Processing of microscopy analysis results in R

Jakub Zahumensky

2024-06-12

Install and load required packages

```r
# the following packages are required for:
# 'data.table' - work with data tables
# 'dplyr' - the %>% piping tool and summarise() function
# 'svDialogs' - interactive dialog windows
# 'ggplot2' - basic graphs
# 'ggstatsplot' - fancy violin graphs with indication of statistical significance
# 'car' - "Companion to Applied Regression" - Lavene's test of equal variance
# 'rstatix' - advanced ANOVA and Tukey's post-test
# 'multcomp' - defining selective contrast definitions for Tukey's post-hoc test
# 'emmeans' - filtering of Tukeys's post-hoc test results

required_packages <- c("here", "knitr", "glue", "data.table", "dplyr", "svDialogs", "ggplot2", "ggstatsp
new_packages <- required_packages[!(required_packages %in% installed.packages()[,"Package"])]
if (length(new_packages))
   install.packages(new_packages)

# load all required packages
for (PACKAGE in required_packages){
    library(PACKAGE, character.only = TRUE)
}
```

This script processes a 'Results' table, e.g., from microscopy image analysis, calculating means and SDs of all quantified parameters across experiments, biological replicate dates, strains, cultivation media and times, and conditions. It automatically extracts the column headers and uses them as variable names to store the calculated values (as data frames) and also saves them in the form of tables. The data from individual biological replicates are written into separate columns. The tables can be directly used for plotting in GraphPad Prism, Systat SigmaPlot etc.

Scripts for plotting in R are provided as well. Here, also more complex plots are available, where all raw data can be plotted for each biological replicate to create "SuperPlots" and Pirateplots.

Load data from the *Results.csv* file into a data table and look inside

The *Results table* is loaded using known patterns, i.e., the columns are comma-separated, period (.) is used as decimal separator and lines starting with the pound sign (#) are comments. For verification, the column names are written in the output by the code.

```r
workingDir <- here()
results_file <- list.files(getwd(), pattern = "Results")
```

```
# if there are multiple Results tables in the current directory, ask which one to process
if (length(results_file) > 0){
  results_file <- dlg_list(
    results_file,
    preselect = NULL,
    multiple = FALSE,
    title = "Multiple Results tables found",
    gui = .GUI
    )$res
}

if (length(results_file) == 0){
    dlg_message(
    "No Results table found. Terminating script.",
    type = c("ok"),
  )
}

data_table <- data.table(read.csv(file = results_file, header = TRUE, sep = ",", quote = "", dec = ".",
# before continuing, we convert all 'NaN' values to 0
for (col in names(data_table))
  set(data_table, i = which(data_table[[col]] == "NaN"), j = col, value = 0)

for (col in names(data_table))
  set(data_table, i = which(data_table[[col]] == "control"), j = col, value = "_control")
```

From the column names, we extract the names of those that describe the data files, including the biological replicate date (all columns between the *exp_code* and *frame* columns) and of those that contain the results of the analysis (all columns after *frame*):

```
# extract the position of important columns
frame_column <- which(colnames(data_table)=="frame") # this columns is always present and "divides" the
last_column <- ncol(data_table)

# columns with parameters that describe the data files; used for filtering into unique combinations
# first column is excluded here as it contains the experiment code, which is handled at a different lev
parameters <- c()
for (i in 3:frame_column-1){
  parameters[i-2] <- colnames(data_table)[i]
}

# columns with quantification data from which means and SDs are calculated below
columns <- c()
for (i in frame_column+1:last_column-frame_column){
  columns[i-frame_column] <- colnames(data_table)[i]
}
```

The following information was loaded from their *Results table*: experiment codes, descriptive parameters and quantification parameters

Data filtering and summarization

Since the analysis can be run across multiple experiments, we first need to separate the *Results table* accordingly and save the summary results, graphs and results of statistical analyses into designated folders. First, we make the necessary folders:

```r
# if it does not exist already, make a folder for each experiment in the 'Results table'
analysis_dir <- file.path(workingDir, "analysis")
if (!file.exists(analysis_dir))
  dir.create(analysis_dir)

for (EXP in unique(data_table$exp_code)){
  EXP_dir <- file.path(workingDir, glue("analysis/{EXP}"))
  if (!file.exists(EXP_dir))
    dir.create(EXP_dir)
}
```

We then group the data based on all possible unique combinations of *experiments*, *biological replicate dates* and *descriptive parameters* extracted from the raw image file names during analysis in Fiji. We then calculate the mean and SD for each group and save the summary tables. Note that the *descriptive parameters*:

- cell types/strains
- growth media
- growth times
- treatment conditions

are optional and only those that are specified are taken into account. The user is asked to select a column according to which the summary tables will be ordered, if multiple choices exist (separately for each experiment). The summary tables are saved and also stored as variables for downstream graphing and statistical analyses.

```r
for (EXP in unique(data_table$exp_code)){
  data_table_exp <- data_table[exp_code == EXP, ] # subset the data_table for the current experiment EX

  # summarize the data table, i.e., calculate mean and SD for all columns with numbers from quantificat
  summary <- data.table(data_table_exp %>%
    group_by(BR_date, across(all_of(parameters))) %>%
    dplyr::summarise(across(all_of(columns), list(mean = ~mean(.), sd = ~sd(.)), .names = "{col}_{fn}")
  assign(paste0("summary_", EXP), summary)

  # ask the user to select a column by which the summary tables will be ordered
  temp_list <- c()
  for (col in parameters){
    if (length(unique(summary[[col]])) > 1){
      temp_list <- c(temp_list, col)
    }
  }
  parameters_multichoice <- assign(paste0("parameters_multichoice_", EXP), temp_list)
  title_parameters <- parameters[! parameters %in% parameters_multichoice]
  assign(paste0("title_parameters_", EXP), title_parameters)

  # ask the user how to order the summary tables
  if (length(parameters_multichoice) > 1){
    prompt <- glue("{EXP} - order summary tables by:")
    sorting_col <- dlg_list(
      parameters_multichoice,
      preselect = NULL,
      multiple = FALSE,
      title = prompt,
      gui = .GUI
    )$res
  } else {
    sorting_col <- parameters_multichoice
  }

  # order the summary table by the category selected and save it as an xlsx file
  setorderv(summary, cols = sorting_col)
  out_file <- glue("analysis/summary_{EXP}.xlsx")
  writexl::write_xlsx(as.data.frame(summary), path = here("", out_file))

  # create and save results tables for each quantified parameter, print data for each biological replic
  # the ordering of the table is kept
  columns_mean <- paste0(columns, "_mean")
  for (COL in columns_mean){
    # write a summary table for each quantification parameter
    subset_columns <- c("BR_date", parameters, COL)
    summary_subset <- summary[, ..subset_columns]
    summary_wide <- reshape(summary_subset, idvar = parameters, timevar = "BR_date", direction = "wide"
    out_file <- glue("analysis/{EXP}/{COL}.xlsx")
    writexl::write_xlsx(as.data.frame(summary_wide), path = here("", out_file))
  }
}
```

Graphs

Definition of groupings

The generation of plots below supports groupings by 3 distinct descriptive parameters: the *independent (x-axis) variable*, the *factor (grouping) variable* and the *faceting variable*. Before the plots are made, the user is prompted to select these variables, if multiple options exist. If more than 3 descriptive variables are changing within an experiment, the data will not be filtered further, resulting in bad graphs. In such a case, either introduce filtering within the code or separate the Results table manually.

```
for (EXP in unique(data_table$exp_code)){
  summary <- get(glue("summary_{EXP}"))
  parameters_multichoice <- get(glue("parameters_multichoice_{EXP}"))
  parameters_multichoice_temp <- parameters_multichoice

  prompts <- c(glue("Ind. variable for {EXP}:"), glue("Factors for {EXP}:", glue("Facet {EXP} by:")))
  terms <- c("", "", "")
  for (i in 1:3){
    if (length(parameters_multichoice_temp) > 1){
      terms[i] <- dlg_list(
        parameters_multichoice_temp,
        preselect = NULL,
        multiple = FALSE,
        title = prompts[i],
        gui = .GUI
      )$res
    } else {
      if (length(parameters_multichoice_temp) > 0){
        terms[i] <- parameters_multichoice_temp
      }
    }
    parameters_multichoice_temp <- parameters_multichoice_temp[! parameters_multichoice_temp %in% terms
  }
  assign(paste0("terms_", EXP), terms)

  terms_extra <- c()
  terms_extra_choices <- c()
    for (col in parameters_multichoice_temp){
      prompt <- glue("Select {col}")
      terms_extra <- c(terms_extra, col)
      terms_extra_choices <- c(
        terms_extra_choices,
        dlg_list(
          unique(data_table_exp[[col]]),
          preselect = NULL,
          multiple = FALSE,
          title = prompt,
          gui = .GUI
        )$res
      )
    }
  assign(paste0("terms_extra_", EXP), terms_extra)
  assign(paste0("terms_extra_choices_", EXP), terms_extra_choices)
}
```

Boxplots

Simple boxplots are generated for each quantification parameter. The graphs show *individual biological replicate values*, *median* and *data interval*. Grouping and faceting are supported, and a single image file will be created for each quantification parameter.

```
for (EXP in unique(data_table$exp_code)){
  summary <- get(glue("summary_{EXP}"))
  terms <- get(glue("terms_{EXP}"))
  terms_extra <- get(glue("terms_extra_{EXP}"))
  title_parameters <- get(glue("title_parameters_{EXP}"))

  plot_title <- ""
  for (col in title_parameters){
      plot_title <- paste(plot_title, summary[1, ..col])
  }

  # the following graphing works for up to 3 descriptive variables changing within the experiment
  # if more than 3 are changing, the plots will be erroneous (reconsider the experiment design if this
  descr <- ""
  for (COL in columns_mean){
    fig_width <- 6
    fig_height <- 4
    figure <- ggplot(summary, aes(x = !!sym(terms[1]), y = !!sym(COL))) +
      geom_boxplot(aes(fill = !!sym(terms[2])), position = position_dodge(width = 0.75)) +
      geom_jitter(aes(fill = !!sym(terms[2])), alpha = 0.75, position = position_dodge(width = 0.75)) +
      guides(fill = guide_legend(title = terms[2]), color = guide_legend(title = terms[2])) +
      theme_bw() +
      labs(title = plot_title) +
      scale_x_discrete(guide = guide_axis(n.dodge = 2))
      if (nchar(terms[3]) > 0) {
        figure <- figure + facet_wrap(as.formula(paste("~", terms[3])))
        fig_width <- fig_width*2
      }
    if (length(terms_extra_choices))
      descr <- paste0("_", terms_extra_choices, collapse = "")
    out_file <- glue("analysis/{EXP}/{COL}{descr}.png")
    ggsave(out_file, figure, device = "png", width = fig_width, height = fig_height)
  }
}
```

Violin plots

The user can select multiple quantification parameters for which they can make more intricate violin plots. These show individual *biological replicate values*, *mean*, *median* and *multiple t-tests*, based on the selected p value adjustment. If three descriptive parameters are changing, there will be multiple graphs made for each quantification parameter, one for each value of the *faceting variable*.

```
library(ggstatsplot)

for (EXP in unique(data_table$exp_code)){
  summary <- get(glue("summary_{EXP}"))
  terms <- get(glue("terms_{EXP}"))
  title_parameters <- get(glue("title_parameters_{EXP}"))

  # as different things are analyzed from the tangential and transversal images, so different plots nee
  if (grepl("transversal", results_file)) {
    graphs_wanted <- c("foci_number", "foci_density", "cell_I.mean",
                       "plasma_membrane_I.div.cell_I.integrated.", "plasma_membrane_I.div.cyt_I.mean.")
  } else {
    graphs_wanted <- c("foci_density.find_maxima.", "foci_density.analyze_particles.",
                "area_fraction.foci_vs_ROI.", "mean_foci_intensity",
                "length.um.","size.um.2.")
  }
  graphs_preselect <- intersect(names(data_table), graphs_wanted)
  graphs <- dlg_list(
    columns,
    preselect = graphs_preselect,
    multiple = TRUE,
    title = "Select violin graph(s) to make",
    rstudio = getOption("svDialogs.rstudio", FALSE),
    gui = .GUI
  )$res
  graphs <- paste0(graphs, "_mean")

  correction <- dlg_list(
        c("holm", "bonferroni"),
        preselect = NULL,
        multiple = FALSE,
        title = "Choose p adjustment method",
        gui = .GUI
      )$res

  if (nchar(terms[2]) > 0){
    grouping_var <- terms[2]
  } else {
    grouping_var <- title_parameters[1]
  }

  if (nchar(terms[3]) > 0){
    facets <- unique(summary[[terms[3]]])
    faceting_var <- terms[3]
  } else {
    facets <- summary[1, get(title_parameters[1])]
    faceting_var <- title_parameters[1]
  }

  # the following graphing works for up to 3 descriptive variables changing within the experiment
  # if more than 3 are changing, the plots will be erroneous (reconsider the experiment design if this
  for (FACET in facets){
    summary_temp <- summary[get(faceting_var) == FACET, ]
    for (GRAPH in graphs){
      fig_width <- 6
      fig_height <- 5
      figure <- grouped_ggbetweenstats(
        data = summary_temp,
        x = !!terms[1],
        y = !!GRAPH,
        grouping.var = !!sym(grouping_var),
        pairwise.display = "significant",
        p.adjust.method = correction,
        xlab = "",
        centrality.point.args = list(size = 4, color = "darkred"),
        point.args = list(position = ggplot2::position_jitterdodge(dodge.width = 1), alpha = 0.75, size
      )
      if (nchar(terms[2]) > 0)
         fig_width <- fig_width*2
      figure_file <- here(glue("analysis/{EXP}_{GRAPH}_{FACET}.png"))
      ggsave(figure_file, figure, device = "png", width = fig_width, height = fig_height)
    }
  }
}
```

Statistical analysis

Independent statistical analysis is performed on data with at least 3 biological replicates. Normality (Shapiro-Wilk) and equal variance (Levene's test) are tested, followed by multiple unpaired t-tests (using Bonferroni and Holm corrections) and ANOVA with Tukey's HSD (honestly significant difference) post-hoc test. Results are reported for each quantified parameter in a separate text file.

```r
for (EXP in unique(data_table$exp_code)){
  summary <- get(glue("summary_{EXP}"))
  parameters_multichoice <- get(glue("parameters_multichoice_{EXP}"))

  # count group sizes for each unique combination of available descriptive parameters
  group_sizes <- summary[, .N, by = parameters]
  assign(glue("group_sizes_{EXP}"), group_sizes)
  # filter the summary table to only keep data pertaining to groups of size >= 3
  groups_filtered <- group_sizes[N >= 3]
  summary_filtered <- summary[groups_filtered, on = parameters]
  assign(glue("summary_filtered_{EXP}"), summary_filtered)

  if (nrow(summary_filtered) > 0){
    for (COL in columns_mean){
      if (length(unique(summary_filtered[[COL]])) > 1){
        p_threshold <- 0.05
        out_file_stat <- glue("analysis/{EXP}/{COL}_statistical_analysis.txt")
        write(paste0("Statistical analysis report for: ", COL, "\n", "p-adj threshold: ", p_threshold,


        separator <- "---------------------------------------- Group size analysis ---------------------
        write(separator, file = out_file_stat, append = TRUE)
        capture.output(list(as.data.frame(group_sizes)), file = out_file_stat, append = TRUE)
        if (all(group_sizes$N >= 3)){
          note <- "\nAll groups are large enough for statistical analysis (i.e., N >= 3 for each group)
        } else {
          note <- "\nThe size of some groups is smaller than 3. These are excluded from statistical anal
        }
        write(note, file = out_file_stat, append = TRUE)

        ########################################################################################
        # As the additional statistical analyses compares various groups, we need to regroup the summary
        # to make the actual groups based on the unique combinations of descriptive parameters.
        # This is easier to work with than trying to force the commands to wrap their heads around the f
        # First, we create a new grouping variable based on unique combinations:
        summary_united <- summary_filtered %>%
          unite("group", all_of(parameters_multichoice), sep = "_", remove = FALSE)
        summary_united$group <- factor(summary_united$group)
        contrast_formula <- as.formula(paste(COL, "~", "group"))
        ########################################################################################


        separator <- "------------------------------------------ Normality test ----------------------
        write(separator, file = out_file_stat, append = TRUE)
        shapiro_results <- summary_united %>%
          group_by(group) %>%
          summarise(p_value = shapiro.test(!!sym(COL))$p.value) %>%
          mutate(test_status = ifelse(p_value > p_threshold, "passed", "failed"))
        shapiro_results <- as.data.frame(shapiro_results)
        if (all(shapiro_results$p_value > p_threshold)) {
          note <- "All groups passed the normality test.\n\n"
        } else {
          note <- "Normality test failed for at least one of the groups. See the table for details and 
        }
        capture.output(list(shapiro_results), file = out_file_stat, append = TRUE)
        write(note, file = out_file_stat, append = TRUE)


        separator <- "----------------- Equal variance test: Levene's Test (homoscedasticity assumption
        write(paste0("\n", separator), file = out_file_stat, append = TRUE)
        levene_results <- leveneTest(contrast_formula, data = summary_united)
        capture.output(levene_results, file = out_file_stat, append = TRUE)
        note1 <- "\nDf - degrees of freedom (i.e., number of groups - 1) and total number of observatio
        if (levene_results$`Pr(>F)`[1] > p_threshold){
          note2 <- "Equal variance test passed.\n\n"
        } else {
          note2 <- "The variances across the groups are not equal.\n\n"
        }
        write(paste0(note1, note2), file = out_file_stat, append = TRUE)



        separator <- "--------------------------------- Multiple t-tests with corrections -------------
        write(separator, file = out_file_stat, append = TRUE)
        t_test_bonferroni <- pairwise.t.test(summary_united[[COL]], summary_united$group, p.adjust.meth
        capture.output(t_test_bonferroni, file = out_file_stat, append = TRUE)
        t_test_holm <- pairwise.t.test(summary_united[[COL]], summary_united$group, p.adjust.method = "
        capture.output(t_test_holm, file = out_file_stat, append = TRUE)
        write("\n", file = out_file_stat, append = TRUE)


        separator <- "-------------------------------------------- ANOVA (base R) ----------------------
        note <- "\n'aov()' function from the 'base R' package.\n"
        write(paste0(separator,note), file = out_file_stat, append = TRUE)
        anova_result <- aov(contrast_formula, data = summary_united)
        aov_summary <- summary(anova_result)
        capture.output(aov_summary, file = out_file_stat, append = TRUE)
        if (any(aov_summary[[1]]$`Pr(>F)` < p_threshold, na.rm = TRUE)) {
          note <- "There is a significant difference among the groups.\n\n"
          separator <- "-------------------- Tukey's HSD (honestly significant difference) test (base R
          note1 <- "\n'TukeyHSD()' function from the 'R base' package run on the ANOVA results above.\n
          note2 <- "Only contrasts that are significantly different are reported.\n"
          write(paste0(note, separator, note1, note2), file = out_file_stat, append = TRUE)
          tukey_res <- setorder(as.data.frame(TukeyHSD(anova_result)$group), `p adj`)
          tukey_res <- filter(tukey_res, `p adj` <= p_threshold)
          capture.output(list(tukey_res), file = out_file_stat, append = TRUE)
          write("\n", file = out_file_stat, append = TRUE)
        } else {
          note <- "The groups are not significantly different.\n\n"
          write(note, file = out_file_stat, append = TRUE)
```

```
        }


        separator <- "----------------------------------------- ANOVA (rstatix) ----------------------
        note <- "\n'anova_test()' function from the 'rstatix' package\n"
        write(paste0(separator, note), file = out_file_stat, append = TRUE)
        anova_results2 <- anova_test(formula = contrast_formula, data = summary_united)
        capture.output(anova_results2, file = out_file_stat, append = TRUE)
        if (anova_results2$p[1] < p_threshold){
          note <- "There is a significant difference among the groups.\n\n"
          separator <- "------------------- Tukey's HSD (honestly significant difference) test (rstatix
          note1 <- "\n'tukey_hsd()' function from the 'rstatix' package.\n"
          note2 <- "Only contrasts that are significantly different are reported.\n"
          write(paste0(note, separator, note1, note2), file = out_file_stat, append = TRUE)
          tukey_res2 <- setorder(as.data.frame(tukey_hsd(x = summary_united, formula = contrast_formula
          tukey_res2 <- filter(tukey_res2, p.adj <= p_threshold)
          capture.output(list(tukey_res2), file = out_file_stat, append = TRUE)
        } else {
          note <- "The groups are not significantly different.\n"
          write(note, file = out_file_stat, append = TRUE)
        }
      }
    }
  }
}
```

Inform the user that the processing has finished and all available smmary tables, graphs, and statistical analyses have been performed.

```
dlg_message(
  "Finito! Press 'OK' to continue.",
  type = c("ok"),
)
```