Bulk RNA seq - UpSet Plot

Brandon Hancock

1/22/2021

This page shows how pca plot and heatmaps were generated for this paper

The dataset can be downloaded from this link

Load the required R packages:

```
library(readxl)
library(readr)
library(DESeq2)
library(ggplot2)
library(mygene)
library(UpSetR)
library(hash)
library(sjmisc)
```

Load in the Bulk RNA seq STAR gene counts file:

```
STAR_gene_counts <- read_csv("C:/Users/bh719/Dropbox (Partners HealthCare)/Harvard CyTof/for Brandon/Sally/STAR_Gene_Counts.csv")
```

Remove genes with duplicate entries (1-Mar and 2-Mar)

```
STAR_gene_counts <- STAR_gene_counts[!duplicated(STAR_gene_counts$Gene_ID),]
```

Define Meta Data

```
colmetadat <- data.frame(Injury = c(rep("Uninjured",9),rep("7D after Injury",10)),CD44 = c(rep("CD44 High",4),rep("CD44 Low",5),rep("CD44 High",4),rep("CD44 Low",6)))
row.names(colmetadat) <- colnames(STAR_gene_counts)[2:length(colnames(STAR_gene_counts))]
```

Define DESeq2 matrix

```
gene_row <- STAR_gene_counts$Gene_ID
cmat <- STAR_gene_counts
cmat <- cmat[,!(names(cmat) %in% c('Gene_ID'))]
row.names(cmat) <- gene_row
```

Create DESeq2 object

```
dds <- DESeqDataSetFromMatrix(countData = cmat,colData = colmetadat,design = ~ CD44 + Injury)
dds <- dds[rowSums(counts(dds)) >= 10,]
dds$group <- factor(paste0(dds$CD44,dds$Injury))
design(dds) <- ~ group
```

Run vst

```
vsd <- vst(dds,blind = FALSE)
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

Define function to calculate variance (stabilized, from vst) of each gene

```
f_get_var <- function(vsd){
  var_list <- c()
  gene_ids <- row.names(vsd)
  for (i in 1:length(gene_ids)){
    gene_row <- as.vector(vsd[gene_ids[i],]) #as vector is a disapointment
    gene_vec <- c()
    for (j in 1:length(gene_row)){
      gene_vec <- c(gene_vec,gene_row[[j]])
    }
    var_list[gene_ids[i]] <- var(gene_vec)
  }
  return(var_list)
}
```

Get the 2000 genes with the highest variance

```
var_list <- f_get_var(assay(vsd))
var_list <- var_list[order(var_list,decreasing = TRUE)]
gene_list <- head(names(var_list),2000)
```

Pull Gene Ontology (GO) Data

```
library(mygene)

res <- queryMany(gene_list,scopes = 'symbol', fields=c('entrezgene','ensembl.gene','go','description'),species = 'mouse')
```

```
## Querying chunk 1
```

```
## Querying chunk 2
```

```
## Finished
## Pass returnall=TRUE to return lists of duplicate or missing query terms.
```

```
res <- res[!duplicated(res$query),]
```

Define and run function to create a list mapping of genes to GO terms

```
f_getBP <- function(res,gene){
  BP <- res[which(res$query == gene),]$go.BP[[1]]
  return(BP)
}

f_genes_term_map <- function(res,genes){
  genes_term_map <- list()
  for (i in 1:length(genes)){
    gene_terms <- c(f_getBP(res,genes[i])$term)
    genes_term_map[[genes[i]]] <- gene_terms
  }
  return(genes_term_map)
}

genes_term_map <- f_genes_term_map(res,gene_list)
```

Define function to provide mapping of custom GO categories for GO terms

```
f_upset_dic <- function(){
  upset_dic <- hash()
  upset_dic[['Phosphorylation']] <- c('Phosphorylation','MAPK')
  upset_dic[['Immune Response']] <- c('immune response','inflammatory response','defense response', 't cell','T cell activation','leukocyte','lymphocyte',' t cell','mast cell','b cell','monocyte','interleukin','response to bacterium','cellular response to lipopolysaccharide','immune system','toll-like receptor')
  upset_dic[['Cell Cycle']] <- c('cell cycle','cell proliferation','cell differentiation','cell division','MAPK cascade','ERK1 and ERK2','chromosome segregation','Ras protein signal transduction','DNA replication','spindle organization','chromosome organization')
  upset_dic[['Transcription']] <- c('transcription','gene expression')
  upset_dic[['Signaling']] <- c('Intracellular Signal','signal transduction','signaling',upset_dic[['Cytokine']],upset_dic[['Phosphorylation']],upset_dic[['Cytokine']],'GTPase')
  upset_dic[['Cell Movement']] <- c('taxis', 'chemotaxis', 'migration', 'motility')
  upset_dic[['Translation']] <- c('translation')

  return(upset_dic)
}
```

Define function to map gene GO terms to custom categories

```
check_func <- function(func,terms,upset_dic){
  for (j in 1:length(terms)){
    t <- terms[j]
    if (any(str_contains(t,func,ignore.case = TRUE))){
      return(TRUE)
    }
  }
  return(FALSE)
}

f_genes_func_map <- function(genes_term_map,upset_dic){
  genes_func_map <- list()
  upset_funcs <- keys(upset_dic)
  for (i in 1:length(genes_term_map)){
    gene_upset_funcs <- c()
    for (j in 1:length(upset_funcs)){
      sterm <- upset_dic[[upset_funcs[[j]]]]
      if (check_func(sterm,genes_term_map[[i]])){
        gene_upset_funcs <- c(gene_upset_funcs, upset_funcs[[j]])
      }
    }
    genes_func_map[[names(genes_term_map)[i]]] <- gene_upset_funcs
  }
  return(genes_func_map)
}
```

Create the gene to function map

```
genes_func_map <- f_genes_func_map(genes_term_map,f_upset_dic())
```

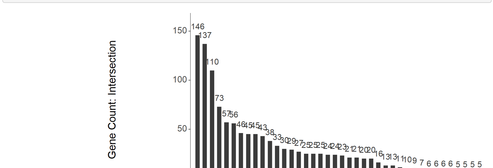
Define function to change format, GO Category: Gene1, Gene 2 ...

```
f_listInput <- function(genes_func_map){
  listInput <- list()
  upset_funcs <- names(f_upset_dic())
  for (i in 1:length(upset_funcs)){
    listInput[upset_funcs[[i]]] <- c()
  }
  for (i in 1:length(upset_funcs)){
    for (j in 1:length(genes_func_map)){
      if (upset_funcs[[i]] %in% genes_func_map[[j]]){
        listInput[[upset_funcs[[i]]]] <- c(listInput[[upset_funcs[[i]]]], names(genes_func_map)[j])
      }
    }
  }
  return(listInput)
}

listInput <- f_listInput(genes_func_map)
```

Plot UpSet

```
upset(fromList(listInput), order.by = "freq",mainbar.y.label = 'Gene Count: Intersection',sets.x.label = 'Gene Count: Gene Ontology',nsets = 9,set_size.scale_max = 1000,set_size.show = TRUE,set_size.angles = 0,text.scale = 1.9)
```



Session Info

```
installed.packages()[names(sessionInfo()$otherPkgs), "Version"]
```

```
##              sjmisc                hash              UpSetR
##             "2.8.7"              "2.2.6.1"              "1.4.0"
##             mygene       GenomicFeatures       AnnotationDbi
##            "1.26.0"              "1.42.3"             "1.52.0"
##             ggplot2                DESeq2 SummarizedExperiment
##             "3.3.3"              "1.30.1"             "1.20.0"
##             Biobase       MatrixGenerics          matrixStats
##             "2.50.0"              "1.2.1"             "0.58.0"
##      GenomicRanges       GenomeInfoDb             IRanges
##            "1.42.0"              "1.26.7"             "2.24.1"
##          S4Vectors         BiocGenerics                readr
##             "0.28.1"              "0.36.1"             "1.4.0"
##              readxl
##             "1.3.1"
```