CODE

Supplementary Protocol 4 – perform manual phenotype randomizations using edgeR followed by GSEA and EM

Ruth Isserlin

2018-10-10

1 Supplementary Protocol 4 – perform manual phenotype randomizations using edgeR followed by GSEA and EM

This protocol demonstrates how to code manual phenotype permutation for RNA-seq data analyzed with edgeR and GSEA.

GSEA calculates for each tested gene-set an enrichment score (ES) which quantifies the enrichment of the gene set. A p-value is needed to estimate if this enrichment (ES) is greater than one that could be obtained by chance only. There are two ways to estimate the p-value and FDR within GSEA: gene-set permutation and phenotype permutation.

First, available as gene set permutation in GSEA,a null distribution can be constructed by randomly shuffling the genes in a given gene-sets: each time a shuffled gene-set is tested an ES is calculated. Then the observed ES for an individual gene set is compared to all the ES scores obtained for the same gene set across all the permutations to obtain an estimate of the gene set permutation FDR value.

An alternate way, available as phenotype permutation in GSEA, is to shuffle the sample labels that constitute the 2 groups to be compared. After shuffling the sample labels differential expression is recalculated. With a random shuffle of the samples not many genes should be found significantly differentially expressed and as a result not many pathways should be found significantly enriched. For each tested gene set, the observed ES is compared to all ES obtained with the shuffled samples and an FDR is estimated.

Note about phenotype permutation: In the GSEA interface, there is one option to upload an entire expression matrix and a class definition file (.cls) and perform the phenotype permutation. However the choice of tests are limited to basic statistical measures (signal to noise, t-test) and tests using non negative binomial distribution or linear models such as the ones implemented in edgeR and DEseq and recommended for RNA-seq counts data are not available. Therefore, in this protocol, we are performing the permutations outside GSEA. Differential expression is calculated within edgeR between the 'immunoreactive' and 'mesenchymal' type and GSEA is run to get the observed ES. Then, 1000 permutations are computed by shuffling the samples. Each permutation run consists of :1) shuffling the samples which results in the formation of 2 random groups. 2) calculate differential expression within edgeR between these 2 groups 3) run GSEA and obtain the ES and NES for this random permutation. Finally, for each gene-set, the p-value is calculated by counting the number of times the observed ES was greater than the ES obtained from the random permutations.

NOTE: Performing this randomization with R is very resource intensive and time consuming. (On a intel i7 computer with 16GB 1000 permutations took more than 9.5 hours)

1.1 Process RNA-seq data

This part of the supplementary protocol demonstrates filtering and scoring RNA-seq data using normalized RNA-seq count data with the edgeR R package. The section prepares the input data for GSEA.

1.1.1 Load required packages

1. Load required Bioconductor packages into R.

HIDE

```
knitr::opts_knit$set(cache = TRUE)

tryCatch(expr = { library("edgeR")},
         error = function(e) {
           source("https://bioconductor.org/biocLite.R")
           biocLite("edgeR")},
         finally = library("edgeR"))

working_dir <- file.path(".","data")
rand_working_dir <- file.path(".","data", "Randomizations_round3")

#The field in the class definition file that defines the classes of the data.
data_classes <- "SUBTYPE"

analysis_name <- "Mesen_Vs_Immuno"
```

1.1.2 Load Expression Data

2. Load the expression dataset of 296 tumours, with 79 classified as Immunoreactive, 71 classified as Mesenchymal, 67 classified as Differentiated, and 79 classified as Proliferative samples. The TCGA counts data was retrieved from the GDC[1] and contained counts per mRNA transcript determined using the RSEM method for 19947 transcripts and 300 samples.

HIDE

```
RNASeq <- read.table(
  file.path(working_dir,"Supplementary_Table10_TCGA_RNASeq_rawcounts.txt"),
  header = TRUE, sep = "\t", quote="\"", stringsAsFactors = FALSE)
```

1.1.3 Load subtype information

3. Load subtype classification of samples. To calculate differential expression, we need to define at least two sample classes. A common experimental design involves cases and controls but any two classes can be used. The current dataset is divided into Mesenchymal and Immunoreactive classes (class definitions were obtained from Verhaak et al.[2] Supplementary Table 1, third column). After loading the matrix, check that the column names of the expression matrix and class definitions are equal.

HIDE

```
classDefinitions_RNASeq <- read.table(
  file.path(working_dir, "Supplementary_Table11_RNASeq_classdefinitions.txt"),
  header = TRUE, sep = "\t", quote="\"", stringsAsFactors = FALSE)
```

1.1.4 Filter Data

4. Filter RNA-seq reads. RNA-seq data are processed following the edgeR protocol that filters reads based on the counts per million (CPM) statistic. RNA-seq read counts are converted to CPM

values and genes with CPM > 1 in at least 50 of the samples are retained for further study (a gene must have at least 50 measurements with more than 1 CPM in one of the classes to be included in the analysis). This step removes genes with very low read counts that are likely not expressed in the majority of samples and cause noise in the data. Note, CPM filtering is used to remove low counts while differential gene expression analysis is based on normalized read counts which are generated below (step 6).

HIDE

```
cpms <- cpm(RNASeq)
keep <- rowSums(cpms > 1) >= 50
counts <- RNASeq[keep,]
```

1.1.5 Normalization and Dispersion

5. Data normalization, dispersion analysis is performed on the entire dataset. Created MDS-plot of all patient samples can be seen in Figure 1.

HIDE

```
# create data structure to hold counts and subtype information for each sample.
d <- DGEList(counts=counts, group=classDefinitions_RNASeq$SUBTYPE)

#Normalize the data
d <- calcNormFactors(d)

#create multidimensional scaling(MDS) plot.  The command below will automatically
# generate the plot containing all samples where each subtype is a different color.
#Ideally there should be a good separation between the different classes.
mds_filename <- file.path(working_dir, "mdsplot_allsamples.png")
png(filename = mds_filename)
mds_output <- plotMDS(d, labels=NULL, pch = 1,
col= c("darkgreen","blue","red", "orange")[factor(classDefinitions_RNASeq$SUBTYPE)],
xlim = c(-2.5,4), ylim = c(-2.5,4))


legend("topright",
       legend=levels(factor(classDefinitions_RNASeq$SUBTYPE)),
       pch=c(1), col= c("darkgreen","blue","red", "orange"),title="Class",
       bty = 'n', cex = 0.75)

dev.off()

#calculate dispersion
d <- estimateCommonDisp(d)
d <- estimateTagwiseDisp(d)
```

1.1.6 Filter unannotated genes

6. (Optional) Exclude genes with missing symbols or uncharacterized genes. In this example gene entries in the dataset containing '?' or starting with LOC are excluded as they represent non-annotated genes or other loci that are not present in pathway databases. The frequency of these and other non protein coding entries in your dataset will depend on the database used to align your RNA-seq data.

HIDE

```
#the below regular expression excludes gene names that are ? or that start with LOC
# any number of additional terms can be added to the regular expresion, for example
# to exclude any genes that start with "His" add |^His to the regular expression
exclude <- grep("\\?|^LOC", rownames(d), value=T)
d <- d[which(!rownames(d) %in% exclude),]
```

1.1.7 Calculate Differential expression

7. Differential expression analysis is performed with a simple design as described in the edgeR protocol[3].

HIDE

```
#calculate differential expression statistics with a simple design
de <- exactTest(d, pair=c("Immunoreactive","Mesenchymal"))
tt_exact_test <- topTags(de,n=nrow(d))

#alternately you can also use the glm model using contrasts.
#For a simple 2 class comparison this is not required but if you want to compare
# 1 class to the remaining 3 classes then this sort of model is useful.
classes <- factor(classDefinitions_RNASeq[,data_classes])
modelDesign <- model.matrix(~ 0 + classes)

contrast_mesenvsimmuno <- makeContrasts(
                  mesenvsimmuno ="classesMesenchymal-classesImmunoreactive",
                  levels=modelDesign)
fit_glm <- glmFit(d,modelDesign)
mesenvsimmuno <- glmLRT(fit_glm , contrast = contrast_mesenvsimmuno)
tt_mesenvsimmuno <- topTags(mesenvsimmuno,n=nrow(d))

tt <- tt_exact_test
#calculate ranks
ranks_RNAseq = sign(tt$table$logFC) * -log10(tt$table$PValue)

#gene names from the TCGA set contain gene name and entrez gene ids separated by '|'
# for all subsequent enrichment analysis we need to have just one id.  Separate the name
s
# into their two ids.
genenames <- unlist(lapply( rownames(tt$table), function(data)
  {unlist(strsplit(data,"\\|"))[1]}))
geneids <- unlist(lapply( rownames(tt$table),  function(data)
  {unlist(strsplit(data,"\\|"))[2]}))

#create ranks file
ranks_RNAseq <- cbind(genenames, ranks_RNAseq)
colnames(ranks_RNAseq) <- c("GeneName","rank")

#sort ranks in decreasing order
ranks_RNAseq <- ranks_RNAseq[order(as.numeric(ranks_RNAseq[,2]),decreasing = TRUE),]
rnk_file <- file.path(rand_working_dir,"MesenvsImmuno_RNASeq_ranks.rnk")
write.table(ranks_RNAseq, rnk_file,
            col.name = TRUE, sep="\t", row.names = FALSE, quote = FALSE)
```

1.1.8 Run GSEA

HIDE

```
#path to GSEA jar
# In order to run GSEA automatically you need to speciry the path to the gsea jar file.
gsea_jar <- "./gsea-3.0.jar"

#Gsea takes a long time to run.  If you have already run GSEA manually or previously
# there is no need to re-run GSEA.  Make sure the gsea results are in the current
# directory and the notebook will be able to find them and use them.
run_gsea = TRUE

# leave blank if you want the notebook to discover the gsea directory for itself
#gsea_directory = paste(working_dir,"Mesen_vs_Immuno.GseaPreranked.1497635459262",sep
="/")
gsea_directory = ""

#TODO: change this to update to the latest gmt file.
dest_gmt_file <- file.path(working_dir,
          "Supplementary_Table3_Human_GOBP_AllPathways_no_GO_iea_July_01_2017_symbol.gm
t" )

num_randomizations <- 1000
```

8. Run GSEA with gene set randomization

Even though we are going to perform our own randomization, we still need to run GSEA to get the observed ES. In addition we will use the FDR from this gene randomization run to compare with our manual phenotype permutations.

HIDE

```
timestamp()
start_gs_perm <- Sys.time()
if(run_gsea){
  command <- paste("java  -Xmx1G -cp",gsea_jar,  "xtools.gsea.GseaPreranked -gmx",
      dest_gmt_file, "-rnk" ,rnk_file,
      "-collapse false -nperm ",num_randomizations,
      " -permute gene_set -scoring_scheme weighted -rpt_label ",
      paste(analysis_name,"gsrand",sep="_"),
" -num 100 -plot_top_x 20 -rnd_seed 12345  -set_max 200 -set_min 15 -zip_report false -o
ut" ,
      rand_working_dir, "-gui false > gsea_output.txt",sep=" ")
  system(command)
}
stop_gs_perm <- Sys.time()
timestamp()
difftime(stop_gs_perm,start_gs_perm,"mins")
```

1.1.9 Manual Phenotype Randomizations

9. Instead of randomizing in GSEA create our own phenotype randomization in R.

Do the following steps:

- Shuffle the class labels using the R sample function. (Even though that data consists of 4 classes we do not limit the shuffling to just the two classes being compared)
- Normalize shuffled data
- Calculate dispersion of shuffled data
- Exclude genes with LOC names
- Calculated differential expression using edgeR
- Run GSEA preranked using resulting ranks (*Needed to specify 1 permutation as if you set permutation to 0 GSEA will not calculate the NES or FDR value. We are not interested in the FDR value but we do need the NES value.*)
- Repeat for X randomizations.

*X = 1000 for this analysis

HIDE

```
timestamp()
start_rand_perm <- Sys.time()
if(run_gsea){
    library("foreach")
    library("doParallel")



    cl <- makeCluster(detectCores() - 1)
    registerDoParallel(cl, cores = detectCores() - 1)



      data = foreach(i = 1:num_randomizations, .packages = c("edgeR"),
                     .combine = rbind) %dopar% {
                       try({
        rand_analysis_name <- paste("Rand",i,sep="_")

        rand_class <- sample(classDefinitions_RNASeq$SUBTYPE)

        # create data structure to hold counts and subtype information for each sample.
        d <- DGEList(counts=counts, group=rand_class)

        #Normalize the data
        d <- calcNormFactors(d)

        #calculate dispersion
        d <- estimateCommonDisp(d)
        d <- estimateTagwiseDisp(d)

        #the below regular expression excludes gene names that are ? or that start with
        # LOC any number of additional terms can be added to the regular expresion, for
        # example to exclude any genes that start with "His" add |^His to the
        # regular expression
        exclude <- grep("\\?|^LOC", rownames(d), value=T)
        d <- d[which(!rownames(d) %in% exclude),]

        #calculate differential expression statistics with a simple design
        de <- exactTest(d, pair=c("Immunoreactive","Mesenchymal"))
        tt_exact_test <- topTags(de,n=nrow(d))

        tt <- tt_exact_test
        #calculate ranks
        ranks_RNAseq = sign(tt$table$logFC) * -log10(tt$table$PValue)

        #gene names from the TCGA set contain gene name and entrez gene ids separated by
        # '|' for all subsequent enrichment analysis we need to have just one id.
        # Separate the names into their two ids.
```

```
        genenames <- unlist(lapply( rownames(tt$table), function(data)
          {unlist(strsplit(data,"\\|"))[1]}))
        geneids <- unlist(lapply( rownames(tt$table),  function(data)
          {unlist(strsplit(data,"\\|"))[2]}))

        #create ranks file
        ranks_RNAseq <- cbind(genenames, ranks_RNAseq)
        colnames(ranks_RNAseq) <- c("GeneName","rank")

        #sort ranks in decreasing order
        ranks_RNAseq <- ranks_RNAseq[order(as.numeric(ranks_RNAseq[,2]),decreasing = TRU
E),]
        rank_filename <- file.path(rand_working_dir,paste("rand",i,"ranks.rnk",sep="_"))
        write.table(ranks_RNAseq, rank_filename,
                    col.name = TRUE, sep="\t", row.names = FALSE, quote = FALSE)

        #need to set permutation number to 1 in order for the NES to be calculated.
        # If you set it to zero the results have no p-value, fdr or NES values.
         command <- paste("java  -Xmx1G -cp",gsea_jar,  "xtools.gsea.GseaPreranked -gmx"
,
                          dest_gmt_file, "-rnk" ,rank_filename,
                          "-collapse false -nperm 1 -permute gene_set",
                          " -scoring_scheme weighted -rpt_label ",
                          rand_analysis_name,
                          " -num 100 -plot_top_x 20 -rnd_seed 12345",
                          " -set_max 200 -set_min 15 -zip_report false -out" ,
                          rand_working_dir, "-gui false > gsea_output.txt",sep=" ")
          system(command)
     })
                  }
      stopCluster(cl)
}
 stop_rand_perm <- Sys.time()
 timestamp()
 difftime(stop_rand_perm,start_rand_perm,"mins")
```

1.1.10 Calculate FDR

10. In order to calculate the FDR load in the GSEA results from each of the randomizations. For each gene set collect its ES and NES values. To calculate the FDR for each gene set:

- For given gene set get all random ES and NES values.
- If given gene set has a negative ES value, count the number of random ES values are less than it. Divide count by number of randomizations to get the FDR.
- If given gene set has a positive ES value, count the number of random ES values are greater than it. Divide count by number of randomizations to get the FDR.

a. Load in Randomization results

HIDE

```
rand_directories <- list.files(path = rand_working_dir, pattern = "Rand")
all_rand_es <- c()
all_rand_nes <- c()

for(i in 1:length(rand_directories)){
    current_rand_files <- list.files(path = file.path(rand_working_dir,rand_directories
[i]),
                                     pattern = "gsea_report_for_na_")
    #restrict to just the xls file.
    current_rand_files <- current_rand_files[grep(current_rand_files, pattern="\\.xls")]

    #load the positive files
    for(j in 1:length(current_rand_files)){
          rand_results <- read.table(  file.path(rand_working_dir,rand_directories[i],
                                    current_rand_files[j]),  header = TRUE,
                                       sep = "\t", quote="\"", stringsAsFactors = FALSE)

          es_values <- data.frame(rand_results$NAME, rand_results$ES)
          nes_values <- data.frame(rand_results$NAME, rand_results$NES)
          colnames(es_values)[2] <- current_rand_files[j]
          colnames(nes_values)[2] <- current_rand_files[j]
          #add es_values to set.
          if(i == 1 && j ==1 ){
            all_rand_es <- es_values
            all_rand_nes <- nes_values
          } else{
            all_rand_es <- merge(all_rand_es, es_values,by.x = 1,by.y = 1, all = TRUE)
            all_rand_nes <- merge(all_rand_nes, nes_values,by.x = 1,by.y = 1, all = TRUE
)
          }

    }
}
```

b. Load in the real rankings

HIDE

```
 gsea_directory <- list.files(path = rand_working_dir,
                              pattern = paste(analysis_name,"gsrand",sep="_"))
gsea_results_files <- list.files(path = file.path(rand_working_dir,
                                    gsea_directory), pattern = "gsea_report_for_na_")
gsea_results_files<- gsea_results_files[grep(gsea_results_files, pattern="\\.xls")]
actual_gsea_results <- c()
    #load the positive files
    for(j in 1:length(gsea_results_files)){
          actual_gsea_results_1 <-  read.table(  file.path(rand_working_dir,
                                                 gsea_directory,gsea_results_files[j]),
                                                 header = TRUE, sep = "\t", quote="\"",
                                                 stringsAsFactors = FALSE)
            actual_gsea_results <- rbind(actual_gsea_results , actual_gsea_results_1)

}

#replace all the zero fdr values with a more precise measurement
actual_gsea_results$FDR.q.val[which(actual_gsea_results$FDR.q.val == 0)] <- 1/1001
gsrand_gsea_results <- actual_gsea_results
```

c. Calculate the nominal p-value using the phenotype permutation results

HIDE

```
#calculate the new fdr values
new_fdr <- c()
nominal_pvalues <- c()
for(m in 1:dim(actual_gsea_results)[1]){
  current_data <- as.numeric(all_rand_es[which(all_rand_es$rand_results.NAME ==
                               actual_gsea_results$NAME[m]),2:ncol(all_rand_es)])
  if(actual_gsea_results$ES[m] < 0 ){
    nominal_pvalues <- c(nominal_pvalues,
    (length( which(current_data < actual_gsea_results$ES[m]))+1)/(num_randomizations + 1
))
  } else {
    nominal_pvalues <- c(nominal_pvalues,
    (length( which(current_data > actual_gsea_results$ES[m]))+1)/(num_randomizations + 1
))
  }
}

new_fdr <- p.adjust(nominal_pvalues, method = "BH")
```

d. Create a new results file containing our computed FDR values

HIDE

```
#create a new fake enrichment results file so we can compare the differences.
new_fdr_gsea_results_files <- actual_gsea_results
new_fdr_gsea_results_files$FDR.q.val <- new_fdr

enrichment_results_filename <- file.path(working_dir,
                                         "new_fdr_gsea_results_files_edger_rand.txt ")

write.table(new_fdr_gsea_results_files[,1:11] ,enrichment_results_filename ,
            col.name = TRUE, sep="\t", row.names = FALSE, quote = FALSE)
```

e. Plot the FDR saturation curve

HIDE

```
saturation_curve <- c()
series_of_randomizations <- c(100,200,300,400,500,600,700,800,900,1000)
fdr_thresh <- 0.01

if(num_randomizations >= 1000){
  for(l in 1:length(series_of_randomizations)){
      saturation_fdr <- c()
      for(m in 1:dim(actual_gsea_results)[1]){
        current_data <- as.numeric(all_rand_es[which(all_rand_es$rand_results.NAME ==
                  actual_gsea_results$NAME[m]),
                  2:(series_of_randomizations[l] *2 + 1)])
        if(actual_gsea_results$ES[m] < 0 ){
          saturation_fdr <- c(saturation_fdr, (length( which(current_data <
                  actual_gsea_results$ES[m]))+1)/
                    (series_of_randomizations[l] + 1))
        } else {
          saturation_fdr <- c(saturation_fdr, (length( which(current_data >
                  actual_gsea_results$ES[m]))+1)/
                    (series_of_randomizations[l] + 1))
        }
      }

      #get the number of gene sets that are less than fdr value threshold
      saturation_curve <- c(saturation_curve, length(which(saturation_fdr <= fdr_thres
h)))
  }
  saturation_curve_filename = file.path(rand_working_dir,"saturation_curve.png")
  png(saturation_curve_filename)
  plot(series_of_randomizations,saturation_curve,
       xlab="number of randomizations",
       ylab = "Number of significant( FDR <0.01) gene sets", main = "FDR Saturation curv
e")
  dev.off()
}
```

1.1.11 Basic comparison of FDR values generated by GSEA gene set randomization and manual phenotype randomization with edgeR

Plot the comparison of the FDR values using gsrand vs phenotype randomization from R using edgeR

HIDE

```
fdr_comparison_filename = file.path(rand_working_dir,"fdr_comparison.png")
png(fdr_comparison_filename)
plot(cbind(gsrand_gsea_results$FDR.q.val, new_fdr), ylab="edgeR randomization FDR",
     xlab="gs randomizations FDR", main="GSEA gs rand FDR vs edgeR manual rand FDR")
dev.off()
```

Plot the comparison of the FDR values using gsrand vs phenotype randomization from R using edgeR - log transform the p-values to better see the differences.

HIDE

```
log_fdr_comparison_filename = file.path(rand_working_dir,"log_fdr_comparison.png")
png(log_fdr_comparison_filename)
plot(cbind(-log10(gsrand_gsea_results$FDR.q.val), -log10(new_fdr)),
     ylab="-log10 edgeR randomization FDR",
     xlab="-log10 gs randomizations FDR",
     main="-log10 transformed - GSEA gs rand FDR vs \nedgeR manual rand FDR")
dev.off()
```

Create a subset of the above plot to include any gene sets where the FDR is significant in at least one of the analyses

HIDE

```
temp<- cbind(gsrand_gsea_results$FDR.q.val, new_fdr)
temp <- temp[which(temp[,1]<0.01 | temp[,2]<0.01),]

top_fdr_comparison_filename = file.path(rand_working_dir,"top_fdr_comparison.png")
png(top_fdr_comparison_filename)
plot(-log10(temp[,1]), -log10(temp[,2]), ylab="-log10 edgeR randomization FDR",
     xlab="-log10 gs randomizations FDR",
main=paste("-log10 transformed GSEA gs rand vs edgeR manual rand" ,
           "- \nsignificant in at least one analysis", sep=""))
dev.off()
```

Venn Diagram of overlaps

HIDE

```
library(VennDiagram)
two_method_overlap = file.path(rand_working_dir,"2_results_overlap_v1.png")
png(two_method_overlap)
  a <- gsrand_gsea_results$NAME[which(gsrand_gsea_results$FDR.q.val < 0.01)]
  b <- new_fdr_gsea_results_files$NAME[which(new_fdr_gsea_results_files$FDR.q.val < 0.01
)]
results <- draw.pairwise.venn(area1 = length(a),
                 area2 = length(b),
                 n12 = length(intersect(a,b)),
                 category = c("gsrand",  "edgeR_rand"), lty = "blank",
    fill = c("skyblue", "pink1"),cross.area = length(intersect(a,b)))
dev.off()
```

Figure 8: Overlap of gsrand and edgeR_rand - Compare the number of significant (FDR < 0.01) gene sets from the two methods, GSEA gs randomization and edgeR manual randomization.

1.2 Launch Cytoscape

Create EM through CyREST interface - make sure you open Cytoscape with a -R 1234 (to enable rest functionality) and allow R to talk directly to Cytoscape.

Launch Cytoscape (by default Cytoscape will automatically enable rest so as long as cytoscape 3.3 or higher is open R should be able to communicate with it)

1.3 Set up connection from R to Cytoscape

HIDE

```
#use easy cyRest library to communicate with cytoscape.
tryCatch(expr = { library("RCy3")},
         error = function(e) { source("https://bioconductor.org/biocLite.R")
           biocLite("RCy3")}, finally = library("RCy3"))

cytoscape.open = TRUE

tryCatch(expr = { cyrestGET("version")},
         error = function(e) { return (cytoscape.open = FALSE)},
         finally =function(r){ return(cytoscape.open = TRUE)})

if(!cytoscape.open){
  #try and launch cytoscape
 print("Cytoscape is not open.  Please launch cytoscape.")
} else{
  cytoscape.version =  cyrestGET("version")
}
```

1.4 Create an enrichment map

HIDE

```
#defined threshold for GSEA enrichments (need to be strings for cyrest call)
pvalue_gsea_threshold <- "0.01"
qvalue_gsea_threshold <-"0.001"

similarity_threshold <- "0.375"
similarity_metric = "COMBINED"

cur_model_name <- paste(analysis_name, "edgeR_rand",sep="_")


#although there is a gmt file in the gsea edb results directory it have been filtered to
#contain only genes represented in the expression set.  If you use this fltered file you
#will get different pathway connectivity depending on the dataset being used.  We recomm
end
#using original gmt file used for the gsea analysis and not the filtered
# one in the results directory.
gmt_gsea_file <- file.path(getwd(),dest_gmt_file)
gsea_ranks_file <- file.path(getwd(),rnk_file)
gsea_results_filename <- file.path(getwd(),enrichment_results_filename)
gsea_expression_filename <- file.path(getwd(),working_dir,
                                      "Supplementary_Table6_TCGA_OV_RNAseq_expression.tx
t")

########################################
#create EM
current_network_name <- paste(cur_model_name,pvalue_gsea_threshold,
                              qvalue_gsea_threshold,sep="_")

em_command = paste('enrichmentmap build analysisType="gsea" gmtFile=',gmt_gsea_file,
                   'pvalue=',pvalue_gsea_threshold, 'qvalue=',qvalue_gsea_threshold,
                   'similaritycutoff=',similarity_threshold,
                   'coefficients=',similarity_metric,'ranksDataset1=',
                   gsea_ranks_file,'enrichmentsDataset1=',gsea_results_filename,
                   'enrichments2Dataset1=',gsea_results_filename,
                   'filterByExpressions=false',
                   'expressionDataset1=',gsea_expression_filename,
                   sep=" ")

#enrichment map command will return the suid of newly created network.
response <- commandsGET(em_command)

current_network_suid <- 0
#enrichment map command will return the suid of newly created network unless it failed.
#If it failed it will contain the word failed
if(grepl(pattern="Failed", response)){
```

```
  paste(response)
} else {
  current_network_suid <- response
}
response <- renameNetwork(title=current_network_name,
                          network = as.numeric(current_network_suid))
```

1.5 Get a screen shot of the initial network.

HIDE

```
output_network_file <- file.path(getwd(),"manual_phenotype_randomization.pdf")

if(file.exists(output_network_file)){
  #cytoscape hangs waiting for user response if file already exists.  Remove it first
  file.remove(output_network_file)
  }

#export the network
response <- exportImage(output_network_file, type = "pdf")
```

Figure 9: Example enrichment map created from manual phenotype randomization in R using edgeR (FDR threshold<0.01)

1. Grossman, R. L. *et al.* Toward a shared vision for cancer genomic data. *N Engl J Med* **375,** 1109–12 (2016).

2. Verhaak, R. G. *et al.* Prognostically relevant gene signatures of high-grade serous ovarian carcinoma. *J Clin Invest* **123,** 517–25 (2013).

3. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. EdgeR: A bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26,** 139–40 (2010).