <- lapply(all_time,
  function(n){
    ms1 <- c(rnorm(3, 3, 2), rnorm(3, 8, 3), rnorm(3, 3, 2))
    ms1 <- ms1[ ms1 > 0 ]
    ms1 <- data.frame(x = 1:length(ms1), xend = 1:length(ms1),
      y = 0, yend = ms1)
    ms1 <- dplyr::mutate(ms1,
      x = x * len, xend = xend * len,
      y = sinpi(anpi) * x + y, yend = sinpi(anpi) * xend + yend,
      x = cospi(anpi) * x + n, xend = x)
    ms1 <- dplyr::bind_rows(ms1,
      c(x = n, xend = max(ms1$xend),
        y = 0, yend = max(ms1$y)))
    dplyr::mutate(ms1, time = n)
  })
```

```r
ms1_set <- data.table::rbindlist(ms1_set)
p <- ggplot(dplyr::filter(data, sample == "sample1"), aes(x = x, y = y * 50)) +
  geom_segment(data = ms1_set,
    aes(x = x, xend = xend, y = y, yend = yend),
    color = "grey30", size = 0.8) +
  geom_area(aes(fill = peak)) +
  geom_line() +
  labs(x = "retention time", y = "intensity") +
  scale_fill_manual(values = palette) +
  theme_void() +
  theme(text = element_text(family = "Times"),
    axis.text.y = element_blank(),
    legend.position = "none"
  )
grob.detect <- as_grob(p)

## project_dataset
## into
grobs.project$dataset %<>% into(grob.collate)

## omit
omit <- circleGrob(seq(.2, .8, , 3), .5, .07, gp = gpar(fill = "grey20"),
                   vp = viewport(, , u(1.5, line), u(1.5, line)))
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

grob.sampleCir <- ggather(circleGrob(), grob.sample)
spArrScale <- ggather(spArr, vp = viewport(, , .2, .7))
grob.rdetect <- ggather(
  into(grectn(bgp_args = list(lty = "solid")), grob.detect),
  vp = viewport(, , .7, .9)
)

frame.project <- frame_row(
  c(grob.sampleCir = .5, samples = .1, spArrScale = .1,
    grob.lcms = 1, `LC-MS/MS` = .1, spArrScale = .1,
    grob.convert = .6, convert_raw_data = .1, spArrScale = .1,
    grob.rdetect = .7, feature_detection = .1, spArrScale = .1,
    grob.sirius = .4, run_SIRIUS = .1, spArrScale = .1,
    MCnebula2 = .2, omit = .1, dataset = 1),
  c(namel(grob.sampleCir, grob.lcms, grob.convert, grob.rdetect,
      grob.sirius, spArrScale, omit),
    dataset = list(grobs.project[[1]]),
    sapply(c("samples", "LC-MS/MS", "convert_raw_data",
        "feature_detection", "run_SIRIUS", "MCnebula2"),
      simplify = F, function(name) { gtext(name) })
  )
)
.gene.vp <- viewport(, , width = unit(8 * 1.5, "line"),
                    height = unit(35 * 1.5, "line"), clip = "off")
.project <- gTree(children = gList(frame.project), vp = .gene.vp)
```

```r
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

baf <- function(x, y, width = u(1, line), height = u(3, line)) {
  rect <- grectn(bgp_args = gpar(lty = "solid"))@grob
  clip <- clipGrob(, , .7)
  ggather(clip, rect, vp = viewport(x, y, width, height))
}
# a1. <- setnullvp("dataset", list(x = 1), .project)
# baf. <- baf(grobX(a1., 0), grobY(a1., 0))
```

75 File: b_mcn_dataset.R

```r
# ==========================================================================
# sub.frame
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
load(paste0(.expath, "/toBinary5.rdata"))
weight.mcn <- weight(mcn_dataset(toBinary5), slotNames(mcn_dataset(toBinary5)))
weight.mcn <-
  sapply(c("dataset", "reference", "backtrack"), simplify = F,
         function(x) weight.mcn[[x]])

grobs.mcn <- lst_grecti(names(weight.mcn), pal, "sub.slot", grecti2)
```

```r
# ==========================================================================
# content
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## dataset
## features
mshn <- 5
mshvp <- viewport(, , .7, .7)
fea <- circleGrob(gp = gpar(fill = "grey85"))
fea <- ggather(fea, vp = mshvp)
grob_feas <- frame_col(fill_list(n(fea, mshn), 1), fill_list(n(fea, mshn), list(fea)))
## function to draw
mglayer <- function(from = 1:5, n = 5, seed = 100) {
  set.seed(seed)
  ns <- sample(from, n, T)
  grobs <- lapply(ns, function(n) {
                    ggather(glayer(n), vp = mshvp)
         })
  sig <- paste0(paste0("glayer", seed), 1:n)
  names(grobs) <- sig
  frame_col(fill_list(sig, 1), grobs)
}
## set
grob_fset <- mglayer(, , 100)
grob_sset <- mglayer(2:7, , 110)
grob_cset <- mglayer(5:10, , 120)
## titles
tit_fea <- gltext("features")
tit_cand <- gltext("candidates", flip = T)
tit_formu <- gtext("molecular formula", gpar(fontface = "plain"))
tit_struc <- gtext("chemical structure", gpar(fontface = "plain"))
tit_class <- gtext("chemical classes", gpar(fontface = "plain"))
## combine
names <- c("grob_feas", "tit_formu", "grob_fset",
           "tit_struc", "grob_sset", "tit_class", "grob_cset")
env <- environment()
mdataset <- frame_row(fill_list(names, 1), sapply(names, get, envir = env))
mdataset <- frame_col(list(tit_cand = .1, mdataset = 1),
                      list(tit_cand = tit_cand, mdataset = mdataset))
mdataset <- frame_row(list(tit_fea = .1, mdataset = 1),
                      list(tit_fea = tit_fea, mdataset = mdataset))
.grob.dataset <- ggather(mdataset, vp = viewport(, , .9, .9))
## into
grobs.mcn$dataset %<>% into(.grob.dataset)

## reference
```

```
ref <- names(reference(mcn_dataset(toBinary5)))
## spe.table
df <- data.frame(.fea_id = n(id., 3), .cand_id = n(cand., 3))
grob_table <- gridExtra::tableGrob(df, theme = gridExtra::ttheme_default(8))
grobs_ref <- sapply(ref, simplify = F,
                    function(name) {
                      if (name == "nebula_index")
                        res <- into(gshiny(xn = 6), gtext(name))
                      else if (name == "specific_candidate") {
                        res <- frame_row(list(tit = .2, df = 1),
                                         list(tit = gtext(name, y = .2), df = grob_table))
                        res <- into(grectn("transparent"), res)
                      } else
                        res <- into(grectn("transparent"), gtext(name))
                      ggather(res, vp = viewport(, , .9))
                    })
weight.ref <- vapply(ref, FUN.VALUE = numeric(1),
                     function(name) {
                       if (name == "nebula_index") .5
                       else if (name == "specific_candidate") 1.3
                       else .5
                     })
grobs_ref <- frame_row(weight.ref, grobs_ref)
grobs.ref <- ggather(grobs_ref, vp = viewport(, , .9, .9))
## into
grobs.mcn$reference %<>% into(grobs.ref)

## backtrack
grob.trash <- ex_grob("trash")
grob_trash <- ggather(grob.trash, vp = viewport(, , .8, .8))
grobs.mcn$backtrack %<>% into(grob_trash)
```

```
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.size_subslot <- u(1, npc) - u(.5, line)
frame.mcn <- frame_row(weight.mcn, grobs.mcn)
frame.mcn <- ggather(frame.mcn, vp = viewport(, , .size_subslot, .size_subslot))
.mcn <- grecti2(form("mcn_dataset"), tfill = pal[[ "slot" ]])
.mcn <- into(.mcn, frame.mcn)
.mcn <- ggather(.mcn, vp = .gene.vp)
```

```
# draw(.mcn)
```

```
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## link
link <- c(
  # "dataset", "stardust_classes", "create_stardust_classes",
  # "dataset", "specific_candidate", "create_reference",
  # "dataset", "features_annotation", "create_features_annotation",
  # "specific_candidate", "features_annotation", "create_features_annotation",
  # "hierarchy", "stardust_classes", "create_stardust_classes",
  # "features_annotation", "nebula_index", "create_nebula_index",
  # "stardust_classes", "nebula_index", "create_nebula_index",
  "stardust_classes", "backtrack", "cross_filter_stardust",
  "backtrack", "stardust_classes", "backtrack_stardust"
)
link <- split(link, rep(1:(length(link) / 3), each = 3))
link <- data.frame(t(do.call(cbind, link)))
names(link) <- c("from", "to", "fun")
## the attributes for arrow
fun_pal <- ggsci::pal_d3("category20")(20)
link <- dplyr::mutate(link,
  group = agroup(to, 1:10, integer(1)),
  group = ifelse(from == "backtrack", max(group) + 1, group),
  color = agroup(group, fun_pal),
  left = ifelse(from %in% c("dataset", "specific_candidate"), F, T),
  left = ifelse(to %in% c("specific_candidate", "stardust_classes"),
    !left, left),
  up = ifelse(from == "backtrack", T, F),
  shift = ifelse(from %in% c("dataset", "backtrack"), 2, 2.5))

## tools
tools <- maparrow(.mcn, link)
link <- tools$data

## complex arr
com_arr <- lapply(1:length(tools$arr_city),
                  function(n) {
                    rect <- c("create_stardust_classes", "cross_filter_stardust",
                              "create_features_annotation", "backtrack_stardust")
                    if (link$fun[[ n ]] %in% rect & !link$dup[[ n ]])
                      tools$arr_city[[n]]
                    else
                      tools$arr_round[[n]]
                  })

## modify bafs
bafs <- sapply(names(tools$bafs), simplify = F,
               function(name){
                 sub_name <- gsub("\\.[a-z]$", "", name)
                 fun <- tools$bafs[[name]]
                 if (sub_name == "backtrack")
                   fun(h = u(2, line))
                 else if (sub_name == "specific_candidate")
                   fun(w = u(3, line))
                 else if (sub_name == "dataset")
                   fun()
                 else
                   fun(u(3, line), u(1, line))
               })

## show
.mcn <- do.call(ggather, c(list(.mcn), bafs, com_arr, tools$sags))
# draw(.mcn)

link_b <- link
```

76 File: c_nebulae.R

```
# ==========================================================================
# sub.frame
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

weight.nebulae <- c(parent_nebula = .7, child_nebulae = 1.5)
grobs.nebulae <- lst_grecti(names(weight.nebulae), pal, "slot", grecti2)
```

```
# ==========================================================================
# content
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## parent_nebula
```

```r
keep <- c("grid_layout", "viewports", "ggset", "...")
name.par <- c("...", slotNames(parent_nebula(toBinary5)))
fun <- function(names) {
  names <- names[ names %in% keep ]
  weight <- rep(10, length(names))
  weight[which(names == "...")] <- 2
  names(weight) <- names
  weight
}
weight.par <- fun(name.par)
## draw
grobs.par <- lst_grecti(names(weight.par), pal, "sub.slot", grecti2)
omit <- circleGrob(seq(.2, .8, , 3), .5, .07, gp = gpar(fill = "grey20"),
                   vp = viewport(, , u(1.5, line), u(1.5, line)))
grobs.par$... <- omit
## ggset
ids <- n(n, 5)
set.seed(99)
graph <- random_graph(ids, layout = "kk")
theme <- theme_void() + theme(legend.position = "none")
layer0 <- ggraph::ggraph(graph)
scale_size <- scale_size(range = c(1, 3))
layer1 <- layer0 + ggraph::geom_node_point(aes(size = size), shape = 21) +
  scale_size + theme
layer1 <- as_grob(layer1)
edge_color <- "lightblue"
layer2 <- layer0 +
  ggraph::geom_edge_fan(aes(edge_width = width), color = edge_color) + theme
layer2 <- as_grob(layer2)
layer4 <- ggplot(data.frame(x = 1:2, y = 1:2, fill = n(g, 2))) +
  geom_tile(aes(x = x, y = y, fill = fill)) +
  ggsci::scale_fill_npg() + theme
layer4 <- as_grob(layer4)
layer5 <- layer0 +
  ggraph::geom_edge_fan(aes(edge_width = width), color = edge_color) +
  ggraph::geom_node_point(aes(size = size, fill = name), shape = 21) +
  scale_size +
  ggsci::scale_fill_npg() +
  theme
layer5 <- layer5_ <- ggather(as_grob(layer5))
## ggset gather
```

```r
layers <- lapply(namel(layer1, layer2, layer4, layer5),
                 function(lay){
                   into(grectn(), lay)
                 })
wei_ggset <- c(layer1 = 1, plus = .5, layer2 = 1, plus = .5,
               layer4 = 1, equals = 1, layer5 = 1)
grob_ggset <- frame_col(wei_ggset, c(layers, list(plus = gtext("+"),
                                                  equals = gtext("=>"))))
grob_ggset <- ggather(grob_ggset, vp = viewport(, , .9, .5))
grobs.par$ggset %<>% into(grob_ggset)
## parent gather
.par <- frame_row(weight.par, grobs.par)
.par <- ggather(.par, vp = viewport(, , .size_subslot, .size_subslot))
## into
grobs.nebulae$parent_nebula %<>% into(.par)

## child_nebulae
name.chi <- c("...", slotNames(child_nebulae(toBinary5)), "...")
weight.chi <- fun(name.chi)
## draw
grobs.chi <- lst_grecti(names(weight.chi), pal, "sub.slot", grecti2)
grobs.chi$... <- omit
## signal grid layout
n <- 3
seq <- seq(.2, .8, length.out = n)
grid <- segmentsGrob(c(rep(0, n), seq),
                     c(seq, rep(1, n)),
                     c(rep(1, n), seq),
                     c(seq, rep(0, n)), gp = gpar(lty = "dashed"))
grid <- ggather(clipGrob(height = .8), rectGrob(), grid, vp = viewport(, , .8, .8))
grobs.chi$grid_layout %<>% into(grid)
## signal viewport
n <- 6
seq <- seq(135, 0, length.out = n)
seq2 <- seq(45, 0, length.out = n)
vps <- lapply(1:length(seq),
              function(n, fx = .5, fy = .5, x = .5) {
                ang <- seq[n]
                ang2 <- seq2[n]
                vp <- viewport(cospi(ang / 180) * fx + x, sinpi(ang / 180) * fy + x,
                               u(2, line), u(2, line), angle = ang2,
                                just = c("centre", "bottom"))
                 ggather(rectGrob(gp = gpar(fill = "lightyellow", col = pal[["sub.slot"]])),
                         gtext(paste0("n", rev(1:length(seq))[n]),
                               gpar(fontface = "plain"), form = F), vp = vp)
               })
vps <- do.call(ggather, c(vps, list(vp = viewport(, .1, .5, .5))))
grobs.chi$viewports %<>% into(vps)
## signal ggset
ggsets <- frame_col(c(n = 1, x = 1, mn = 1),
                    list(n = gtext("n", list(cex = 2.5), form = F),
                         x = gtext("×", list(cex = 2, font = c(plain = 1))),
                         mn = into(glayer(3), layer5_)))
ggsets <- ggather(ggsets, vp = viewport(, , .7, .7))
grobs.chi$ggset %<>% into(ggsets)
## child... gather
.chi <- frame_row(weight.chi, grobs.chi)
.chi <- ggather(.chi, vp = viewport(, , .size_subslot, .size_subslot))
## ino
grobs.nebulae$child_nebulae %<>% into(.chi)

## visualization
load(paste0(.expath, "/toActiv30.rdata"))
test1 <- toActiv30
set.seed(10)
## filter
reference(mcn_dataset(test1))$nebula_index %<>%
  dplyr::filter(class.name %in% sample(unique(class.name), 9))
test1 <- create_parent_nebula(test1, 0.01, 5, T)
test1 <- create_child_nebulae(test1, 0.01, 5)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)
## gather child grobs
chAsGrob <- function(ch, x) {
  ggset <- modify_default_child(ch)
  ggset@layers$ggtitle@command_args$label %<>%
    gsub("[a-zA-Z]", ".", .) %>%
    gsub("^....", "Class", .)
  as_grob(call_command(ggset))
}
sets <- lapply(ggset(child_nebulae(test1)), chAsGrob, x = test1)
sets <- lapply(names(sets),
               function(name) {
                 ggather(sets[[name]],
                         vp = viewports(child_nebulae(test1))[[name]])
               })
sets_vp <- viewport(layout = grid_layout(child_nebulae(test1)))
sets <- do.call(ggather, c(sets, list(vp = sets_vp)))
legendH <- MCnebula2:::.legend_hierarchy(child_nebulae(test1), test1)
export_name(test1)[["tani.score"]] <- "TS"
export_name(test1)[["similarity"]] <- "SS"
legendG <- ggset(child_nebulae(test1))[[1]] %>%
  modify_default_child(x = test1) %>%
  call_command() %>%
  MCnebula2:::.get_legend()
## integrate
vis <- frame_row(list(sets = 5, legendH = .5), namel(sets, legendH))
vis <- frame_col(list(vis = 4, legendG = 1), namel(vis, legendG))
## final
.grob.vis <- vis

## report
rept <- grecti("Report: Workflow", tfill = pal[["report"]],
               tgp_args = list(col = "transparent"), borderF = 1.5)
head <- gtext("4.4 Create Nebulae", x = 0, hjust = 0)
head2 <- gtext("4.5. Visualize Nebulae", x = 0, hjust = 0)
fig <- into(grectn(), gtext("Fig.", list(cex = 3)))
des <- gtext("Description...", x = .1, hjust = 0)
code <- into(grectn("grey95", , list(col = "transparent")),
             gtext("## code",
                   list(font = c(plain = 1)), x = .1, hjust = 0))
mores <- gltext("More sections")
cont <- frame_row(c(mores = .3, head = .3, des = .2, code = .3,
                    head2 = .3, fig = 1, des = .2, code = .3, mores = .3),
                  namel(head2, head, fig, des, code, mores))
cont <- ggather(cont, vp = viewport(, , .8, .8))
rept <- into(rept, cont)
## final
.grob.report <- rept
```

```
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
## nebulae # .size_subslot <- u(1, npc) - u(.5, line)
weight.nebulae[[ "ne_null" ]] <- .1
weight.nebulae <- weight.nebulae[c(1, 3, 2)]
.nebulae <- frame_row(weight.nebulae, c(grobs.nebulae, list(ne_null = nullGrob())))
## more
.vis <- frame_col(c(slotNebula = 1, null = .3, report = 1),
                  list(slotNebula = .nebulae, report = .grob.report, null = nullGrob()))
.vis <- frame_row(c(vis1 = 1, vis_null = .05, vis2 = 1),
                  list(vis1 = .vis, vis2 = .grob.vis, vis_null = nullGrob()))
## panel vp
.gene.vp.2w <- .gene.vp
.gene.vp.2w$width <- .gene.vp$width * 2
.nebulae <- ggather(.vis, vp = .gene.vp.2w)

## show
# x11(, 7, 11)
# draw(.nebulae)
```

```
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## link
link <- c(
  "grid_layout", "vis2", "visualize_all",
  "viewports", "vis2", "visualize_all",
  "ggset", "vis2", "visualize_all"
)
link <- split(link, rep(1:(length(link) / 3), each = 3))
link <- data.frame(t(do.call(cbind, link)))
names(link) <- c("from", "to", "fun")
## attributes
fun_pal <- ggsci::pal_d3("category20")(20)
fun_pal <- fun_pal[!fun_pal %in% unique(link_b$color)]
link <- dplyr::mutate(link,
  group = agroup(to, 1:10, integer(1)),
  color = agroup(group, fun_pal),
  left = F, up = F,
  shift = 2,
  dup = duplicated(group)
)
```

```r
## tools
tools_c <- maparrow(.nebulae, link, list(ggset = "child_nebulae.*::ggset"))
link <- tools_c$data

## bafs
bafs <- unlist(tools_c$bafs, F)
bafs <- lapply(bafs, function(fun) fun(u(1, line), u(1, line)))
bafs$vis2.r <- NULL
bafs$vis2 <- ({
  function(){
    g <- tools_c$nulls$vis2$t
    rectGrob(grobX(g, 0), grobY(g, 0), u(25, line), u(.05, line),
             gp = gpar(col = "transparent", fill = link$color[1]))
  }})()

## arrow
arr <- apply(link, 1, simplify = F,
             function(v) {
               pos <- ifelse(as.logical(v[[ "left" ]]), "l", "r")
               cur <- if (pos == "l") 1 else -1
               cur <- if (as.logical(v[[ "up" ]])) -cur else cur
               garrow(tools_c$nulls[[ v[["from"]] ]][[ pos ]],
                      tools_c$nulls[[ v[["to"]] ]][[ "t" ]],
                      list(curvature = cur, square = T, ncp = 1,
                           gp = gpar(fill = v[["color"]], col = v[["color"]],
                                     lwd = u(1, line)))
               )
             })

## sagnage
cvp <- garrow_city(tools_c$nulls$ggset$r, tools_c$nulls$ggset$t, F, F,
                   u(link$shift[1], line), gpar())@cvp
sag <- tools_c$sags[[1]]
sag$vp[[1]] <- cvp

.nebulae <- do.call(ggather, c(list(.nebulae), bafs, arr, list(sag)))
# draw(.nebulae)

link_c <- link
```

77 File: d_across.R

```
# ==========================================================================
# all grobs
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.over <- frame_col(list(a = .5, b = .5, c = 1),
                   list(a = .project, b = .mcn, c = .nebulae))
```

```
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## pal
fun_pal <- ggsci::pal_d3("category20")(20)
fun_pal <- fun_pal[!fun_pal %in% unique(c(link_b$color, link_c$color))]
fun_pal <- fun_pal[-1]

## a_dataset to b_dataset
# col <- fun_pal[1]
# a_dataset <- setnullvp("a::.*::dataset", list(x = 1, y = .5), .over)
# b_dataset <- setnullvp("b::.*::dataset", list(x = 0, y = .55), .over)
# arr <- garrow_snake(a_dataset, b_dataset, col)
# sag <- sagnage_shiny("Fil.", F, col)
# sag <- ggather(sag, vp = viewport(grobX(arr, 90), grobY(arr, 90)))

## a_dataset to b_spectral_similarity
# col <- fun_pal[2]
# b_spec <- setnullvp("b::.*::spectral_similarity", list(x = 0, y = .5), .over)
# baf2 <- baf(grobX(b_spec, 0), grobY(b_spec, 0), u(3, line), u(1, line))
# arr2 <- garrow_snake(a_dataset, b_spec, col, cur = -1)
# sag2 <- sagnage_shiny("Com.", F, col)
# sag2 <- ggather(sag2, vp = viewport(grobX(arr2, 270), grobY(arr2, 270)))

## b_nebula_index to c_child_nebula
col <- fun_pal[3]
b_index <- setnullvp("b::.*::nebula_index", list(x = 1, y = .5), .over)
b_index. <- setnullvp("b::.*::nebula_index", list(x = 1.2, y = .5), .over)
c_child <- setnullvp("c::.*::child_nebulae", list(x = 0, y = .95), .over)
baf3 <- baf(grobX(b_index, 0), grobY(b_index, 0), u(3, line), u(1, line))
arr3 <- garrow_snake(b_index., c_child, col)
sag3 <- sagnage_shiny("Cre.", F, col)
sag3 <- ggather(sag3, vp = viewport(grobX(arr3, 90), grobY(arr3, 90)))
```

```r
## b ... to c_vis2
col <- "grey50"
b_spec <- setnullvp("b::.*::spectral_similarity", list(x = 1.2, y = .5), .over)
c_vis2 <- setnullvp("c::.*::vis2", list(x = 0, y = .1), .over)
c_vis2. <- setnullvp("c::.*::vis2", list(x = 0, y = .6), .over)
seg.x <- ruler(b_spec, c_vis2.)
seg.y <- ruler(c_vis2, c_vis2.)
arr4 <- parrow(, col)
vp <- viewport(grobX(seg.y, 0), grobY(seg.y, 0), grobWidth(seg.x), grobHeight(seg.y),
               just = c("right", "centre"))
arr4$vp <- vp
sag4 <- sagnage_shiny("Cus.", F, col)
sag4 <- ggather(sag4, vp = vp)
```

```r
# ==========================================================================
# legend
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

le1 <- grecti2("...", tfill = pal[[ "slot" ]])
le2 <- grecti2("...", tfill = pal[[ "sub.slot" ]])
le <- frame_row(c(le1 = 1, leNull = .2, le2 = 1),
                namel(le1, leNull = nullGrob(), le2))
le.text <- frame_row(c(txt1 = 1, txtNull = .2, txt2 = 1),
                     list(txt1 = gtextp("slot", x = 0, hjust = 0),
                          txtNull = nullGrob(),
                          txt2 = gtextp("sub slot", x = 0, hjust = 0)))
le <- frame_col(c(leNull2 = .2, le = .6, leNull2 = .2, le.text = 1),
                namel(le, leNull2 = nullGrob(), le.text))

dsf_fpal <- dplyr::select(link_b, color, fun) %>%
  rbind(., dplyr::select(link_c, color, fun)) %>%
  rbind(.,
    # c(fun_pal[1], "filter"), c(fun_pal[2], "compute"),
    c(fun_pal[3], "create"), c("grey50", "custom")) %>%
  dplyr::mutate(des = stringr::str_extract(fun, "^[a-zA-Z]{1,}")) %>%
  dplyr::distinct(color, des) %>%
  split(~ des)
le3 <- sapply(dsf_fpal, simplify = F,
  function(df) {
    n <- if (nrow(df) > 1) 2 else 1
    sags <- lapply(1:n,
      function(n) {
        text <- paste0(substr(df$des[n], 1, 3), ".")
        sag <- sagnage_shiny(text, F, df$color[n])
        arr <- garrow_snake(list(x1 = .5, y1 = 1),
          list(x2 = .5, y2 = 0),
          df$color[n])
        size <- grobHeight(gtext("text")) * 5
        ggather(arr, sag, vp = viewport(, , size, size))
      })
    if (n  == 2)
      sags <- frame_col(c(c1 = 1, c2 = 2),
        list(c1 = sags[[1]], c2 = sags[[2]]))
    else
      sags[[1]]
  })
le3.obj <- sapply(sort(names(le3)), function(name) le3[[ name ]], simplify = F)
le3 <- frame_row(fill_list(names(le3.obj), 1), le3.obj)
dup <- vapply(dsf_fpal, function(df) if (nrow(df) > 1) T else F, T)
texts <- ifelse(dup, paste0("...  ", form(names(le3.obj))), names(le3.obj))
le3.text <- sapply(texts, simplify = F,
  function(lab) gtextp(lab, x = 0, hjust = 0))
le3.text <- frame_row(fill_list(names(le3.text), 1), le3.text)
le3 <- frame_col(c(le3 = 1, le3.text = 1), namel(le3, le3.text))

legend <- frame_row(c(tit1 = .1, le = .3, legendNull = .1,
                      tit2 = .1, le3 = 1),
                    namel(tit1 = gtext("Object", list(cex = 1.2)), le,
                          tit2 = gtext("Function", list(cex = 1.2)), le3,
                          legendNull = nullGrob()))
legend <- ggather(legend, vp = viewport(.2, , , .8))
```

```
# ==========================================================================
# Altogether
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

figure <- ggather(.over, baf3, arr3, sag3, arr4, sag4)
figure <- frame_col(c(figure = 1, legend = .15),
                    namel(figure, legend))
```

```
# ==========================================================================
# output
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```r
## gather all
pdf(tmp_pdf(), 15, 11)
draw(figure)
dev.off()
# op(tmp_pdf())
# file.copy(tmp_pdf(), "~/Documents/figure.pdf")
```

78 File: a_project.R

```r
# ==========================================================================
# external grob
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## external element
grob.path <- ex_grob("path")
grob.dir <- ex_grob("dir")
grob.api <- ex_grob("api")
grob.files <- ex_grob("files")
# draw(grob.table)
```

```r
# ==========================================================================
# sub.frame
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## melody: palette
type <- c("class", "slot", "sub.slot", "function", "custom")
pal <- MCnebula2:::.as_dic(palette_set(mcn_5features)[-3], type, na.rm = T)
pal[[ "report" ]] <- "black"

## weight
name.slots <- slotNames(mcn_5features)
name.project <- name.slots[grepl("^project", name.slots)]
weight.project <- weight(mcn_5features, name.project)
weight.project$project_metadata <- 3
names(weight.project) %<>% gsub("^project_", "", .)
weight.project <-
  sapply(c("dataset", "api", "metadata", "conformation", "version", "path"),
         simplify = F,
         function(x) weight.project[[ x ]])

grobs.project <- lst_grecti(names(weight.project), pal, , grecti2)
```

```
# ==========================================================================
# content
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## project_path
grobs.project$path <- into(grobs.project$path, grob.path)

## project_metadata
grectn <- grectn(, list(width = .9, height = .9),
                 gpar(lty = "solid"), b = match.fun(roundrectGrob))
grob.menu <- frame_row(fill_list(n(f, 3), 1), fill_list(n(f, 3), list(grectn@grob)))
grob.menu <- frame_col(fill_list(n(fc, 3), 1), fill_list(n(fc, 3), list(grob.menu)))
grob.menu <- gTree(children = gList(grob.menu), vp = viewport(, , .8, .8))
grob_files <- frame_row(c(labels = .1, files = 1),
                        list(labels = gtext("\n......\n", gpar(cex = 1.2), x = .5, y = .4),
                             files = grob.menu))
.grob.meta <- frame_row(fill_list(n(x, 3), 1), fill_list(n(x, 3), list(grob.dir)))
.grob.meta <- frame_col(c(dirs = 1, table = 2), list(dirs = .grob.meta, table = grob_files))
## arrow 1
a1 <- setnullvp("x2", list(x = .8, y = .8), x = .grob.meta)
a1. <- setnullvp("files", list(x = 0, y = 1), x = .grob.meta)
a1 <- garrow(a1, a1., list(curvature = 0, gp = .gpar_dashed_line), fun_arrow = NULL)
## arrow 2
a2 <- setnullvp("x2", list(x = .8, y = .2), .grob.meta)
a2. <- setnullvp("files", list(x = 0, y = 0), .grob.meta)
a2 <- garrow(a2, a2., list(curvature = 0, gp = .gpar_dashed_line), fun_arrow = NULL)
.grob.meta <- gTree(children = gList(.grob.meta, a1, a2))
## into
grobs.project$metadata %<>% into(.grob.meta)

## project_dataset
load(paste0(.expath, "/toAnno5.rdata"))
anno <- features_annotation(toAnno5)
## formula
fml <- anno$mol.formula[5]
fml <- gtext(paste0("Molecular Formula\n", fml), gpar(fontface = "plain"))
grob.formu <- into(glayer(), fml)
# draw(grob.formu)
## structure
## get grob structure
smi <- anno$smiles[5]
grob.struc <- sym_chem(smi)
```

```r
grob.struc <- ggather(into(glayer(6), grob.struc), vp = viewport(, , .7))
## classes
gp.cir <- gpar(lwd = u(3, line), lty = "dotted")
grob.gla <- gTree(children = gList(gtext("?", list(cex = 1.5, col = "red"), x = .7, y = .3),
                                   circleGrob(, , .4, gp = gp.cir)))
## Benzene
smi <- "C1=CC=CC=C1"
grob.ben <- sym_chem(smi)
## group
grob.ben_gla <- gTree(children = gList(grob.ben, grob.gla))
class.n <- 5
grob.classes <- into(glayer(class.n), grob.ben_gla)
## omit
grob.omit <- circleGrob(seq(.2, .8, , 3), .5, .07, gp = gpar(fill = "grey20"))
grob.omit <- into(glayer(class.n), grob.omit)
## groups
grob.groups <- frame_col(list(gomit = 1, gclasses = 1, gomit = 1),
                         list(gomit = grob.omit, gclasses = grob.classes))
## gather
more <- ggather(gltext("More data"), vp = viewport(, , .7))
.grob.dataset <- frame_row(list(formu = 1, null = .2, struc = 1, null = .2, class = 1,
                                null = .2, more = .3),
                          list(formu = grob.formu, struc = grob.struc, class = grob.groups,
                               null = nullGrob(), more = more))
.grob.dataset <- gTree(children = gList(.grob.dataset), vp = viewport(, , .8, .8))
## into
grobs.project$dataset %<>% into(.grob.dataset)

## project_conformation
text <- c(.id = "regex", .f2_formula = "*.tsv", .f3_fingerid = "*.tsv",
          .f3_canopus = "*.fpt", ... = "...")
text.title <- c(.id = "id", .f2_formula = "form.", .f3_fingerid = "struc.",
                .f3_canopus = "class", ... = "...")
nodes.text <- sapply(names(text), simplify = F,
                     function(name) {
                       grectN(text.title[[ name ]], text[[ name ]], bfill = "lightblue")
                     })
## nodes
nodes.1 <- frame_col(list(Nnull = 1.5, id = 1, Nnull = 1.5),
                     list(Nnull = nullGrob(), id = nodes.text[[ ".id" ]]))
nodes.2 <- frame_col(list(formu = 1, Nnull = .5, fing = 1, Nnull = .5, cano = 1),
                     list(Nnull = nullGrob(), formu = nodes.text[[ ".f2_formula" ]],
                          fing = nodes.text[[ ".f3_fingerid" ]],
                          cano = nodes.text[[ ".f3_canopus" ]]))
nodes <- frame_row(list(nodes.1 = 1, Nnull = .7, nodes.2 = 1),
                   list(nodes.1 = nodes.1, nodes.2 = nodes.2, Nnull = nullGrob()))
## add arrow
a1 <- setnullvp("id", list(x = .4, y = 0), nodes)
a1. <- setnullvp("formu", list(x = .6, y = 1), nodes)
a1 <- garrow(a1, a1., list(inflect = T, curvature = -.1))
a2 <- setnullvp("id", list(x = .5, y = 0), nodes)
a2. <- setnullvp("fing", list(x = .5, y = 1), nodes)
a2 <- garrow(a2, a2., list(inflect = T, curvature = .1, gp = .gpar_dotted_line))
a3 <- setnullvp("id", list(x = .6, y = 0), nodes)
a3. <- setnullvp("cano", list(x = .4, y = 1), nodes)
a3 <- garrow(a3, a3., list(inflect = T, curvature = .1, gp = .gpar_dotted_line))
.grob.confo <- gTree(children = gList(nodes, a1, a2, a3), vp = viewport(, , .8, .8))
## into
grobs.project$conformation %<>% into(.grob.confo)

## project_api
df <- data.frame(ID = n(ID., 3), Form. = n(C.H.O., 3), Struc. = n("CO=CC.", 3))
grob_table <- gridExtra::tableGrob(df, theme = gridExtra::ttheme_default(8))
grob_api <- frame_row(list(pfiles = 1, plug = 1),
                      list(pfiles = grob.files, plug = grob.api))
.grob.api <- frame_col(list(extract = .2, gtable = 1),
                       list(extract = grob_api, gtable = grob_table))
a1 <- setnullvp("pfiles", list(x = .5, y = .2), .grob.api)
a1. <- setnullvp("plug", list(x = .5, y = .8), .grob.api)
a1 <- garrow(a1, a1.)
a2 <- setnullvp("plug", list(x = .9), .grob.api)
a2. <- setnullvp("gtable", list(x = .1), .grob.api)
a2 <- garrow(a2, a2., list(inflect = T))
.grob.api <- gTree(children = gList(.grob.api, a1, a2))
## into
grobs.project$api %<>% into(.grob.api)

## project_version
.grob.version <- gtext("sirius.v4\n...", list(cex = 1.5))
## into
grobs.project$version %<>% into(.grob.version)
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if.ex <- grepl("version|path", names(weight.project))
frame.project <- frame_row(weight.project, grobs.project, if.ex)
.gene.vp <- viewport(, , width = unit(8 * 1.5, "line"),
                     height = unit(35 * 1.5, "line"), clip = "off")
.project <- gTree(children = gList(frame.project), vp = .gene.vp)
```

```r
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
pal.collate <- colorRampPalette(c("white", "lightblue"))(6)[c(-1, -2)]
a_gpar <-
  lapply(pal.collate,
         function(col) {
           gpar(fill = col, col = col, lwd = u(2, line))
         })

a.set <- c("path", "conformation", "metadata", "api", "dataset")
baf <- function(x, y, width = u(1, line), height = u(3, line)) {
  rect <- grectn(bgp_args = gpar(lty = "solid"))@grob
  clip <- clipGrob(, , .7)
  ggather(clip, rect, vp = viewport(x, y, width, height))
}
for (i in 2:length(a.set)) {
  a1 <- setnullvp(a.set[i - 1], list(x = 1), .project)
  a1. <- setnullvp(a.set[i], list(x = 1), .project)
  assign(paste0("arr", i - 1),
         garrow(a1, a1., list(gp = a_gpar[[ i - 1 ]]), city = list(shift = u(1, line))))
  assign(paste0("baf", i - 1), baf(grobX(a1, 0), grobY(a1, 0)))
}
baf. <- baf(grobX(a1., 0), grobY(a1., 0))

## gather
.project <- ggather(.project,
                    baf1, baf2, baf3, baf4, baf.,
                    arr1, arr2, arr3, arr4)
```

79 File: b_mcn_dataset.R

```r
# ==========================================================================
# sub.frame
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
load(paste0(.expath, "/toBinary5.rdata"))
weight.mcn <- weight(mcn_dataset(toBinary5), slotNames(mcn_dataset(toBinary5)))
weight.mcn <-
  sapply(c("dataset", "reference", "backtrack"), simplify = F,
         function(x) weight.mcn[[x]])

grobs.mcn <- lst_grecti(names(weight.mcn), pal, "sub.slot", grecti2)
```

```r
# ==========================================================================
# content
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## dataset
## features
mshn <- 5
mshvp <- viewport(, , .7, .7)
fea <- circleGrob(gp = gpar(fill = "grey85"))
fea <- ggather(fea, vp = mshvp)
grob_feas <- frame_col(fill_list(n(fea, mshn), 1), fill_list(n(fea, mshn), list(fea)))
## function to draw
mglayer <- function(from = 1:5, n = 5, seed = 100) {
  set.seed(seed)
  ns <- sample(from, n, T)
  grobs <- lapply(ns, function(n) {
                    ggather(glayer(n), vp = mshvp)
         })
  sig <- paste0(paste0("glayer", seed), 1:5)
  names(grobs) <- sig
  frame_col(fill_list(sig, 1), grobs)
}
## set
grob_fset <- mglayer(, , 100)
grob_sset <- mglayer(2:7, , 110)
grob_cset <- mglayer(5:10, , 120)
## titles
tit_fea <- gltext("features")
tit_cand <- gltext("candidates", flip = T)
tit_formu <- gtext("molecular formula", gpar(fontface = "plain"))
```

```r
tit_struc <- gtext("chemical structure", gpar(fontface = "plain"))
tit_class <- gtext("chemical classes", gpar(fontface = "plain"))
## combine
names <- c("grob_feas", "tit_formu", "grob_fset",
           "tit_struc", "grob_sset", "tit_class", "grob_cset")
env <- environment()
mdataset <- frame_row(fill_list(names, 1), sapply(names, get, envir = env))
mdataset <- frame_col(list(tit_cand = .1, mdataset = 1),
                      list(tit_cand = tit_cand, mdataset = mdataset))
mdataset <- frame_row(list(tit_fea = .1, mdataset = 1),
                      list(tit_fea = tit_fea, mdataset = mdataset))
.grob.dataset <- ggather(mdataset, vp = viewport(, , .9, .9))
## into
grobs.mcn$dataset %<>% into(.grob.dataset)

## reference
ref <- names(reference(mcn_dataset(toBinary5)))
## spe.table
df <- data.frame(.fea_id = n(id., 3), .cand_id = n(cand., 3))
grob_table <- gridExtra::tableGrob(df, theme = gridExtra::ttheme_default(8))
grobs_ref <- sapply(ref, simplify = F,
                    function(name) {
                      if (name == "nebula_index")
                        res <- into(gshiny(xn = 6), gtext(name))
                      else if (name == "specific_candidate") {
                        res <- frame_row(list(tit = .2, df = 1),
                                         list(tit = gtext(name, y = .2), df = grob_table))
                        res <- into(grectn("transparent"), res)
                      } else
                        res <- into(grectn("transparent"), gtext(name))
                      ggather(res, vp = viewport(, , .9))
                    })
weight.ref <- vapply(ref, FUN.VALUE = numeric(1),
                     function(name) {
                       if (name == "nebula_index") .5
                       else if (name == "specific_candidate") 1.3
                       else .5
                     })
grobs_ref <- frame_row(weight.ref, grobs_ref)
grobs.ref <- ggather(grobs_ref, vp = viewport(, , .9, .9))
## into
```

```r
grobs.mcn$reference %<>% into(grobs.ref)

## backtrack
grob.trash <- ex_grob("trash")
grob_trash <- ggather(grob.trash, vp = viewport(, , .8, .8))
grobs.mcn$backtrack %<>% into(grob_trash)
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.size_subslot <- u(1, npc) - u(.5, line)
frame.mcn <- frame_row(weight.mcn, grobs.mcn)
frame.mcn <- ggather(frame.mcn, vp = viewport(, , .size_subslot, .size_subslot))
.mcn <- grecti2(form("mcn_dataset"), tfill = pal[[ "slot" ]])
.mcn <- into(.mcn, frame.mcn)
.mcn <- ggather(.mcn, vp = .gene.vp)

## show
# x11(, 7, 11)
# draw(.mcn)
```

```r
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## link
link <- c("dataset", "stardust_classes", "create_stardust_classes",
          "dataset", "specific_candidate", "create_reference",
          "dataset", "features_annotation", "create_features_annotation",
          "specific_candidate", "features_annotation", "create_features_annotation",
          "hierarchy", "stardust_classes", "create_stardust_classes",
          "features_annotation", "nebula_index", "create_nebula_index",
          "stardust_classes", "nebula_index", "create_nebula_index",
          "stardust_classes", "backtrack", "cross_filter_stardust",
          "backtrack", "stardust_classes", "backtrack_stardust"
)
link <- split(link, rep(1:(length(link) / 3), each = 3))
link <- data.frame(t(do.call(cbind, link)))
names(link) <- c("from", "to", "fun")
## the attributes for arrow
fun_pal <- ggsci::pal_d3("category20")(20)
link <- dplyr::mutate(link,
                      group = agroup(to, 1:10, integer(1)),
                      group = ifelse(from == "backtrack", max(group) + 1, group),
                      color = agroup(group, fun_pal),
                      left = ifelse(from %in% c("dataset", "specific_candidate"), F, T),
                      left = ifelse(to %in% c("specific_candidate", "stardust_classes"),
                                    !left, left),
                      up = ifelse(from == "backtrack", T, F),
                      shift = ifelse(from %in% c("dataset", "backtrack"), 2, 2.5))

## tools
tools <- maparrow(.mcn, link)
link <- tools$data

## complex arr
com_arr <- lapply(1:length(tools$arr_city),
                  function(n) {
                    rect <- c("create_stardust_classes", "cross_filter_stardust",
                              "create_features_annotation", "backtrack_stardust")
                    if (link$fun[[ n ]] %in% rect & !link$dup[[ n ]])
                      tools$arr_city[[n]]
                    else
                      tools$arr_round[[n]]
                  })

## modify bafs
bafs <- sapply(names(tools$bafs), simplify = F,
               function(name){
                 sub_name <- gsub("\\.[a-z]$", "", name)
                 fun <- tools$bafs[[name]]
                 if (sub_name == "backtrack")
                   fun(h = u(2, line))
                 else if (sub_name == "specific_candidate")
                   fun(w = u(3, line))
                 else if (sub_name == "dataset")
                   fun()
                 else
                   fun(u(3, line), u(1, line))
               })

## show
.mcn <- do.call(ggather, c(list(.mcn), bafs, com_arr, tools$sags))
```

```r
# draw(.mcn)

link_b <- link
```

80 File: c_nebulae.R

```r
# ===========================================================================
# sub.frame
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

weight.nebulae <- c(parent_nebula = .7, child_nebulae = 1.5)
grobs.nebulae <- lst_grecti(names(weight.nebulae), pal, "slot", grecti2)
```

```r
# ===========================================================================
# content
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## parent_nebula
keep <- c("grid_layout", "viewports", "ggset", "...")
name.par <- c("...", slotNames(parent_nebula(toBinary5)))
fun <- function(names) {
  names <- names[ names %in% keep ]
  weight <- rep(10, length(names))
  weight[which(names == "...")] <- 2
  names(weight) <- names
  weight
}
weight.par <- fun(name.par)
## draw
grobs.par <- lst_grecti(names(weight.par), pal, "sub.slot", grecti2)
omit <- circleGrob(seq(.2, .8, , 3), .5, .07, gp = gpar(fill = "grey20"),
                   vp = viewport(, , u(1.5, line), u(1.5, line)))
grobs.par$... <- omit
## ggset
ids <- n(n, 5)
set.seed(99)
graph <- random_graph(ids, layout = "kk")
theme <- theme_void() + theme(legend.position = "none")
layer0 <- ggraph::ggraph(graph)
scale_size <- scale_size(range = c(1, 3))
layer1 <- layer0 + ggraph::geom_node_point(aes(size = size), shape = 21) +
  scale_size + theme
layer1 <- as_grob(layer1)
edge_color <- "lightblue"
layer2 <- layer0 +
  ggraph::geom_edge_fan(aes(edge_width = width), color = edge_color) + theme
layer2 <- as_grob(layer2)
layer4 <- ggplot(data.frame(x = 1:2, y = 1:2, fill = n(g, 2))) +
  geom_tile(aes(x = x, y = y, fill = fill)) +
  ggsci::scale_fill_npg() + theme
layer4 <- as_grob(layer4)
layer5 <- layer0 +
  ggraph::geom_edge_fan(aes(edge_width = width), color = edge_color) +
  ggraph::geom_node_point(aes(size = size, fill = name), shape = 21) +
  scale_size +
  ggsci::scale_fill_npg() +
  theme
layer5 <- layer5_ <- ggather(as_grob(layer5))
## ggset gather
layers <- lapply(namel(layer1, layer2, layer4, layer5),
                 function(lay){
                   into(grectn(), lay)
                 })
wei_ggset <- c(layer1 = 1, plus = .5, layer2 = 1, plus = .5,
               layer4 = 1, equals = 1, layer5 = 1)
grob_ggset <- frame_col(wei_ggset, c(layers, list(plus = gtext("+"),
                                                  equals = gtext("=>"))))
grob_ggset <- ggather(grob_ggset, vp = viewport(, , .9, .5))
grobs.par$ggset %<>% into(grob_ggset)
## parent gather
.par <- frame_row(weight.par, grobs.par)
.par <- ggather(.par, vp = viewport(, , .size_subslot, .size_subslot))
## into
grobs.nebulae$parent_nebula %<>% into(.par)

## child_nebulae
name.chi <- c("...", slotNames(child_nebulae(toBinary5)), "...")
weight.chi <- fun(name.chi)
## draw
grobs.chi <- lst_grecti(names(weight.chi), pal, "sub.slot", grecti2)
grobs.chi$... <- omit
## signal grid layout
```

```r
n <- 3
seq <- seq(.2, .8, length.out = n)
grid <- segmentsGrob(c(rep(0, n), seq),
                     c(seq, rep(1, n)),
                     c(rep(1, n), seq),
                     c(seq, rep(0, n)), gp = gpar(lty = "dashed"))
grid <- ggather(clipGrob(height = .8), rectGrob(), grid, vp = viewport(, , .8, .8))
grobs.chi$grid_layout %<>% into(grid)
## signal viewport
n <- 6
seq <- seq(135, 0, length.out = n)
seq2 <- seq(45, 0, length.out = n)
vps <- lapply(1:length(seq),
              function(n, fx = .5, fy = .5, x = .5) {
                ang <- seq[n]
                ang2 <- seq2[n]
                vp <- viewport(cospi(ang / 180) * fx + x, sinpi(ang / 180) * fy + x,
                               u(2, line), u(2, line), angle = ang2,
                               just = c("centre", "bottom"))
                ggather(rectGrob(gp = gpar(fill = "lightyellow", col = pal[["sub.slot"]])),
                        gtext(paste0("n", rev(1:length(seq))[n]),
                              gpar(fontface = "plain"), form = F), vp = vp)
              })
vps <- do.call(ggather, c(vps, list(vp = viewport(, .1, .5, .5))))
grobs.chi$viewports %<>% into(vps)
## signal ggset
ggsets <- frame_col(c(n = 1, x = 1, mn = 1),
                    list(n = gtext("n", list(cex = 2.5), form = F),
                         x = gtext("×", list(cex = 2, font = c(plain = 1))),
                         mn = into(glayer(3), layer5_)))
ggsets <- ggather(ggsets, vp = viewport(, , .7, .7))
grobs.chi$ggset %<>% into(ggsets)
## child... gather
.chi <- frame_row(weight.chi, grobs.chi)
.chi <- ggather(.chi, vp = viewport(, , .size_subslot, .size_subslot))
## ino
grobs.nebulae$child_nebulae %<>% into(.chi)

## visualization
load(paste0(.expath, "/toActiv30.rdata"))
test1 <- toActiv30
```

```r
set.seed(10)
## filter
reference(mcn_dataset(test1))$nebula_index %<>%
  dplyr::filter(class.name %in% sample(unique(class.name), 9))
test1 <- create_parent_nebula(test1, 0.01, T)
test1 <- create_child_nebulae(test1, 0.01, 5)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)
## gather child grobs
chAsGrob <- function(ch, x) {
  ggset <- modify_default_child(ch)
  ggset@layers$ggtitle@command_args$label %<>%
    gsub("[a-zA-Z]", ".", .) %>%
    gsub("^....", "Class", .)
  as_grob(call_command(ggset))
}
sets <- lapply(ggset(child_nebulae(test1)), chAsGrob, x = test1)
sets <- lapply(names(sets),
               function(name) {
                 ggather(sets[[name]],
                         vp = viewports(child_nebulae(test1))[[name]])
               })
sets_vp <- viewport(layout = grid_layout(child_nebulae(test1)))
sets <- do.call(ggather, c(sets, list(vp = sets_vp)))
legendH <- MCnebula2:::.legend_hierarchy(child_nebulae(test1), test1)
export_name(test1)[["tani.score"]] <- "TS"
export_name(test1)[["similarity"]] <- "SS"
legendG <- ggset(child_nebulae(test1))[[1]] %>%
  modify_default_child(x = test1) %>%
  call_command() %>%
  MCnebula2:::.get_legend()
## integrate
vis <- frame_row(list(sets = 5, legendH = .5), namel(sets, legendH))
vis <- frame_col(list(vis = 4, legendG = 1), namel(vis, legendG))
## final
.grob.vis <- vis

## report
rept <- grecti("Report: Title", tfill = pal[["report"]],
               tgp_args = list(col = "transparent"), borderF = 1.5)
```

```r
head <- gtext("1. Heading", x = 0, hjust = 0)
head2 <- gtext("2. Heading", x = 0, hjust = 0)
fig <- into(grectn(), gtext("Fig.", list(cex = 3)))
des <- gtext("Description...", x = .1, hjust = 0)
code <- into(grectn("grey95", , list(col = "transparent")),
             gtext("## code",
                   list(font = c(plain = 1)), x = .1, hjust = 0))
mores <- gltext("More sections")
cont <- frame_row(c(head = .3, des = .2, code = .3,
                    head2 = .3, fig = 1, des = .2, code = .3, mores = .3),
                  namel(head2, head, fig, des, code, mores))
cont <- ggather(cont, vp = viewport(, , .8, .8))
rept <- into(rept, cont)
## final
.grob.report <- rept
```

```r
# ==========================================================================
# gather
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## nebulae # .size_subslot <- u(1, npc) - u(.5, line)
weight.nebulae[[ "ne_null" ]] <- .1
weight.nebulae <- weight.nebulae[c(1, 3, 2)]
.nebulae <- frame_row(weight.nebulae, c(grobs.nebulae, list(ne_null = nullGrob())))
## more
.vis <- frame_col(c(slotNebula = 1, null = .3, report = 1),
                  list(slotNebula = .nebulae, report = .grob.report, null = nullGrob()))
.vis <- frame_row(c(vis1 = 1, vis_null = .05, vis2 = 1),
                  list(vis1 = .vis, vis2 = .grob.vis, vis_null = nullGrob()))
## panel vp
.gene.vp.2w <- .gene.vp
.gene.vp.2w$width <- .gene.vp$width * 2
.nebulae <- ggather(.vis, vp = .gene.vp.2w)

## show
# x11(, 7, 11)
# draw(.nebulae)
```

```r
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


## link
```

```
link <- c(
          "grid_layout", "vis2", "visualize_all",
          "viewports", "vis2", "visualize_all",
          "ggset", "vis2", "visualize_all"
)
link <- split(link, rep(1:(length(link) / 3), each = 3))
link <- data.frame(t(do.call(cbind, link)))
names(link) <- c("from", "to", "fun")
## attributes
fun_pal <- ggsci::pal_d3("category20")(20)
fun_pal <- fun_pal[!fun_pal %in% unique(link_b$color)]
link <- dplyr::mutate(link,
                      group = agroup(to, 1:10, integer(1)),
                      color = agroup(group, fun_pal),
                      left = F, up = F,
                      shift = 2,
                      dup = duplicated(group))

## tools
tools_c <- maparrow(.nebulae, link, list(ggset = "child_nebulae.*::ggset"))
link <- tools_c$data

## bafs
bafs <- unlist(tools_c$bafs, F)
bafs <- lapply(bafs, function(fun) fun(u(1, line), u(1, line)))
bafs$vis2.r <- NULL
bafs$vis2 <- ({
  function(){
    g <- tools_c$nulls$vis2$t
    rectGrob(grobX(g, 0), grobY(g, 0), u(25, line), u(.05, line),
             gp = gpar(col = "transparent", fill = link$color[1]))
  }})()

## arrow
arr <- apply(link, 1, simplify = F,
             function(v) {
               pos <- ifelse(as.logical(v[[ "left" ]]), "l", "r")
               cur <- if (pos == "l") 1 else -1
               cur <- if (as.logical(v[[ "up" ]])) -cur else cur
               garrow(tools_c$nulls[[ v[["from"]] ]][[ pos ]],
                      tools_c$nulls[[ v[["to"]] ]][[ "t" ]],
                      list(curvature = cur, square = T, ncp = 1,
                           gp = gpar(fill = v[["color"]], col = v[["color"]],
                                     lwd = u(1, line)))
               )
             })

## sagnage
cvp <- garrow_city(tools_c$nulls$ggset$r, tools_c$nulls$ggset$t, F, F,
                   u(link$shift[1], line), gpar())@cvp
sag <- tools_c$sags[[1]]
sag$vp[[1]] <- cvp

## show
.nebulae <- do.call(ggather, c(list(.nebulae), bafs, arr, list(sag)))
# draw(tmp.nebulae)

link_c <- link
```

81 File: d_across.R

```
# ==========================================================================
# all grobs
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.over <- frame_col(list(a = .5, b = .5, c = 1),
                   list(a = .project, b = .mcn, c = .nebulae))
```

```
# ==========================================================================
# line and arrow
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## pal
fun_pal <- ggsci::pal_d3("category20")(20)
fun_pal <- fun_pal[!fun_pal %in% unique(c(link_b$color, link_c$color))]
fun_pal <- fun_pal[-1]

## a_dataset to b_dataset
col <- fun_pal[1]
a_dataset <- setnullvp("a::.*::dataset", list(x = 1, y = .5), .over)
b_dataset <- setnullvp("b::.*::dataset", list(x = 0, y = .55), .over)
arr <- garrow_snake(a_dataset, b_dataset, col)
sag <- sagnage_shiny("Fil.", F, col)
```

```r
sag <- ggather(sag, vp = viewport(grobX(arr, 90), grobY(arr, 90)))

## a_dataset to b_spectral_similarity
col <- fun_pal[2]
b_spec <- setnullvp("b::.*::spectral_similarity", list(x = 0, y = .5), .over)
baf2 <- baf(grobX(b_spec, 0), grobY(b_spec, 0), u(3, line), u(1, line))
arr2 <- garrow_snake(a_dataset, b_spec, col, cur = -1)
sag2 <- sagnage_shiny("Com.", F, col)
sag2 <- ggather(sag2, vp = viewport(grobX(arr2, 270), grobY(arr2, 270)))

## b_nebula_index to c_child_nebula
col <- fun_pal[4]
b_index <- setnullvp("b::.*::nebula_index", list(x = 1, y = .5), .over)
b_index. <- setnullvp("b::.*::nebula_index", list(x = 1.2, y = .5), .over)
c_child <- setnullvp("c::.*::child_nebulae", list(x = 0, y = .95), .over)
baf3 <- baf(grobX(b_index, 0), grobY(b_index, 0), u(3, line), u(1, line))
arr3 <- garrow_snake(b_index., c_child, col)
sag3 <- sagnage_shiny("Cre.", F, col)
sag3 <- ggather(sag3, vp = viewport(grobX(arr3, 90), grobY(arr3, 90)))

## b ... to c_vis2
col <- "grey50"
b_spec <- setnullvp("b::.*::spectral_similarity", list(x = 1.2, y = .5), .over)
c_vis2 <- setnullvp("c::.*::vis2", list(x = 0, y = .1), .over)
c_vis2. <- setnullvp("c::.*::vis2", list(x = 0, y = .6), .over)
seg.x <- ruler(b_spec, c_vis2.)
seg.y <- ruler(c_vis2, c_vis2.)
arr4 <- parrow(, col)
vp <- viewport(grobX(seg.y, 0), grobY(seg.y, 0), grobWidth(seg.x), grobHeight(seg.y),
               just = c("right", "centre"))
arr4$vp <- vp
sag4 <- sagnage_shiny("Cus.", F, col)
sag4 <- ggather(sag4, vp = vp)
```

```r
# ==========================================================================
# legend
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

le1 <- grecti2("...", tfill = pal[[ "slot" ]])
le2 <- grecti2("...", tfill = pal[[ "sub.slot" ]])
le <- frame_row(c(le1 = 1, leNull = .2, le2 = 1),
                namel(le1, leNull = nullGrob(), le2))
```

```r
le.text <- frame_row(c(txt1 = 1, txtNull = .2, txt2 = 1),
                     list(txt1 = gtextp("slot", x = 0, hjust = 0),
                          txtNull = nullGrob(),
                          txt2 = gtextp("sub slot", x = 0, hjust = 0)))
le <- frame_col(c(leNull2 = .2, le = .6, leNull2 = .2, le.text = 1),
                namel(le, leNull2 = nullGrob(), le.text))

dsf_fpal <- dplyr::select(link_b, color, fun) %>%
  rbind(., dplyr::select(link_c, color, fun)) %>%
  rbind(., c(fun_pal[1], "filter"), c(fun_pal[2], "compute"),
        c(fun_pal[4], "create"), c("grey50", "custom")) %>%
  dplyr::mutate(des = stringr::str_extract(fun, "^[a-zA-Z]{1,}")) %>%
  dplyr::distinct(color, des) %>%
  split(~ des)
le3 <- sapply(dsf_fpal, simplify = F,
              function(df) {
                n <- if (nrow(df) > 1) 2 else 1
                sags <- lapply(1:n,
                               function(n) {
                                 text <- paste0(substr(df$des[n], 1, 3), ".")
                                 sag <- sagnage_shiny(text, F, df$color[n])
                                 arr <- garrow_snake(list(x1 = .5, y1 = 1),
                                                     list(x2 = .5, y2 = 0),
                                                     df$color[n])
                                 size <- grobHeight(gtext("text")) * 5
                                 ggather(arr, sag, vp = viewport(, , size, size))
                               })
                if (n  == 2)
                  sags <- frame_col(c(c1 = 1, c2 = 2),
                                    list(c1 = sags[[1]], c2 = sags[[2]]))
                else
                  sags[[1]]
              })
le3.obj <- sapply(sort(names(le3)), function(name) le3[[ name ]], simplify = F)
le3 <- frame_row(fill_list(names(le3.obj), 1), le3.obj)
dup <- vapply(dsf_fpal, function(df) if (nrow(df) > 1) T else F, T)
texts <- ifelse(dup, paste0("...  ", form(names(le3.obj))), names(le3.obj))
le3.text <- sapply(texts, simplify = F,
                   function(lab) gtextp(lab, x = 0, hjust = 0))
le3.text <- frame_row(fill_list(names(le3.text), 1), le3.text)
le3 <- frame_col(c(le3 = 1, le3.text = 1), namel(le3, le3.text))
```

```r
legend <- frame_row(c(tit1 = .1, le = .3, legendNull = .1,
                      tit2 = .1, le3 = 1),
                    namel(tit1 = gtext("Object", list(cex = 1.2)), le,
                          tit2 = gtext("Function", list(cex = 1.2)), le3,
                          legendNull = nullGrob()))
legend <- ggather(legend, vp = viewport(.2, , , .8))
```

```r
# ==========================================================================
# Altogether
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

figure <- ggather(.over, arr, sag, baf2, arr2, sag2,
                  baf3, arr3, sag3, arr4, sag4)
figure <- frame_col(c(figure = 1, legend = .15),
                    namel(figure, legend))
```

```r
# ==========================================================================
# output
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
## gather all
pdf(tmp_pdf(), 17, 11)
draw(figure)
dev.off()
op(tmp_pdf())

# file.copy(tmp_pdf(), "~/Documents/figure.pdf")
```

82 File: merge_into_examples.R

```r
setwd("~/MCnebula2/R")
library(magrittr)

script <- list.files(".", pattern = "\\.R$") %>%
  sapply(readLines)

ex_path <- "/home/echo/outline/mc.test/example/"

dontrun <- function(codes){
  codes <- c("#' @examples",
             "#' \\dontrun{",
             paste0("#'   ", codes),
             "#' }"
  )
  paste0(codes, collapse = "\n")
}

target <-
  lapply(script,
         function(text){
           sig <- 0
           record <- list()
           for (i in 1:length(text)) {
             if (!grepl("^#'", text[i])) {
               if (sig == 3) {
                 if (grepl("^[a-z|A-Z|.]", text[i])) {
                   if (!is.null(record[[tar]])) {
                     sig <- 1
                     next
                   }
                   record[[tar]] <- T
                   file <- paste0(ex_path, tar, ".R")
                   codes <- readLines(file)
                   start <- grep("^## codes", codes)[1] + 1
                   if (length(codes) < start) {
                     next
                   }
                   codes <- codes[start:length(codes)]
                   codes <- dontrun(codes)
                   if (!grepl("#'\\s{0,}$", text[i - 1])) {
                     codes <- paste0("#'\n", codes)
                   }
                   text[i] <- paste0(codes, "\n", text[i])
                 }
               }
               sig <- 1
               next
             }
             if (grepl("^#' @.{0,2}name", text[i])) {
               if (sig == 2) {
                 sig <- 3
                 tar <- stringr::str_extract(text[i], "(?<=^#' @[a-z]{0,2}name ).{1,}(?=$)")
               }
```

```r
              next
            }
            if (grepl("^#' @param|^#' @inheritParams", text[i])) {
              sig <- 2
              next
            }
          }
          text
        })

lapply(target, writeLines)

path <- "~/code_backup/mcnebula2"
dir.create(path, recursive = T)

lapply(names(target),
       function(name){
         writeLines(target[[name]], paste0(path, "/", name))
       })

system("cp ~/code_backup/mcnebula2/* -t ~/MCnebula2/R")
```

83 File: new_examples.R

```r
setwd("~/MCnebula2/R")
library(magrittr)

script <- list.files(".", pattern = "\\.R$") %>%
  sapply(readLines)

target <-
  lapply(script,
         function(text){
           tar <- stringr::str_extract(text, "(?<=^#' @[a-z]{0,2}name ).{1,}(?=$)")
           tar <- unique(tar[!is.na(tar)])
         })

target

ex_path <- "/home/echo/outline/mc.test/example/"
lapply(unique(unlist(target)),
       function(rd){
         path <- paste0(ex_path, rd, ".R")
         if (!file.exists(path))
           writeLines(c("", "## codes"), path)
       })
```

84 File: report_documents.R

```
# ==========================================================================
# Description of the preset
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

reportDoc <-
  list(introduction = readLines("~/Documents/mcnebula2/introduction.Rmd"),
       setup = readLines("~/Documents/mcnebula2/setup.Rmd"),
       abstract = readLines("~/Documents/mcnebula2/abstract.Rmd"),
       filter = readLines("~/Documents/mcnebula2/filter.Rmd"),
       stardust = readLines("~/Documents/mcnebula2/stardust.Rmd"),
       nebulae = readLines("~/Documents/mcnebula2/nebulae.Rmd"),
       statistic = readLines("~/Documents/mcnebula2/statistic.Rmd"),
       tracer = readLines("~/Documents/mcnebula2/tracer.Rmd"),
       annotate = readLines("~/Documents/mcnebula2/annotate.Rmd")
  )

# usethis::use_data(reportDoc)
```

85 File: separate_functions.R

```
setwd("~/MCnebula2/R")
library(magrittr)

script <- list.files(".", pattern = "\\.R$") %>%
  sapply(readLines)

target <- lapply(script,
                 function(text){
                   sig <- 0
                   for (i in 1:length(text)) {
                     if (grepl("^#", text[i])) {
                       if (sig == 2) {
```

```r
                          sig <- 1
                          text[i] <- paste0("\n", text[i])
                        }
                        next
                      }
                      if (grepl("^NULL", text[i]))
                        next
                      if (grepl("^$", text[i])) {
                        sig <- 1
                        next
                      }
                      if (grepl("^[a-z|A-Z|.]", text[i])) {
                        if (grepl("^setGeneric", text[i]))
                          break
                        if (sig == 2) {
                          text[i] <- paste0("\n", text[i])
                        }
                        sig <- 2
                      }
                    }
                    return(text)
                  })

writeLines(target[[3]])

path <- "~/code_backup/mcnebula2"
dir.create(path, recursive = T)

lapply(names(target),
       function(name){
         writeLines(target[[name]], paste0(path, "/", name))
       })
```

86 File: serum_workflow.R

```r
# ==========================================================================
# workflow to process data and output report (Serum)
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## Initialize
s0 <- new_heading("Initialize analysis", 2)
```

```r
s0.02 <- new_section2(
  c("Set SIRIUS project path and its version to initialize mcnebula object."),
  rblock({
    mcn <- mcnebula()
    mcn <- initialize_mcnebula(mcn, "sirius.v4", ".")
    ion_mode(mcn) <- "pos"
  }, eval = F)
)

s0.1 <- new_section2(
  c("Create a temporary folder to store the output data."),
  rblock({
    tmp <- paste0(tempdir(), "/temp_data")
    dir.create(tmp, F)
    export_path(mcn) <- tmp
  })
)

s0.2 <- new_section2(
  c("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 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>"),
  rblock({
    exfiles <- system.file("extdata", package = "exMCnebula2")
  }))

s0.3 <- new_section2(
  c("Load the '.rdata' file."),
  rblock({
    load(paste0(exfiles, "/mcn_serum6501.rdata"))
    mcn <- mcn_serum6501
    rm(mcn_serum6501)
  })
```

```
)

## Filter candidates
s1 <- new_heading("Filter candidates", 2)

s1.1 <- new_section2(
  reportDoc$filter,
  rblock({
    mcn <- filter_structure(mcn)
    mcn <- create_reference(mcn)
    mcn <- filter_formula(mcn, by_reference = T)
  })
)

## Filter chemcial classes
s2 <- new_heading("Filter chemical classes", 2)

s2.1 <- new_section2(
  reportDoc$stardust,
  rblock({
    mcn <- create_stardust_classes(mcn)
    mcn <- create_features_annotation(mcn)
    mcn <- cross_filter_stardust(mcn, max_ratio = .05, cutoff = .4, identical_factor = .6)
    classes <- unique(stardust_classes(mcn)$class.name)
    table.filtered.classes <- backtrack_stardust(mcn)
  })
)

s2.5 <- new_section2(
  c("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."),
  rblock({
    classes
    pattern <- c("stero", "fatty acid", "pyr", "hydroxy", "^orga")
    dis <- unlist(lapply(pattern, grep, x = classes, ignore.case = T))
    dis <- classes[dis]
    dis
    dis <- dis[-1]
  }, args = list(eval = T))
```

```r
)

## Create Nebulae
s3 <- new_heading("Create Nebulae", 2)

s3.1 <- new_section2(
  c("Create Nebula-Index data. This data created based on 'stardust_classes' data."),
  rblock({
    mcn <- backtrack_stardust(mcn, dis, remove = T)
    mcn <- create_nebula_index(mcn)
  })
)

s3.2 <- new_section2(
  reportDoc$nebulae,
  rblock({
    mcn <- compute_spectral_similarity(mcn)
    mcn <- create_parent_nebula(mcn)
    mcn <- create_child_nebulae(mcn)
  })
)

## Visualize Nebulae
s4 <- new_heading("Visualize Nebulae", 2)

s4.1 <- new_section2(
  c("Create layouts for Parent-Nebula or Child-Nebulae visualizations."),
  rblock({
    mcn <- create_parent_layout(mcn)
    mcn <- create_child_layouts(mcn)
    mcn <- activate_nebulae(mcn)
  })
)

s4.5 <- new_section2(
  c("The available chemical classes for visualization and its",
    "sequence in storage."),
  rblock({
    table.nebulae <- visualize(mcn)
    table.nebulae
  }, args = list(eval = T))
```

```
)

s4.6 <- new_section2(
  c("Draw and save as .png or .pdf image files."),
  rblock({
    ## Parent-Nebula
    p <- visualize(mcn, "parent")
    ggsave(f4.61 <- paste0(tmp, "/parent_nebula.png"), p)
    ## Child-Nebulae
    pdf(f4.62 <- paste0(tmp, "/child_nebula.pdf"), 12, 14)
    visualize_all(mcn)
    dev.off()
  })
)

s4.6.fig1 <- include_figure(f4.61, "parent", "Parent-Nebula")
s4.6.fig2 <- include_figure(f4.62, "child", "Child-Nebulae")

ref <- function(x) {paste0("(Fig. ", get_ref(x), ")")}

s4.7 <- c("In general, Parent-Nebulae", ref(s4.6.fig1),
          "is too informative to show, so Child-Nebulae", ref(s4.6.fig2),
          "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.")

## Statistic analysis
s5 <- new_heading("Statistic analysis", 2)

s5.1 <- new_section2(
  c("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."),
  rblock({
    utils::untar(paste0(exfiles, "/serum.tar.gz"), exdir = tmp)
    origin <- data.table::fread(paste0(tmp, "/serum_origin_mmc3.tsv"), skip = 1)
    origin <- tibble::as_tibble(origin)
  })
```

```
)

s5.2 <- new_section2(
  c("Its original data can obtained from:",
    paste0("<https://www.cell.com/cms/10.1016/j.cell.2020.07.040/",
      "attachment/f13178d1-d1ee-4179-9d33-227a02e604f1/",
      "mmc3.xlsx>."),
    "Now, let's check the columns in the table."),
  rblock({
    origin
  }, args = list(eval = T))
)

s5.3 <- new_section2(
  c("Remove the rest of the columns and keep only the columns for ID,",
    "m/z, retention time, and quantification."),
  rblock({
    keep <- grep("^[A-Z]{2}[0-9]{1,3}$", colnames(origin))
    quant <- dplyr::select(origin, oid = 1, mz = 2, rt = 3, dplyr::all_of(keep))
  })
)

s5.4 <- new_section2(
  c("The IDs in the data `quant` are",
    "different from the IDs in the object `mcn`, so we need to align them",
    "first, according to mz and rt (they originate from the same batch of samples).",
    "In the following, we have allowed the two sets of data to be merged as",
    "accurately as possible in the form of evaluation of score:", "",
    "- Score = (1 - rt.difference / rt.tolerance) * rt.weight +",
    "(1 - mz.defference / mz.tolerance) * mz.weight"),
  rblock({
    meta_col <- dplyr::select(features_annotation(mcn), .features_id, mz, rt.secound)
    meta_col$rt.min <- meta_col$rt.secound / 60
    merged <- align_merge(meta_col, quant, ".features_id", rt.main = "rt.min", rt.sub = "rt")
    merged <- dplyr::select(merged, -mz.main, -mz.sub, -rt.min, -rt, -rt.secound)
  })
)

s5.5 <- new_section2(
  c("Due to the differences in feature detection algorithms,",
    "some of the features inevitably do not get matched."),
  rblock({
    merged
  }, args = list(eval = T))
)

s5.6 <- new_section2(
  c("Create the metadata table and store it in the `mcn` object",
    "along with the quantification data."),
  rblock({
    gp <- c(NN = "^NN", HN = "^HN", HS = "^HS", HM = "^HM")
    metadata <- MCnebula2:::group_strings(colnames(merged), gp, "sample")
    metadata$group_name <-
      vapply(metadata$group, switch, FUN.VALUE = character(1),
        NN = "non-hospital & non-infected",
        HN = "hospital & non-infected",
        HS = "hospital & survival",
        HM = "hospital & mortality")
    metadata$supergroup <-
      vapply(metadata$group, switch, FUN.VALUE = character(1),
        NN = "control groups",
        HN = "control groups",
        HS = "infection groups",
        HM = "infection groups")
    features_quantification(mcn) <- dplyr::select(merged, -oid)
    sample_metadata(mcn) <- metadata
  })
)

s5.7 <- new_section2(
  c(reportDoc$statistic, "", "In the following we use the",
    "`binary_comparison` function for variance analysis. Note that",
    "the quantification data in `origin` has been normalized.",
    "To accommodate the downstream analysis of gene",
    "expression that the `limma` package was originally used for, we",
    "should log2-transform and centralize this data."),
  rblock({
    mcn <- binary_comparison(mcn, (HS + HM) - (NN + HN), HM - HS,
      fun_norm = function(x) scale(log2(x), scale = F))
    top.list <- top_table(statistic_set(mcn))
  })
)
```

```r
s5.8 <- new_section2(
  c("To verify the validity of the above variance analysis, the data",
    "columns were merged to obtain the IDs from the original analysis."),
  rblock({
    top.list <- lapply(top.list, merge,
      y = dplyr::select(merged, .features_id, oid),
      by = ".features_id", all.x = T, sort = F)
    top.list <- lapply(top.list, tibble::as_tibble)
  })
)

s5.81 <- new_section2(
  c("Verify with the `EFS_Rank` and `MW_Rank` column in the `origin` data.",
    "(The original authors used the two methods to rank the features.)"),
  rblock({
    origin_top50 <- dplyr::filter(origin, EFS_Rank <= 50 | MW_Rank <= 50)
    inter. <- lapply(top.list,
      function(df) {
        match <- head(df, n = 50)$oid %in% origin_top50$Unique_ID
        oid <- head(df, n = 50)$oid[match]
        list(table.match = table(match), oid = oid)
      })
    lapply(inter., function(x) x$table.match)
  }, args = list(eval = T))
)

s5.82 <- new_section2(
  c("Let's see which compounds were identified that intersected our",
    "ranking and the original ranking of features."),
  rblock({
    inter.compound <- dplyr::filter(origin, Unique_ID %in% inter.[[2]]$oid)
    table(inter.compound$Spectral_Library_Match, useNA = "if")
  }, args = list(eval = T))
)

s5.9 <- c("Interestingly, these two compounds were critical compounds",
            "in the original study.")

## Set tracer in Child-Nebulae
s6 <- new_heading("Set tracer in Child-Nebulae", 2)
```

```r
s6.1 <- new_section2(
  reportDoc$tracer,
  rblock({
    n <- 50
    tops <- unique(unlist(lapply(top.list, function(df) df$.features_id[1:n])))
    palette_set(melody(mcn)) <- colorRampPalette(palette_set(mcn))(length(tops))
    mcn2 <- set_tracer(mcn, tops)
    mcn2 <- create_child_nebulae(mcn2)
    mcn2 <- create_child_layouts(mcn2)
    mcn2 <- activate_nebulae(mcn2)
    mcn2 <- set_nodes_color(mcn2, use_tracer = T)
  })
)

s6.2 <- new_section2(
  c("Draw and save the image."),
  rblock({
    pdf(f6.2 <- paste0(tmp, "/tracer_child_nebula.pdf"), 12, 14)
    visualize_all(mcn2)
    dev.off()
  })
)

s6.2.fig1 <- include_figure(f6.2, "tracer", "Tracing top features in Child-Nebulae")

s6.3 <- c("A part of the top features are marked with colored nodes in",
          "Child-Nebulae", ref(s6.2.fig1), ".",
          "These features are at least identified with chemical molecular",
          "formula. Those that are not identified, or the Nebula-Index data do",
          "not contain the chemical class to which these features belong, are",
          "absent from the Figure.")

## Quantification in Child-Nebulae
s7 <- new_heading("Quantification in Child-Nebulae", 2)

s7.1 <- new_section2(
  c("Show Fold Change (HM versus HS) in Child-Nebulae."),
  rblock({
    palette_gradient(melody(mcn2)) <- c("blue", "grey90", "red")
    mcn2 <- set_nodes_color(mcn2, "logFC", top.list[[2]])
    pdf(f7.1 <- paste0(tmp, "/logFC_child_nebula.pdf"), 12, 14)
    visualize_all(mcn2, fun_modify = modify_stat_child)
    dev.off()
  })
)

s7.1.fig1 <- include_figure(f7.1, "logFC", "Show log2(FC) in Child-Nebulae")

s7.2 <- c("Each Child-Nebula separately shows the overall content variation of",
          "the chemical class to which it belongs", ref(s7.1.fig1), ".")

## Annotate Nebulae
s8 <- new_heading("Annotate Nebulae", 2)

s8.0 <- new_section2(
  c("Now, the available Nebulae contains:"),
  rblock({
    table.nebulae2 <- visualize(mcn2)
    table.nebulae2
  }, args = list(eval = T))
)

s8.1 <- new_section2(
  c("Next, let us focus on Acyl carnitines, a class that was highlighted",
    "in the original research and also appears in Child-Nebulae, marked by",
    "plural top features (Likewise, we annotated two other chemical",
    "classes of Nebulae)."),
  rblock({
    mcn2 <- set_nodes_color(mcn2, use_tracer = T)
    palette_stat(melody(mcn2)) <- c(
      NN = "#B6DFB6", HN = "#ACDFEE", HS = "#EBA9A7", HM = "grey70"
    )
    ac <- "Acyl carnitines"
    lpc <- "Lysophosphatidylcholines"
    ba <- "Bile acids, alcohols and derivatives"
    mcn2 <- annotate_nebula(mcn2, ac)
    mcn2 <- annotate_nebula(mcn2, lpc)
    mcn2 <- annotate_nebula(mcn2, ba)
  })
)

s8.2 <- new_section2(
  c("Draw and save the image."),
  rblock({
    p <- visualize(mcn2, ac, annotate = T)
    ggsave(f8.2 <- paste0(tmp, "/ac_child.pdf"), p, height = 5)
    p <- visualize(mcn2, lpc, annotate = T)
    ggsave(f8.2.2 <- paste0(tmp, "/lpc_child.pdf"), p, height = 5)
    p <- visualize(mcn2, ba, annotate = T)
    ggsave(f8.2.3 <- paste0(tmp, "/ba_child.pdf"), p, height = 5)
  })
)

s8.2.fig1 <- include_figure(f8.2, "ac", paste0("Annotated Nebulae: ", ac))

s8.3 <- c("See results", ref(s8.2.fig1), ".", reportDoc$annotate)

s8.4 <- new_section2(
  c("Use the `show_node` function to get the annotation details",
    "for a feature. For example:"),
  rblock({
    ef <- "2068"
    pdf(f8.4 <- paste0(tmp, "/features_", ef, ".pdf"), 10, 4)
    show_node(mcn2, ef)
    dev.off()
  })
)

s8.4.fig1 <- include_figure(f8.4, "ef", "The annotated feature of ID: 2068")

s8.5 <- c("See results", ref(s8.4.fig1), ".")

notShow1 <- new_section2(
  c("Combine the images."),
  rblock({
    p.ac <- into(grecta("a"), as_grob(visualize(mcn2, ac, annotate = T)))
    p.lpc <- into(grecta("a"), as_grob(visualize(mcn2, lpc, annotate = T)))
    p.ba <- into(grecta("b"), as_grob(visualize(mcn2, ba, annotate = T)))
    p.node <- into(grecta("b"), grid::grid.grabExpr(show_node(mcn2, ef)))
    dev.off()
    grob.ac_node <- frame_row(c(p.ac = 1, p.node = .5), namel(p.ac, p.node))
    grob.ac_node <- ggather(grob.ac_node, vp = viewport(, , .95, .95))
    grob.lpc_ba <- frame_row(c(p.lpc = 1, p.ba = 1), namel(p.lpc, p.ba))
```

```r
    grob.lpc_ba <- ggather(grob.lpc_ba, vp = viewport(, , .95, .95))
    pdf(paste0(tmp, "/fig.ac_node.pdf"), 9, 10)
    draw(grob.ac_node)
    dev.off()
    pdf(paste0(tmp, "/fig.lpc_ba.pdf"), 9, 13)
    draw(grob.lpc_ba)
    dev.off()
  })
)

notShow1.5 <- new_section2(
  c("Combine the image..."),
  rblock({
    ## b plot: ac
    mcn2 <- set_nodes_color(mcn2, "logFC", top.list[[2]])
    p.logfc.ac <- visualize(mcn2, ac, modify_stat_child)
    p.logfc.ac <- into(grecta("b"), as_grob(p.logfc.ac))
    ## a plot: ac
    mcn2 <- set_nodes_color(mcn2, use_tracer = T)
    p.tracer.ac <- visualize(mcn2, ac, modify_set_labs_and_unify_scale_limits)
    ## legend for a and b
    child_legend <- MCnebula2:::.get_legend(p.tracer.ac + guides(fill = "none"))
    ## a plot: ac revise
    p.tracer.ac <- visualize(mcn2, ac, modify_default_child)
    p.tracer.ac <- into(grecta("a"), as_grob(p.tracer.ac))
    ## c plot: annotated ac
    p.ac <- into(grecta("c"), as_grob(visualize(mcn2, ac, annotate = T)))
    ## d plot: node
    p.node <- into(grecta("d"), grid::grid.grabExpr(show_node(mcn2, ef)))
    ## gather theme
    vis1 <- frame_col(
      c(p.tracer.ac = 1, p.logfc.ac = 1, child_legend = .5),
      namel(p.tracer.ac, p.logfc.ac, child_legend)
    )
    vis2 <- frame_row(
      c(vis1 = 1, p.ac = 1.5, p.node = 1),
      namel(vis1, p.ac, p.node)
    )
    vis2 <- ggather(vis2, vp = viewport(, , .95, .95))
    pdf(paste0(tmp, "/fig.ac_node2.pdf"), 10, 14)
    draw(vis2)
    dev.off()
  })
)

## Output identification table
s10 <- new_heading("Query compounds", 2)

s10.1 <- c("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.")

s10.2 <- new_section2(
  c("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."),
  rblock({
    feas <- features_annotation(mcn2)
    feas <- merge(feas, top.list[[2]], by = ".features_id", all.x = T)
    feas <- dplyr::mutate(feas, arrange.rank = adj.P.Val)
    feas <- format_table(feas, export_name = NULL)
    key2d <- feas$inchikey2d
  })
)

s10.3 <- new_section2(
  c("Create a folder to store the acquired data."),
  rblock({
    tmp2 <- paste0(tmp, "/query")
    dir.create(tmp2, F)
  })
)
```

```
s10.4 <- new_section2(
  c("Query the compound's InChIKey, chemical class, IUPUA name.",
    "If your system is not Linux, the multithreading below may pose some problems,",
    "please remove the parameters `curl_cl = 4` and `classyfire_cl = 4`."),
  rblock({
    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)
  })
)

s10.5 <- new_section2(
  c("We will also query for synonyms of compounds, but this is done in",
    "'CID' (PubChem's ID), so some transformation is required."),
  rblock({
    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)
  })
)

s10.6 <- new_section2(
  c("Screen for unique synonyms and chemical classes for all compounds."),
  rblock({
    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"))
  })
)

s10.7 <- new_section2(
  c("The formatted table as following:"),
  rblock({
    feas.table
  }, args = list(eval = T))
)
```

```
s10.8 <- new_section2(
  c( "Filtering our results based on the ranking of 'features' (top 25 of EFS",
     "and MWU) and identification by Wozniak et al."),
  rblock({
    origin_top50 <- dplyr::select(
      origin_top50, oid = 1, EFS_Rank, MW_Rank,
      Spectral_Library_Match
    )
    feas.otop <- dplyr::select(merged, .features_id, oid)
    feas.otop <- merge(origin_top50, feas.otop, all.x = T, by = "oid")
    feas.otop <- merge(
      feas.otop, features_annotation(mcn2),
      all.x = T, by = ".features_id"
    )
    feas.otop.format <- format_table(
      feas.otop, filter = NULL,
      select = c("oid", "EFS_Rank", "MW_Rank", "Spectral_Library_Match", .select_format),
      export_name = c(oid = "# Original ID", EFS_Rank = "# EFS_Rank",
        MW_Rank = "# MW_Rank",
        Spectral_Library_Match = "# Spectral_Library_Match",
        .export_name)
    )
    write_tsv(feas.otop.format, paste0(tmp, "/oTop50_compounds_format.tsv"))
  })
)

## Pathway enrichment
s11 <- new_heading("Pathway enrichment", 2)

s11.1 <- new_section2(
  c("A plural number of chemical classes of interest were selected and",
    "the IDs of their features were obtained. These",
    "features were filtered with the statistic analysis data",
    "(Q value < 0.05)."),
  rblock({
    focus <- c("Acyl carnitines",
      "Lysophosphatidylcholines",
      "Bile acids, alcohols and derivatives")
    focus <- select_features(mcn, focus, q.value = .05, logfc = .3, coef = 1:2)
  })
)
```

```r
s11.2 <- new_section2(
  c("Get the InChIkey 2D of these features, then get the possible",
    "InChIkey, then get the CID, and finally get the KEGG ID by the",
    "CID."),
  rblock({
    focus.key2d <- maps(feas, focus, ".features_id", "inchikey2d")
    focus.cid <- lapply(focus.key2d,
      function(key2d) {
        set <- key.set[ names(key.set) %in% key2d ]
        unlist(lapply(set, function(df) df$CID),
          use.names = F)
      })
    keggids <- cid.to.kegg(unlist(focus.cid, use.names = F))
    focus.kegg <- maps(keggids, lapply(focus.cid, as.character), "Query", "KEGG")
  })
)

s11.3 <- new_section2(
  c("Let's see what we get."),
  rblock({
    focus.kegg
  }, args = list(eval = T))
)

s11.4 <- new_section2(
  c("Using package 'FELLA' for pathway enrichment analysis.",
    "The following step create a 'database' for enrichment",
    "(It is quite time consuming and can take up to 30 minutes).",
    "Subsequently, load the data."),
  rblock({
    db.dir <- init_fella(tmp, "hsa")
    db.data <- load_fella(db.dir)
  })
)

s11.5 <- new_section2(
  c("Perform enrichment."),
  rblock({
    focus.enrich <- enrich_fella(focus.kegg, db.data)
    focus.graph <- graph_fella(focus.enrich, db.data, "pagerank")
    names(focus.graph) <- names(focus.kegg)
  })
)

s11.6 <- new_section2(
  c("Some compounds are not present in the KEGG graph and are only background",
    "compounds. Let's check the enrichment results."),
  rblock({
    !vapply(focus.graph, is.null, logical(1))
  }, args = list(eval = T))
)

s11.7 <- new_section2(
  c("Visualization of enrichment results."),
  rblock({
    p2 <- plotGraph_fella(focus.graph[[2]])
    ggsave(f11.71 <- paste0(tmp, "/ba_enrich.pdf"), p2)
    p3 <- plotGraph_fella(focus.graph[[3]])
    ggsave(f11.72 <- paste0(tmp, "/lpc_enrich.pdf"), p3)
  })
)

s11.7.fig1 <- include_figure(f11.71, "ba", "Enrichment of Pagerank of BA compounds")
s11.7.fig2 <- include_figure(f11.72, "lpc", "Enrichment of Pagerank of LPC compounds")

## Heatmap analysis
s12 <- new_heading("Heatmap analysis", 2)

s12.1 <- new_section2(
  c("Since the quantitative data obtained by merging contain many missing values,",
    "they need to be processed before plotting the heat map:",
    "For each subset of data, the missing values will be filled with the average",
    "value; if the set is all missing values, they will be filled with zero."),
  rblock({
    hp.data <- handling_na(
      features_quantification(mcn2),
      metadata = sample_metadata(mcn2)
    )
  })
)

s12.2 <- new_section2(
  c("Convert wide data to long data; log transform the values; if there is a",
    "value 0, replace it with 1/10 of the minimum value of the value column."),
  rblock({
    hp.data <- log_trans(hp.data)
  })
)

s12.3 <- new_section2(
  c("Draw heat maps for three chemical classes."),
  rblock({
    names(focus) <- c("ACs", "BAs", "LPCs")
    hp.lst <- plot_heatmap(focus, hp.data, metadata, pal_group = palette_stat(mcn2))
    ggsave(f12.31 <- paste0(tmp, "/ac_heatmap.pdf"), hp.lst[[1]], width = 13, height = 4)
    ggsave(f12.32 <- paste0(tmp, "/ba_heatmap.pdf"), hp.lst[[2]], width = 13, height = 7)
    ggsave(f12.33 <- paste0(tmp, "/lpc_heatmap.pdf"), hp.lst[[3]], width = 13, height = 4)
  })
)

# test <- plot_heatmap(focus, hp.data, metadata, pal_group = palette_stat(mcn2), clust_col = F)
# test <- lapply(test, function(x) grid.grabExpr(print(x)))
# names(test)
# # [1] "ACs"  "BAs"  "LPCs"
# test.p <- frame_row(c(ACs = 1, BAs = 1.5, LPCs = 1), test)
# pdf(paste0(tmp, "/test_heatmap.pdf"), width = 13, height = 18)
# draw(test.p)
# dev.off()

s12.4.fig1 <- include_figure(f12.31, "acHp", "Heatmap of ACs")
s12.4.fig2 <- include_figure(f12.32, "baHp", "Heatmap of BAs")
s12.4.fig3 <- include_figure(f12.33, "lpcHp", "Heatmap of LPCs")

s12.5 <- c("As shown in figure,",
  paste0("Fig. ", get_ref(s12.4.fig1), ", ", get_ref(s12.4.fig2), ", and ",
    get_ref(s12.4.fig3)),
  "ACs and BAs implied a high correlation with disease,",
  "while LPCs showed a relatively weak correlation."
)

notShow2 <- new_section2(
  c("Combined heatmap."),
  rblock({
    grobs.hp <- lapply(hp.lst, function(p) grid::grid.grabExpr(print(p)))
    grobs.hp <- lapply(1:3,
      function(n, x = c("a", "b", "c")) {
        into(grecta(x[n]), grobs.hp[[ n ]])
      })
    names(grobs.hp) <- names(hp.lst)
    grobs.hp <- frame_row(c(ACs = 1, BAs = 1.5, LPCs = 1), grobs.hp)
    grobs.hp <- ggather(grobs.hp, vp = viewport(, , .95, .95))
    pdf(paste0(tmp, "/hps.pdf"), 13, 15)
    draw(grobs.hp)
    dev.off()
  })
)

s15 <- new_heading("Verify Identification", 1)

s15.1 <- new_section2(
  c("In research of Wozniak et al, a subset of ACs compounds were identified. In",
    "addition, four top rank metabolites were identified. However, Most of the",
    "top 25 metabolites (EFS rank) were not identified."),
  rblock({
    ac_names <- c("Palmitoyl-carnitine", "Octanoyl-carnitine",
      "Acetyl-carnitine", "Hexanoyl-carnitine",
      "Decanoyl-carnitine")
    ac_inchikey2d <- c("XOMRRQXKHMYMOC", "CXTATJFJDMJMIY",
      "RDHQFKQIGNGIED", "VVPRQWTYSNDTEA",
      "LZOSYCMHQXPBFU")
    names(ac_inchikey2d) <- ac_names
    other_4m <- c(
      "Hepc", "D-erythro-Sphingosine-1-phosphate", "L-THYROXINE",
      "Decanoyl-L-carnitine"
    )
    other_4m_inchikey2d <- c(
      "BDPQVGIMLZYZQA", "DUYSYHSSBDVJSM", "XUIIKFGFIJCVMT",
      "LZOSYCMHQXPBFU"
    )
    other_4m.seq <- unlist(lapply(other_4m, grep,
        x = origin$Spectral_Library_Match, ignore.case = T))
    other_4m <- origin[other_4m.seq, ]
    origin.top25 <- c(349, 746, 854, 228, 320, 971, 2532, 670, 92, 1363,
      13, 798, 1379, 1947, 4146, 736, 1656, 464, 731, 289,
      4431, 3865, 476, other_4m$Unique_ID)
    origin.top25.featureID <-
      dplyr::filter(merged, oid %in% origin.top25)$.features_id
  })
)

s15.2 <- new_section2(
  c("Confirm which compounds were identified:"),
  rblock({
    ac_inchikey2d %in% features_annotation(mcn2)$inchikey2d
    other_4m_inchikey2d %in% features_annotation(mcn2)$inchikey2d
    reIdentify.origin.top25 <- dplyr::filter(features_annotation(mcn2),
      .features_id %in% origin.top25.featureID)
    reIdentify.origin.top25 <- dplyr::select(
      reIdentify.origin.top25, .features_id, tani.score, inchikey2d
    )
    print(reIdentify.origin.top25, n = Inf)
  }, args = list(eval = T))
)

## Session infomation
s100 <- new_heading("Session infomation", 1)

s100.1 <- rblock({
  sessionInfo()
}, args = list(eval = T))
```

```
# ==========================================================================
# output report
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

sections <- gather_sections()
prologue <- list(
  new_section("Abstract", 1, reportDoc$abstract, NULL),
  new_section("Introduction", 1, reportDoc$introduction, NULL),
  new_section("Set-up", 1, reportDoc$setup,
    rblock({
      library(MCnebula2)
      library(exMCnebula2)
    }, F)
  )
)
```

```r
report <- do.call(new_report, c(prologue, sections))
yaml(report)[1] <- c("title: Analysis on serum dataset")

## post-modify, add heading
h1 <- new_heading("Integrate data and Create Nebulae", 1)
h2 <- new_heading("Nebulae for Downstream analysis", 1)
seqs <- search_heading(report, "^Initialize|^Statistic")
report <- insert_layers(report, seqs[1], h1)
report <- insert_layers(report, seqs[2], h2)

render_report(report, file.report <- paste0(tmp, "/report.rmd"))
rmarkdown::render(file.report)
```

```r
# ==========================================================================
# convert as html with biocStyle
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# library(MCnebula2)

# write_biocStyle(report, file2 <- paste0(tmp, "/report_biocStyle_nofloat.Rmd"),
#   title <- paste0(yaml(report)[1], "\nauther: 'LiChuang Huang'")
# )

# rmarkdown::render(file2)
```

```r
# ==========================================================================
# convert to docx version
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# write_thesisDocx(report, file3 <- paste0(tmp, "/report_thesisStyle.Rmd"),
#   title <- yaml(report)[1]
# )

# rmarkdown::render(file3)
# file.copy(gsub("\\.Rmd", ".docx", file3),
#   "/mnt/data/wizard/Documents/article/MCnebula2/others/report_serum.docx", T)
```

```r
# ==========================================================================
# extra picture
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# pgg <- modify_set_labs(mcn@parent_nebula@ggset, mcn)
# pgg <- mutate_layer(pgg, 3, stroke = 0, color = "transparent")
# # ggsave("herbal_parent.pdf", call_command(pgg), width = 7, height = 7)

# mcn@child_nebulae@ggset %<>%
#   lapply(function(ggset) {
#     mutate_layer(ggset, 3, stroke = 0, color = "transparent")
# })

# pc <- nebulae_as_grob(mcn)
# frame <- frame_col(c(p = 1.2, pc = 1), namel(p = as_grob(call_command(pgg)), pc))
# # dev.new(width = 23, height = 14)

# pdf("parent_and_child.pdf", 23 * .8, 14 * .8)
# draw(frame)
# dev.off()
```

87 File: site_mcnebula2.R

```
# install.packages("blogdown")
setwd("~/MCnebula2/")

# web_config <- readLines(file <- "config/production/config.toml")
# web_config[2] <- "baseURL = 'https://mcnebula.netlify.app'"
# writeLines(web_config, file)

devtools::load_all("~/utils.tool")
set_hugoDir("~/MCnebula2")
options(blogdown.method = "markdown")
blogdown::serve_site()
# blogdown::stop_server()

description <- paste0(strwrap("Critical chemical classes to classify and boost
    identification by visualization for untargeted LC-MS/MS data analysis"),
    collapse = " ")
addition <- "R package for analysis of non-targeted LC-MS/MS"
```

```
# ==========================================================================
# basic setting (text)
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

set_home(c(title = "MCnebula2",
           titleSeparator = " | ",
           titleAddition = addition,
           description = description))

set_index(c(title = paste0("MCnebula2: ", addition),
            lead = description,
            date = record_time(),
            lastmod = record_time()),
          "en/_index.Rmd"
)
```

```
# ==========================================================================
# create scene
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

new_scene(c(prologue = "introduction",
    prologue = "super_quick_start",
    prologue = "quick_start",
    prologue = "usage"),
  rep(110, 4), c(100, 40, 50, 110))

new_scene(c(installation = "linux",
            installation = "windows",
            installation = "macOS"),
          rep(100, 3), c(10, 20, 30))

new_scene(c(recommendation = "mzmine2",
            recommendation = "sirius_4",
            recommendation = "proteowizard"),
          rep(200, 3), c(220, 230, 210))

new_scene(c(workflow = "basic_workflow"), 150)
new_scene(c(workflow = "metabolic_workflow"), 152)
new_scene(c(workflow = "chemical_workflow"), 153)

new_scene(c(herbal_eucommia = "index",
            metabolites_serum = "index"),
          rep(500, 3), c(100, 200), tar = "news", index_Draft = F)

new_scene(c(help = "R_tips"), 5000)
```

```
# ==========================================================================
# contact and about
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

new_scene(c(contact = "index"), 1000, 1100, tar = "en", index_Draft = F)
new_scene(c(about = "index"), 2000, 2100, tar = "en", index_Draft = F)
```

88 File: subjective_evaluation.R

```
# ==========================================================================
# create the table for subjective_evaluation
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

menu <- list(
  identificaiton = c("MS1", "library match", "machine prediction"),
  classifying = c("structure based", "MS/MS based", "select classes"),
  visualize_dataset = c("spectral based", "classes based", "indepth annotation"),
  others = c("preprocessing", "statistics", "path_enrichment", "report"),
  usage = c("software", "availability", "difficulty")
)

methods <- c("MCnebula", "SIRIUS", "GNPS", "MZmine", "XCMS", "MetaboAnalyst", "MS-DIAL")

data <- lapply(menu, function(x) data.frame(item = x))
data <- data.table::rbindlist(data, idcol = T)
data <- dplyr::rename(data, group = .id)

data <- sapply(methods, function(name) data, simplify = F)
data <- data.table::rbindlist(data, idcol = T)
data <- dplyr::rename(data, method = .id)
data <- dplyr::mutate(
  data, eval = integer(1),
  item = factor(item, levels = unique(item))
)
data <- tidyr::spread(data, method, eval)
data <- dplyr::arrange(data, item)
data <- dplyr::select(data, group, item, dplyr::all_of(methods))

filename <- paste0("subEval_", Sys.Date(), ".csv")
data.table::fwrite(data, filename)
```

```
# ==========================================================================
# format the table
```

```r
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

filename <- "subEval_2023-02-21.csv"
data <- data.table::fread(filename)
data <- dplyr::filter(data, item != "software")
data <- dplyr::mutate_if(
  data, is.integer,
  function(x) {
    ifelse(x == 0, "-",
      vapply(x, function(x) paste0(rep("*", abs(x)), collapse = ""),
        character(1)))
  }
)
data <- dplyr::mutate(data, group = form(group), item = form(item))
colnames(data) <- form(colnames(data))

table <- pretty_table(
  dplyr::group_by(data, Group), title = "Functional evaluation",
  subtitle = "Evaluation for software of analysis of LC-MS/MS"
)
data.table::fwrite(data, "subEval.csv")
```

89 File: Abstract.R

```r
s0.1 <- new_section("Abstract", 1, reportDoc$abstract, NULL)
```

90 File: Annotate Nebulae.R

91 File: Create Nebulae.R

```r
s4 <- new_heading("Create Nebulae", 2)

s4.1 <- new_section2(
  c("Create Nebula-Index data. This data created based on 'stardust_classes' data."),
  rblock({
    mcn <- create_nebula_index(mcn)
  })
)

s4.5 <- new_section2(
  reportDoc$nebulae,
  rblock({
    mcn <- compute_spectral_similarity(mcn)
    mcn <- create_parent_nebula(mcn)
    mcn <- create_child_nebulae(mcn)
  })
)
```

92 File: Filter candidates.R

```
s2 <- new_heading("Filter candidates", 2)

s2.1 <- new_section2(
  reportDoc$filter,
  rblock({
    mcn <- filter_structure(mcn)
    mcn <- create_reference(mcn)
    mcn <- filter_formula(mcn, by_reference = T)
  })
)
```

93 File: Filter chemical classes.R

```
s3 <- new_heading("Filter chemical classes", 2)

s3.1 <- new_section2(
  reportDoc$stardust,
  rblock({
    mcn <- create_stardust_classes(mcn)
    mcn <- create_features_annotation(mcn)
    mcn <- cross_filter_stardust(mcn)
    classes <- unique(stardust_classes(mcn)$class.name)
    table.filtered.classes <- backtrack_stardust(mcn)
  })
)

s3.5 <- new_section2(
  c("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",
```

```r
    "as their sub-structure."),
  rblock({
    classes
    pattern <- c("stero", "fatty acid", "pyr", "hydroxy", "^orga")
    dis <- unlist(lapply(pattern, grep, x = classes, ignore.case = T))
    dis <- classes[dis]
    dis
    dis <- dis[-1]
  }, args = list(eval = T))
)

s3.6 <- new_section2(
  c("Remove these classes."),
  rblock({
    mcn <- backtrack_stardust(mcn, dis, remove = T)
  })
)
```

94 File: Initialize analysis.R

```r
s1 <- new_heading("Initialize analysis", 2)

s1.1 <- new_section2(
  c("Set SIRIUS project path and its version to initialize mcnebula object."),
  rblock({
    mcn <- mcnebula()
    mcn <- initialize_mcnebula(mcn, "<<<sirius_version>>>", "<<<sirius_project>>>")
    ion_mode(mcn) <- "<<<ion_mode>>>"
  })
)

s1.5 <- new_section2(
  c("Create a temporary folder to store the output data."),
  rblock({
    tmp <- paste0(tempdir(), "/temp_data")
    dir.create(tmp, F)
    export_path(mcn) <- tmp
  })
)
```

95 File: Integrate data and Create Nebulae.R

```r
s0.9 <- new_heading("Integrate data and Create Nebulae", 1)
```

96 File: Introduction.R

```r
s0.2 <- new_section("Introduction", 1, reportDoc$introduction, NULL)
```

97 File: Pathway enrichment.R

98 File: Quantification in Child-Nebulae.R

99 File: Query compounds.R

100 File: Session infomation.R

```r
s100 <- new_heading("Session infomation", 1)

s100.1 <- rblock({
  sessionInfo()
}, args = list(eval = T))
```

101 File: Set tracer in Child-Nebulae.R

102 File: Set-up.R

```r
s0.3 <- new_section("Set-up", 1, reportDoc$setup,
  rblock({
    library(MCnebula2)
  }, F)
)
```

103 File: Statistic analysis.R

104 File: Visualize Nebulae.R

```r
s5 <- new_heading("Visualize Nebulae", 2)
```

```r
s5.1 <- new_section2(
  c("Create layouts for Parent-Nebula or Child-Nebulae visualizations."),
  rblock({
    mcn <- create_parent_layout(mcn)
    mcn <- create_child_layouts(mcn)
    mcn <- activate_nebulae(mcn)
  })
)

s5.3 <- new_section2(
  c("The available chemical classes for visualization and its",
    "sequence in storage."),
  rblock({
    table.nebulae <- visualize(mcn)
    table.nebulae
  }, args = list(eval = T))
)

s5.6 <- new_section2(
  c("Draw and save as .png or .pdf image files."),
  rblock({
    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()
  })
)

s5.6.fig1 <- include_figure(f5.61, "parent", "Parent-Nebula")
s5.6.fig2 <- include_figure(f5.62, "child", "Child-Nebulae")

ref <- function(x) {
  paste0("(Fig. ", get_ref(x), ")")
}

s5.8 <- c(
  "In general, Parent-Nebulae", ref(s5.6.fig1),
  "is too informative to show, so Child-Nebulae", ref(s5.6.fig2),
  "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."
)
```

105 File: test_collate_sirius.5.R

```
df <- data.table::fread("./canopus_compound_summary.tsv")
t <- tolower(gsub("#| ", "_", colnames(df)))

lapply(1:length(t),
  function(n) {
    x <- paste0(t[n], " = \"", colnames(df)[n], "\",")
    writeLines(x)
  })

utils::unzip("./1_instance5_gnps1234/test.zip", list = T)
t <- list.files("./1_instance5_gnps1234/ff/")[c(2, 4)]
tt <- paste0("./1_instance5_gnps1234/fingerid", "/", t)




setwd("~/operation/sirius_5_test/test/")

mcn <- initialize_mcnebula(mcnebula(), "sirius.v5", ".")

mcn1 <- collate_used(mcn)
latest(filter_ppcp(mcn1))
```

106 File: TEST_compareSpectra.R

```
# ## from package ProtGenerics
# setGeneric("compareSpectra", function(x, y, ...)
#     standardGeneric("compareSpectra"))
# ## from MSnbase
# setClass("Spectrum",
#          representation = representation(
#              msLevel="integer",
#              peaksCount="integer",
#              rt="numeric",
#               acquisitionNum="integer",
#               scanIndex = "integer",
#               tic = "numeric",
#               mz = "numeric",
#               intensity = "numeric",
#               fromFile = "integer",
#               centroided = "logical",
#               smoothed = "logical",
#               polarity="integer",
#               "VIRTUAL"),
#           contains=c("Versioned"),
#           prototype = prototype(
#               rt = numeric(),
#               polarity = NA_integer_,
#               acquisitionNum = NA_integer_,
#               msLevel = NA_integer_,
#               centroided = NA,
#               smoothed = NA,
#               peaksCount = 0L,
#               tic = 0,
#               scanIndex = integer(),
#               mz = numeric(),
#               intensity = numeric()),
#           validity = function(object)
#               validSpectrum(object)
#           )

# setClass("Spectrum2",
#           representation = representation(
#               merged="numeric",
#               precScanNum="integer",
#               precursorMz="numeric",
#               precursorIntensity = "numeric",
#               precursorCharge = "integer",
#               collisionEnergy = "numeric"),
#           contains=c("Spectrum"),
#           prototype = prototype(
#               merged = 1,
#               acquisitionNum = integer(),
#               precScanNum = integer(),
#               precursorMz = numeric(),
#               precursorIntensity = numeric(),
#               msLevel = as.integer(2),
#               precursorCharge = integer(),
#               collisionEnergy = numeric()),
#           validity = function(object) {
#               msg <- validMsg(NULL, NULL)
#               msl <- object@msLevel
#               if (msl < as.integer(2))
#                   msg <- validMsg(msg,
#                                   paste0("Object of class ",
#                                          class(object),
#                                          " but msLevel is ", msl,
#                                          " (should be > 1)"))
#               if (is.null(msg)) TRUE
#               else msg
#           })


# setMethod("compareSpectra", c("Spectrum", "Spectrum"),
#            function(x, y, fun=c("common", "cor", "dotproduct"),
#                     ...) {
#                compare_Spectra(x, y, fun = fun, ...)
#            })

# #' calculate similarity between spectra (between their intensity profile)
# #' @param x spectrum1 (MSnbase::Spectrum)
# #' @param y spectrum2 (MSnbase::Spectrum)
# #' @param fun similarity function (must take two spectra and ... as arguments)
# #' @param ... further arguments passed to "fun"
# #' @return double, similarity score
# #' @noRd
# compare_Spectra <- function(x, y,
#                              fun=c("common", "cor", "dotproduct"),
#                              ...) {
#   if (is.character(fun)) {
#     fun <- match.arg(fun)
#     if (fun == "cor" || fun == "dotproduct") {
#       binnedSpectra <- bin_Spectra(x, y, ...)
#       inten <- lapply(binnedSpectra, intensity)
#       return(do.call(fun, inten))
#     } else if (fun == "common") {
```

```
#       return(numberOfCommonPeaks(x, y, ...))
#     }
#   } else if (is.function(fun)) {
#     return(fun(x, y, ...))
#   }
#   return(NA)
# }


# setMethod("mz", "Spectrum", function(object) object@mz)
# setMethod("intensity", "Spectrum", function(object) object@intensity)

# bin_Spectra <- function(object1, object2, binSize = 1L,
#                          breaks = seq(floor(min(c(mz(object1), mz(object2)))),
#                                       ceiling(max(c(mz(object1), mz(object2)))),
#                                       by = binSize)) {
#     breaks <- .fix_breaks(breaks, range(mz(object1), mz(object2)))
#     list(bin_Spectrum(object1, breaks = breaks),
#          bin_Spectrum(object2, breaks = breaks))
# }

# bin_Spectrum <- function(object, binSize = 1L,
#                           breaks = seq(floor(min(mz(object))),
#                                        ceiling(max(mz(object))),
#                                        by = binSize),
#                           fun = sum,
#                           msLevel.) {
#     ## If msLevel. not missing, perform the trimming only if the msLevel
#     ## of the spectrum matches (any of) the specified msLevels.
#     if (!missing(msLevel.)) {
#         if (!(msLevel(object) %in% msLevel.))
#             return(object)
#     }
#     bins <- .bin_values(object@intensity, object@mz, binSize = binSize,
#                          breaks = breaks, fun = fun)
#     object@mz <- bins$mids
#     object@intensity <- bins$x
#     object@tic <- sum(object@intensity)
#     object@peaksCount <- length(object@mz)
#     if (validObject(object))
#         return(object)
# }

# #' The function aggregates `x` for `toBin` falling into bins defined
# #' by `breaks` using the `fun` function.
# #'
# #' @details
# #'
# #' This is a combination of the code from the former bin_Spectrum.
# #'
# #' @param x `numeric` with the values that should be binned.
# #'
# #' @param toBin `numeric`, same length than `x`, with values to be used for the
# #'     binning.
# #'
# #' @param binSize `numeric(1)` with the size of the bins.
# #'
# #' @param breaks `numeric` defining the breaks/bins.
# #'
# #' @param fun `function` to be used to aggregate values of `x` falling into the
# #'     bins defined by `breaks`.
# #'
# #' @return `list` with elements `x` and `mids` being the aggregated values
# #'     of `x` for values in `toBin` falling within each bin and the bin mid
# #'     points.
# #'
# #' @author Johannes Rainer, Sebastian Gibb
# #'
# #' @noRd
# .bin_values <- function(x, toBin, binSize = 1, breaks = seq(floor(min(toBin)),
#                                                              ceiling(max(toBin)),
#                                                              by = binSize),
#                         fun = max) {
#     if (length(x) != length(toBin))
#         stop("lengths of 'x' and 'toBin' have to match.")
#     fun <- match.fun(fun)
#     breaks <- .fix_breaks(breaks, range(toBin))
#     nbrks <- length(breaks)
#     idx <- findInterval(toBin, breaks)
#     ## Ensure that indices are within breaks.
#     idx[which(idx < 1L)] <- 1L
#     idx[which(idx >= nbrks)] <- nbrks - 1L
#     ints <- double(nbrks - 1L)
#     ints[unique(idx)] <- unlist(lapply(base::split(x, idx), fun),
#                                  use.names = FALSE)
#     list(x = ints, mids = (breaks[-nbrks] + breaks[-1L]) / 2L)
# }



# #' calculate the dot product between two vectors
# #'
# #' Stein, S. E., and Scott, D. R. (1994).
# #' Optimization and testing of mass spectral library search algorithms for
# #' compound identification.
# #' Journal of the American Society for Mass Spectrometry, 5(9), 859-866.
# #' doi: https://doi.org/10.1016/1044-0305(94)87009-8
# #'
# #' Lam, H., Deutsch, E. W., Eddes, J. S., Eng, J. K., King, N., Stein, S. E.
# #' and Aebersold, R. (2007)
# #' Development and validation of a spectral library searching method for peptide
# #' identification from MS/MS.
# #' Proteomics, 7: 655-667.
# #' doi: https://doi.org/10.1002/pmic.200600625
# #'
# #' @param x double
# #' @param y double
# #' @return double, length == 1
# #' @noRd
# dotproduct <- function(x, y) {
#   as.vector(x %*% y) / (sqrt(sum(x*x)) * sqrt(sum(y*y)))
# }



# #' Simple function to ensure that breaks (for binning) are span al leat the
# #' expected range.
# #'
# #' @param brks `numeric` with *breaks* such as calculated by `seq`.
# #'
# #' @param rng `numeric(2)` with the range of original numeric values on which
# #'     the breaks were calculated.
# #'
# #' @noRd
# .fix_breaks <- function(brks, rng) {
#     ## Assuming breaks being sorted.
#     if (brks[length(brks)] <= rng[2])
#         brks <- c(brks, max((rng[2] + 1e-6),
#                             brks[length(brks)] + mean(diff(brks))))
#     brks
# }

# ## ---------------------------------------------------------------------

# numberOfCommonPeaks <- function(x, y, tolerance=25e-6, relative=TRUE) {
#   sum(commonPeaks(x, y, tolerance=tolerance, relative=relative))
# }
# commonPeaks <- function(x, y, method=c("highest", "closest"),
#                          tolerance=25e-6, relative=TRUE) {
#   m <- matchPeaks(x, y, method=match.arg(method), tolerance=tolerance,
#                   relative=relative)

#   m[which(is.na(m))] <- 0L

#   as.logical(m)
# }

# matchPeaks <- function(x, y, method=c("highest", "closest", "all"),
#                        tolerance=25e-6, relative=TRUE) {
#   method <- match.arg(method)

#   if (inherits(y, "Spectrum")) {
#     y <- mz(y)
#   }

#   if (peaksCount(x) == 0 || length(y) == 0) {
#     return(integer(peaksCount(x)))
#   }

#   m <- relaxedMatch(mz(x), y, nomatch=NA, tolerance=tolerance,
#                     relative=relative)

#   if (anyDuplicated(m)) {
#     if (method == "highest") {
```

```
#       o <- order(intensity(x), decreasing=TRUE)
#     } else if (method == "closest") {
#       o <- order(abs(mz(x)-y[m]))
#     } else {
#       o <- 1:length(x)
#     }
#     sortedMatches <- m[o]

#     if (method != "all") {
#       sortedMatches[which(duplicated(sortedMatches))] <- NA
#     }
#     m[o] <- sortedMatches
#   }

#   as.integer(m)
# }

# relaxedMatch <- function(x, table, nomatch=NA_integer_, tolerance=25e-6,
#                          relative=TRUE) {

#   if (relative) {
#     if (tolerance > 1L) {
#       stop(sQuote("tolerance"),
#            " must be smaller than 1 for relative deviations.")
#     }
#     tolerance <- table*tolerance
#   }

#   match.closest(x, table, tolerance=tolerance, nomatch=nomatch)
# }
```

107 File: TEST_mcnebula2.R

```
setwd("~/operation/sirius.mcn")
# library(MCnebula2)
mcn <- initialize_mcnebula(mcnebula())

mcn1 <- filter_structure(mcn)
mcn1 <- create_reference(mcn1)
mcn1 <- filter_formula(mcn1, by_reference=T)
```

```r
mcn1 <- create_stardust_classes(mcn1)
mcn1 <- create_features_annotation(mcn1)
mcn1 <- cross_filter_stardust(mcn1, 5, 1)

ids <- sample(features_annotation(mcn1)$.features_id, 8)
mcn1 <- draw_structures(mcn1, .features_id = ids)
# plot_msms_mirrors(mcn1, ids)

mcn1 <- create_nebula_index(mcn1)
mcn1 <- compute_spectral_similarity(mcn1)
mcn1 <- create_parent_nebula(mcn1, 0.01, 5, T)
mcn1 <- create_child_nebulae(mcn1, 0.01, 5)

mcn1 <- create_parent_layout(mcn1)
mcn1 <- create_child_layouts(mcn1)
mcn1 <- activate_nebulae(mcn1)

pdf("instance.pdf", width = 8, height = 8.5)
visualize_all(mcn1)
dev.off()

pdftools::pdf_convert("instance.pdf", dpi = 150)

re <- history_rblock(, "initialize_mcnebula\\(", "activate_nebulae\\(")

mcn1 <- .simulate_quant_set(mcn1)
mcn1 <- set_ppcp_data(mcn1)
mcn1 <- set_ration_data(mcn1)
mcn1 <- binary_comparison(mcn1, control - model,
                           model - control, 2 * model - control)

mcn1 <- draw_structures(mcn1, "Fatty Acyls")
mcn1 <- draw_nodes(mcn1, "Fatty Acyls")
mcn1 <- annotate_nebula(mcn1, "Fatty Acyls")
visualize(mcn1, "Fatty Acyls", annotate = T)

# pdf("child_nebulae.pdf", width = 10, height = 12)
# visualize_all(mcn1)
# dev.off()

des <- "see Table \\??"
```

```
re <- new_report(yaml = .yaml_default("de"),
                 document_mc_workflow("abstract"),
                 document_mc_workflow("introduction"),
                 document_mc_workflow("setup"),
                 new_heading("analysis", 1),
                 new_section("step1"),
                 new_section("step2"),
                 new_heading("statistic", 1),
                 new_section(NULL, paragraph = "the flowing is figure..."),
                 new_section(NULL, paragraph = des),
                 # include_figure("child_8.pdf", "plot1", "child-nebulae"),
                 # include_table(df, "table1", "top features"),
                 new_section(NULL, code_block =
                             new_code_block(codes = "re"))
)
```

108 File: TEST__test__pkg.R

```
setwd("~/MCnebula2")
devtools::check()
devtools::install()
devtools::test()
usethis::use_package("methods")
usethis::use_test("baseWorkflow.R")
```

109 File: utools__external__collate__evaluation.R

```
Sys.time()
# [1] "2023-01-30 14:44:43 CST"
Sys.info()
#                                                            sysname
#                                                            "Linux"
#                                                            release
#                                          "5.17.5-76051705-generic"
#                                                            version
# "#202204271406~1653440576~22.04~6277a18 SMP PREEMPT Wed May 25 01"
#                                                           nodename
#                                                           "pop-os"
#                                                            machine
#                                                           "x86_64"
#                                                                   login
#                                                                  "echo"
#                                                                    user
#                                                                  "echo"
#                                                          effective_user
#                                                                  "echo"

devtools::load_all("~/MCnebula2")

dirs <- c(
  "/media/echo/DATA/yellow/iso_gnps_pos",
  "/media/echo/DATA/yellow/noise_gnps_pos",
  "/media/echo/DATA/yellow/h_noise_gnps_pos"
)

lst <- lapply(dirs,
  function(dir) {
    mcn <- mcnebula()
    mcn <- initialize_mcnebula(mcn, "sirius.v4", dir)
    mcn <- collate_used(mcn)
    ## ------------------------------------- decrease size
    ## structure
    df <- entity(dataset(project_dataset(mcn))[[ ".f3_fingerid" ]])
    df <- dplyr::mutate(df, links = character(1), pubmed.ids = character(1))
    df <- dplyr::mutate_if(df, is.numeric, function(x) round(x, 2))
    entity(dataset(project_dataset(mcn))[[ ".f3_fingerid" ]]) <- df
    ## formula
    df2 <- entity(dataset(project_dataset(mcn))[[ ".f2_formula" ]])
    df2 <- dplyr::mutate_if(df2, is.numeric, function(x) round(x, 2))
    entity(dataset(project_dataset(mcn))[[ ".f2_formula" ]]) <- df2
    ## class
    df3 <- entity(dataset(project_dataset(mcn))[[ ".f3_canopus" ]])
    df3 <- dplyr::mutate_if(df3, is.numeric, function(x) round(x, 2))
    df3 <- dplyr::mutate(df3, description = character(1),
      rel.index = as.integer(rel.index))
    entity(dataset(project_dataset(mcn))[[ ".f3_canopus" ]]) <- df3
    mcn
  })

names(lst) <- paste0(c("", "median_noise_", "high_noise_"), "gnps_pos.rdata")
```

```
format(object.size(lst), units = "MB")
## ------------------------------------- save

lapply(names(lst),
  function(name) {
    mcn <- lst[[ name ]]
    save(mcn, file = paste0("~/utils.tool/inst/extdata/evaluation/", name))
  })
```

110 File: utools_external_collate_herbal.R

```
Sys.time()
# [1] "2023-01-02 15:33:34 CST"
Sys.info()
#                                                              sysname
#                                                              "Linux"
#                                                              release
#                                            "5.17.5-76051705-generic"
#                                                              version
# "#202204271406~1653440576~22.04~6277a18 SMP PREEMPT Wed May 25 01"
#                                                             nodename
#                                                             "pop-os"
#                                                              machine
#                                                             "x86_64"
#                                                                login
#                                                               "echo"
#                                                                 user
#                                                               "echo"
#                                                       effective_user
#                                                               "echo"

devtools::load_all("~/MCnebula2")
mcn_herbal <- mcnebula()
mcn_herbal <- initialize_mcnebula(mcn_herbal, "sirius.v4", "/media/echo/DATA/yellow/eucommia/neg/")
mcn_herbal <- collate_used(mcn_herbal)
# latest(mcn_herbal, , ".f2_formula")

## ------------------------------------- decrease size
## structure
df <- entity(dataset(project_dataset(mcn_herbal))[[ ".f3_fingerid" ]])
df <- dplyr::mutate(df, links = character(1), pubmed.ids = character(1))
```

```r
df <- dplyr::mutate_if(df, is.numeric, function(x) round(x, 2))
entity(dataset(project_dataset(mcn_herbal))[[ ".f3_fingerid" ]]) <- df

## formula
df2 <- entity(dataset(project_dataset(mcn_herbal))[[ ".f2_formula" ]])
df2 <- dplyr::mutate_if(df2, is.numeric, function(x) round(x, 2))
entity(dataset(project_dataset(mcn_herbal))[[ ".f2_formula" ]]) <- df2

## class
df3 <- entity(dataset(project_dataset(mcn_herbal))[[ ".f3_canopus" ]])
df3 <- dplyr::mutate_if(df3, is.numeric, function(x) round(x, 2))
df3 <- dplyr::mutate(df3, description = character(1),
                     rel.index = as.integer(rel.index))
entity(dataset(project_dataset(mcn_herbal))[[ ".f3_canopus" ]]) <- df3

## ------------------------------------- save
mcn_herbal1612 <- mcn_herbal
format(object.size(mcn_herbal1612), units = "MB")
save(mcn_herbal1612, file = "~/utils.tool/inst/extdata/mcn_herbal1612.rdata")
# load("~/utils.tool/inst/extdata/mcn_herbal1612.rdata")
```

111 File: utools_external_collate_serum.R

```r
Sys.time()
# [1] "2022-12-12 14:56:25 CST"
Sys.info()
#                                                           sysname
#                                                           "Linux"
#                                                           release
#                                           "5.17.5-76051705-generic"
#                                                           version
# "#202204271406~1653440576~22.04~6277a18 SMP PREEMPT Wed May 25 01"
#                                                          nodename
#                                                          "pop-os"
#                                                           machine
#                                                          "x86_64"
#                                                             login
#                                                            "echo"
#                                                              user
#                                                            "echo"
#                                                     effective_user
#                                                             "echo"

devtools::load_all("~/MCnebula2")
mcn_serum <- mcnebula()
mcn_serum <- initialize_mcnebula(mcn_serum, "sirius.v4", "/media/echo/DATA/yellow/massive/cell/massive.
mcn_serum <- collate_used(mcn_serum)
# latest(mcn_serum, , ".f2_formula")

## ------------------------------------- decrease size
## structure
df <- entity(dataset(project_dataset(mcn_serum))[[ ".f3_fingerid" ]])
df <- dplyr::mutate(df, links = character(1), pubmed.ids = character(1))
df <- dplyr::mutate_if(df, is.numeric, function(x) round(x, 2))
entity(dataset(project_dataset(mcn_serum))[[ ".f3_fingerid" ]]) <- df

## formula
df2 <- entity(dataset(project_dataset(mcn_serum))[[ ".f2_formula" ]])
df2 <- dplyr::mutate_if(df2, is.numeric, function(x) round(x, 2))
entity(dataset(project_dataset(mcn_serum))[[ ".f2_formula" ]]) <- df2

## class
df3 <- entity(dataset(project_dataset(mcn_serum))[[ ".f3_canopus" ]])
df3 <- dplyr::mutate_if(df3, is.numeric, function(x) round(x, 2))
df3 <- dplyr::mutate(df3, description = character(1),
                     rel.index = as.integer(rel.index))
entity(dataset(project_dataset(mcn_serum))[[ ".f3_canopus" ]]) <- df3

## ------------------------------------- save
mcn_serum6501 <- mcn_serum
format(object.size(mcn_serum6501), units = "MB")
save(mcn_serum6501, file = "~/utils.tool/inst/extdata/mcn_serum6501.rdata")
# load("~/utils.tool/inst/extdata/mcn_serum6501.rdata")
```

112 File: utools_external_mcn.R

```
test <- mcn_5features

e1 <- substitute({
  test1 <- filter_structure(test)
  test1 <- create_reference(test1)
```

```r
  test1 <- filter_formula(test1, by_reference=T)
  test1 <- create_stardust_classes(test1)
  test1 <- create_features_annotation(test1)
})
eval(e1)

toAnno <- test1
save(toAnno5, "~/utils.tool/inst/extdata/toAnno5.rdata")

test1 <- cross_filter_stardust(test1, 1, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_parent_nebula(test1, 0.01, T)
test1 <- create_child_nebulae(test1, 0.01, 5)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

test1 <- .simulate_quant_set(test1)
test1 <- set_ppcp_data(test1)
test1 <- set_ration_data(test1)
test1 <- binary_comparison(test1, control - model,
                           model - control, 2 * model - control)

toBinary5 <- test1
save(toBinary5, file = "~/utils.tool/inst/extdata/toBinary5.rdata")

# test1 <- draw_structures(test1, "Fatty Acyls")
# test1 <- draw_nodes(test1, "Fatty Acyls")
# test1 <- annotate_nebula(test1, "Fatty Acyls")
```

```r
# ==========================================================================
# 30 features
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

setwd("~/operation/sirius.test")

test <- initialize_mcnebula(mcnebula())
test1 <- filter_structure(test)
test1 <- create_reference(test1)
test1 <- filter_formula(test1, by_reference=T)
test1 <- create_stardust_classes(test1)
```

```
test1 <- create_features_annotation(test1)
test1 <- cross_filter_stardust(test1, 5, 1)
test1 <- create_nebula_index(test1)
test1 <- compute_spectral_similarity(test1)
test1 <- create_parent_nebula(test1, 0.01, T)
test1 <- create_child_nebulae(test1, 0.01, 5)
test1 <- create_parent_layout(test1)
test1 <- create_child_layouts(test1)
test1 <- activate_nebulae(test1)

toActiv30 <- test1
save(toActiv30, file = "~/utils.tool/inst/extdata/toActiv30.rdata")
```