}

Ingenuity Pathway Analysis Filtering

Timothy Smyth

2024-04-02

Ingenuity Pathway Analysis Filtering

Ingenuity pathway analysis (IPA) is a powerful tool for the analysis of differential expression data. Here, results of IPA are ingested and canonical pathway activity results are extracted and visualized. Similar methods can be employed to isolate and visualize other IPA readouts such as Master Regulators of pathways which can help identify key elemental, chemical, and biological agents which may impact pathway activity.

Prepare environment

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

# Load packages
library(tidyverse) # for data cleaning
library(dplyr)
library(formattable)
library(ComplexHeatmap)
library(circlize)
library(readxl)
library(parallelpam)
```

Import IPA results

IPA results can be downloaded as excel/csv files. This section imports all files in a single folder and names them accordingly.

```
wd = paste0(getwd(), '/Output/IPA Results')

# Create a list of the files from your target directory
file_list <- list.files(path = wd)

list <- list()

# Read each .csv and save to list
for (i in 1:length(file_list)){

  list[[i]] <- read_excel(paste0(wd,
                                 '/',
                                 file_list[i]),
                          col_types = 'text')
  }
```

```
# Set the names in the list to match the IPA files without .csv
names(list) <- substr(file_list,
                      1,
                      nchar(file_list) - 4)

# Remove the first 21 rows of each object which contains IPA metadata
list <- lapply(list, function(x){

  tmp <- data.frame(x[-c(1:21), ]) # Remove first 21 rows
  row.names(tmp) = seq(1, nrow(tmp), by = 1) # Set rownames to sequential 
order of 1 to nrow
  tmp

})
```

First, check to see if all of the normal IPA readouts are present in your files.

Sometimes, I have had specific readouts not present in my data. This checks if every subheading is present. If any are not present, manual filtering must be employed. If the same readout is missing from all data as with this data set, it can simply be removed from downstream analysis and run normally.

```
IPA_Results <- lapply(list, function(x){

  # ID the row and column number with the indicated text
  # These correspond to the IPA readouts
  Analysis <- which(x == 'Analysis', arr.ind = TRUE)
  Master_Regulator <- which(x == 'Master Regulator', arr.ind = TRUE)
  Categories <- which(x == 'Categories', arr.ind = TRUE)
  Consistency_Score <- which(x == 'Consistency Score', arr.ind = TRUE)
  ID <- which(x == 'ID', arr.ind = TRUE)
  Ingenuity_Toxicity_Lists <- which(x == 'Ingenuity Toxicity Lists', arr.ind 
= TRUE)
  ID2 <- as.numeric(ID[2]) # ID2 is the second indexed position of ID above

  # Put all indexed positions into one list
  x <- list(Analysis,
            Master_Regulator,
            Categories,
            Consistency_Score,
            ID,
            Ingenuity_Toxicity_Lists,
            ID2)

  # Rename list objects to match what it is
  names(x) <- c('Analysis',
                'Master_Regulator',
                'Categories',
                'Consistency_Score',
                'ID',
                'Ingenuity_Toxicity_Lists',
                'ID2')


  # Save the length of each object in the list
  y <- data.frame((lengths(x)))

  # Print any rownames in the list which have no indexed positions
  print(rownames(y %>% filter_all(any_vars(. %in% c(0)))))

  # Save list x to IPA_Results[[x]]
  x

})

## [1] "Consistency_Score"
## [1] "Consistency_Score"
## [1] "Consistency_Score"
## [1] "Consistency_Score"
## [1] "Consistency_Score"
## [1] "Consistency_Score"

# If any files have to be removed for manual filtering, add the name here
list <- list[list != '']
```

Segregate IPA readouts based on subheading

As consistency scores are not present in the IPA readouts for this example, we have to work around it below.

```
IPA_Results <- lapply(list, function(x){

  # ID the row and column number with the indicated text
  # These correspond to the IPA readouts
  Analysis <- which(x == 'Analysis', arr.ind = TRUE)
  Master_Regulator <- which(x == 'Master Regulator', arr.ind = TRUE)
  Categories <- which(x == 'Categories', arr.ind = TRUE)

  # Consistency_Score <- which(x == 'Consistency Score', arr.ind = TRUE)

  ID <- which(x == 'ID', arr.ind = TRUE)
  Ingenuity_Toxicity_Lists <- which(x == 'Ingenuity Toxicity Lists', arr.ind
= TRUE)

  # ID2 is the second indexed position of ID above
  ID2 <- as.numeric(ID[2])

  # Subset from row 1 to 2 rows before 'Analysis' text (last data point)
  Ingenuity_Canonical_Pathways <- x[c(1:as.numeric(Analysis[1]-
2)),c(1:ncol(x))]
```

```
  # Subset from first row matching 'Analysis' to 2 rows before 'Master
Regulator
  Analysis <- x[c(as.numeric(Analysis[1]):as.numeric(Master_Regulator[1]-2)),
c(1:ncol(x))]

  Master_Regulator <-
x[c(as.numeric(Master_Regulator[1]):as.numeric(Categories[1]-2)),
c(1:ncol(x))]

  # Categories <-
x[c(as.numeric(Categories[1]):as.numeric(Consistency_Score[1]-2)),
c(1:ncol(x))]
  Categories <- x[c(as.numeric(Categories[1]):as.numeric(ID[1]-2)),
c(1:ncol(x))]

  # Consistency_Score <-
x[c(as.numeric(Consistency_Score[1]):as.numeric(ID[1]-2)),c(1:ncol(x))]

  ID <- x[c(as.numeric(ID[1]):as.numeric(Ingenuity_Toxicity_Lists[1]-2)),
c(1:ncol(x))]

  Ingenuity_Toxicity_Lists <-
x[c(as.numeric(Ingenuity_Toxicity_Lists[1]):as.numeric(ID2-2)), c(1:ncol(x))]

  ID2 <- x[c(as.numeric(ID2):nrow(x)), c(1:ncol(x))]

  # Combine all isolated readouts into one list
  tmp <- list(Ingenuity_Canonical_Pathways,
              Analysis,
              Master_Regulator,
              Categories,
              # Consistency_Score,
              ID,
              Ingenuity_Toxicity_Lists,
              ID2)

  # Rename list objects to match what it is
  names(tmp) <- c('Ingenuity_Canonical_Pathways',
                  'Analysis',
                  'Master_Regulator',
                  'Categories',
                  # 'Consistency_Score',
                  'ID',
                  'Ingenuity_Toxicity_Lists',
                  'ID2')

  # Organize each object from the list
  tmp <- lapply(names(tmp), function(y){
    names(tmp[[y]]) <- lapply(tmp[[y]][1, ], as.character) # Set colnames of
object to first row of object
    tmp[[y]] <- as.data.frame(tmp[[y]][-1, ]) # Remove first row, which
contained the colnames
    keep.cols <- names(tmp[[y]]) %in% c("") # Keep columns with names
(contain data)
    tmp[[y]] <- tmp[[y]][!keep.cols] # Keep columns with names (contain data)
    row.names(tmp[[y]]) = seq(1, nrow(tmp[[y]]), by = 1) # Set rownames to 1
to nrow in sequential order
    tmp[[y]] # Save modified object to list

  })

  # Rename list objects to match what it is
  names(tmp) <- c('Ingenuity_Canonical_Pathways',
                  'Analysis',
                  'Master_Regulator',
                  'Categories',
                 # 'Consistency_Score',
                  'ID',
                  'Ingenuity_Toxicity_Lists',
                  'ID2')

  tmp

})
```

Isolate the top canonical pathways for each group

This section assumes you plan to analyze and visualize the canonical pathway results. Other sections can be analyzed, but the section below will have to be modified to isolate the correct columns, filter on different metrics, and break up data based on activity.

tmp <- x[[1]] = 'Ingenuity_Canonical_Pathways' tmp <- x[[2]] = 'Analysis' tmp <- x[[3]] = 'Master_Regulator' tmp <- x[[4]] = 'Categories' tmp <- x[[5]] = 'Consistency_Score' tmp <- x[[6]] = 'ID' tmp <- x[[7]] = 'Ingenuity_Toxicity_Lists' tmp <- x[[8]] = 'ID2'

Some data sets can yield dozens of significantly impacted canonical pathways. Here, you can isolate the top 10 pathways by absolute z-score from each group instead for easier visualization of results.

```
# # If any csv files had to be manually sorted, import it here
# sorted <- read.csv('.csv')
#
# # Remake IPA_Results list
# # Here, IPA_Results[[x]][[y]] corresponds to data frame x, readout y (as
described above)
# IPA_Results <- list(IPA_Results[[1]][[1]],
#                      # sorted,
```

```r
#                      IPA_Results[[2]][[1]],
#                      IPA_Results[[3]][[1]],
#                      IPA_Results[[4]][[1]],
#                      IPA_Results[[5]][[1]]
#                      # ...
#                      IPA_Results[[n]][[1]])

Results <- lapply(names(IPA_Results), function(x){

  # Isolate the first position of the IPA results (Canonical Pathways)
  tmp <- IPA_Results[[x]][[1]]

  colnames(tmp) <- c('Pathway',
                     'Neg.log.10.p',
                     'zScore',
                     'Ratio')

  # Set to numeric
  tmp$zScore <- as.numeric(tmp$zScore)
  tmp$Neg.log.10.p <- as.numeric(tmp$Neg.log.10.p)
  tmp$Ratio <- as.numeric(tmp$Ratio)

  # Round to 3 digits
  tmp$zScore <- round(tmp$zScore, 3)

  # Reorganize data and keep only significantly different pathways
  tmp <- tmp[, c(1, 3, 4, 2)] %>%
    subset(tmp$Neg.log.10.p >= 1.3)

  tmp <- tmp %>% arrange(desc(abs(zScore)))

  # Set names of columns
  colnames(tmp) <- c('Pathway',
                     paste0(as.character(x), ' zScore'),
                     paste0(as.character(x), ' Ratio'),
                     paste0(as.character(x), ' Neg.log.10.p'))

  # Save the results to Canonical list
  tmp[1:10, ]

})

# Set names of list object to corresponding source
names(Results) <- substr(file_list,
                         1,
                         nchar(file_list) - 4)
```

With the top 10 pathways for each comparison, we can merge them together and visualize the activation z-scores between groups.

Since this is an artificial data set with a limited number of input genes, a large number of pathways have NA z-scores across multiple groups. To make visualization a little bit more interesting, NA values are replaced with random values. If you are working with a real data set, be sure to remove this step.

```
# Merge results together
zScore <- Reduce(function(x, y) merge(x, y, all = TRUE), Results)
zScore <- data.frame(zScore[, c(2, 5, 8, 11, 14, 17)], row.names =
zScore$Pathway)

# Replace NA z-score values with random values
# If you have a real data set, remove this step!
zScore[is.na(zScore)] <- sample(seq(-4, 4, by = 0.1), sum(is.na(zScore)),
replace = TRUE) %>% round(2)

# Remove groups with no variance between comparisons
Remove <- zScore %>% filter_all(all_vars(is.na(.))) %>% rownames()
zScore <- zScore[!(row.names(zScore) %in% Remove), ]

var <- as.data.frame(sapply(as.data.frame(t(zScore)), var))
var <- var %>% subset(var[, 1] == 0)

zScore <- zScore[!(row.names(zScore) %in% row.names(var)), ]

# Rename and reorder columns if desired
colnames(zScore) <- names(Results)

# zScore <- zScore[, c()]

# Run partition around medoid clustering to create clusters by pathway
similarities
parallelpam::JWriteBin(as.matrix(zScore),
                       "group_pam.bin",
                       dtype="float",
                       dmtype="full")

CalcAndWriteDissimilarityMatrix("group_pam.bin",
                                "group_pam2.bin",
                                distype = "L2",
                                restype = "float",
                                comment = "L2 distance for vectors in jmatrix
file vst_results_group.bin",
                                nthreads = 8) # 8 threads on desktop, 16 on
laptop

## Loading required package: memuse
```

```r
JMatInfo("group_pam2.bin")

## File:                 group_pam2.bin
## Matrix type:          SymmetricMatrix
## Number of elements: 729 (378 really stored)
## Data type:            float
## Endianness:           little endian (same as this machine)
## Number of rows:       27
## Number of columns:    27
## Metadata:             Stored only names of rows.
## Metadata comment:   "L2 distance for vectors in jmatrix file
vst_results_group.bin"

group_pam = ApplyPAM("group_pam2.bin",
                     k = 4,
                     init_method = "BUILD",
                     max_iter = 1000,
                     nthreads = 8)

# Set color scheme and breaks
myCol <- colorRamp2(c(-5, 0, 5), hcl_palette = "Vik")

hmap1 <- Heatmap(as.matrix(zScore),

                 # Split the rows according to the PAM clusters
                 split = group_pam$clasif,
                 cluster_row_slices = TRUE,
                 gap = unit(2, "mm"),
                 border = TRUE,
                 width = ncol(zScore) * unit(50, 'mm'),
                 height = ncol(zScore) * unit(100, 'mm'),

                 name = 'Canonical Pathway\nZ-Score',

                 # Set the color scheme
                 col = myCol,

                 # Create a color for NA values if present
                 na_col = "pink",

                 # Add Z-score values to cells
                 # If z-scores are above or below certain values,
                 # change the color of the text to make the text more visible
                 cell_fun = function(j, i, x, y, width, height, fill) {
                   grid.text(sprintf("%.1f", zScore[i, j]), x, y, gp =
gpar(fontsize = 24,

col = if_else(zScore[i, j] < 3 &

zScore[i, j] > -3 |
is.na(zScore[i, j]) == TRUE,

'black',

'white'),

fontface = 'bold'))},

                 # parameters for the color-bar that represents gradient of 
expression
                 heatmap_legend_param = list(
                   color_bar = 'continuous',
                   at = c(-5, -2.5, 0, 2.5, 5),
                   legend_direction = 'horizontal',
                   legend_width = unit(15, 'cm'),
                   legend_height = unit(10.0, '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_side = 'right',
                 row_title_gp = gpar(fontsize = 12,  fontface = 'bold'),
                 show_row_names = TRUE,
                 row_names_gp = gpar(fontsize = 20, fontface = 'bold'),
                 row_names_side = 'right',
                 row_dend_width = unit(25,'mm'),

                 # column (sample) parameters
                 cluster_columns = TRUE,
                 show_column_dend = TRUE,
                 column_title = '',
                 column_title_side = 'bottom',
                 column_title_gp = gpar(fontsize = 12, fontface = 'bold'),
                 show_column_names = TRUE,
                 column_names_gp = gpar(fontsize = 30, fontface = 'bold'),
                 column_names_rot = 45,
                 column_names_max_height = unit(10, 'cm'),
                 column_dend_height = unit(25,'mm'))

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

draw(hmap1,
     heatmap_legend_side = 'bottom',
     annotation_legend_side = 'right')
```

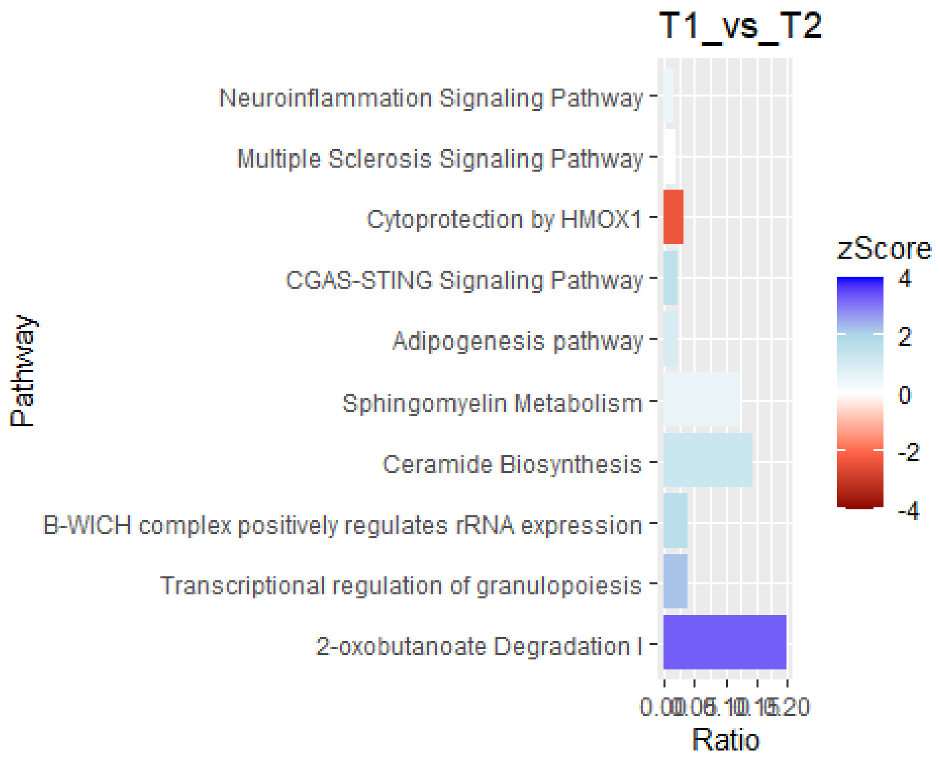
```
dev.off()

## png
##   2
```

Another way to visualize the pathway results are to make bar graphs for each comparison. This gives the added advantage of presenting two metrics (any combination of zScore, gene ratio, and p-value) by placing one on the axis and using one to fill the bar.

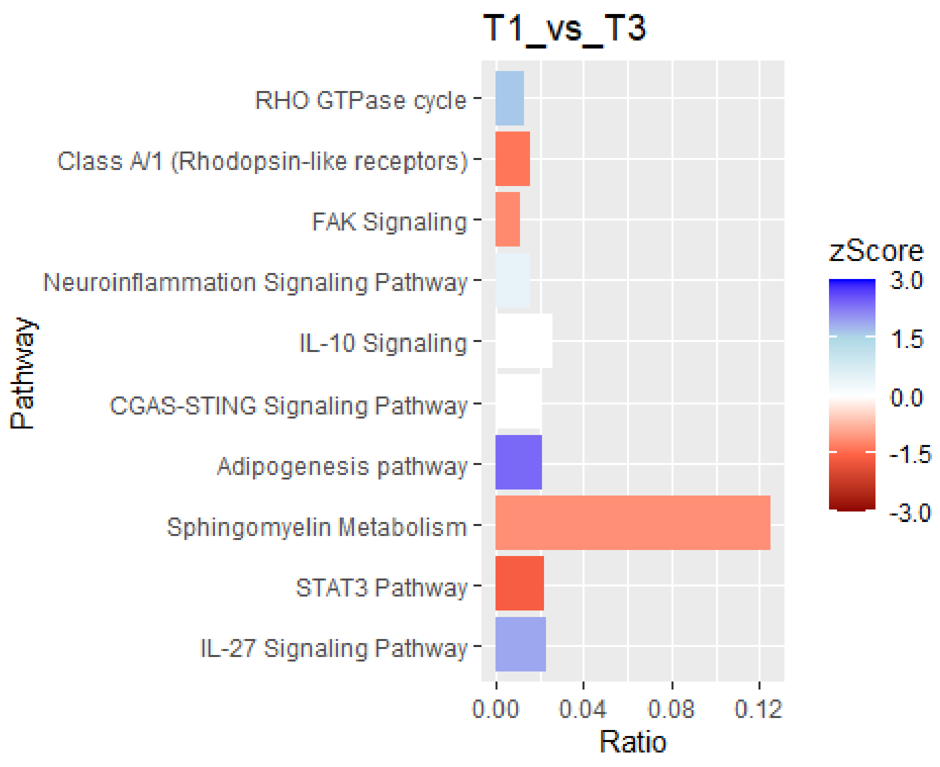
```
lapply(names(Results), function(x){

  tmp <- Results[[x]]

  names(tmp) <- c('Pathway', 'zScore', 'Ratio', 'Neg.log.10.p')

  tmp$zScore <- zScore[tmp$Pathway, x]

  max <- ceiling(max(abs(tmp$zScore)))

  tmp$Pathway <- factor(tmp$Pathway,
                        levels = tmp$Pathway)

  plot <- ggplot(tmp,
                 aes(x = Ratio,
                     y = Pathway,
                     fill = zScore)) +

    geom_bar(stat = 'identity') +

    scale_fill_gradientn(
      colours = c("darkred", "tomato", "white", "lightblue", "blue"),
      breaks = c(-max, -max/2, 0, max/2, max),
      limits = c(-max, max)) +

    scale_y_discrete(limits = rev) +

    labs(title = paste0(as.character(x)))

})

## [[1]]
```

T1_vs_
IL-10 Signaling
O-linked glycosylation
Cytoprotection by HMOX1
Role of JAK2 in Hormone-like Cytokine Signaling
Adipogenesis pathway
Sphingomyelin Metabolism
EIF2 Signaling
SNARE Signaling Pathway
Role of Osteoblasts, Osteoclasts and Chondrocytes in Rheumatoid Arthritis
STAT3 Pathway
0.00
Ratio
zScor
4
2
0
-2
-4

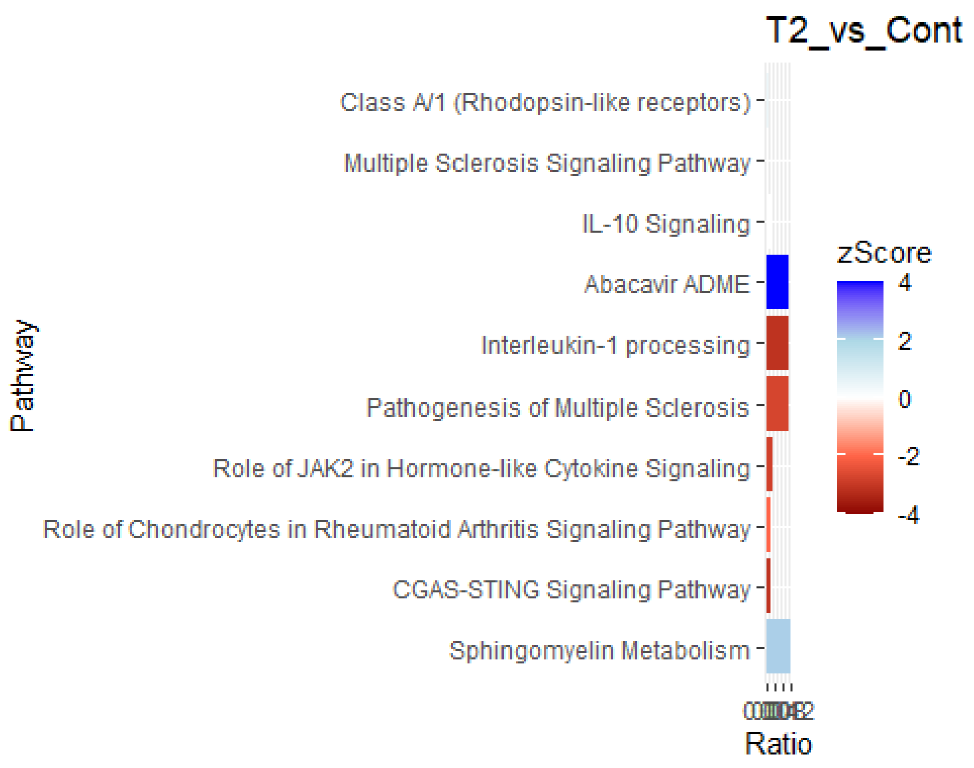
```
##
## [[2]]
```


T1_vs_T2
Pathway
Neuroinflammation Signaling Pathway
Multiple Sclerosis Signaling Pathway
Cytoprotection by HMOX1
CGAS-STING Signaling Pathway
Adipogenesis pathway
Sphingomyelin Metabolism
Ceramide Biosynthesis
B-WICH complex positively regulates rRNA expression
Transcriptional regulation of granulopoiesis
2-oxobutanoate Degradation I
0.00 0.05 0.10 0.15 0.20
Ratio
zScore
4
2
0
-2
-4

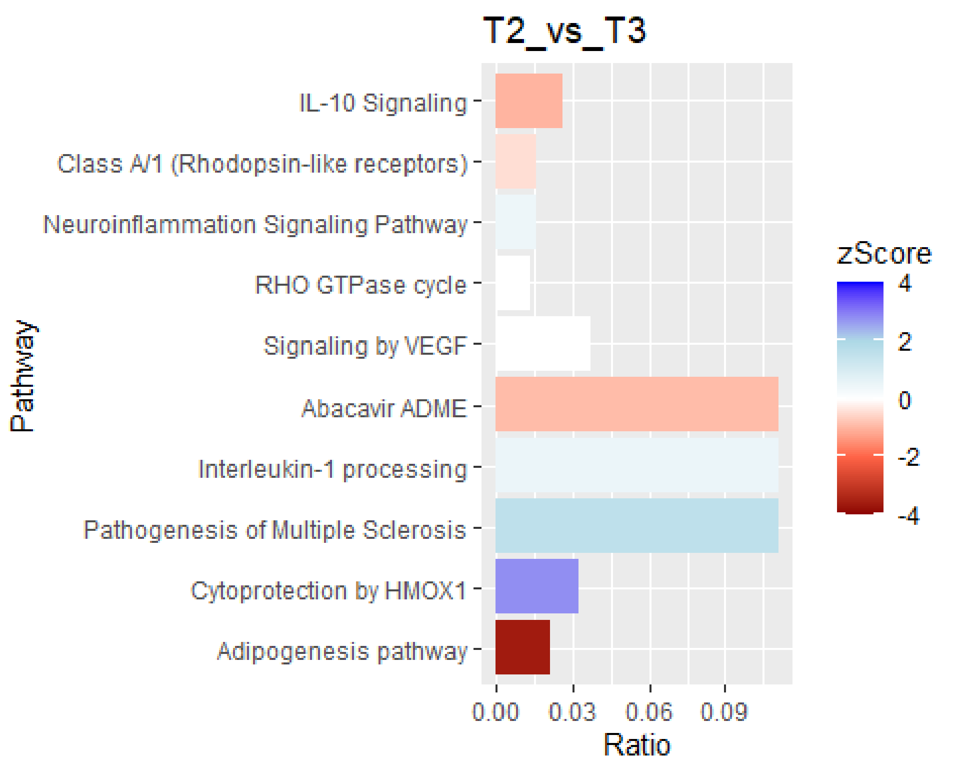
```
##
## [[3]]
```


T1_vs_T3
RHO GTPase cycle
Class A/1 (Rhodopsin-like receptors)
FAK Signaling
Neuroinflammation Signaling Pathway
IL-10 Signaling
CGAS-STING Signaling Pathway
Adipogenesis pathway
Sphingomyelin Metabolism
STAT3 Pathway
IL-27 Signaling Pathway
Pathway
0.00
0.04
0.08
0.12
Ratio
zScore
3.0
1.5
0.0
-1.5
-3.0

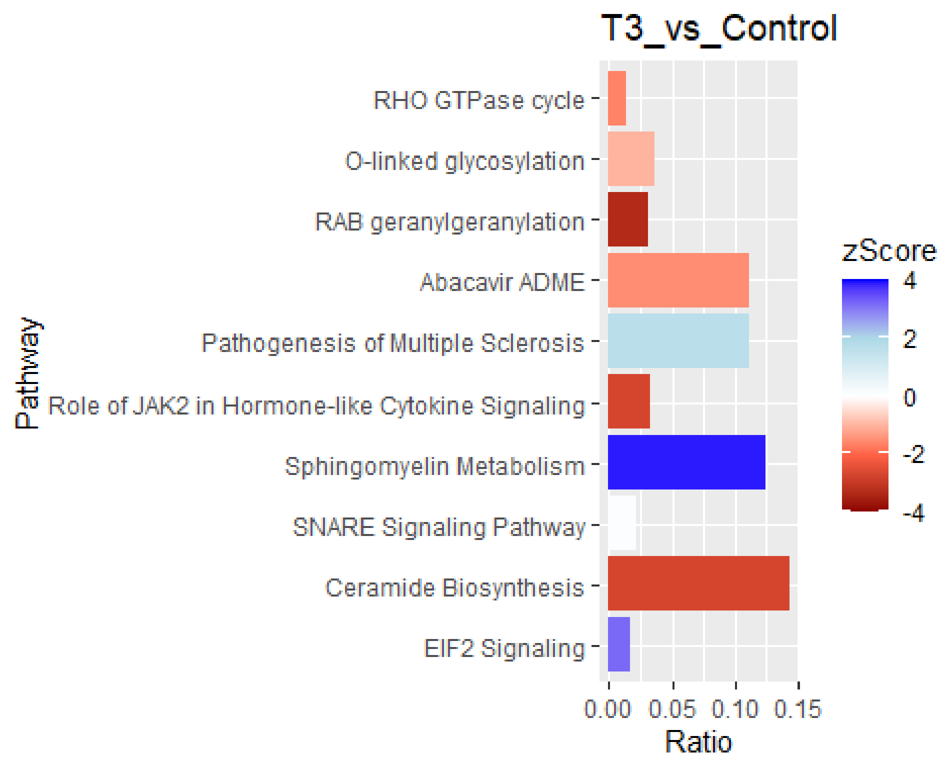
```
##
## [[4]]
```


T2_vs_Cont
Pathway
Class A/1 (Rhodopsin-like receptors)
Multiple Sclerosis Signaling Pathway
IL-10 Signaling
Abacavir ADME
Interleukin-1 processing
Pathogenesis of Multiple Sclerosis
Role of JAK2 in Hormone-like Cytokine Signaling
Role of Chondrocytes in Rheumatoid Arthritis Signaling Pathway
CGAS-STING Signaling Pathway
Sphingomyelin Metabolism
Ratio
zScore
4
2
0
-2
-4

```
##
## [[5]]
```


T2_vs_T3
Pathway
IL-10 Signaling
Class A/1 (Rhodopsin-like receptors)
Neuroinflammation Signaling Pathway
RHO GTPase cycle
Signaling by VEGF
Abacavir ADME
Interleukin-1 processing
Pathogenesis of Multiple Sclerosis
Cytoprotection by HMOX1
Adipogenesis pathway
0.00
0.03
0.06
0.09
Ratio
zScore
4
2
0
-2
-4

```
##
## [[6]]
```


T3_vs_Control
Pathway
RHO GTPase cycle
O-linked glycosylation
RAB geranylgeranylation
Abacavir ADME
Pathogenesis of Multiple Sclerosis
Role of JAK2 in Hormone-like Cytokine Signaling
Sphingomyelin Metabolism
SNARE Signaling Pathway
Ceramide Biosynthesis
EIF2 Signaling
0.00
0.05
0.10
0.15
Ratio
zScore
4
2
0
-2
-4