Code for: Yung T et al, "SUFU and SPOP mediated downregulation of Hh signaling promotes pancreatic beta cell differentiation through the regulation of organ-specific stromal niche signals"

Minggao Liang

May 24, 2019

Contents

1 Reading, selecting, and preprocessing count data

1.1 Libraries, count data, annotation

```
## Libraries used for plotting/analysis
library(DESeq2)

## Warning in is.na(x[[i]]):  is.na() applied to non-(list or vector) of type 'environment'

library(scales)
library(ggrepel)
library(ggplot2)
library(reshape2)
```

```
library(gtools)
library(pcaMethods)
library(gplots)
library(RColorBrewer)
library(gProfileR)
library(Vennerable)
library(fgsea)
library(BiocParallel)
register(SerialParam())
library(ggpubr)
library(pandaR)
library(biomaRt)
library(GenomicRanges)

plot.dir <- "./output/plots/"
tab.dir <- "./output/tables/"

## Read in a table with annotations for genes. Used to get strand and start/end info for genes.
annotation <-
    read.table("./reference_data/gencode.vM4.genLocs.bed")

head(annotation)

##     V1      V2      V3 V4            V5     V6
## 1 chr1 3073253 3074322  + RP23-271O17.1   1069
## 2 chr1 3102016 3102125  +       Gm26206    109
## 3 chr1 3205901 3671498  -          Xkr4 465597
## 4 chr1 3252757 3253236  + RP23-317L18.1    479
## 5 chr1 3365731 3368549  - RP23-317L18.4   2818
## 6 chr1 3375556 3377788  - RP23-317L18.3   2232

## Column 5 corresponds to the gene names. This script is written under the assumption that genes are
## identified using their NAMES (not ENSGID, etc.)
annotation.uniq <- annotation[!duplicated(annotation$V5),]
row.names(annotation.uniq) <- annotation.uniq$V5

## A curated list of all mouse TFs from RIKEN http://genome.gsc.riken.jp/TFdb/tf_list.html
mmTFList <- scan("./reference_data/uniq_all_mgi.txt", what=character())

## Set this variable to the directory where raw counts are stored
sample.M.dir <- "./counts_data/"

## Select only a subset of the files in the directory to consider.
sample.M.files <- list.files(sample.M.dir, pattern=".counts")
```

1.2 Setting up the DESeq2 object

Setup the contrast table for analysis with DESeq2

```
##Parses file names to leave only the characters preceding a number. In this case, the tissue identity
sample.M.type <- sub(".counts", "", sub(".*_","",sample.M.files))
sample.M.condition <- gsub("_.+","",sample.M.files)

## Sets up groups from the above.
groups <- paste0(sample.M.type,"_",sample.M.condition)

## Constructs a table to build the DESeq2 dataset with.
sample.M.table <- data.frame(sampleName = sample.M.files,
                             fileName = sample.M.files,
```

```
                            type=sample.M.type,
                            condition=sample.M.condition)
## DESeq2 built-in function to read in htseq-counts table.
## Counts table should be tab delimited 2 columns of geneName\tcount
## The contrast matrix is by type (ie. tissue type)
ddsHTSeq.raw <-  DESeqDataSetFromHTSeqCount(sampleTable = sample.M.table,
                                            directory = sample.M.dir,
                                            design = ~type)

raw.counts <- counts(ddsHTSeq.raw)

## A filtering step removes genes that are very lowly expressed across all samples
## Genes that have on avg <1 count per sample are removed. The filtered matrix is used to reconstruct
## DESeq2 object.
adj.counts <- raw.counts[rowSums(raw.counts) > ncol(raw.counts),]
ddsHTSeq.M <- DESeqDataSetFromMatrix(adj.counts,colData=ddsHTSeq.raw@colData,
                                     design=ddsHTSeq.raw@design)

## Step that does the actual DE analysis
dds <- DESeq(ddsHTSeq.M, betaPrior = FALSE)
mcols(dds)$basepairs <- annotation.uniq[rownames(dds),6]
```

1.2.1 Supplementary Table 1,2

```
## Generates supplementary counts tables and fpkm tables
write.table(cbind(data.frame(gene = row.names(counts(dds))), counts(dds)),
            paste0(tab.dir,"/rawCounts.txt"), sep="\t", row.names=F, quote=F)
write.table(cbind(data.frame(gene = row.names(fpkm(dds))), fpkm(dds)), paste0(tab.dir,"/fpkm.txt"),
            sep="\t", row.names=F, quote=F)
```

2 Analysis and Figures

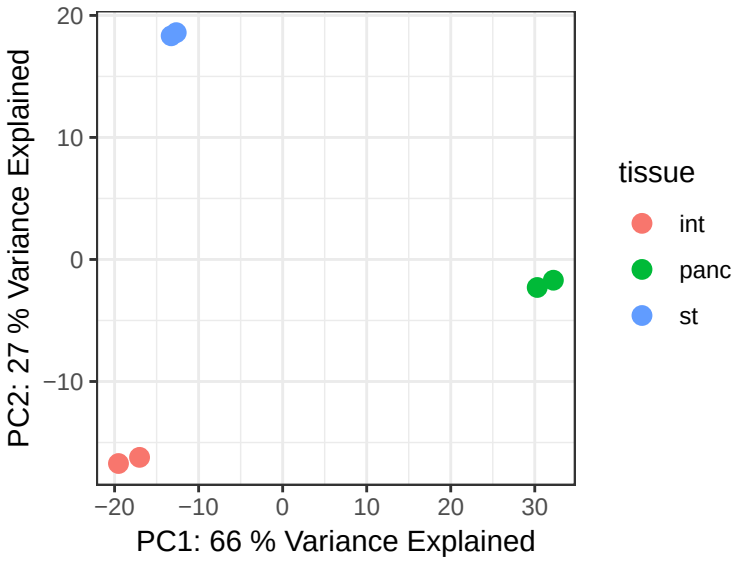
2.1 PCA Analysis of raw counts

```
## Process counts data for PCA, as per the DESeq2 vignette
rld <- varianceStabilizingTransformation(dds)
rld.mat <- assay(rld)

## Identify top 1000 most variable genes in WT organs (exclude mutant panc)
rld.mat.noHH <- rld.mat[, !grepl("pancHH",colnames(rld.mat))]
rld.rv <- apply(rld.mat.noHH, 1, var)
select <- order(rld.rv, decreasing=T)[seq_len(min(1000, length(rld.rv)))]
variableGenes <- row.names(rld.mat)[select]
```

2.1.1 Figure 1C

```
## Perform pca with top 1000 most variable genes
rld.pca.noHH <- pca(t(rld.mat.noHH[select,]),method="svd",nPcs=2)
rld.scores <- as.data.frame(scores(rld.pca.noHH))
rld.scores$sample <- gsub(".counts","", row.names(rld.scores))
rld.scores$tissue <- gsub(".*_","", rld.scores$sample)
rld.summary <- summary(rld.pca.noHH)

## svd calculated PCA
```

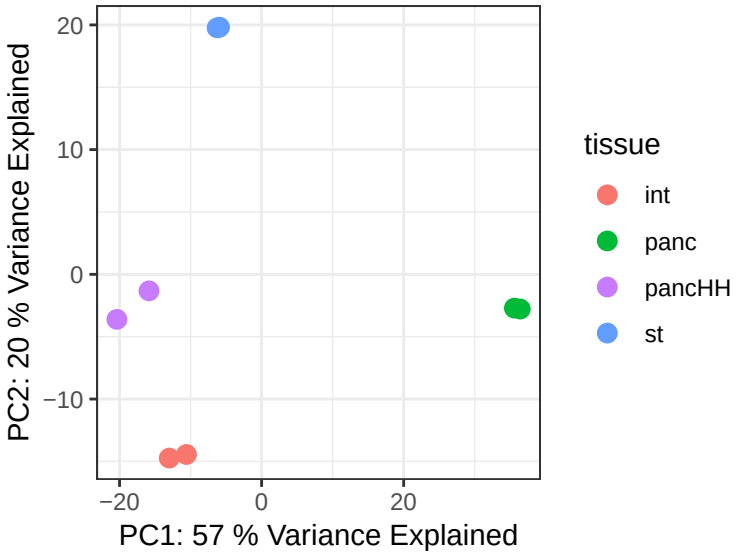
```
## Importance of component(s):
##                     PC1    PC2
## R2            0.6564 0.2730
## Cumulative R2 0.6564 0.9294

## pdf(paste0(plot.dir,"Fig1C_noMut_pca.pdf"),width=3,height=3)
ggplot(rld.scores, aes(x=PC1, y=PC2, color = tissue)) + geom_point(size=3) +
    xlab(paste("PC1:",round(rld.summary[1,"PC1"]*100),"% Variance Explained")) +
    ylab(paste("PC2:",round(rld.summary[1,"PC2"]*100),"% Variance Explained")) +
    theme_bw() #+ theme(legend.position="none")#+ ggtitle("PCA of WT tissues")
```




```
## dev.off()
```

2.1.2 Figure 5B

```
## PCA using same 1000 most variable genes above but including mutant samples
rld.all.pca <- pca(t(rld.mat[select,]),method="svd",nPcs=2)
rld.all.scores <- as.data.frame(scores(rld.all.pca))
rld.all.scores$sample <- gsub(".counts","", row.names(rld.all.scores))
rld.all.scores$tissue <- gsub(".*_","", rld.all.scores$sample)
rld.all.summary <- summary(rld.all.pca)

## svd calculated PCA
## Importance of component(s):
##                     PC1    PC2
## R2            0.5691 0.1951
## Cumulative R2 0.5691 0.7641

## pdf(paste0(plot.dir,"Fig5A_withMut_pca.pdf"),width=3,height=3)
ggplot(rld.all.scores, aes(x=PC1, y=PC2, color = tissue)) + geom_point(size=3) +
    xlab(paste("PC1:",round(rld.all.summary[1,"PC1"]*100),"% Variance Explained")) +
    ylab(paste("PC2:",round(rld.all.summary[1,"PC2"]*100),"% Variance Explained")) +
    theme_bw() + scale_color_manual(values=c("#F8766D","#00BA38","#C77CFF","#619CFF")) #+
```




```
    #theme(legend.position="none")
## dev.off()
```

2.2 DESeq2 analysis per contrast

```
## Set up a list of non-redundant tissue-tissue contrasts
list.of.contrasts <- list(c("panc", "int"),
                          c("panc", "st"),
                          c("panc", "pancHH"),
                          c("st", "int"),
                          c("st", "pancHH"),
                          c("int", "pancHH"))

list.of.top <- lapply(list.of.contrasts, function(contrast) {
    top <- results(dds, contrast=c("type",contrast[1], contrast[2]), pAdjustMethod="bonferroni")

    ## Assign a column corresponding to gene names.
    top$gene <- row.names(top)

    ## Drop any rows that contain NAs (shouldn't be any at this point, but just to make sure).
    top$pval[is.na(top$pval)] <- 1
    top$padj[is.na(top$padj)] <- 1
    ## print(range(log(top£baseMean)))

    ## Append info about strand, and chr start end positions
    annotation.df <- annotation.uniq[top$gene,c(1:4,6)]
    colnames(annotation.df) <- c("chr","start","end","strand","size")
    top <- cbind(top,annotation.df)

    ## Throw out chrM
    top <- subset(top, chr!="M")
    ## Sort in a manner that makes sense for chrs
    top$chr <- factor(as.character(top$chr), levels=levels(top$chr)[mixedorder(levels(top$chr))])
    ## Extract cell-type specific average counts (these make more sense than total averages)
    return(as.data.frame(top))
})

names(list.of.top) <- sapply(list.of.contrasts, function(x) paste0(x, collapse="_vs_"))
```

2.2.1 Supplementary Table 3

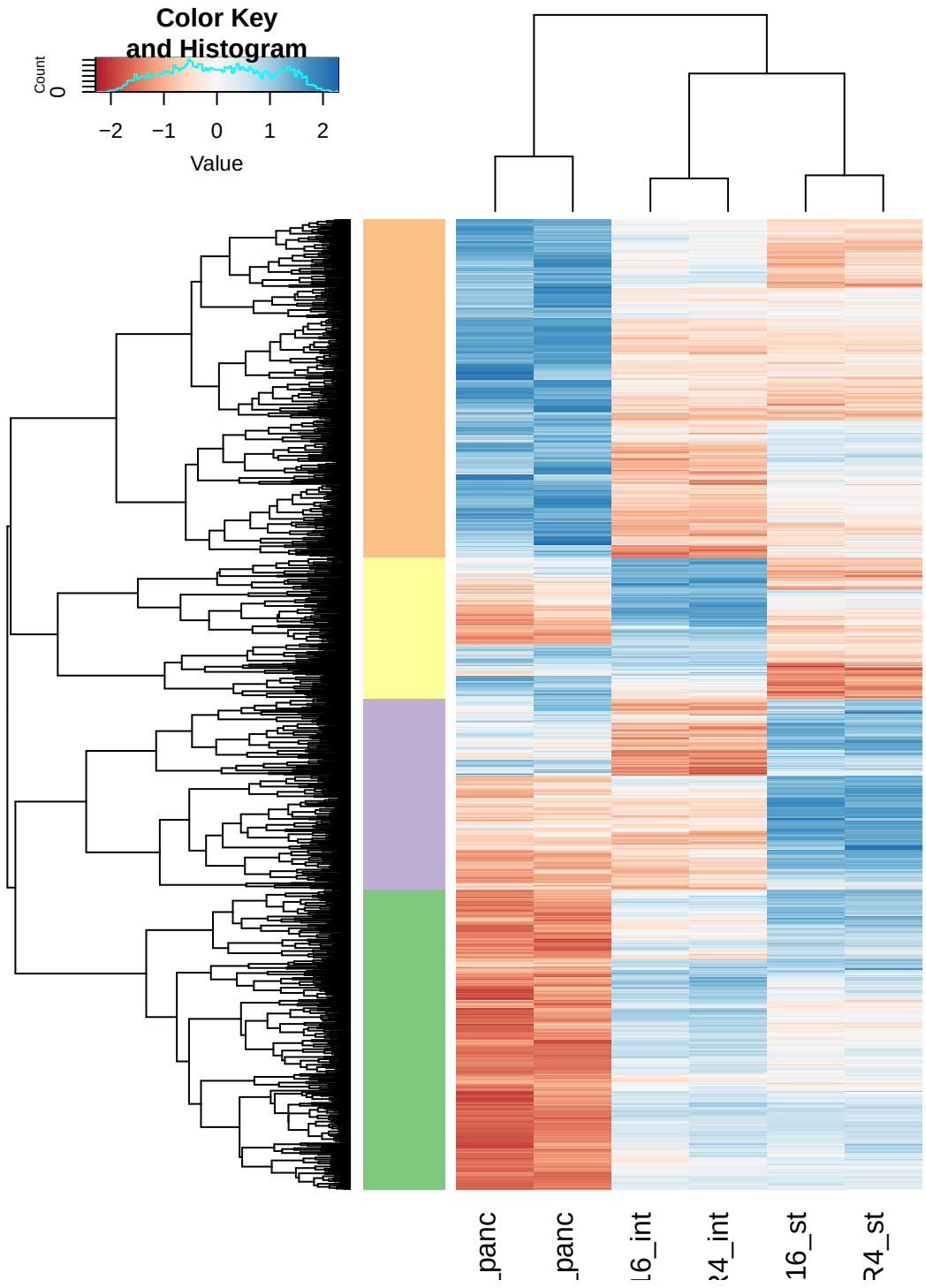
```
for(n in names(list.of.top)){
    write.table(list.of.top[[n]], paste0(tab.dir,n,"_deTab.txt"),quote=F,sep="\t",row.names=F)
}
```

2.3 Unsupervised clustering and heatmap

```
## Select significantly DE genes from any comparison (padj < 0.05 and log2FC > 1)
any.sig.gene.mat <- do.call(rbind, lapply(names(list.of.top), function(n){
    top <- list.of.top[[n]]
    top.sub <- subset(top, padj < 0.05 & abs(log2FoldChange) > 1)
    return(data.frame(genes = top.sub$gene,
                      padj = top.sub$padj,
                      log2FC = top.sub$log2FoldChange,
                      contrast=n))
}))

## Keep only those DE in WT tissue comparisons
any.sig.gene.noMut <- as.character(unique(any.sig.gene.mat[!grepl("pancHH",any.sig.gene.mat[,4]),1]))

## Get a normalized count table for comparable expression
norm.counts <- counts(dds, normalize=T)*1000/mcols(dds)$basepairs
colnames(norm.counts) <- gsub("\\..+","",colnames(norm.counts))
to.plot <- log(as.matrix(as.data.frame(norm.counts)) + 1)

## Scale the counts so that we're looking at differences rather than driven by high/low expressers
to.plot.scale <- t(apply(to.plot,1,scale))
to.plot.noHH <- to.plot.scale[any.sig.gene.noMut,!grepl("pancHH", colnames(to.plot))]
colnames(to.plot.noHH) <- colnames(to.plot)[!grepl("pancHH", colnames(to.plot))]

## Clustering for the heatmap
mandist <- function(x){ dist(x, method="euc")}
manhclust <- function(x){hclust(x, method="complete")}
to.plot.noHH.dist <- mandist(to.plot.noHH)
dendro <- manhclust(to.plot.noHH.dist)
##plot(dendro)
row.cut <- cutree(dendro, k=4)
row.colors <- factor(row.cut)
levels(row.colors) <- brewer.pal(4,"Accent")

## Group genes based on clusters defined above
genes.by.grp <- lapply(levels(row.colors), function(color) {
    any.sig.gene.noMut[row.colors == color]
})
names(genes.by.grp) <- levels(row.colors)
```

2.3.1 Figure 1B

```
## pdf(paste0(plot.dir,"Fig1B_noMut_HM.pdf"),width=6,height=7.5)
heatmap.2(to.plot.noHH, distfun=mandist, hclust=manhclust, trace="none",
          col=colorRampPalette(brewer.pal(9, "RdBu"))(100), labRow=F,
          lhei=c(.2,1), lwid=c(0.8,1.2), margins=c(4,4), RowSideColors=as.character(row.colors))
```




```
## dev.off()
```

2.4 GO enrichments and Construction of EnrichmentMap networks Fig1D

Tables for enrichment map were generated using gene identifiers grouped as above with gProfiler rev 1741 build date 2017-10-19 using the web portal with the following settings:
Organism: Mus musculus
Options: Significant only, No electronic GO annotations, Hierarchical sorting, Hierarchical filtering = Best per parent group (strong), user p-value = 0.01, Size of functional category Max = 350
Ontologies: Gene Ontology BP CC MF; KEGG
Output format: Generic Enrichment Map
This was originally done in R but will need to be done manually via the web portal as the gProfiler R package does not support archived versions.

Network was constructed in cytoscape v3.6.0 using the EnrichmentMap app v2.0.1 using the following parameters:
P-value Cutoff = 1
FDR Q-value Cutoff = 1
Similarity Cutoff: Overlap Coefficient = 0.2
GMT file was generated by downloading the corresponding mmusculus.GO.NAME.gmt from gProfiler GMT dataset and appending the archived KEGG pathways. The final compiled GMT is included as the file mouse_combi.gmt.

2.5 Network analysis of niche TFs (Fig S1 A-D)

To identify TF regulatory networks responsible for tissue-specific niches, we used a previously published package Panda, which integrates gene expression and motif information to build TF-gene edges. The idea is to ask which TFs are enriched for edges to genes that are overexpressed in a particular tissue or cluster.

This analysis requires some extra datasets which include
1. Collection of mouse TF motifs (reference_datasets/merged_mmuscentric_nonempty.tf; taken from CisBP build 20171201) In the case where no PWM was available for mouse, the corresponding human PWM was selected (made based on gene name). 2. Gene expression data across multple tissues (Fantom5 mouse, downloaded 20141111)
3. Promoter motif predictions (generated using matrix-scan from Regulatory Sequence Analysis Tools RSAT)

```
## Assemble a table of the per tissue mean expression of each gene
to.plot.tissueMean <- as.data.frame(sapply(unique(sample.M.type), function(type){
    to.plot.sub <- log(counts(dds, normalize=T)+1)[, grepl(paste0(type,"$"), colnames(to.plot))]
    return(rowMeans(to.plot.sub))
}))

## Identify TFs in the dataset (significantly DE and expressed, based on mean CPM
de.tf.list <- any.sig.gene.noMut[any.sig.gene.noMut %in% mmTFList]
expr.tf.list <- row.names(to.plot.tissueMean)[
    row.names(to.plot.tissueMean) %in% mmTFList & rowSums(to.plot.tissueMean > 3.5) > 0]
cat(expr.tf.list, file="reference_data/exprtflist.txt", sep="\n")
```

The below code retrieves promoter regions from genes of interest, ie. our list of expressed TFs The default range used is 2.5kb upstream, 0.5 kb downstream

2.5.1 Retrieve promoter regions via biomaRt

```
ensembl.version <- "feb2014.archive.ensembl.org"
mart.name <- "ENSEMBL_MART_ENSEMBL"
genes <- any.sig.gene.noMut ## Any sig DE gene identified by comparing WT tissues

mmus <- useMart(mart.name, dataset = "mmusculus_gene_ensembl", host=ensembl.version)

regions <- getBM(attributes = c("chromosome_name","transcript_start","transcript_end",
                                "mgi_symbol","strand","transcript_status","transcript_biotype",
                                "status","gene_biotype"),
      filters="mgi_symbol",values=genes,
      mart = mmus)
colnames(regions) <- c("chr","start","end","gene","strand","tstatus","tbiotype","gstatus","gbiotype")

promoters <- unique(do.call(rbind, apply(regions, 1, function(x){
    chr=x[1]
    starto = as.numeric(x[2])
    endo = as.numeric(x[3])
    gene = x[4]
    strand = x[5]
    tstatus = x[6]
    tbiotype = x[7]
    gstatus = x[8]
    gbiotype = x[9]
```

```
    extend.range = c(2500,500)
    if(starto > 0){
        start = max(1, starto - extend.range[1])
        end = starto + extend.range[2]
    } else {
        start = max(1, endo - extend.range[2])
        end = endo + extend.range[1]
    }
    output.bed <- data.frame(chr = paste("chr", chr, sep=""),
                             start = start,
                             end = end,
                             gene = gene,
                             strand = strand,
                             tstatus=tstatus,
                             tbiotype=tbiotype,
                             gstatus=gstatus,
                             gbiotype=gbiotype
                             )
    return(output.bed)
}))

promoters.filt <- subset(promoters, tstatus == "KNOWN" & tbiotype == "protein_coding" &
                                    gstatus=="KNOWN" & gbiotype=="protein_coding")

promoters.range <- GRanges(seqnames=promoters.filt$chr,
                                  ranges=IRanges(start=promoters.filt$start,
                                                 end=promoters.filt$end),
                                  gene=promoters.filt$gene)
genes.keep <- unique(mcols(promoters.range))$gene
promoters.collapse <- do.call(c,lapply(genes.keep, function(a.gene){
    a.range <- reduce(promoters.range[mcols(promoters.range)$gene == a.gene])
    mcols(a.range)$gene <- a.gene
    return(a.range)
}))

out.bed <- data.frame(chr=seqnames(promoters.collapse),
                      start = start(promoters.collapse),
                      end = end(promoters.collapse),
                      gene = mcols(promoters.collapse)$gene)
out.bed$gene <- with(out.bed, paste0(chr,":",start,"-",end,".",gene))

write.table(out.bed, "./reference_data/anySigGeneNoMut_promoter.bed", row.names=F,col.names=F,
            quote=F,sep="\t")
```

2.5.2 Promoter motif occurrence using RSAT

To get motif occurences within promoter regions of DE genes, we obtained fasta sequences using fastaFromBed (genome.fa version is Mus_musculus.GRCm38.68; not included) from the bedtools suite of tools and matrix-scan from RSAT as below:

```
## Retrieve fasta sequences for promoter regions
fastaFromBed -fi mm10_genome.fa -fo anySigGene_noMut_pr.fa -bed anySigGeneNoMut_promoter.bed\
-name

## Extract PWMs for DE tfs from the master list (not necessary but reduces run time)
awk 'FNR==NR{tokeep[$1]=1; next}; $1~/^AC/{if(out==1){print toprint};out=0;toprint=$0;next};\
{toprint=toprint"\n"$0};$1~/^ID/&&tokeep[$2]==1{out=1};END{if(out==1)print toprint}'\
exprtflist.txt merged_mmuscentric_nonempty.tf > exprtfmatrices.tf

## Scan fasta sequences for motif occurrences
matrix-scan  -v 1 -quick -i anySigGene_noMut_pr.fa -m exprtfmatrices.tf\
```

```
-matrix_format transfac -origin start -bginput -markov 1 -2str -uth pval 0.00001\
-return pval > anySigGene_noMut_pr.mscan
```

Matrix scan can be run on the RSAT Web Portal using the appropriate fasta and matrix files with the following settings:
Estimate residue probabilities from input sequences with Markov order = 1
Sequence Origin = 1
Return = sites plus pval
Threshold = pval ¡= 0.00001

2.5.3 Construction of regulatory network with PandaR

```
## Wrapper function for parsing the output from matrix-scan
## Takes a raw matrix scan file, p-value cutoff, gene set, and expression matrix
condenseMscan <- function(a.mscan, p.cut, gene.set, expr){
    ## Clean up gene names to match mgi symbols
    levels(a.mscan$X.seq_id) <- gsub(".*\\.","",levels(a.mscan$X.seq_id))

    ## Keep only genes/TFs that are common to the promoter gene set and the expression dataset used
    ## This is necessary to avoid errors with PandaR later
    common.g <- intersect(gene.set, row.names(expr))
    a.reduced <-
        subset(a.mscan, ft_name %in% common.g & Pval <= p.cut & X.seq_id %in% common.g)

    ## Take only the TF name, gene name, and weight scores. In case of duplicated TF-gene pairings,
    ## take best weight score.
    a.condense <- unique(a.reduced[,c("ft_name", "X.seq_id","weight")])
    a.condense$pairing <- paste0(a.condense$ft_name, a.condense$X.seq_id)
    a.condense <- do.call(rbind, lapply(unique(a.condense$pairing), function(x){
        a.sub <- subset(a.condense, pairing==x)
        return(data.frame(ft_name=a.sub[1,1],
                          X.seq_id=a.sub[1,2],
                          score = max(a.sub[,3])))
    }))
}

## Extract and scale expression data for TFs for network analysis w/ panda
to.plot.tf.tissueMean <- as.data.frame(t(apply(to.plot.tissueMean[de.tf.list,],1,scale)))
colnames(to.plot.tf.tissueMean) <- colnames(to.plot.tissueMean)
to.plot.tf.tissueMean$gene <- row.names(to.plot.tf.tissueMean)
to.plot.tf.tissueMean$group <- factor(row.cut[row.names(to.plot.tf.tissueMean)])

## load a simplified mouse fantom5 expression data matrix
## Gene counts are the sum of all individual promoters; all non-annotated promoters removed
load("reference_data/simple.fantom.expr.RData")
simple.fantom.scale <- t(apply(simple.fantom.expr,1, scale))

## Read in and parse results from matrix-scan
mouse.de.mscan <- read.table("reference_data/anySigGene_noMut_pr.mscan",header=T,comment.char=";")
mouse.fantomT.condense <- condenseMscan(mouse.de.mscan, 0.00001, row.names(to.plot.tissueMean),
                                        simple.fantom.expr)

## construct network w/ panda
mouse.fantomT.panda <-
    panda(mouse.fantomT.condense,
          simple.fantom.expr[unique(as.character(mouse.fantomT.condense$X.seq_id)),])

## Retrieve regulatory network component
mouse.fantomT.subnet <- mouse.fantomT.panda@regNet
```

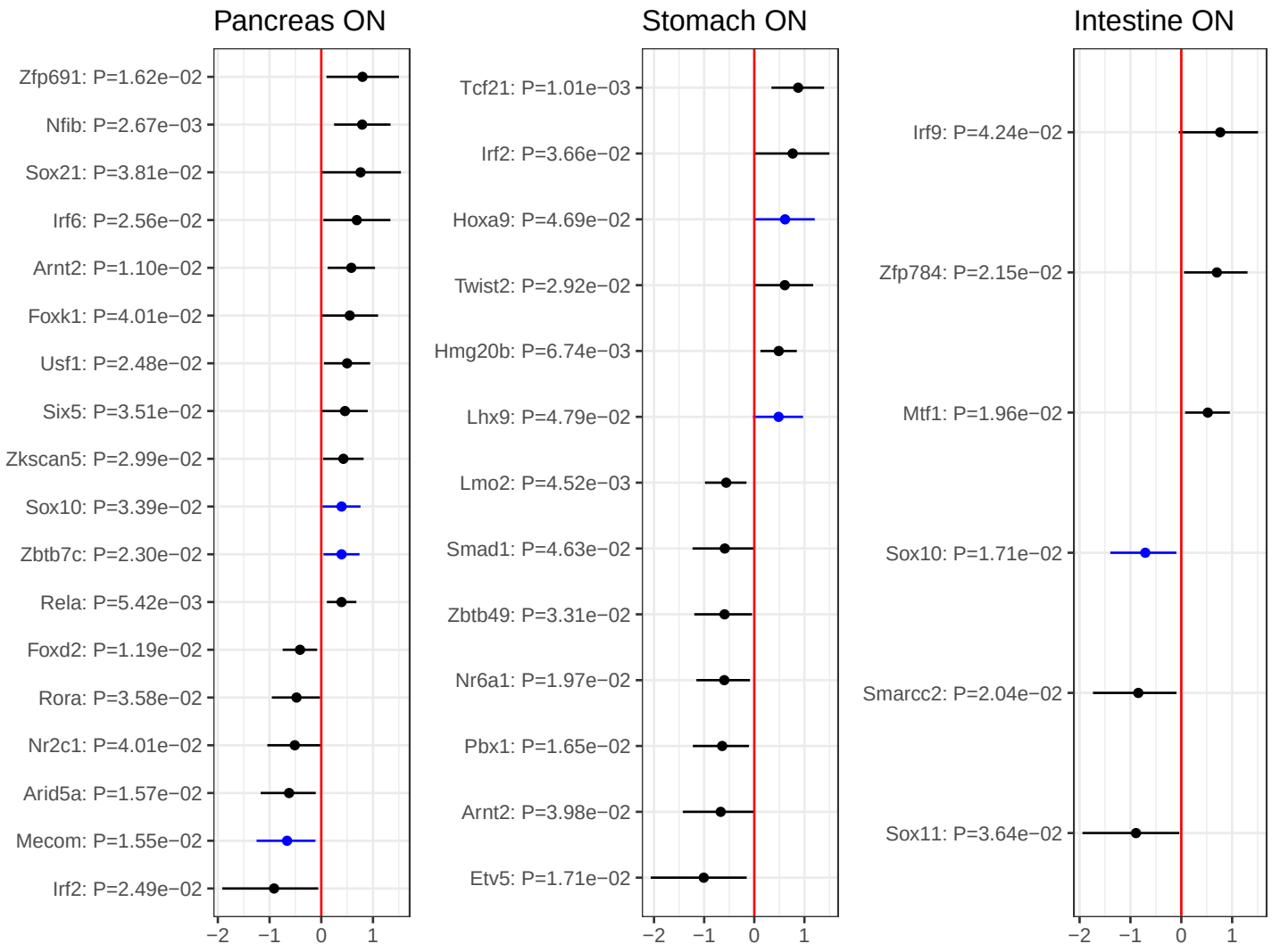
```
## Label genes based on which group/cluster they belong to (from heatmap/clustering)
grp.tab <- to.plot.tf.tissueMean$group
names(grp.tab) <- as.character(to.plot.tf.tissueMean$gene)
mouse.fantomT.melt <- subset(melt(mouse.fantomT.subnet))

mouse.fantomT.melt$grp <- grp.tab[as.character(mouse.fantomT.melt$X1)] #Which group does TF belong to
mouse.fantomT.melt$grp2 <- row.cut[as.character(mouse.fantomT.melt$X2)] #Which group target belong to
```

2.5.4 Calculate enrichment of regulatory edges per gene cluster

```
## Wrapper function to test for enrichment of edges using a Fisher's exact test (Fig Supp 1A-D)
fish.plot <- function(melted.network, a.cutoff, grpA, n=10, pv=0.1, addTFs=FALSE){
    cut.sub <- subset(melted.network, value >= a.cutoff)
    list.of.fish <- lapply(unique(cut.sub$X1), function(a.tf){
        if(addTFs){
            tfs <- unique(intersect(cut.sub$X1, cut.sub$X2[cut.sub$X1==a.tf]))
        } else {
            tfs <- a.tf
        }
        a.row <- c(length(unique(subset(cut.sub, X1%in%tfs & grp2 == grpA)$X2)),
                   length(unique(subset(cut.sub, X1%in%tfs & grp2 != grpA)$X2)))
        n1 <- nrow(unique(subset(melted.network, grp2==grpA)[,c("X2","grp2")]))
        n2 <- nrow(unique(subset(melted.network, grp2!=grpA)[,c("X2","grp2")]))
        fisher.test(matrix(c(a.row, c(n1,n2) - a.row), nrow=2, byrow=T))
    })
    names(list.of.fish) <- unique(cut.sub$X1)
    fish.to.plot <- do.call(rbind,lapply(list.of.fish, function(f){
        p <- f$p
        est <- log(f$estimate)
        ci <- log(f$conf.int)
        return(data.frame(pval = p,
                          est = est,
                          ci.min = ci[1],
                          ci.max = ci[2]))
    }))
    fish.to.plot$tf <- paste0(row.names(fish.to.plot), ": P=",
                              formatC(fish.to.plot$pval, format = "e", digits = 2))
    fish.to.plot$tf <- factor(fish.to.plot$tf,
                              levels = as.character(fish.to.plot$tf)[order(fish.to.plot$est)])
    fish.to.plot.top <-
        fish.to.plot[order(fish.to.plot$est),c("tf","est")][
            c(1:n,(nrow(fish.to.plot)-n):nrow(fish.to.plot)),]
    fish.to.plot <- subset(fish.to.plot, tf %in% fish.to.plot.top$tf & pval <= pv) #+
}

cutoff <-1
list.of.fish <- lapply(1:4,function(n) fish.plot(mouse.fantomT.melt, cutoff, grpA=n, n=25, pv=0.05))
names(list.of.fish) <- c("Pancreas OFF", "Stomach ON", "Pancreas ON", "Intestine ON")

list.of.fish.2 <- lapply(names(list.of.fish), function(n){
    fish.to.plot <- list.of.fish[[n]]
    ggplot(fish.to.plot, aes(y=tf, color = gsub(":.*","",tf) %in% de.tf.list)) +
        geom_point(aes(x=est)) +
        geom_segment(aes(x=ci.min, xend=ci.max, yend=tf)) + geom_vline(xintercept=0, color="red") +
        scale_color_manual(values=c("black","blue")) + guides(color=guide_legend(title="isDE")) +
        ylab(NULL) + xlab(NULL) + theme_bw() + theme(legend.position="none") + ggtitle(n)
})
names(list.of.fish.2) <- names(list.of.fish)
```

2.5.5 Figure S1 B-D

```
##pdf(paste0(plot.dir,"Fig_S1BCD_niche_TFs.pdf"),width=10,height=7.5)
print(ggarrange(list.of.fish.2[["Pancreas ON"]],list.of.fish.2[["Stomach ON"]],
                list.of.fish.2[["Intestine ON"]], ncol=3))
```




```
##dev.off()
```

2.6 Analysis of WT vs Mutant pancreas gene lists (Fig 5A, Fig5C)

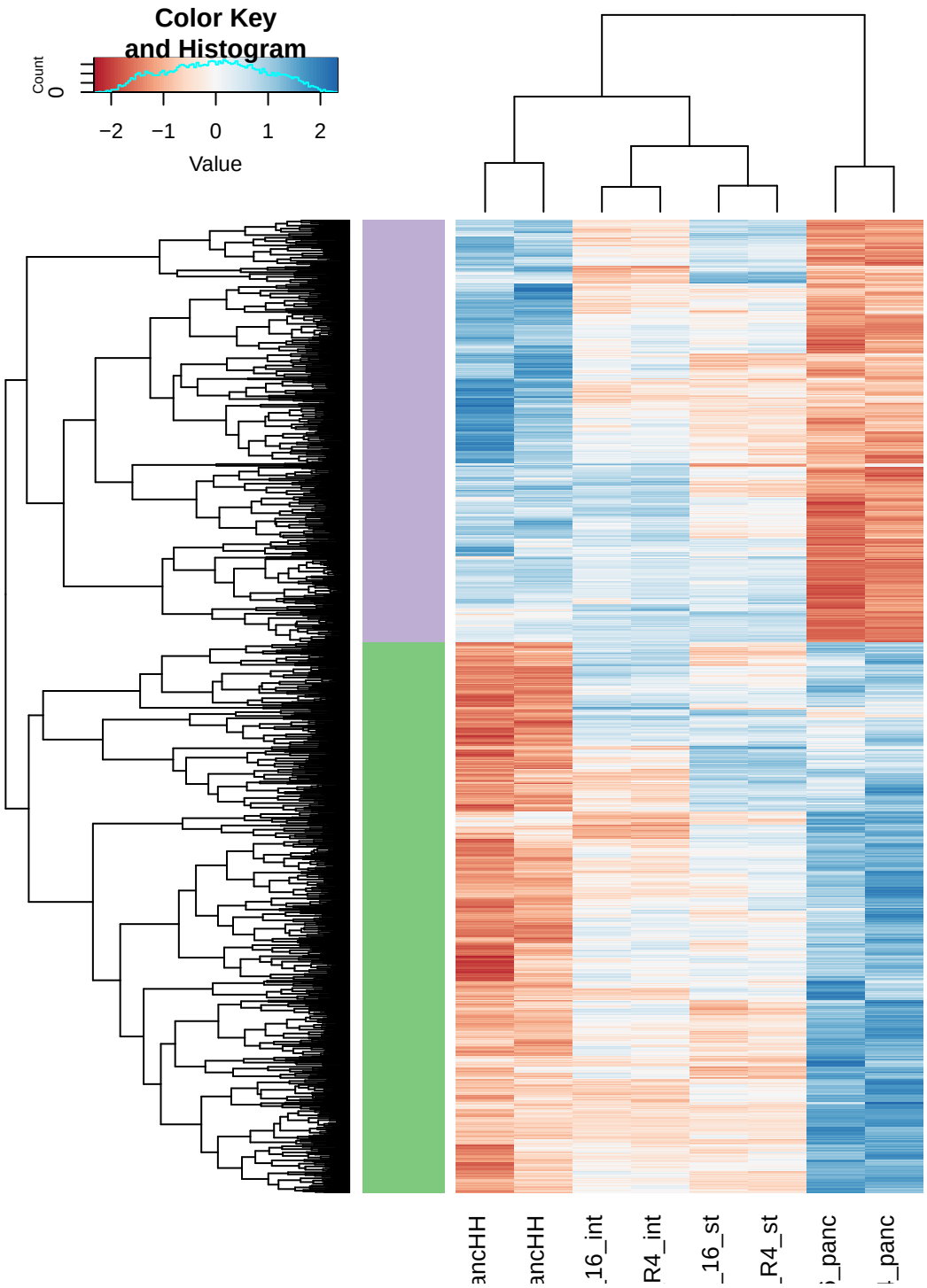
```
## Setup gene lists for making Venn
any.sig.gene.all <- as.character(unique(any.sig.gene.mat[,1]))
any.sig.gene.panc <- as.character(unique(any.sig.gene.mat[grepl("^panc",any.sig.gene.mat[,2]),1]))
any.sig.gene.panc.vMut <-
    as.character(unique(any.sig.gene.mat[any.sig.gene.mat$contrast == "panc_vs_pancHH",1]))
any.sig.gene.panc.vSt <-
    as.character(unique(any.sig.gene.mat[any.sig.gene.mat$contrast == "panc_vs_st",1]))
any.sig.gene.panc.vInt <-
    as.character(unique(any.sig.gene.mat[any.sig.gene.mat$contrast == "panc_vs_int",1]))
any.sig.gene.panc.nMut <-
    any.sig.gene.panc[!any.sig.gene.panc %in% any.sig.gene.panc.vMut]
```

2.6.1 Clustering of WTvsMut panc genes

```
## Scale the counts so that we're looking at differences rather than driven by high/low expressers
to.plot.panc <- to.plot.scale[any.sig.gene.panc.vMut,]
colnames(to.plot.panc) <- colnames(to.plot)

to.plot.panc.dist <- mandist(to.plot.panc)
dendro.panc <- manhclust(to.plot.panc.dist)
row.cut.panc <- cutree(dendro.panc, k=2)
row.colors.panc <- factor(row.cut.panc)
suppressWarnings(levels(row.colors.panc) <- brewer.pal(2,"Accent"))

any.sig.gene.panc.vMut.list <- lapply(c(1,2), function(n){
    any.sig.gene.panc.vMut[row.cut.panc == n]
})
names(any.sig.gene.panc.vMut.list) <- c("TurnedOFF","TurnedON")
```

2.6.2 Figure 5A

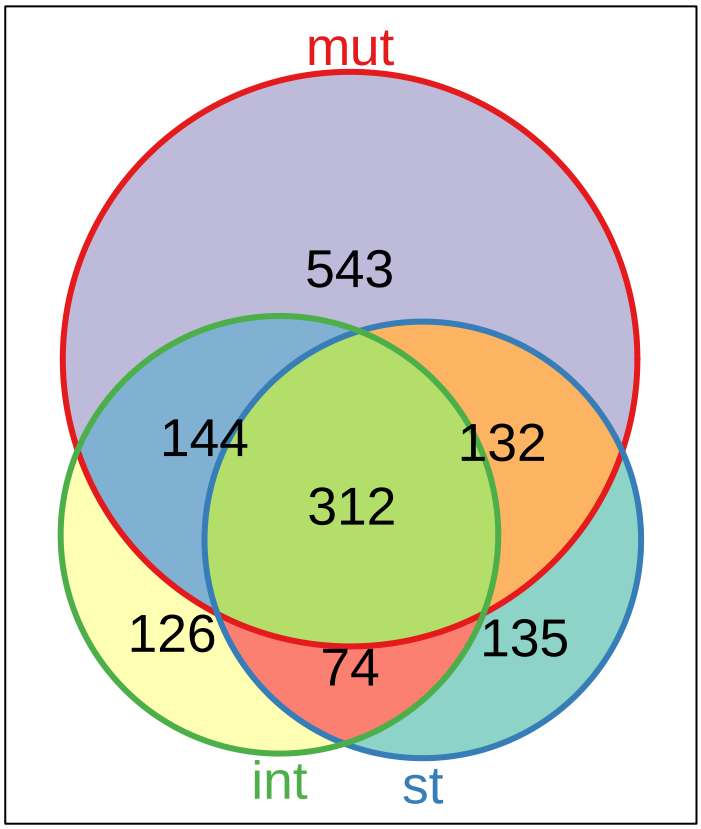
```
## Figure 5A
## pdf(paste0(plot.dir,"Fig5A_withMut_HM.pdf"),width=6,height=7.5)
heatmap.2(to.plot.panc, distfun=mandist, hclust=manhclust, trace="none",
          col=colorRampPalette(brewer.pal(9, "RdBu"))(100), labRow=F,
          lhei=c(.2,1), lwid=c(0.8,1.2), margins=c(4,4), RowSideColors=as.character(row.colors.panc))
```




```
## dev.off()
```

2.6.3 Figure 5C

```
## Venn diagram of genes DE in each contrast w/ wt pancreas
##pdf(paste0(plot.dir,"Fig5C_pancVENN.pdf"),width=4,height=6)
plot(Venn(Sets=list(mut=any.sig.gene.panc.vMut, st=any.sig.gene.panc.vSt,
                    int=any.sig.gene.panc.vInt)))
```




```
##dev.off()
```

2.7 Enrichment analysis for HH and BMP pathway genes

```
list.of.HH <- c("Gli3", "Gli2", "Myc", "Boc", "Cdk6", "Sfrp1", #Provided by Theo
                "Ptch2", "Ptch1", "Gli1", "Hhip", "Bmp4", "Foxl1",
                "Foxf1", "Jag2","Ccnd2","Ccnd1","Foxm1",
                "Bcl2", "Cflar", "Prdm1", "Grem1","Snai1","Snai2",
                "Zeb1","Zeb2","Twist2","Foxc2","Fst","Mycn",
                "Wnt2b","Wnt5a","Bmi1","Lgr5","Cd44","Prom1","Pthlh") #PMID: 19860666

bmp.15 <- read.table("reference_data/GO_lists/SMAD1_5.csv",sep="\t", header=F, skip=1)
bmp.15 <- as.character(subset(bmp.15, !is.na(V4) & !is.na(V5))$V2)
bmp.4 <- read.table("reference_data/GO_lists/SMAD4.txt.csv",sep="\t", header=F, skip=1)
bmp.4 <- as.character(subset(bmp.4, !is.na(V4) & !is.na(V5))$V2)
#SMAD gene lists obtained from PMID:19926752 Supp Table S3 and S4

bmp.targets <- unique(c(bmp.15, bmp.4))

urow.cut <- unique(row.cut)
names(urow.cut) <- c("Pancreas OFF", "Stomach ON", "Pancreas ON", "Intestine ON")
```

```
upanc.cut <- unique(row.cut.panc)
names(upanc.cut) <- c("Turned OFF", "Turned ON")


## Phyper for HedgeHog target genes for WT tissue contrasts
sapply(urow.cut, function(i){
    phyper(q=sum(list.of.HH %in% any.sig.gene.noMut[row.cut==i]), m=length(list.of.HH),
           n=(nrow(to.plot.scale) - length(list.of.HH)), k=length(any.sig.gene.noMut[row.cut==i]),
           lower.tail =F)
})
```

```
## Pancreas OFF   Stomach ON  Pancreas ON Intestine ON
## 1.058211e-13 2.944539e-01 1.246967e-01 2.130821e-03
```

```
## Phyper for Hedgehog target genes for Mutant vs WT panc
sapply(upanc.cut, function(i){
    phyper(q=sum(list.of.HH %in% any.sig.gene.panc.vMut[row.cut.panc==i]), m=length(list.of.HH),
           n=(nrow(to.plot.scale) - length(list.of.HH)),
           k=length(any.sig.gene.panc.vMut[row.cut.panc==i]), lower.tail =F)
})
```

```
##   Turned OFF    Turned ON
## 3.136023e-01 1.551838e-15
```

```
## Phyper for BMP target genes for WT tissue contrasts
sapply(urow.cut, function(i){
    phyper(q=sum(bmp.targets %in% any.sig.gene.noMut[row.cut==i]), m=length(bmp.targets),
           n=(nrow(to.plot.scale) - length(bmp.targets)),
           k=length(any.sig.gene.noMut[row.cut==i]), lower.tail =F)
})
```

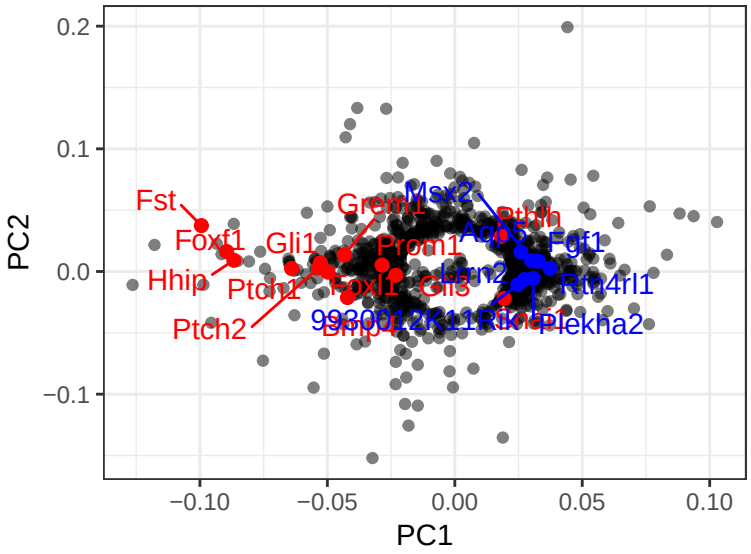
```
## Pancreas OFF   Stomach ON  Pancreas ON Intestine ON
##  0.693274869  0.784632094  0.001566592  0.678401009
```

```
## Phyper for BMP target genes for Mutant vs WT panc
sapply(upanc.cut, function(i){
    phyper(q=sum(bmp.targets %in% any.sig.gene.panc.vMut[row.cut.panc==i]), m=length(bmp.targets),
           n=(nrow(to.plot.scale) - length(bmp.targets)),
           k=length(any.sig.gene.panc.vMut[row.cut.panc==i]), lower.tail =F)
})
```

```
## Turned OFF  Turned ON
##  0.2304137  0.5318209
```

2.7.1 Figure 1E

```
## Label PCA loadings plot with HH or BMP downstream targets (Fig1E)
rld.loadings <- as.data.frame(loadings(rld.pca.noHH))
rld.loadings$gene <- row.names(rld.loadings)
rld.loadings$isHH <- rld.loadings$gene %in% list.of.HH
rld.loadings$isBMP <- rld.loadings$gene %in% bmp.targets
## pdf(paste0(plot.dir,"Fig1E_HH_BMP_pca_loadings.pdf"),width=3,height=3)
ggplot(mapping=aes(x=PC1,y=PC2)) + geom_point(data=rld.loadings,alpha=0.5) +
    geom_point(data=subset(rld.loadings,isHH), color = "red", size=2) +
    geom_point(data=subset(rld.loadings,isBMP), color = "blue", size=2) +
    geom_text_repel(data=subset(rld.loadings, isHH), aes(label = gene),color ="red", size=4) +
    geom_text_repel(data=subset(rld.loadings, isBMP), aes(label = gene),color ="blue", size=4) +
    theme_bw()
```




```
## dev.off()
```

2.8 GO Enrichment for mutant ON and OFF genes

GO Enrichment for mutant ON and OFF genes was obtained using gProfiler as described above in the section "GO enrichments and Construction of EnrichmentMap networks"

2.9 Session Info

```
sessionInfo()

## R version 3.4.1 (2017-06-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.2 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
##  [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
##  [7] LC_PAPER=en_CA.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] biomaRt_2.30.0             pandaR_1.6.0
##  [3] ggpubr_0.2                 magrittr_1.5
##  [5] BiocParallel_1.8.2         fgsea_1.0.2
##  [7] Rcpp_1.0.1                 Vennerable_3.0
##  [9] xtable_1.8-3               reshape_0.8.8
## [11] lattice_0.20-38            RBGL_1.50.0
## [13] graph_1.52.0               gProfileR_0.6.7
```

```
## [15] RColorBrewer_1.1-2         gplots_3.0.1.1
## [17] pcaMethods_1.66.0          gtools_3.8.1
## [19] reshape2_1.4.3             ggrepel_0.8.0
## [21] ggplot2_3.1.0              scales_1.0.0
## [23] DESeq2_1.14.1              SummarizedExperiment_1.4.0
## [25] Biobase_2.34.0             GenomicRanges_1.26.4
## [27] GenomeInfoDb_1.10.3        IRanges_2.8.2
## [29] S4Vectors_0.12.2           BiocGenerics_0.20.0
## [31] knitr_1.22
##
## loaded via a namespace (and not attached):
##  [1] matrixStats_0.54.0   bitops_1.0-6         bit64_0.9-7
##  [4] tools_3.4.1          backports_1.1.3      R6_2.4.0
##  [7] rpart_4.1-13         KernSmooth_2.23-15   Hmisc_4.2-0
## [10] DBI_1.0.0            lazyeval_0.2.2       colorspace_1.4-1
## [13] nnet_7.3-12          withr_2.1.2          tidyselect_0.2.5
## [16] gridExtra_2.3        bit_1.1-14           compiler_3.4.1
## [19] htmlTable_1.13.1     labeling_0.3         caTools_1.17.1.2
## [22] checkmate_1.9.1      hexbin_1.27.2        genefilter_1.56.0
## [25] stringr_1.4.0        digest_0.6.18        foreign_0.8-71
## [28] XVector_0.14.1       base64enc_0.1-3      pkgconfig_2.0.2
## [31] htmltools_0.3.6      highr_0.8            htmlwidgets_1.3
## [34] rlang_0.3.3          rstudioapi_0.10      RSQLite_2.1.1
## [37] bindr_0.1.1          acepack_1.4.1        dplyr_0.7.6
## [40] RCurl_1.95-4.12      Formula_1.2-3        Matrix_1.2-17
## [43] munsell_0.5.0        stringi_1.4.3        zlibbioc_1.20.0
## [46] plyr_1.8.4           blob_1.1.1           gdata_2.18.0
## [49] crayon_1.3.4         cowplot_0.9.4        splines_3.4.1
## [52] annotate_1.52.1      locfit_1.5-9.1       pillar_1.3.0
## [55] igraph_1.2.4         RUnit_0.4.32         geneplotter_1.52.0
## [58] fastmatch_1.1-0      XML_3.98-1.19        glue_1.3.1
## [61] evaluate_0.13        latticeExtra_0.6-28  data.table_1.12.0
## [64] gtable_0.3.0         purrr_0.3.2          assertthat_0.2.1
## [67] xfun_0.6             survival_2.44-1.1    tibble_1.4.2
## [70] AnnotationDbi_1.36.2 memoise_1.1.0        bindrcpp_0.2.2
## [73] cluster_2.0.8
```