Two_factors

```
# Package names
packages <- c("ggrepel", "pheatmap", "DOSE", "clusterProfiler", "ensembldb", "annotables", "apeglm", "R

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  BiocManager::install(packages[!installed_packages])
}

# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

```
countData <- read_excel("Gene_count_matrix_DEseq2_V4_NOT normalized.xlsx")
colnames(countData)=c("gene_id","WT0_1","WT0_2","WT0_3","WT4_1","WT4_2","WT4_3","WT8_1","WT8_2","WT8_3"
countData=countData[,c(1,5:13,17:24)]
countData=aggregate(. ~gene_id , data = countData, sum)
countData=column_to_rownames(countData,'gene_id')
treatment=c("WT","WT","WT","WT","WT","WT","WT","WT","WT","KO","KO","KO","KO","KO","KO","KO","KO")
timee=c(rep("4",3),rep("8",3),rep("24",3),rep("4",3),rep("8",3),rep("24",2))
#colData=as_tibble(cbind(colnames(countData),time,treatment))


colData=as_tibble(cbind(colnames(countData),treatment,timee))
```

```
## Warning: The ‘x‘ argument of ‘as_tibble.matrix()‘ must have unique column names if
## ‘.name_repair‘ is omitted as of tibble 2.0.0.
## i Using compatibility ‘.name_repair‘.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
colData$group=str_c(treatment,timee)
colnames(colData)[1]="samplename"
dds=DESeqDataSetFromMatrix(countData = countData, colData = colData, design = ~timee+treatment+timee*tr
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
dds=DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
dds =dds[rowSums(counts(dds))>50,]
sizeFactors(dds)
```

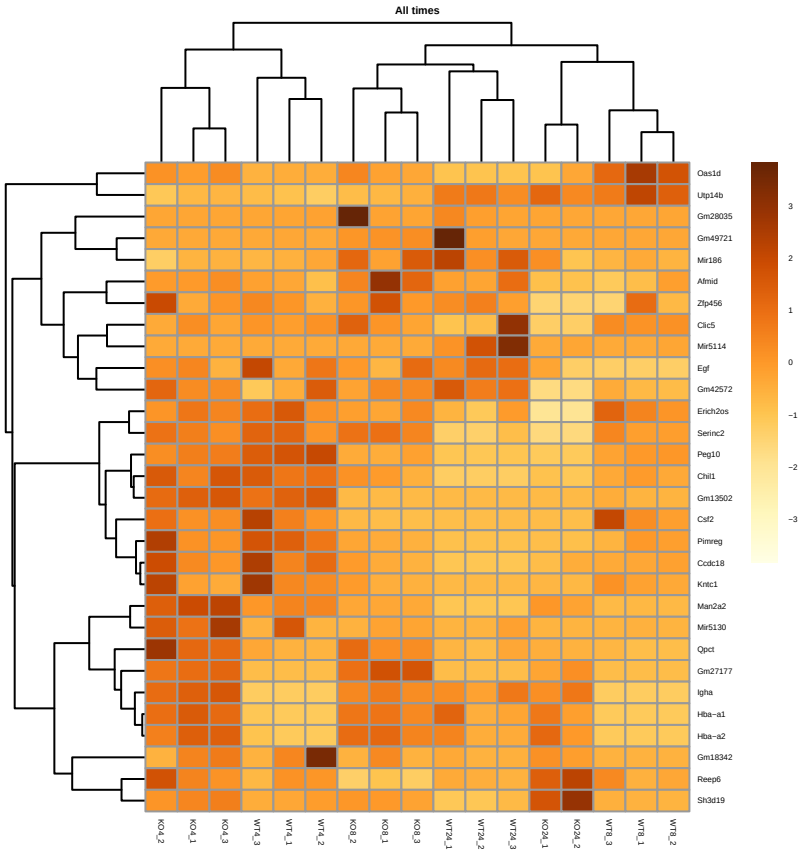
```
##     WT4_1     WT4_2     WT4_3     WT8_1     WT8_2     WT8_3    WT24_1    WT24_2
## 0.9308537 0.9145382 1.0197690 0.9064397 0.9623465 0.9282227 1.1713776 1.2439343
##    WT24_3     KO4_1     KO4_2     KO4_3     KO8_1     KO8_2     KO8_3    KO24_1
## 1.1017722 0.8764220 0.9986679 0.8848651 1.0101009 0.9991292 0.9867509 1.2645652
##    KO24_2
## 1.1876410
```

```
rld=vst(dds)
pca=plotPCA(rld, intgroup=c("treatment"))


res_table=results(dds)

normalized_counts=counts(dds, normalized=TRUE)


res_table_tb <- res_table %>%
  data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()


sig <- res_table_tb[which(res_table_tb[,7]<0.05 & abs(res_table_tb[,3])>1.5),]


normalized_counts_c=counts(dds, normalized=TRUE)


normalized_counts_c <- normalized_counts_c %>%
  data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()

norm_sig <- normalized_counts_c[,] %>%
              dplyr::filter(normalized_counts_c$gene %in% sig$gene) %>%
               data.frame() %>%
               column_to_rownames(var = "gene")
```

```
pdf("PCA.pdf")
print(pca)
dev.off()
```

```
## pdf
##   2
```

```
phea=pheatmap(norm_sig, scale = "row", clustering_distance_rows = "correlation", fontsize = 3, color = c
```




```
rownames(norm_sig[phea$tree_row[["order"]],])
```

```
##  [1] "Oas1d"    "Utp14b"   "Gm28035"  "Gm49721"  "Mir186"   "Afmid"
##  [7] "Zfp456"   "Clic5"    "Mir5114"  "Egf"      "Gm42572"  "Erich2os"
## [13] "Serinc2"  "Peg10"    "Chil1"    "Gm13502"  "Csf2"     "Pimreg"
## [19] "Ccdc18"   "Kntc1"    "Man2a2"   "Mir5130"  "Qpct"     "Gm27177"
## [25] "Igha"     "Hba-a1"   "Hba-a2"   "Gm18342"  "Reep6"    "Sh3d19"
```

```
a=sort(cutree(phea$tree_row, k=2))

WT_only=names(a[a==1])
```

```
write.csv(WT_only,"WT_only.csv")

KO_only=names(a[a==2])
write.csv(KO_only,"KO_only.csv")

pdf("heatmap.pdf", height = 27)
print(phea)
dev.off()
```

```
## pdf
##   2
```

```
idx = grcm38$symbol %in% rownames(res_table)

ids <- grcm38[idx, ]

non_duplicates <- which(duplicated(ids$symbol) == FALSE)

ids <- ids[non_duplicates, ]

res_ids <- inner_join(res_table_tb, ids, by=c("gene"="symbol"))

all_genes <- as.character(res_ids$entrez)

sig <- dplyr::filter(res_ids, padj < 0.05)

sig_genes <- as.character(sig$entrez)


res_entrez <- filter(res_ids, entrez != "NA")
res_entrez <- res_entrez[which(duplicated(res_entrez$entrez) == F), ]
all_foldchanges <- res_entrez$log2FoldChange
names(all_foldchanges) <- res_entrez$entrez
all_foldchanges <- sort(all_foldchanges, decreasing = TRUE)

symbol_foldchanges=sig$log2FoldChange

names(symbol_foldchanges)=sig$gene


all_foldchanges=sort(all_foldchanges,decreasing=TRUE)



go <- enrichGO(gene = sig_genes,
               universe = all_genes,
               keyType = "ENTREZID",
               OrgDb = org.Mm.eg.db,
               ont = "ALL",
               pAdjustMethod = "BH",
               qvalueCutoff = 0.05,
               readable = TRUE)
```

```
go_dp=dotplot(go, showCategory=32)
```

```
netplot=cnetplot(go, foldchange=all_foldchanges)
```

```
ids<-bitr(rownames(norm_sig), fromType = "SYMBOL", toType = "ENTREZID", OrgDb=org.Mm.eg.db)
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
## Warning in bitr(rownames(norm_sig), fromType = "SYMBOL", toType = "ENTREZID", :
## 13.33% of input gene IDs are fail to map...
```

```
dedup_ids = ids[!duplicated(ids[c("ENTREZID")]),]
```

```
df2 = res_table_tb[res_table_tb$gene %in% dedup_ids$SYMBOL,]

df2$Y = dedup_ids$ENTREZID
```

```
# Name vector with ENTREZ ids
kegg_gene_list <- df2$log2FoldChange

names(kegg_gene_list) <- df2$Y

# omit any NA values
kegg_gene_list<-na.omit(kegg_gene_list)

# sort the list in decreasing order (required for clusterProfiler)
kegg_gene_list = sort(kegg_gene_list, decreasing = TRUE)
```

```
gse <- gseGO(gene = kegg_gene_list,
               keyType = "ENTREZID",
               OrgDb = org.Mm.eg.db,
               ont = "ALL",     pAdjustMethod = "BH",)
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## no term enriched under specific pvalueCutoff...
```

```
gsea=dotplot(gse, showCategory=10)
```

```
## Error in `$<-.data.frame`(`*tmp*`, ".sign", value = "activated"): replacement has 1 row, data has 0
```

```
gseap=gseaplot(gse, by = "all", title = gse$Description[35], geneSetID = 35)
```

```
## Error in if (abs(max.ES) > abs(min.ES)) {: missing value where TRUE/FALSE needed
```

```
kgg <- gseKEGG(geneList= kegg_gene_list, organism= "mmu", minGSSize    = 3,maxGSSize    = 1800, pvalueC
```

```
## Reading KEGG annotation online: "https://rest.kegg.jp/link/mmu/pathway"...
```

```
## Reading KEGG annotation online: "https://rest.kegg.jp/list/pathway/mmu"...
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## no term enriched under specific pvalueCutoff...
```

```
kegg=dotplot(kgg)
```

```
## Error in `$<-.data.frame`(`*tmp*`, ".sign", value = "activated"): replacement has 1 row, data has 0
```

```
pdf("GO_dotplot.pdf", height=13)
print(go_dp)
dev.off()
```

```
## pdf
##   2
```

```
pdf("gsea_plot.pdf")
print(gseap)
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in selecting a method for fu
```

```
dev.off()
```

```
## pdf
##   2
```

```
pdf("gsea_dotplot.pdf")
print(gsea)
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in selecting a method for fu
```

```
dev.off()
```

```
## pdf
##   2
```

```
pdf("kegg_dotplot.pdf")
print(kegg)
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in selecting a method for fu
```

```
dev.off()
```

```
## pdf
##   2
```

```
pdf("network.pdf",width = 10)
print(netplot)
dev.off()
```

```
## pdf
##   2
```

```
normalized_counts=rlog(counts(dds))
```

```
normalized_counts <- normalized_counts %>%
  data.frame() %>%
  rownames_to_column(var="gene") %>%
  as_tibble()

normalized_counts$entrez=mapIds(org.Mm.eg.db,key,keys = normalized_counts$gene,column = "ENTREZID", key
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
normalized_counts$gene=normalized_counts$entrez

normalized_counts=normalized_counts[,1:18]

normalized_counts=aggregate(. ~gene , data = normalized_counts, sum)

normalized_counts <- normalized_counts %>%
  data.frame() %>%
 column_to_rownames(var = "gene")


require(Biobase)
normalized_counts1<-new("ExpressionSet", exprs=as.matrix(normalized_counts))
```

```r
normalized_counts1$time= colData$group
```

```r
pathwaysDF <- msigdbr("mouse", category=c("H"))
pathways <- split(as.character(pathwaysDF$entrez_gene), pathwaysDF$gs_name)
```

```r
set.seed(1)
gesecaRes <- geseca(pathways, exprs(normalized_counts1), minSize = 10, maxSize = 1000, eps = 0,nPermSim

geseres=plotGesecaTable(gesecaRes |> head(5), pathways,E=exprs(normalized_counts1),)

IFNG=plotCoregulationProfile(pathway=pathways[["HALLMARK_INTERFERON_GAMMA_RESPONSE"]],
                        E=exprs(normalized_counts1), conditions=normalized_counts1$time)


INF=plotCoregulationProfile(pathway=pathways[["HALLMARK_INFLAMMATORY_RESPONSE"]],
                        E=exprs(normalized_counts1), conditions=normalized_counts1$time)


TNF_NFKB=plotCoregulationProfile(pathway=pathways[["HALLMARK_TNFA_SIGNALING_VIA_NFKB"]],
                        E=exprs(normalized_counts1), conditions=normalized_counts1$time)


TNFA=plotCoregulationProfile(pathway=pathways[["HALLMARK_INTERFERON_ALPHA_RESPONSE"]],
                        E=exprs(normalized_counts1), conditions=normalized_counts1$time)

E2F=plotCoregulationProfile(pathway=pathways[["HALLMARK_E2F_TARGETS"]],
                        E=exprs(normalized_counts1), conditions=normalized_counts1$time)



pdf("IFNG.pdf",width = 10)
print(IFNG)
dev.off()
```

```
## pdf
##   2
```

```r
pdf("IFN.pdf",width = 10)
print(INF)
dev.off()
```

```
## pdf
##   2
```

```r
pdf("TNF_NFKB.pdf",width = 10)
print(TNF_NFKB)
dev.off()
```

```
## pdf
##   2
```

```
pdf("TNFA.pdf",width = 10)
print(TNFA)
dev.off()
```

```
## pdf
##   2
```

```
pdf("E2F.pdf",width = 10)
print(E2F)
dev.off()
```

```
## pdf
##   2
```

```
pdf("geseca.pdf",width = 10)
print(geseres)
dev.off()
```

```
## pdf
##   2
```