Statistical Analysis and Visualization

Timothy Smyth

2024-04-09

Statistical Analysis and Visualization

In addition to differential expression analysis, other methods of statistical analysis and visualization may be desired. As DE occurs between two sepcific groups, it can be difficult to determine overall patterns between multiple treatment groups. Here, genes are selected for visualization and downstream analysis based on significant differences between treatment groups. Depending on the results of normality testing (shapiro-wilk test), repeated measures (RM) one-way ANOVA (parametric data) or Friedman tests (non-parametric data) are employed to determine if there is a significant difference in log count per million (logCPM) data between any groups. Genes with significant differences between at least two groups are retained, expression patterns are visualized, and downstream analysis is performed.

Prepare the environment

```
rm(list = ls(all.names = TRUE)) # clears global environ.

# Load packages
library(tidyverse) # for data cleaning
library(dplyr)
library(ggplot2) # for plotting
library(broom)
library(tibble)
library(edgeR)
library(future.apply)
library(factoextra)
library(parallelpam)
library(ComplexHeatmap)
library(circlize)
```

Perform statistical testing and gene identification

Data is tested for normality using the shapiro-wilk test. If all groups have p >= 0.05, a RM one-way ANOVA is conducted. If not, the Friedman test, an alternative for RM one-way ANOVA for non-parametric data, is conducted.

P-values are adjusted using the Benjamini-Hochberg Procedure (FDR) and genes with adjusted p-values < 0.05 are retained.

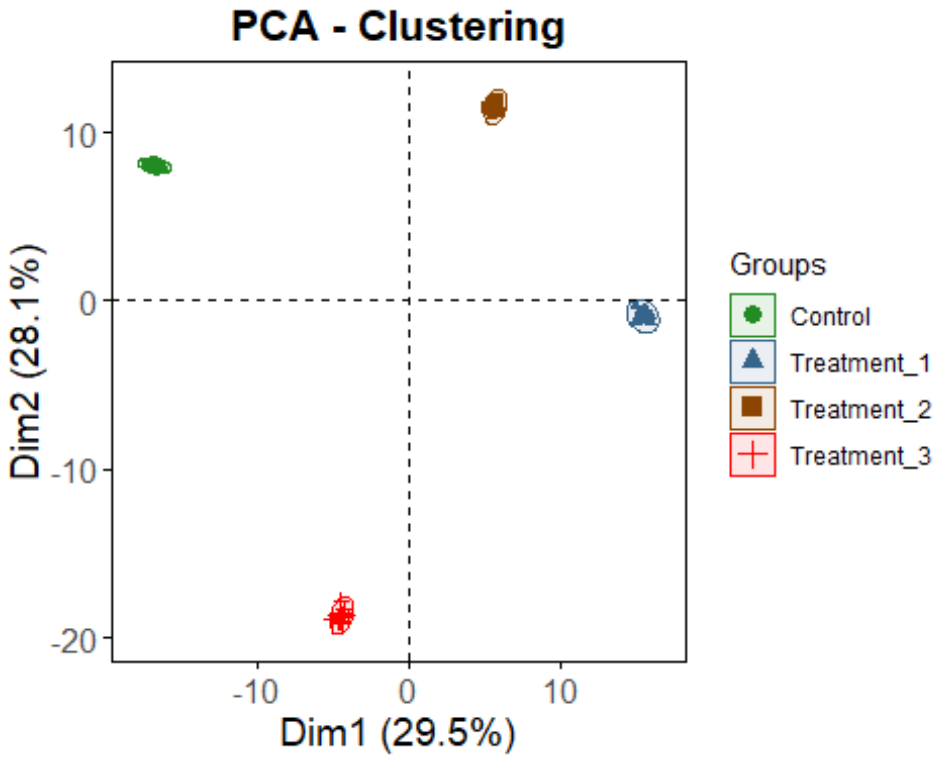
```
load("Selected_lcpm_Data.RData")

# Set maximum size allowed to be passed to future to 850 MB
options(future.globals.maxsize = 891289600)
```

```r
# Begin future multisession for parallelization
plan(multisession)

pivot <- lcpm %>% pivot_longer(cols = colnames(lcpm[3:ncol(lcpm)]),
                               names_to = 'Gene',
                               values_to = 'lcpm')

statistics <- future_lapply(colnames(lcpm[3:ncol(lcpm)]), function(x) {

  tmp <- pivot %>% subset(Gene == x)

  normality <- tmp %>%
      group_by(Treatment) %>%
      do(broom::tidy(shapiro.test(.$lcpm))) %>% data.frame()

  if(min(normality[['p.value']]) >= 0.05) {

  ANOVA <- broom::tidy(aov(lcpm[[x]] ~ Treatment +
Error(Biological_Sample/Treatment), data = lcpm))

  ANOVA <- ANOVA[[2, 'p.value']]

  } else{

  Friedman <- broom::tidy(friedman.test(y = lcpm[[x]],
                                        groups = lcpm$Treatment,
                                        blocks = lcpm$Biological_Sample)) %>%
data.frame()

  Friedman <- Friedman[['p.value']]

  }

})

# Return to sequential to reduce memory usage
plan(sequential)

# Bind p-values to one location
statistics_results <- do.call(rbind, statistics)

# Rename columns to associated genes
rownames(statistics_results) <- colnames(lcpm[3:ncol(lcpm)])
colnames(statistics_results) <- 'p'

# Order by ascending p value and rank
statistics_results <- data.frame(statistics_results) %>% arrange(p)
```

```
statistics_results$Rank <- seq_along(1:nrow(statistics_results))

# Calculate adjusted p-value using BH method (FDR)
statistics_results$BH <-  p.adjust(statistics_results$p,
                                   method = 'BH')

Genes <- statistics_results %>% subset(BH < 0.05) %>% rownames()

save(file = 'Sig_Genes.RData',
     Genes)
```

Principal component analysis by treatment group following removal of genes without adjusted p-values < 0.05.

```
# Save metadata
meta <- lcpm[, 1:2]

# Run PCA
pca.res <- prcomp(lcpm[, Genes], center = TRUE, scale = TRUE)

# Set theme
theme_set(theme_bw())

colors <- c("forestgreen",
            "steelblue4",
            "darkorange4",
            "red")

# All groups cluster plot
fviz_pca_ind(pca.res,
             label = "none",
             habillage = meta$Treatment,
             palette = colors,
             addEllipses = TRUE) +

  ggtitle("PCA - Clustering") +

  theme(axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 16),
        panel.border = element_rect(fill = NA, color = "black", size = 0.3),
        panel.grid.minor = element_blank(),
        panel.grid.major = element_blank())

## Warning: The `size` argument of `element_rect()` is deprecated as of
ggplot2 3.4.0.
## ℹ Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Partition around medoid clustering

The optimal number of clusters for PAM clustering is determined and PAM clustering is performed

```
# Determine the optimal number of clusters
silhouette <- fviz_nbclust(lcpm[3:ncol(lcpm)], cluster::pam, method =
'silhouette')

# Perform partition around medoid (PAM) clustering
parallelpam::JWriteBin(as.matrix(t(scale(lcpm[3:ncol(lcpm)]))),
                       "group.bin",
                       dtype = "float",
                       dmtype = "full")

CalcAndWriteDissimilarityMatrix("group.bin",
                                "group2.bin",
                                distype = "L2",
                                restype = "float",
                                comment = "L2 distance for vectors in jmatrix
file vst_results_group.bin",
                                nthreads = 8)

## Loading required package: memuse

JMatInfo("group2.bin")
```

```
## File:               group2.bin
## Matrix type:        SymmetricMatrix
## Number of elements: 253009 (126756 really stored)
## Data type:          float
## Endianness:         little endian (same as this machine)
## Number of rows:     503
## Number of columns:  503
## Metadata:           Stored only names of rows.
## Metadata comment:  "L2 distance for vectors in jmatrix file
vst_results_group.bin"

pam = ApplyPAM("group2.bin",
               k = length(silhouette[["data"]][["y"]]),
               init_method = "BUILD",
               max_iter = 1000,
               nthreads = 8)
```

Resulting genes are presented using a heatmap with gene clustering based on PAM clustering results.

```
# Scale results
heat <- t(scale(lcpm[3:ncol(lcpm)]))

levels <- paste0('Cluster ', seq_along(1:max(pam$clasif)))

# Define clusters calculated above for sorting
pam$clasif <- paste0('Cluster ', pam$clasif)

pam$clasif <- factor(pam$clasif,
                     levels = levels)

# Create annotation data frame
annotation <- data.frame(
  Treatment = meta$Treatment,
  stringsAsFactors = FALSE)

colors <- list(

  Treatment = c('Control' = 'forestgreen',
                'Treatment_1' = 'steelblue4',
                'Treatment_2' = 'darkorange4',
                'Treatment_3' = 'red'))

colAnn <- HeatmapAnnotation(
  df = annotation,
  name = ' ',
  which = 'col', # 'col' (samples) or 'row' (gene) annotation?
  col = colors,
  annotation_height = 0.6,
  annotation_width = unit(1, 'cm'),
```

```r
  gap = unit(1, 'mm'),
  show_legend = TRUE)

hmap <- Heatmap(heat,

                # split the genes / rows according to the PAM clusters
                row_split = pam$clasif,
                column_split = meta$Treatment,
                cluster_row_slices = TRUE,
                gap = unit(2.5, "mm"),
                row_gap = unit(2.5, 'mm'),
                border = TRUE,
                height = ncol(heat) * unit(5, 'mm'),
                width = ncol(heat) * unit(4, 'mm'),

                name = 'Gene\nZ-Score',

                col = colorRamp2(c(-2, 0, 2), hcl_palette = "Vik"),

                # parameters for the color-bar that represents gradient of
expression
                heatmap_legend_param = list(
                  color_bar = 'continuous',
                  legend_direction = 'horizontal',
                  legend_width = unit(15, 'cm'),
                  legend_height = unit(15, 'cm'),
                  title_position = 'topcenter',
                  title_gp = gpar(fontsize = 16, fontface = 'bold'),
                  labels_gp = gpar(fontsize = 16, fontface = 'bold')),

                # row (gene) parameters
                cluster_rows = TRUE,
                show_row_dend = TRUE,
                row_title = NULL,
                show_row_names = FALSE,
                row_names_gp = gpar(fontsize = 18, fontface = 'bold'),
                row_names_side = 'left',
                row_dend_width = unit(25,'mm'),

                # column (sample) parameters
                cluster_columns = TRUE,
                show_column_dend = TRUE,
                column_title = NULL,
                show_column_names = FALSE,
                column_names_gp = gpar(fontsize = 10, fontface = 'bold'),
                column_names_max_height = unit(10, 'cm'),
                column_dend_height = unit(25,'mm'),
                column_names_rot = 45,
```

```
                # cluster methods for rows and columns
                clustering_distance_columns = function(x) as.dist(1 -
cor(t(x))),
                clustering_method_columns = 'ward.D2',
                clustering_distance_rows = function(x) as.dist(1 -
cor(t(x))),
                clustering_method_rows = 'ward.D2',

                # specify top and bottom annotations
                top_annotation = colAnn)


# png(file = "Heatmap.png", height = 1200, width = 1000)
#
# draw(hmap,
#      heatmap_legend_side = 'bottom',
#      annotation_legend_side = 'right')
#
# dev.off()
```