Gene Set Variation Analysis

Timothy Smyth

2024-03-27

Gene Set Variation Analysis (GSVA)

After removing genes without significant differences in log cpm data between any two groups, we can test for pathway enrichment to determine how different treatments impact different cellular processes. Here, Gene Ontology (GO) Biological Process (BP) term enrichment and subsequent GSVA of enriched terms is conducted. GSVA allows gene sets, which represent genes associated with specific biological processes, to be tested for overall expression and collapsed into a single enrichment value. Following GSVA calculations, differential expression of these single GSVA values can be performed using limma-voom. Top enriched gene sets can then be extracted and compared between groups using a heatmap describing GSVA values and a yes/no significance heatmap.

Prepare environment

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

library(tibble)
library(tidyverse)
library(dplyr)
library(topGO)
library(org.Hs.eg.db)
library(GO.db)
library(biomaRt)
library(GSVA)
library(GSEABase)
library(edgeR)
library(ComplexHeatmap)
library(RColorBrewer)
library(circlize)
library(factoextra)
library(parallelpam)
library(variancePartition)
```

Prepare GO IDs using the human ENSEMBL gene BioMart database

This process has some issues. Namely, the Ensembl web interface is a public resource that does not work well with requests of this size. Timeout will frequently be reached, which will abort the process and force you to try again, which further leads to Ensembl trying to handle a large request it isn't really meant to handle. The chunk below is code that allows for the query, but the next section loads a completed query, allowing for this request to be skipped.

```
# # load gene universe for GO term assignment
# gene_universe <- read.csv('Ensembl_IDs.csv')
#
# # Select human ENSEMBL gene BioMart database and data set
# db = useMart('ENSEMBL_MART_ENSEMBL',
#              dataset = 'hsapiens_gene_ensembl',
#              # host = "https://www.ensembl.org"
#              host = "https://useast.ensembl.org"
#              )
#
# # Assign GO IDs to genes
# go_ids = getBM(attributes = c('go_id',
#                               'name_1006',
#                               'ensembl_gene_id',
#                               'namespace_1003'),
#                filters = 'ensembl_gene_id',
#                values = gene_universe$ID,
#                mart = db)
```

Perform GO BP term enrichment

```
# Load data
load("GO_Data.RData")
load("Sig_Genes.RData")
load("Selected_lcpm_Data.RData")

meta <- lcpm[1:2]

Results <- t(lcpm[, Genes])

# Create annotation list for TopGo object
gene_2_GO = unstack(go_ids[, c(1, 3)])

# Turn off scientific notation
options(scipen=999)

# Save gene names
names <- rownames(Results)

# remove any candidate genes without GO annotation
keep = names %in% go_ids[, 3]
keep = which(keep == TRUE)
names = names[keep]

# make named factor showing which genes are of interest
geneList = factor(as.integer(gene_universe %in% names))
names(geneList) = gene_universe

# Create TopGo object
GOdata = new('topGOdata',
```

```
              ontology = 'BP',
              allGenes = geneList,
              annot = annFUN.gene2GO,
              gene2GO = gene_2_GO)

##
## Building most specific GOs .....

##  ( 12329 GO terms found. )

##
## Build GO DAG topology ..........

##  ( 15417 GO terms and 34686 relations. )

##
## Annotating nodes ...............

##  ( 19098 genes annotated to the GO terms. )
```

```
# Define test using the weight01 algorithm with fisher
weight_fisher_result = runTest(GOdata,
                               algorithm = 'weight01',
                               statistic = 'fisher')

##
##           -- Weight01 Algorithm --
##
##       the algorithm is scoring 3012 nontrivial nodes
##       parameters:
##           test statistic: fisher

##
##   Level 17:  4 nodes to be scored    (0 eliminated genes)

##
##   Level 16:  10 nodes to be scored   (0 eliminated genes)

##
##   Level 15:  22 nodes to be scored   (84 eliminated genes)

##
##   Level 14:  40 nodes to be scored   (178 eliminated genes)

##
##   Level 13:  64 nodes to be scored   (703 eliminated genes)

##
##   Level 12:  97 nodes to be scored   (1528 eliminated genes)

##
##   Level 11:  179 nodes to be scored  (2492 eliminated genes)
```

```
##
##   Level 10:  291 nodes to be scored  (5291 eliminated genes)

##
##   Level 9:   365 nodes to be scored  (8179 eliminated genes)

##
##   Level 8:   447 nodes to be scored  (10761 eliminated genes)

##
##   Level 7:   452 nodes to be scored  (12949 eliminated genes)

##
##   Level 6:   455 nodes to be scored  (15391 eliminated genes)

##
##   Level 5:   314 nodes to be scored  (17082 eliminated genes)

##
##   Level 4:   174 nodes to be scored  (18118 eliminated genes)

##
##   Level 3:   81 nodes to be scored   (18507 eliminated genes)

##
##   Level 2:   16 nodes to be scored   (18784 eliminated genes)

##
##   Level 1:   1 nodes to be scored    (18898 eliminated genes)
```

```r
# Generate a table of results
allGO = usedGO(GOdata)

all_res = GenTable(GOdata,
                   weightFisher = weight_fisher_result,
                   orderBy = 'weightFisher',
                   topNodes = length(allGO),
                   numChar = 1000)

# Performing BH correction on our p values
p.adj = round(p.adjust(all_res$weightFisher,
                       method = "BH"), digits = 4)

# Add adjusted p value to results and order based on adjusted p value
all_res_final = cbind(all_res, p.adj)
all_res_final = all_res_final[order(all_res_final$p.adj),]

##############################################################################
##
##############################################################################
##
```

```
# Annotated:
# number of genes (in our gene list) that are annotated with the term

# Significant:
# Number of significantly DE genes annotated with that term (i.e. genes where
geneList = 1)

# Expected:
# Under random chance, number of genes that would be expected to be
significantly DE and annotated with that term

# p.value:
# P-value from Fishers Exact Test, testing for association between
significance and pathway membership.

################################################################################
##
################################################################################
##
```

Assign genes affiliated to each GO BP term to the linked term

```
# Isolate significant GO pathways and add matching gene names for GSVA below
Sig_GO <- all_res_final %>% subset(weightFisher < 0.05)
Sig_GO$Genes <- ''

tmp <- as.list(Sig_GO$GO.ID)

tmp2 <- lapply(tmp, function(y){

  tmp3 <- go_ids %>% subset(go_id == y)
  tmp3 <- as.character(tmp3$ensembl_gene_id)

})

Sig_GO$Genes <- tmp2
```

Create gene sets and perform GSVA

```
# Store treatment group data
phenoData <- new("AnnotatedDataFrame",
                 data = meta)

lcpm <- ExpressionSet(assayData = as.matrix(Results),
                      phenoData = phenoData)

Gene_sets <- list()

for(i in 1:length(Sig_GO[["Term"]])){

  Gene_sets[[i]] <- GeneSet(Sig_GO[["Genes"]][[i]],
```

```
                              setName = Sig_GO[["Term"]][[i]])

}

names(Gene_sets) <- Sig_GO[["Term"]]

library(BiocParallel)

param = SnowParam(10, "SOCK", progressbar = TRUE, exportglobals = FALSE)

# Create gsva parameter object
Parameters <- gsvaParam(lcpm, GeneSetCollection(Gene_sets), minSize = 2)

# Run GSVA on each pathways
GSVA_results <- gsva(Parameters, verbose = FALSE, BPPARAM = param)

## No annotation package name available in the input data object.
## Attempting to directly match identifiers in data to gene sets.
##   |
|                                                                      |   0%
|
|===                                                                   |   4%
|
|======                                                                |   8%
|
|=========                                                             |  12%
|
|============                                                          |  17%
|
|===============                                                       |  21%
|
|==================                                                    |  25%
|
|====================                                                  |  29%
|
|=======================                                               |  33%
|
|===========================                                           |  38%
|
|===============================                                       |  42%
|
|==================================                                    |  46%
|
|======================================                                |  50%
|
|=========================================                             |  54%
|
|=============================================                         |  58%
|
```

```
|=========================================                             |  62%
|
|=============================================                         |  67%
|
|=================================================                     |  71%
|
|=====================================================                 |  75%
|
|=========================================================             |  79%
|
|=============================================================         |  83%
|
|=================================================================     |  88%
|
|=====================================================================|  92%
|
|=====================================================================|  96%
|
|=====================================================================| 100%
```

```
GSVA_results <- cbind(GSVA_results@phenoData@data,
                      t(GSVA_results@assayData[["exprs"]]))
```

Perform differential expression analysis of GSVA scores

```
Treatment <- factor(meta$Treatment,
                    level = c("Control",
                              "Treatment_1", "Treatment_2", "Treatment_3"))

# Convert into a matrix
counts <- as.matrix(t(GSVA_results[3:ncol(GSVA_results)]))

model <- ~ 0 + Treatment + (1 | Biological_Sample)

list <- makeContrastsDream(model,
                           meta,
                           contrasts = c(Treatment_1 = "(TreatmentTreatment_1
- TreatmentControl)",
                                         Treatment_2 = "(TreatmentTreatment_2
- TreatmentControl)",
                                         Treatment_3 = "(TreatmentTreatment_3
- TreatmentControl)",

                                         Treatment_1_vs_Treatment_2 =
"(TreatmentTreatment_1 - TreatmentTreatment_2)",
                                         Treatment_1_vs_Treatment_3 =
"(TreatmentTreatment_1 - TreatmentTreatment_3)",

                                         Treatment_2_vs_Treatment_3 =
"(TreatmentTreatment_2 - TreatmentTreatment_3)"))
```

```
# Fit linear model using Dream
dream_fit <- dream(counts,
                   model,
                   meta,
                   list)
```

Calculate pathway differential expression p-values

```
# Empirical Bayes Statistics for Differential Expression
fit <- eBayes(dream_fit)

contrasts <- c('Treatment_1', 'Treatment_2', 'Treatment_3',
               'Treatment_1_vs_Treatment_2', 'Treatment_1_vs_Treatment_3',
               'Treatment_2_vs_Treatment_3')

# Calculate DEGs
DEGs <- lapply(contrasts, function(x){

  topTable(fit, coef = x, sort.by = "P", n = Inf)

})


# Results are the following information:

# logFC: log0 fold change of group1/group0
# AveExpr: Average expression across all samples, in log0 CPM
# t: logFC divided by its standard error
# P.Value: Raw p-value (based on t) from test that logFC differs from 0
# adj.P.Val: Benjamini-Hochberg false discovery rate adjusted p-value
# B: log-odds that gene is DE

# Isolate key DEG readouts
DEGs <- lapply(DEGs, function(x){

  tmp <- data.frame(cbind(x[["logFC"]],
                          x[["P.Value"]],
                          x[["adj.P.Val"]],
                          x["AveExpr"]),
                    row.names = rownames(x))

  colnames(tmp) <- c('log2FC', 'P.Value', 'adj.P.Val', 'Expression')
  tmp

})

names(DEGs) <- c('Treatment_1', 'Treatment_2', 'Treatment_3',
                 'Treatment_1_vs_Treatment_2', 'Treatment_1_vs_Treatment_3',
                 'Treatment_2_vs_Treatment_3')
```

```
list <- names(DEGs[1:3])
```

Isolate the top 20 pathway names for each comparison versus M0 and calculate mean GSVA scores by group

```
# Isolate the pathway names which are significantly differentially expressed
pathway_names <- lapply(list, function(x){

  tmp <- DEGs[[x]] %>% arrange(adj.P.Val) %>% subset(adj.P.Val < 0.05)
  tmp <- rownames(tmp)

  # If there are any pathways differentially expressed, isolate the top 20
  # Otherwise, mark the comparison with 'Remove'
  if(length(tmp) > 0){
    tmp <- tmp[1:20]
  }
  else{
    tmp <- 'Remove'
    }

})

# Remove comparisons without DE pathways
pathway_names <- pathway_names[pathway_names != 'Remove']
pathway_names <- as.list(unique(unlist(pathway_names)))
pathway_names <- pathway_names[pathway_names != 'NA'] # Remove NA results
pathway_names <- as.character(unique(unlist(pathway_names)))

# Isolate GSVA results for DE pathways
DEG_results <- GSVA_results[, pathway_names]
meta <- meta[rownames(DEG_results), ]
DEG_results <- data.frame(meta, DEG_results)

# Calculate mean GSVA scores
DEG_results <- DEG_results %>%
  group_by(Treatment) %>%
  summarize(across(everything(), mean))
```

```
## Warning: There were 4 warnings in `summarize()`.
## The first warning was:
## ℹ In argument: `across(everything(), mean)`.
## ℹ In group 1: `Treatment = Control`.
## Caused by warning in `mean.default()`:
## ! argument is not numeric or logical: returning NA
## ℹ Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.
```

```
DEG_results <- data.frame(DEG_results[3:ncol(DEG_results)], row.names =
DEG_results$Treatment)
colnames(DEG_results) <- pathway_names
```

```r
colnames(DEG_results) <- gsub('\\.', ' ', colnames(DEG_results)) %>%
stringr::str_to_title() # Reformat pathway names
```

Generate a heatmap of mean GSVA values for top pathways

```r
# Determine the optimal number of clusters
silhouette <- fviz_nbclust(t(DEG_results), cluster::pam, method =
'silhouette')

# Perform partition around medoid (PAM) clustering
JWriteBin(as.matrix(t(DEG_results)),
                        "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: 576 (300 really stored)
## Data type:          float
## Endianness:         little endian (same as this machine)
## Number of rows:     24
## Number of columns:  24
## 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)

# Set color palette
myCol <- colorRamp2(c(-1, 0, 1), hcl_palette = "Vik")

DEG_results <- DEG_results[c("Control", "Treatment_1", "Treatment_2",
"Treatment_3"), ]

hmap <- Heatmap(t(DEG_results),
```

```
                # split the rows according to the PAM clusters
                row_split = pam[["clasif"]],
                row_gap = unit(5, 'mm'),
                rect_gp = gpar(col = "black"),
                border = FALSE,
                width = nrow(DEG_results) * unit(50, 'mm'),
                height = ncol(DEG_results) * unit(15, 'mm'),

                name = 'GSVA\nEnrichment Score',

                col = myCol,

                # Add mean GSVA values
                # Change text color based on value
                cell_fun = function(j, i, x, y, width, height, fill) {
                  grid.text(sprintf("%.2f", t(DEG_results)[i, j]), x, y, gp =
gpar(fontsize = 20,

col = if_else(abs(t(DEG_results)[i, j]) > 0.5,

'white',

'black'),

fontface = 'bold'))},

                # parameters for the color-bar that represents gradient of
expression
                heatmap_legend_param = list(
                  color_bar = 'continuous',
                  at = c(-1, -0.5, 0, .5, 1),
                  legend_direction = 'horizontal',
                  legend_width = unit(12, 'cm'),
                  legend_height = unit(12, 'cm'),
                  title_position = 'topcenter',
                  title_gp = gpar(fontsize = 24, fontface = 'bold'),
                  labels_gp = gpar(fontsize = 20, fontface = 'bold')),

                # row (gene) parameters
                cluster_rows = TRUE,
                show_row_dend = TRUE,
                row_title = NULL,
                row_title_side = 'left',
                row_title_gp = gpar(fontsize = 12,  fontface = 'bold'),
                show_row_names = TRUE,
                row_names_gp = gpar(fontsize = 24, fontface = 'bold'),
                row_names_side = 'left',
                row_dend_width = unit(25,'mm'),
```

```
                # column (sample) parameters
                cluster_columns = TRUE,
                show_column_dend = TRUE,
                show_column_names = TRUE,
                column_title = NULL,
                column_names_gp = gpar(fontsize = 30, fontface = 'bold'),
                column_names_rot = 45,
                column_names_max_height = unit(20, 'cm'),
                column_dend_height = unit(25,'mm'))

png(file = "GSVA_DE_Heatmap.png", height = 3000, width = 3000)

draw(hmap,
     heatmap_legend_side = 'bottom',
     row_dend_side = 'right')

dev.off()

## png
##   2
```

Generate a heatmap indicating whether each pathway is differentially expressed between treatment groups

As the selected pathways are the top 20 differentially expressed pathways between single treatments and the control group, this allows us to determine if the GSVA values presented in the heatmap above are significantly different between treatment groups as well.

```
list <- as.list(names(DEGs)[4:6])

# Mark pathways as significant or non-significant DE
significant <- lapply(list, function(x){

  tmp <- DEGs[[x]][pathway_names, ]
  tmp$Sig <- if_else(tmp$adj.P.Val >= 0.05, 'Not_Significant', 'Significant')
  tmp$Sig

})

significant <- do.call(cbind, significant)
colnames(significant) <- list
rownames(significant) <- pathway_names
rownames(significant) <- gsub('\\.', ' ', rownames(significant)) %>%
stringr::str_to_title()
significant <- significant[colnames(DEG_results), ]

hmap <- Heatmap(significant,
```

```
                # split the rows according to the PAM clusters
                row_split = pam[["clasif"]],
                row_gap = unit(5, 'mm'),
                column_gap = unit(0, 'mm'),
                rect_gp = gpar(col = "black"),
                border = FALSE,
                width = nrow(DEG_results) * unit(50, 'mm'),
                height = ncol(DEG_results) * unit(15, 'mm'),

                col = c('tomato', 'lightblue'),

                name = ' ',

                # parameters for the color-bar that represents gradient of
expression
                heatmap_legend_param = list(
                  legend_width = unit(12, 'cm'),
                  legend_height = unit(12, 'cm'),
                  title_position = 'topcenter',
                  title_gp = gpar(fontsize = 24, fontface = 'bold'),
                  labels_gp = gpar(fontsize = 24, fontface = 'bold')),

                # row (gene) parameters
                cluster_rows = FALSE,
                show_row_dend = FALSE,
                row_title = NULL,
                row_title_side = 'left',
                row_title_gp = gpar(fontsize = 12,  fontface = 'bold'),
                show_row_names = TRUE,
                row_names_gp = gpar(fontsize = 24, fontface = 'bold'),
                row_names_side = 'right',
                row_dend_width = unit(25,'mm'),

                # column (sample) parameters
                cluster_columns = FALSE,
                cluster_column_slices = FALSE,
                show_column_dend = FALSE,
                show_column_names = TRUE,
                column_title = NULL,
                column_names_gp = gpar(fontsize = 30, fontface = 'bold'),
                column_names_rot = 45,
                column_names_max_height = unit(20, 'cm'),
                column_dend_height = unit(25,'mm'))

png(file = "GSVA_DE_Significance_Heatmap.png", height = 3000, width = 2500)

draw(hmap,
     heatmap_legend_side = 'bottom')
```

```
dev.off()
```

```
## png
##   2
```