```
\documentclass{article}

\begin{document}
\SweaveOpts{concordance=TRUE}

<<>>=
library(tximportData)
library(TxDb.Mmusculus.UCSC.mm10.knownGene)
library(tximport)
library(readr)
library(DESeq2)
library(RColorBrewer)
library(pheatmap)
library(PoiClaClu)
library(ggplot2)
library(biomaRt)
library(org.Mm.eg.db)
library(WGCNA)
library(genefilter)
library(EnsDb.Mmusculus.v79)

setwd("~/OneDrive - University of Texas Southwestern/Research/Cleaver/")

# Specify location of alignment files
ngs.root <- "/Volumes/CarrollNGS/"
#ngs.root <- "~/LabNGS/"
csv.file <- file.path("master.sample.table.csv")
sample.table <- read.csv(csv.file, row.names=1)

sample.files <- file.path(paste0(ngs.root,
                                 sample.table$dir,
                                 sample.table$name,
                                 "_quant/quant.sf"))
file.exists(sample.files)
names(sample.files) <- row.names(sample.table)

edb <- EnsDb.Mmusculus.v79
txs <- transcripts(edb)
tx2gene <- data.frame(TXNAME=txs$tx_name, GENEID=txs$gene_id)

txi <- tximport(sample.files,
                type = "salmon",
                tx2gene=tx2gene,
                dropInfReps=TRUE)

dds <- DESeqDataSetFromTximport(txi, sample.table, ~condition)

## Exploratory analysis and visualization
# Pre-filtering the dataset
nrow(dds)
```

```
# apply the most minimal filtering rule: removing rows of the
DESeqDataSet
# that have no counts, or only a single count across all samples
dds <- dds[rowSums(counts(dds)) > 1, ]
nrow(dds)

rld <- rlog(dds, blind=FALSE)
vsd <- vst(dds, blind=FALSE)
par(mfrow=c(1,3))
dds <- estimateSizeFactors(dds)
lims <- c(-2, 20)
plot(log2(counts(dds, normalized=TRUE)[, 1:2] + 1),
     pch=16,
     cex=0.3,
     main="log2(x + 1)",
     xlim=lims,
     ylim=lims)
plot(assay(rld)[,1:2],
     pch=16,
     cex=0.3,
     main="rlog",
     xlim=lims,
     ylim=lims)
plot(assay(vsd)[,1:2],
     pch=16,
     cex=0.3,
     main="VST",
     xlim=lims,
     ylim=lims)

# Sample distances
sampleDists <- dist(t(assay(rld)))
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rld$tissue,
                                    rld$age,
                                    rld$cell.type,
                                    sep=":")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)

poisd <- PoissonDistance(t(counts(dds)))
samplePoisDistMatrix <- as.matrix(poisd$dd)
rownames(samplePoisDistMatrix) <- paste(rld$tissue,
                                        rld$age,
                                        rld$cell.type,
                                        sep=":")
colnames(samplePoisDistMatrix) <- NULL
```

```
pheatmap(samplePoisDistMatrix,
         clustering_distance_rows = poisd$dd,
         clustering_distance_cols = poisd$dd,
         col=colors)

# PCA plot
plotPCA(rld, intgroup=c('tissue', 'age', 'cell.type'))

pcaData <- plotPCA(rld,
                   intgroup=c('tissue', 'age', 'cell.type'),
                   returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=age, shape=cell.type)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "%variance")) +
  coord_fixed()

# MDS plot
mdsData <- data.frame(cmdscale(sampleDistMatrix))
mds <- cbind(mdsData, as.data.frame(colData(rld)))
ggplot(mds, aes(X1, X2, color=tissue, shape=cell.type)) +
  geom_point(size=3) +
  coord_fixed()

mdsPoisData <- data.frame(cmdscale(samplePoisDistMatrix))
mdsPois <- cbind(mdsPoisData, as.data.frame(colData(dds)))
ggplot(mdsPois, aes(X1, X2, color=tissue, shape=cell.type)) +
  geom_point(size=3) +
  coord_fixed()

# WGCNA analysis
options(stringsAsFactors = FALSE);
expression.data <- t(assay(rld))
gsg = goodSamplesGenes(expression.data, verbose = 3);
gsg$allOK
if (!gsg$allOK) {
  # Optionally, print the gene and sample names that were removed:
  if (sum(!gsg$goodGenes)>0)
    printFlush(paste("Removing genes:",
                     paste(names(expression.data)[!gsg$goodGenes],
                           collapse = ", ")));
  if (sum(!gsg$goodSamples)>0)
    printFlush(paste("Removing samples:",
                     paste(rownames(expression.data)[!gsg$goodSamples],
                           collapse = ", ")));
  # Remove the offending genes and samples from the data:
  expression.data = expression.data[gsg$goodSamples, gsg$goodGenes]
}
sample.tree = hclust(dist(expression.data), method = "average");
# Plot the sample tree: Open a graphic output window of size 12 by 9
```

```
inches
# The user should change the dimensions if the window is too large or
too small.
sizeGrWindow(12,9)
#pdf(file = "Plots/sampleClustering.pdf", width = 12, height = 9);
par(cex = 0.6);
par(mar = c(0,4,2,0))
plot(sample.tree,
     main = "Sample clustering to detect outliers",
     sub="",
     xlab="",
     cex.lab = 1.5,
     cex.axis = 1.5,
     cex.main = 2)
n.genes = ncol(expression.data)
n.samples = nrow(expression.data)
sample.data <- data.frame(row.names=row.names(sample.table))
for(tissue in unique(sample.table$tissue)) {
  sample.data[, tissue] <- as.integer(sample.table$tissue == tissue)
}
for(age in unique(sample.table$age)) {
  sample.data[, age] <- as.integer(sample.table$age == age)
}
for(cell.type in unique(sample.table$cell.type)) {
  sample.data[, cell.type] <- as.integer(sample.table$cell.type ==
cell.type)
}
sample.data$kidney_endothelium <- sample.data$Kidney *
  sample.data$Endothelium
sample.data$kidney_endothelium_e12.5 <- sample.data$kidney_endothelium *
  sample.data$E12.5
sample.data$kidney_endothelium_e15.5 <- sample.data$kidney_endothelium *
  sample.data$E15.5
sample.data$kidney_endothelium_e18.5 <- sample.data$kidney_endothelium *
  sample.data$E18.5

trait.colors <- numbers2colors(sample.data, signed=FALSE)
plotDendroAndColors(sample.tree, trait.colors,
                    groupLabels = names(sample.data),
                    main = "Sample dendrogram and trait heatmap")

# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=20, by=2))
# Call the network topology analysis function
soft.threshold = pickSoftThreshold(expression.data,
                                   powerVector = powers,
                                   verbose = 5)

# Plot the results:
sizeGrWindow(9, 5)
par(mfrow = c(1,2));
```

```
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding
power
plot(soft.threshold$fitIndices[,1],
     -sign(soft.threshold$fitIndices[,3])*soft.threshold$fitIndices[,2],
     xlab="Soft Threshold (power)",
     ylab="Scale Free Topology Model Fit,signed R^2",
     type="n",
     main = paste("Scale independence"));
text(soft.threshold$fitIndices[,1],
     -sign(soft.threshold$fitIndices[,3])*soft.threshold$fitIndices[,2],
     labels=powers,
     cex=cex1,
     col="red");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.90,col="red")
# Mean connectivity as a function of the soft-thresholding power
plot(soft.threshold$fitIndices[,1],
     soft.threshold$fitIndices[,5],
     xlab="Soft Threshold (power)",
     ylab="Mean Connectivity", type="n",
     main = paste("Mean connectivity"))
text(soft.threshold$fitIndices[,1],
     soft.threshold$fitIndices[,5],
     labels=powers,
     cex=cex1,
     col="red")
soft.power = 10;
adjacency.matrix = adjacency(expression.data,
                             power = soft.power);

top.variable.genes <- rownames(rld[head(order(rowVars(assay(rld)),
                                              decreasing=TRUE),
                                        4830), ])
reduced.expression.data <- t(assay(rld[top.variable.genes, ]))

reduced.adjacency.matrix <- adjacency.matrix[top.variable.genes,
                                             top.variable.genes]

similarity = TOMsimilarity(reduced.adjacency.matrix);
dissimilarity = 1-similarity

# Call the hierarchical clustering function
gene.tree = hclust(as.dist(dissimilarity), method = "average");
# Plot the resulting clustering tree (dendrogram)
sizeGrWindow(12,9)
plot(gene.tree,
     xlab="",
     sub="",
     main = "Gene clustering on TOM-based dissimilarity",
     labels = FALSE,
```

```
     hang = 0.04);

# We like large modules, so we set the minimum module size relatively
high:
min.module.size = 30;
# Module identification using dynamic tree cut:
dynamic.modules = cutreeDynamic(dendro = gene.tree,
                                distM = dissimilarity,
                                deepSplit = 2,
                                pamRespectsDendro = FALSE,
                                minClusterSize = min.module.size);
table(dynamic.modules)

# Convert numeric lables into colors
dynamic.colors = labels2colors(dynamic.modules)
table(dynamic.colors)
# Plot the dendrogram and colors underneath
sizeGrWindow(8,6)
plotDendroAndColors(gene.tree, dynamic.colors, "Dynamic Tree Cut",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05,
                    main = "Gene dendrogram and module colors")

# Calculate eigengenes
module.eigengene.list = moduleEigengenes(reduced.expression.data,
                                         colors = dynamic.colors)
module.eigengenes = module.eigengene.list$eigengenes
# Calculate dissimilarity of module eigengenes
module.eigengene.dissimilarity = 1-cor(module.eigengenes);
# Cluster module eigengenes
module.eigengene.tree = hclust(as.dist(module.eigengene.dissimilarity),
                               method = "average");
# Plot the result
sizeGrWindow(7, 6)
plot(module.eigengene.tree, main = "Clustering of module eigengenes",
     xlab = "", sub = "")
module.eigengene.dissimilarity.threshold = 0.25
# Plot the cut line into the dendrogram
abline(h=module.eigengene.dissimilarity.threshold, col = "red")
# Call an automatic merging function
merged.modules = mergeCloseModules(reduced.expression.data,
                                   dynamic.colors,
                                   cutHeight =

module.eigengene.dissimilarity.threshold,
                                   verbose = 3)
# The merged module colors
merged.colors = merged.modules$colors;
# Eigengenes of the new merged modules:
merged.module.eigengenes = merged.modules$newMEs;
```

```
sizeGrWindow(12, 9)
plotDendroAndColors(gene.tree,
                    cbind(dynamic.colors, merged.colors),
                    c("Dynamic Tree Cut", "Merged dynamic"),
                    dendroLabels = FALSE,
                    hang = 0.03,
                    addGuide = TRUE,
                    guideHang = 0.05)

# Rename to moduleColors
module.colors= merged.colors
# Construct numerical labels corresponding to the colors
color.order = c("grey", standardColors(50));
module.labels = match(module.colors, color.order) - 1;
module.eigengenes = merged.module.eigengenes;

n.genes <- ncol(reduced.expression.data)

# Quantifying module-genotype associations
module.eigengenes <- orderMEs(module.eigengenes)
module.sample.cor <- cor(module.eigengenes, sample.data, use="p")
module.sample.p <- corPvalueStudent(module.sample.cor, n.samples)
sizeGrWindow(12,8)
# Will display correlations and their p-values
text.matrix = paste(signif(module.sample.cor, 2), "\n(",
                    signif(module.sample.p, 1), ")", sep = "");
dim(text.matrix) <- dim(module.sample.cor)
# Display the correlation values within a heatmap plot
labeledHeatmap(Matrix = module.sample.cor,
               xLabels = names(sample.data),
               yLabels = names(module.eigengenes),
               ySymbols = names(module.eigengenes),
               colorLabels = FALSE,
               colors = blueWhiteRed(50),
               textMatrix = text.matrix,
               setStdMargins = FALSE,
               cex.text = 0.5,
               zlim = c(-1,1),
               main = paste("Module-sample relationships"))

kidney_endothelium <- as.data.frame(sample.data$kidney_endothelium)
endothelium <- as.data.frame(sample.data$Endothelium)
names(kidney_endothelium) <- "Kidney Endothelium"
names(endothelium) <- "Endothelium"
module.names <- substring(names(module.eigengenes), 3)
gene.module.membership <- as.data.frame(cor(reduced.expression.data,
                                            module.eigengenes,
                                            use="p"))
module.membership.p <- as.data.frame(
  corPvalueStudent(
    as.matrix(gene.module.membership),
```

```
    n.samples))
names(gene.module.membership) <- paste("MM",
                                       module.names,
                                       sep="")
names(module.membership.p) <- paste("p.MM",
                                    module.names,
                                    sep="")
gene.kidney_endothelium.significance <- as.data.frame(
  cor(reduced.expression.data,
      kidney_endothelium,
      use="p"))
gene.endothelium.significance <- as.data.frame(
  cor(reduced.expression.data,
      endothelium,
      use="p"))
gene.kidney_endothelium.significance.p <- as.data.frame(
  corPvalueStudent(
    as.matrix(
      gene.kidney_endothelium.significance),
    n.samples))
gene.endothelium.significance.p <- as.data.frame(
  corPvalueStudent(
    as.matrix(gene.endothelium.significance),
    n.samples))
names(gene.kidney_endothelium.significance) <- paste("GS.",

names(kidney_endothelium),
                                                      sep="")
names(gene.endothelium.significance) <- paste("GS.",
                                              names(endothelium),
                                              sep="")
names(gene.kidney_endothelium.significance.p) <- paste("p.GS.",

names(kidney_endothelium),
                                                        sep="")
names(gene.endothelium.significance.p) <- paste("p.GS.",
                                                names(endothelium),
                                                sep="")

module <- "royalblue"
column <- match(module, module.names)
module.genes <- module.colors == module
sizeGrWindow(7, 7);
par(mfrow = c(1,1));
verboseScatterplot(abs(gene.module.membership[module.genes, column]),

abs(gene.kidney_endothelium.significance[module.genes, 1]),
                   xlab = paste("Module Membership in", module,
"module"),
                   ylab = "Gene significance for kidney endothelium",
                   main = paste("Module membership vs. gene
```

```
significance\n"),
                    cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col =
module)

colnames(reduced.expression.data)[module.colors == "royalblue"]
gene2symbol <- select(org.Mm.eg.db,
                      keys=colnames(reduced.expression.data),
                      columns=c("ENSEMBL", "SYMBOL"),
                      keytype="ENSEMBL",
                      multiVals="first")
orphaned <- gene2symbol[which(is.na(gene2symbol$SYMBOL)), ]$ENSEMBL
genes <- colnames(reduced.expression.data)
genes.to.annotation <- match(genes,
                             gene2symbol$ENSEMBL)
gene.info <- data.frame(ensembl_gene_id=
                          gene2symbol$ENSEMBL[genes.to.annotation],
                        symbol=
                          gene2symbol$SYMBOL[genes.to.annotation],
                        module.color=module.colors,
                        gene.endothelium.significance,
                        gene.endothelium.significance.p,
                        gene.kidney_endothelium.significance,
                        gene.kidney_endothelium.significance.p)

# Order modules by their significance for kidney endothelium cell type
module.order <- order(-abs(cor(module.eigengenes,
                               endothelium,
                               use="p")))
# Add module membership information in the chosen order
for (module in 1:ncol(gene.module.membership)) {
  old.names = names(gene.info)
  gene.info = data.frame(gene.info,
                         gene.module.membership[, module.order[module]],
                         module.membership.p[, module.order[module]]);
  names(gene.info) = c(old.names,
                       paste("MM.",
                             module.names[module.order[module]],
                             sep=""),
                       paste("p.MM.",
                             module.names[module.order[module]],
                             sep=""))
}

gene.order <- order(gene.info$module.color,
                    -abs(gene.info$GS.Endothelium))
gene.info <- gene.info[gene.order, ]

#sink("sessionInfo_wgcna.txt")
sessionInfo()
#sink()
@
```

```
\end{document}
```