counts_analysis_report

Mary T. Yohannes

8/14/2020

1) First, import needed libraries

More info:

Data was obtained from "Sex, scavengers and chaperones" paper by Levin et al

The paper looks at different natural populations of Symbiodiniaceae

2 populations (South Molle (SM) = sensitive, Magnetic Island (MI) = tolerant) -> 6 samples each -> 4 replicates = 24 samples each -> two temperatutres (27 and 32C) – day 1 doesn't have 32C -> three time points (days 1, 9, 13)

Tutorials used to help with the analyses:

edgeR, limma - https://www.bioconductor.org/packages/devel/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

WGCNA - https://github.com/iscb-dc-rsg/2016-summer-workshop/tree/master/3B-Hughitt-RNASeq-Coex-Network-Analysis/tutorial#co-expression-network-construction & https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/Rpackages/WGCNA/Tutorials/index.html

Unless we needed to change them due to the nature of our dataset and results, the values used for the options/parameters were taken from these tutorials

2) Import samples' info

```
# make sure results are reproducible
set.seed(1)

# set the working directory
setwd("/projectnb2/incrna/mary_lncrna/R/repeat")

# originally downloaded from ncbi web from Run Selector (edited metadata with
only the columns we are interested in)
samples <- read.table("/projectnb/incrna/mary_lncrna/R/samples.txt", header =
TRUE, sep=",", stringsAsFactors = F)

# rename columns
colnames(samples)[1:5] <-
c("sample_id","population","time_point","temp","tolerance")

# substitute "day -1" with "day 1" in the time_point column
samples$time_point <- gsub("-1", "1", samples$time_point)

# unzip the count files and place them in another directory - run on the
command line/terminal and only need to run once
#for f in *.gz ; do gunzip -c "$f" >
/projectnb/incrna/mary_lncrna/R/repeat/"${f%.*}" ; done
```

3-a) Access the salmon output files and read them in as matrices of their abundance, counts, and length

```
# set up the names of the unzipped files so they can be imported/read
file_names <- paste(samples$sample_id, "_1.transcripts_quant.quant.sf", sep =
"")

# run tximport - outputs abundance, counts, and length matrices (this
basically goes through each count file for each sample, takes the columns:
EffectiveLength, TPM, NumReads, and puts them in separate matrices with
samples as columns and genes as rows)
txi <- tximport(file_names, type="salmon", txOut=TRUE,
countsFromAbundance="lengthScaledTPM")

## reading in files with read_tsv

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48

# create a DGEList object from the "counts" matrix
cts <- DGEList(txi$counts)
#cts$counts

# rename the column names using the "sample_id" column in the "samples" file
that was imported at the beginning
colnames(cts) <- samples$sample_id
```

3-b) Another way to do step 3-a by only accessing the number of reads coulmn (outputs just a count matrix) and create a DGEList-object

```
# # read 5 rows of the first file in the "file_names" list (file located in
the current directory)
# read.delim(file_names[1], nrow=5)
#
# # only interested in the first (gene name) and 5th (number of reads)
columns - readDGE only takes 2 columns
# dgeObj <- readDGE(file_names, columns=c(1,5))
#
# # dgeObj is the same as cts except cts count values have more decimal
points and the samples element doesn't include the files column
#
# # shorten the sample names and set it to be the same as the "sample_id"
column in the "samples" file that was imported at the beginning
# colnames(dgeObj) <- samples$sample_id
```

cts is the file that is used for the rest of the analyses

4) Edit the “samples” element in the DGEList-object to count for the different conditions of the experiment (according to the samples tables imported earlier)

```
# update "group" column to categorize the samples into the two populations
# convert the column into a character type in order to replace it
cts$samples$group <- as.character(cts$samples$group)
# update
cts$samples$group[colnames(cts) %in% samples$sample_id] <-
str_sub(samples$population, -3, -2)
# convert back to a factor
cts$samples$group <- as.factor(cts$samples$group)

# add a column to separate samples into three time points
cts$samples$day <- as.factor(str_sub(samples$time_point, -2,-1))

# add a column to separate samples into two temperatures
cts$samples$temp <- as.factor(samples$temp)

# variables used in step 6 - this line needs to be run prior to filtering the
data
pre_lcpm <- cpm(cts, log=TRUE)

# before filtering, the average library size of our dataset was about 3.8
million, so L approximates to 3.76 and the minimum log-CPM value for each
```

```
sample becomes log2(2/3.76) = -0.91; a count of zero for this data maps to a
log-CPM value of -0.91 after adding the prior count or offset
L <- mean(cts$samples$lib.size) * 1e-6
M <- median(cts$samples$lib.size) * 1e-6
c(L, M) # 3.758108 3.837114

## [1] 3.758108 3.837114
```

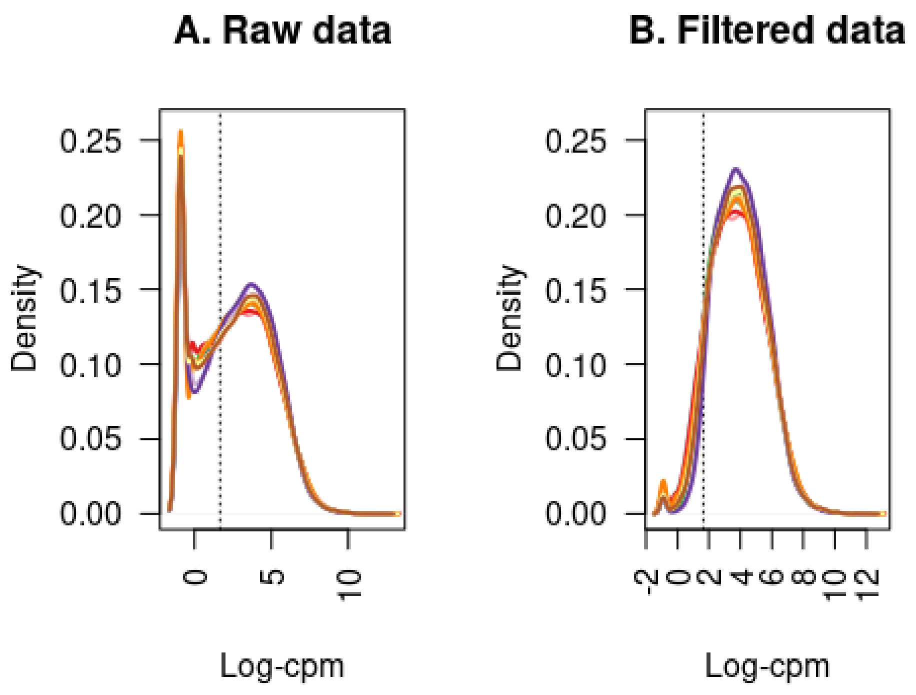
5) Removing genes that are lowly expressed

```
# 1.4% of genes in the dataset have zero counts across all 48 samples
table(rowSums(cts$counts==0)==48)

##
## FALSE  TRUE
## 47491   664

# filterByExpr (function in the edgeR package) filters out genes while
keeping the ones with worthwhile counts
dim(cts) # 48155 genes

## [1] 48155    48

keep <- filterByExpr(cts, group=cts$samples$group)
cts <- cts[keep,, keep.lib.sizes=FALSE]
dim(cts) # 32095 genes - about 2/3 (67%) of the number that we started with

## [1] 32095    48
```

6) Density plots of log-CPM values of each sample for raw pre-filtered data and post-filtered data

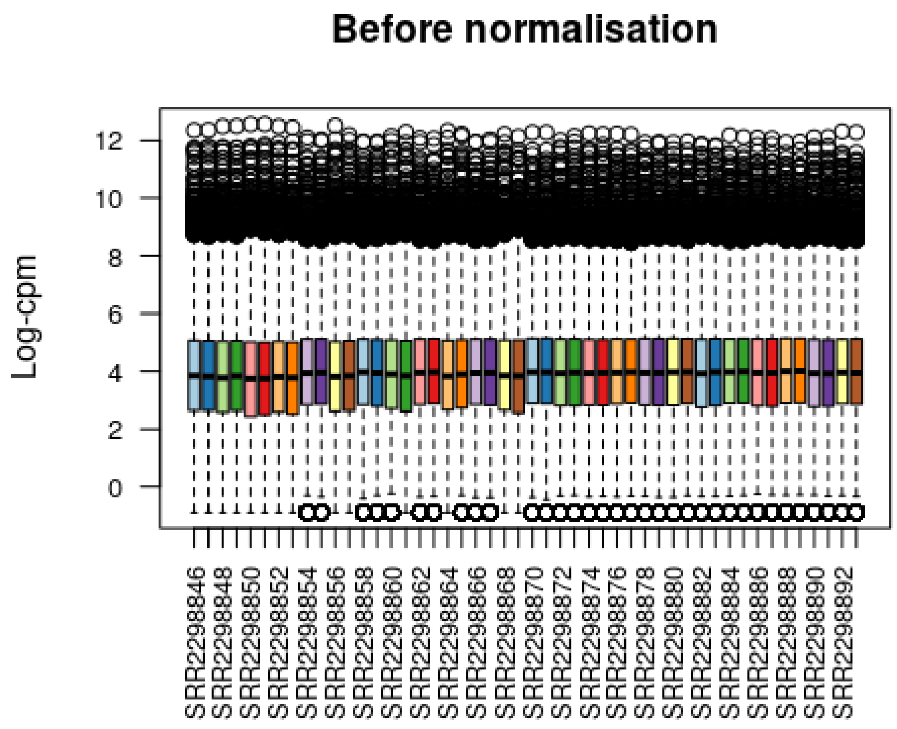
```
# code taken from the online tutorial

# raw data plot
lcpm.cutoff <- log2(10/M + 2/L) # log-CPM threshold used in the filtering
step (dotted vertical lines in the graphs)
col <- brewer.pal(12, "Paired")
par(mfrow=c(1,2))
plot(density(pre_lcpm[,1]), col=col[1], lwd=2, ylim=c(0,0.26), las=2,
main="", xlab="")
title(main="A. Raw data", xlab="Log-cpm")
abline(v=lcpm.cutoff, lty=3)
for (i in 2:ncol(cts)){
den <- density(pre_lcpm[,i])
lines(den$x, den$y, col=col[i], lwd=2)
}

# filtered data plot
lcpm <- cpm(cts, log=TRUE)
```

```
plot(density(lcpm[,1]), col=col[1], lwd=2, ylim=c(0,0.26), las=2, main="",
xlab="")
title(main="B. Filtered data", xlab="Log-cpm")
abline(v=lcpm.cutoff, lty=3)
for (i in 2:ncol(cts)){
den <- density(lcpm[,i])
lines(den$x, den$y, col=col[i], lwd=2)
}
```



7) Normalising gene expression distributions by the method of trimmed mean of M-values (TMM) - using calcNormFactors function in edgeR

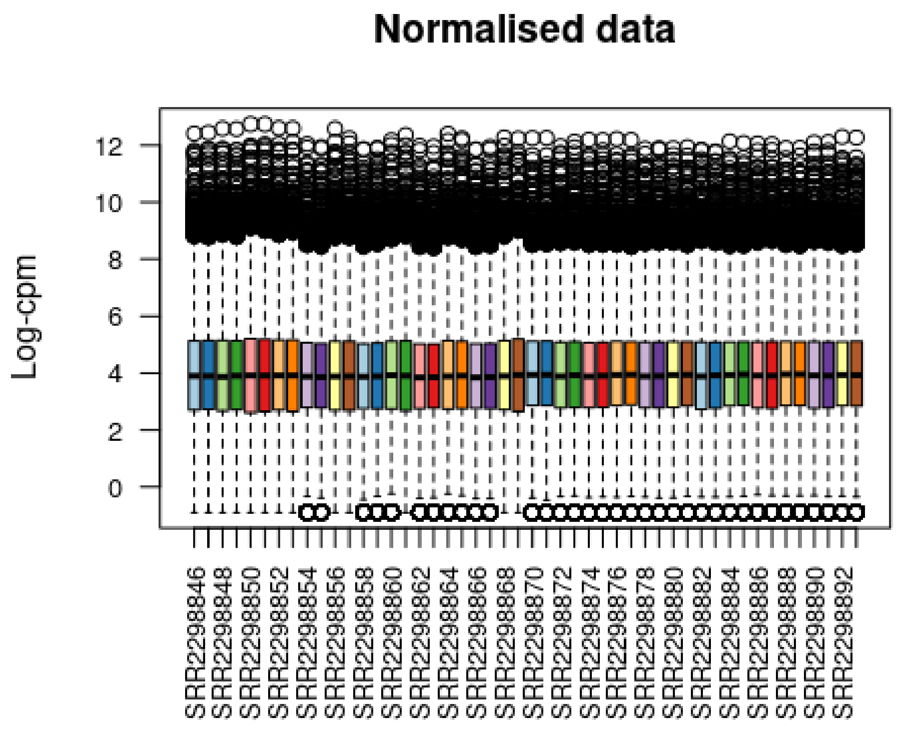
```
# before normalisation
boxplot(lcpm, las=2, col=col, main="", cex.axis = 0.8)
title(main="Before normalisation",ylab="Log-cpm")
```



```
# normalisation factors = scaling factors for the library sizes
# for our dataset, the effect of TMM-normalisation is mild - the scaling
factors are all relatively close to 1
cts <- calcNormFactors(cts, method = "TMM")
cts$samples$norm.factors
```

```
##  [1] 0.9584700 0.9429469 0.9288287 0.9461696 0.8802422 0.8843035 0.9192387
##  [8] 0.9013255 1.0409520 1.0663160 0.9464189 0.9615709 1.0674357 1.0355979
## [15] 0.9803898 0.9522853 1.0752784 1.0823649 0.9546578 0.9888886 1.0683324
## [22] 1.0615210 0.9626648 0.9308574 1.0195543 1.0144365 1.0277840 1.0169167
## [29] 1.0356674 1.0247569 1.0074077 1.0141186 1.0369555 1.0323246 1.0241358
## [36] 1.0208614 1.0222057 1.0346429 1.0284305 1.0263776 1.0231414 1.0211482
## [43] 1.0263104 1.0332684 1.0072699 1.0068750 1.0122947 1.0067903
```

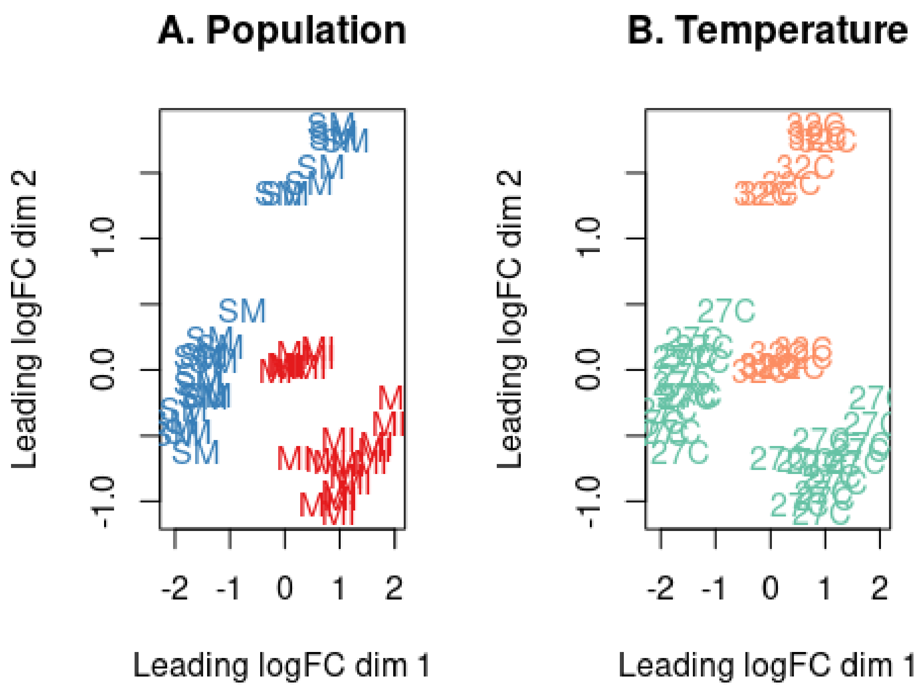
```
# after normalisation - very slight difference
lcpm <- cpm(cts, log=TRUE)
boxplot(lcpm, las=2, col=col, main="", cex.axis = 0.8)
title(main="Normalised data",ylab="Log-cpm")
```



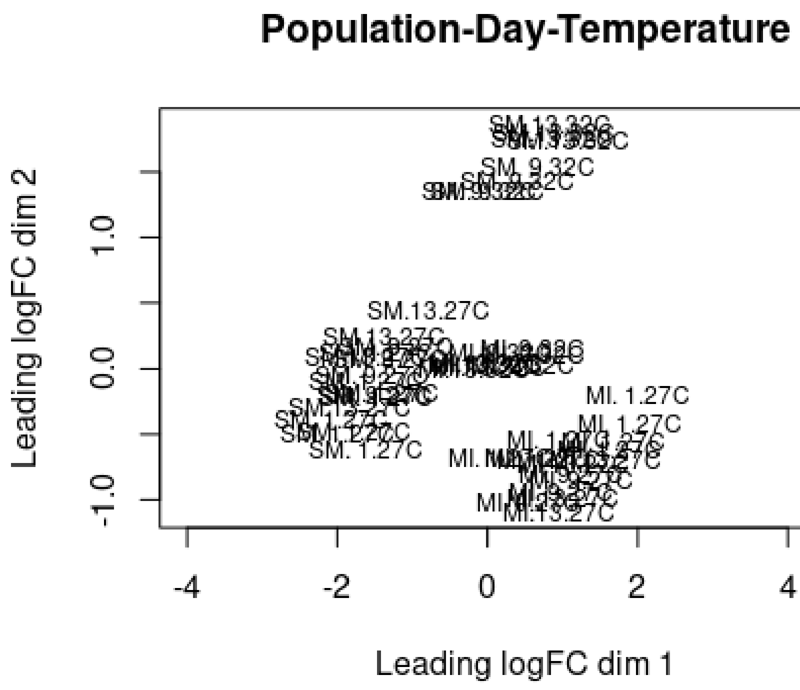
8-a) Unsupervised clustering of samples

```
# observing similarities and dissimilarities between samples in an
unsupervised manner can give us an idea of the extent to which differential
expression can be detected before carrying out formal tests
# first dimension = the leading-fold-change that best separates samples and
explains the largest proportion of variation in the data
# subsequent dimensions = have a smaller effect and are orthogonal to the
dimension before
# samples seem to cluster well within population and temperature groups over
dimension 1 and 2 - mostly similar
par(mfrow=c(1,2))

# population plot
col.group <- cts$samples$group
levels(col.group) <-  brewer.pal(nlevels(col.group), "Set1")

## Warning in brewer.pal(nlevels(col.group), "Set1"): minimal value for n is
3, returning requested palette with 3 different levels

col.group <- as.character(col.group)
plotMDS(lcpm, labels=cts$samples$group, col=col.group)
title(main="A. Population")

# temperature plot
```

```
col.temp <- cts$samples$temp
levels(col.temp) <-  brewer.pal(nlevels(col.temp), "Set2")

## Warning in brewer.pal(nlevels(col.temp), "Set2"): minimal value for n is
3, returning requested palette with 3 different levels

col.temp <- as.character(col.temp)
plotMDS(lcpm, labels=cts$samples$temp, col=col.temp)
title(main="B. Temperature")
```



```
# # day has no effect on expression differences -- no cluster -- left out of
downstream anayses
# col.day <- cts$samples$day
# levels(col.day) <-  brewer.pal(nlevels(col.day), "Set3")
# col.day <- as.character(col.day)
# plotMDS(lcpm, labels=cts$samples$day, col=col.day)
# title(main="B. Time Point (day)")

# all three conditions together
par(mfrow=c(1,1))
plotMDS(lcpm, labels=paste(cts$samples$group, cts$samples$day,
cts$samples$temp, sep="."), cex=0.75, xlim=c(-4, 5))
title(main="Population-Day-Temperature")
```



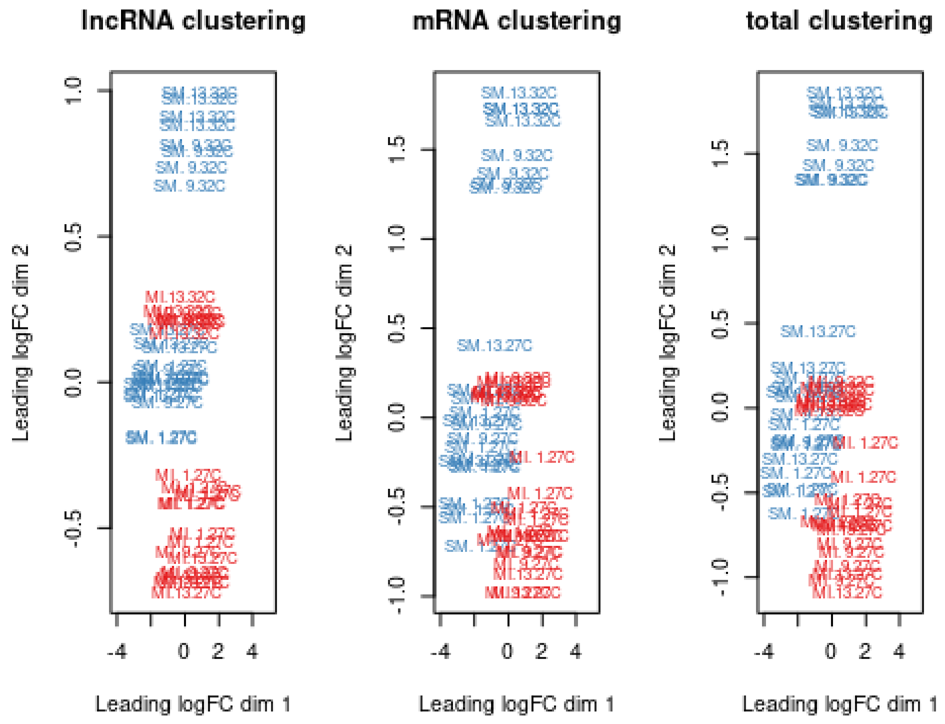
8-b) Separating lncRNAs and mRNAs and creating dgeObj for each

```
# list of genes
gene_id <- data.frame(rownames(cts))

# filter the gene ids that don't start with "mRNA" -- 7,897 lncRNAs
lncRNA <- filter(gene_id, !grepl("mRNA",rownames.cts.))

# filter the gene ids that start with "mRNA" -- 24,198 mRNAs
mRNA <- filter(gene_id, grepl("mRNA",rownames.cts.))

# remove the ".1" at the end of the gene ids and convert to a string of
characters
gene_id1 <- sub('^([^.]+.[^.]+).*', '\\1', lncRNA$rownames.cts.)
gene_id2 <- sub('^([^.]+.[^.]+.[^.]+).*', '\\1', mRNA$rownames.cts.)

# combine the above gene ids: one large character vector with lncRNAs first
and mRNAs next
gene_id <- c(gene_id1,gene_id2)

# separate count data into lncRNAs and mRNAs
lds <- subset(cts$counts, rownames(cts) %in% lncRNA$rownames.cts.) # lncRNA
expression matrix
mds <- subset(cts$counts, rownames(cts) %in% mRNA$rownames.cts.) # mRNA
expression matrix
```

```r
# creating dgeObj for lncRNA and mRNA count files
lcts <- DGEList(lds)
mcts <- DGEList(mds)

# similar to step 4 - account for the different conditions

# lncRNA dgeObj
# update "group" column to categorize the samples into the two populations
lcts$samples$group <- as.character(lcts$samples$group)
lcts$samples$group[colnames(lcts) %in% samples$sample_id] <-
str_sub(samples$population, -3, -2)
lcts$samples$group <- as.factor(lcts$samples$group)
# add a column to separate samples into three time points
lcts$samples$day <- as.factor(str_sub(samples$time_point, -2,-1))
# add a column to separate samples into two temperatures
lcts$samples$temp <- as.factor(samples$temp)

# mRNA dgeObj
mcts$samples$group <- as.character(mcts$samples$group)
mcts$samples$group[colnames(mcts) %in% samples$sample_id] <-
str_sub(samples$population, -3, -2)
mcts$samples$group <- as.factor(mcts$samples$group)
# add a column to separate samples into three time points
mcts$samples$day <- as.factor(str_sub(samples$time_point, -2,-1))
# add a column to separate samples into two temperatures
mcts$samples$temp <- as.factor(samples$temp)
```

8-c) MDS plots for lncRNAs and mRNAs separately…also plotting the total again for comparison

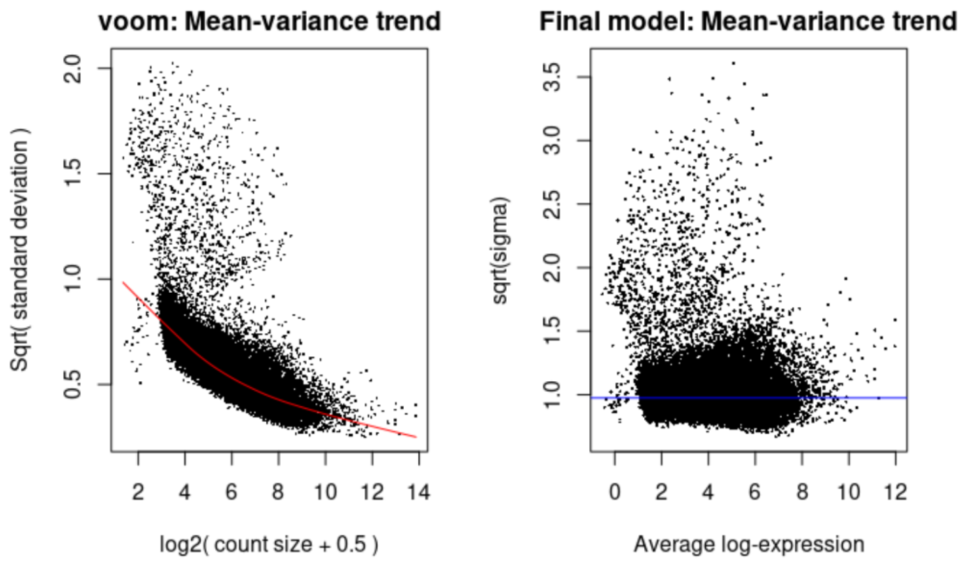
```r
par(mfrow=c(1,3))

# lncRNA - different from the other two clusterings
log_lcts <- cpm(lcts, log=TRUE)
plotMDS(log_lcts, labels=paste(lcts$samples$group, lcts$samples$day,
lcts$samples$temp, sep="."), cex=0.75, xlim=c(-4, 5), col=col.group)
title(main = "lncRNA clustering")

# mRNA - similar to the total clustering
log_mcts <- cpm(mcts, log=TRUE)
plotMDS(log_mcts, labels=paste(mcts$samples$group, mcts$samples$day,
mcts$samples$temp, sep="."), cex=0.75, xlim=c(-4, 5), col=col.group)
title(main = "mRNA clustering")

# everything
plotMDS(lcpm, labels=paste(cts$samples$group, cts$samples$day,
cts$samples$temp, sep="."), cex=0.75, xlim=c(-4, 5), col=col.group)
title(main = "total clustering")
```



9-a) Differential expression analysis

```
# creating a design matrix - day was ignored
g <- cts$samples$group
t <- cts$samples$temp
#d <- cts$samples$day
gt <- interaction(g,t)
design <- model.matrix(~0+gt)
colnames(design) <- gsub("gt", "", colnames(design))
#colnames(design)

# contrasts
contr.matrix <- makeContrasts(
  MI.27CvsSM.27C = MI.27C - SM.27C,
  MI.27CvsMI.32C = MI.27C - MI.32C,
  MI.27CvsSM.32C = MI.27C - SM.32C,
  SM.27CvsMI.32C = SM.27C - MI.32C,
  SM.27CvsSM.32C = SM.27C - SM.32C,
  MI.32CvsSM.32C = MI.32C - SM.32C,
  levels = colnames(design))
#contr.matrix

# removing heteroscedascity from count data so that variance is no longer
```

```
dependent on the mean expression level - the difference can be seen in the
plots
par(mfrow=c(1,2))
v <- voom(cts, design, plot=TRUE)
#v
vfit <- lmFit(v, design)
vfit <- contrasts.fit(vfit, contrasts=contr.matrix)
efit <- eBayes(vfit)
plotSA(efit, main="Final model: Mean-variance trend")
```

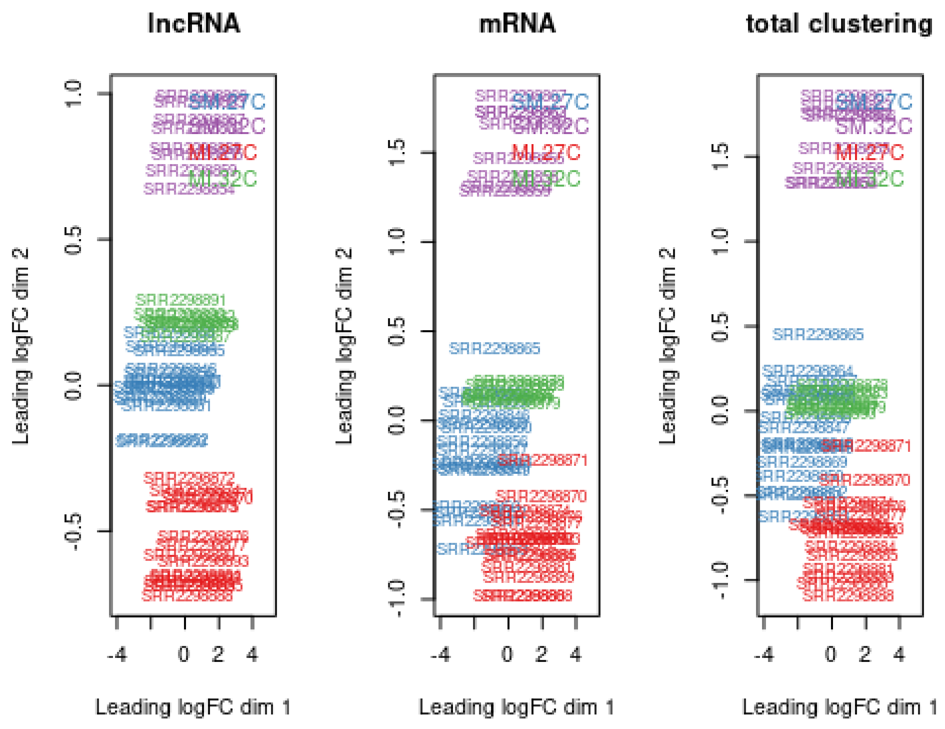


```
# examining the number of DE genes among all six comparisons
dt <- decideTests(efit)
pander(summary(dt), style='rmarkdown')
```

	MI.27CvsSM.27C	MI.27CvsMI.32C	MI.27CvsSM.32C	SM.27CvsMI.32C	SM.27CvsSM.32C	MI.32CvsSM.32C
Down	13086	8587	7287	9372	12774	5365
NotSig	6715	14455	15429	13928	7819	19519
Up	12294	9053	9379	8795	11502	7211

```
# color grouping defined by group and temp
# unsupervised clustering of samples that is similar to the clusterings done
before (step 8-c) except now we are only considering population and
temperature, and points are labeled by sample names - similar results as
before
par(mfrow=c(1,3))
col.g <- gt
```

```
levels(col.g) <-  brewer.pal(nlevels(col.g), "Set1")
col.g <- as.character(col.g)

# lncRNAs
plotMDS(log_lcts, cex=0.75, xlim=c(-4, 5), col=col.g)
legend("topright", legend = unique(gt), text.col = unique(col.g), bty = "n")
title(main = "lncRNA")

# mRNAs
plotMDS(log_mcts, cex=0.75, xlim=c(-4, 5), col=col.g)
legend("topright", legend = unique(gt), text.col = unique(col.g), bty = "n")
title(main = "mRNA")

# all samples
plotMDS(lcpm, cex=0.75, xlim=c(-4, 5), col=col.g)
legend("topright", legend = unique(gt), text.col = unique(col.g), bty = "n")
title(main = "total clustering")
```

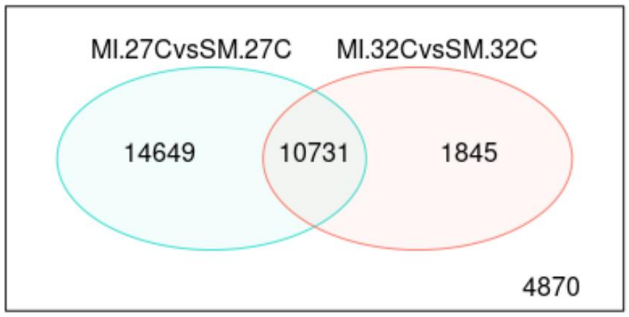


9-b) Differential expression analysis for MI.27CvsSM.27C and MI.32CvsSM.32C

This analysis can be done for/between any of the six comparisons

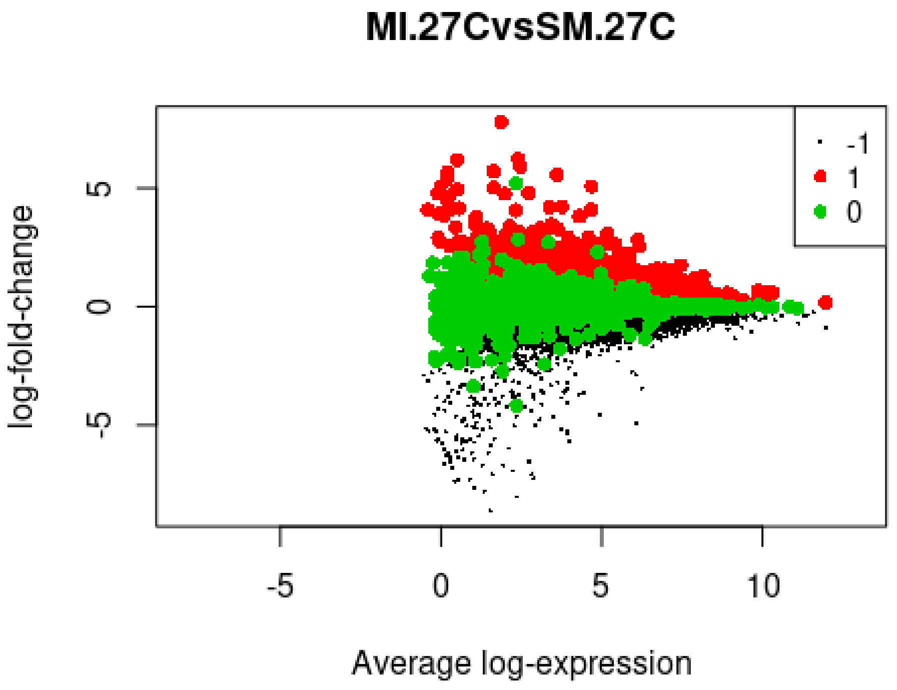
```
# DE genes between MI.27CvsSM.27C and MI.32CvsSM.32C - how many of the genes
are common among these comparisons
```

```
de.common <- which(dt[,1]!=0 & dt[,6]!=0) # 10731 gens
#length(de.common)

# venn diagram showing the number of genes DE in the comparison between
MI.27CvsSM.27C (left) and MI.32CvsSM.32C (right), the number of genes that
are DE in both comparisons (center), and the number of genes that are not DE
in either comparison (bottom-right)
vennDiagram(dt[,c(1,6)], circle.col=c("turquoise", "salmon"), cex = 1)
```

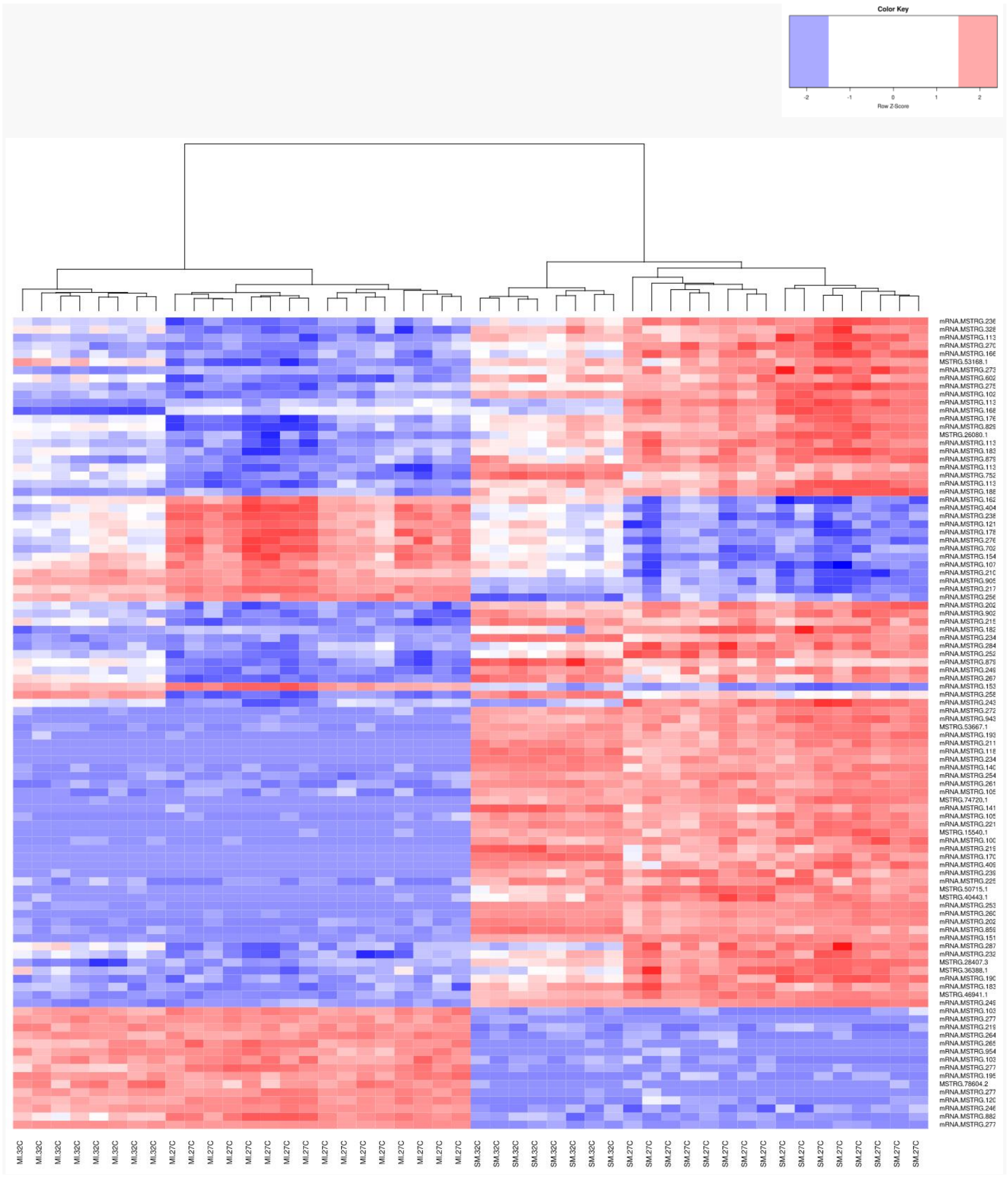


```
# if we want to extract and write the results for all the comparisons to a
single output file
#write.fit(efit, dt, file="results.txt")

# examine individual DE genes from top to bottom in MI.27CvsSM.27C - up+down
expressed
MI.27CvsSM.27C <- topTable(efit, coef=1, n=Inf, p.value = 0.05)
#dim(MI.27CvsSM.27C)

# each comparison
# cp1 <- topTable(efit, coef=1, n=Inf, p.value = 0.05)
# cp1$rn <- rownames(cp1)
#
# cp2 <- topTable(efit, coef=2, n=Inf, p.value = 0.05)
# cp2$rn <- rownames(cp2)
#
# cp3 <- topTable(efit, coef=3, n=Inf, p.value = 0.05)
# cp3$rn <- rownames(cp3)
#
# cp4 <- topTable(efit, coef=4, n=Inf, p.value = 0.05)
# cp4$rn <- rownames(cp4)
#
# cp5 <- topTable(efit, coef=5, n=Inf, p.value = 0.05)
# cp5$rn <- rownames(cp5)
#
# cp6 <- topTable(efit, coef=6, n=Inf, p.value = 0.05)
# cp6$rn <- rownames(cp6)

# separate DG genes into lncRNAs and mRNAs in MI.27CvsSM.27C
```

```
onlylncrna <- MI.27CvsSM.27C[grep("^MSTRG", rownames(MI.27CvsSM.27C)), ] # 
6218 DE lncRNAs
onlymrna <- MI.27CvsSM.27C[grep("^mRNA", rownames(MI.27CvsSM.27C)), ] # 19162
DE mRNAs
### sanity check- 6218 + 19162 = 25380 total DG genes

# useful graphical representations of differential expression results

# mean-difference plot displaying the log-fold change against average log-
counts per million with DE genes highlighted
plotMD(efit, column=1, status=dt[,1], main=colnames(efit)[1], xlim=c(-8,13))
```



```
# heatmap - top 100 DE genes from MI.27C vs SM.27C - both lncRNAs and mRNAs
MI.27CvsSM.27C.topgenes <- rownames(MI.27CvsSM.27C)[1:100]
i <- which(rownames(v) %in% MI.27CvsSM.27C.topgenes)
mycol <- colorpanel(1000,"blue","white","red")
#heatmap.2(lcpm[i,], scale="row", labRow=rownames(v)[i], labCol=gt,
col=mycol, trace="none", density.info="none", margin=c(8,6), lhei=c(2,10),
dendrogram="column")

# since the margins of the heatmap are too large, the columns and rows are
not true representations; it is better to save the heatmap to your computer
as follows and then open it online (it will take a bit time to load)
svg("myheatmap.svg", width=20, height=16)
heatmap.2(lcpm[i,], scale="row", labRow=rownames(v)[i], labCol=gt, col=mycol,
trace="none", density.info="none", margin=c(8,6), lhei=c(2,10),
dendrogram="column")
```



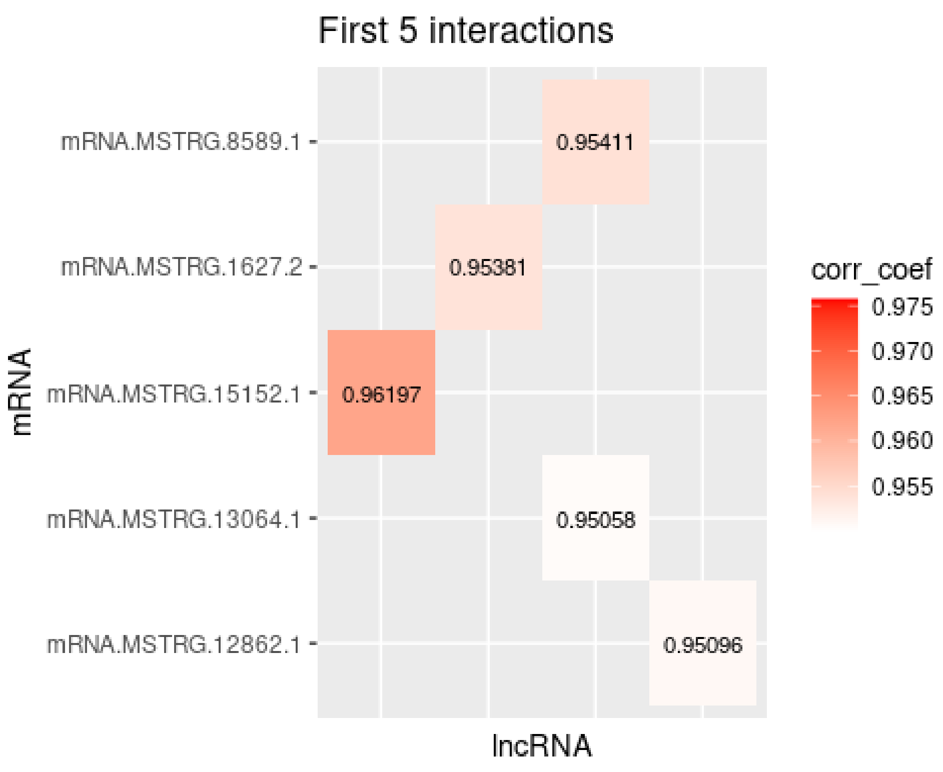
10-a) Pearson correlation between mRNA vs lncRNA - produces a table with gene IDs, corr values, and p-value

```
# this is a test run with smaller datasets - once I made sure the output was
how I wanted it to be, I created an Rscript (corr_test.R) and submitted it as
```

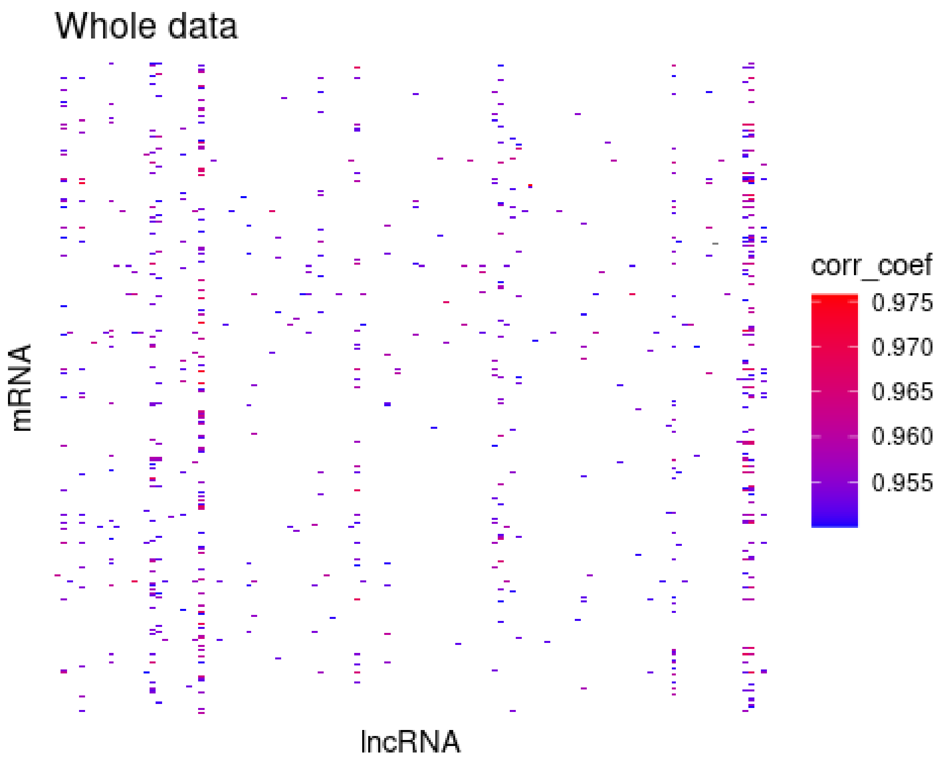
```
a job. I had to download the lncRNA (lds) and mRNA (mds) files to the
terminal before I could run the job. The .qsub file is called
"corr_test.qsub" and can be found in the "repeat" directory (same as this
document). It has all the bash options and the command to load the R module.
The .R file does not have the bash options. Once the .cvs file from that
script is produced, I imported it onto here and did further analyses

# test run - first 500 genes
x <- lds[1:500,]
y <- mds[1:500,]

# columns for the final matrix
p_value <- list(); corr_coef <- list(); gene1 <- list(); gene2 <- list()

# pearson correlation test (1-2 min)
for (i in 1:nrow(x)) {
  for (j in 1:nrow(y)){
    c <- cor.test(x[i,], y[j,], method = "pearson")
    # filter out the values based on the cutoff values: corr coef >= |0.95|
and a P-value < 0.05
    if (abs(c$estimate) >= 0.95 & c$p.value < 0.05) {
      gene1 <- c(gene1,rownames(x)[i]) # get name of first gene
      gene2 <- c(gene2,rownames(y)[j]) # get name of second gene
      p_value <- c(p_value,c$p.value) # get p-value
      corr_coef <- c(corr_coef,c$estimate) # get correlation coefficient
    }
  }}

ds <- cbind(gene1,gene2,corr_coef,p_value)
# convert to dataframe
ds <- as.data.frame(ds)
# convert elements into character type
ds <- apply(ds,2,as.character)

# write result into a table
#write.csv(ds, "corr_result_1hr.csv")

# what the output file looks like
ds

##      gene1          gene2                corr_coef
## [1,] "MSTRG.3855.1" "mRNA.MSTRG.276.1" "0.953884304324871"
## [2,] "MSTRG.4966.1" "mRNA.MSTRG.426.1" "0.95203748302883"
## [3,] "MSTRG.4966.1" "mRNA.MSTRG.581.1" "0.95075258236401"
##      p_value
## [1,] "1.11431688272589e-25"
## [2,] "2.69519292725329e-25"
## [3,] "4.88255293435593e-25"
```

10-b) Correlation heatmap

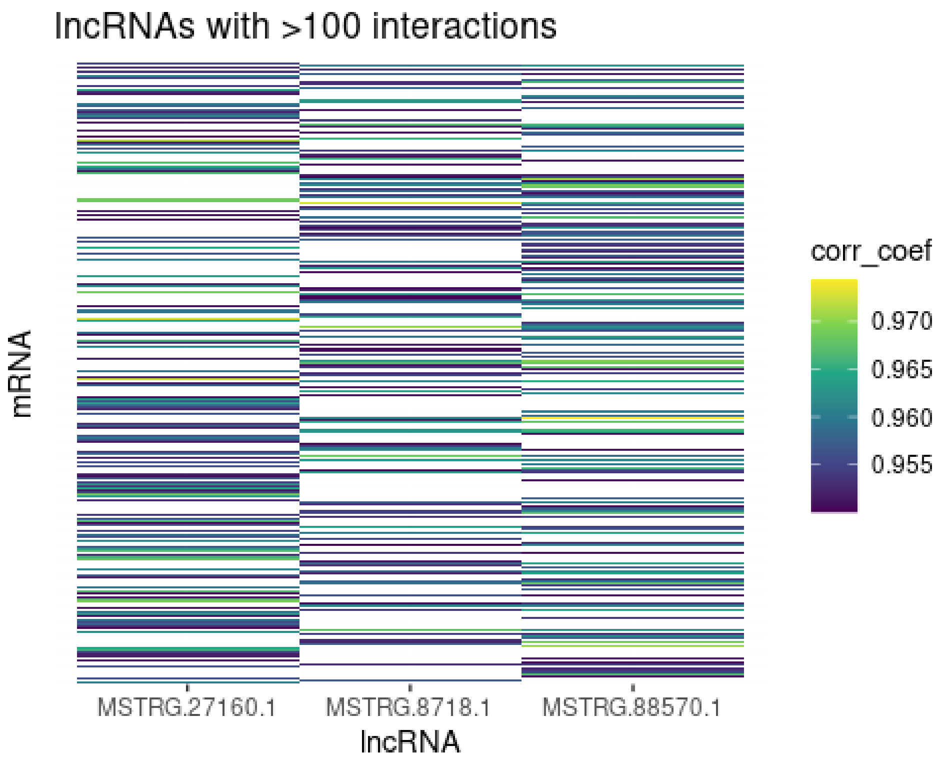
```r
# read in the pearson correlation test output
corr_result <-
read.table("/projectnb/incrna/mary_lncrna/R/repeat/output.csv", header =
TRUE, sep="", stringsAsFactors = F)

# 1345 interactions
#dim(corr_result)

# edit column names to show which ones are mRNAs and lncRNAs
colnames(corr_result)[1:2] <- c("lncRNA", "mRNA")

# check that the mRNA and lncRNA columns are labeled correctly
#sapply(colnames(corr_result[]), function(x) grep("mRNA", corr_result[,x]))

# 119 lncRNAs and 670 mRNAs are involved in the interactions
#c(length(unique(corr_result$lncRNA)), length(unique(corr_result$mRNA)))

# exclude p-value column
corr1 <- corr_result[1:3]

# heatmap - first 5 interactions
ggplot(data = corr1[1:5,], aes(x=lncRNA, y=mRNA, fill=corr_coef)) +
  geom_tile() +
  geom_text(aes(lncRNA, mRNA, label = round(corr_coef, digits = 5)), color =
"black", size = 3)+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())+
  scale_fill_gradient2(low="blue", high="red",
              midpoint=mean(0.9500036, 0.9757489),
              limits=c(0.9500036, 0.9757489))+
  ggtitle("First 5 interactions")
```



```
# heatmap - whole data
ggplot(data = corr1, aes(x=lncRNA, y=mRNA, fill=corr_coef)) +
  geom_tile() +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank())+
  scale_fill_gradient2(mid = "blue", high="red",
              midpoint=mean(0.9500036, 0.9757489),
              limits=c(0.9500036, 0.9757489))+
  ggtitle("Whole data")
```



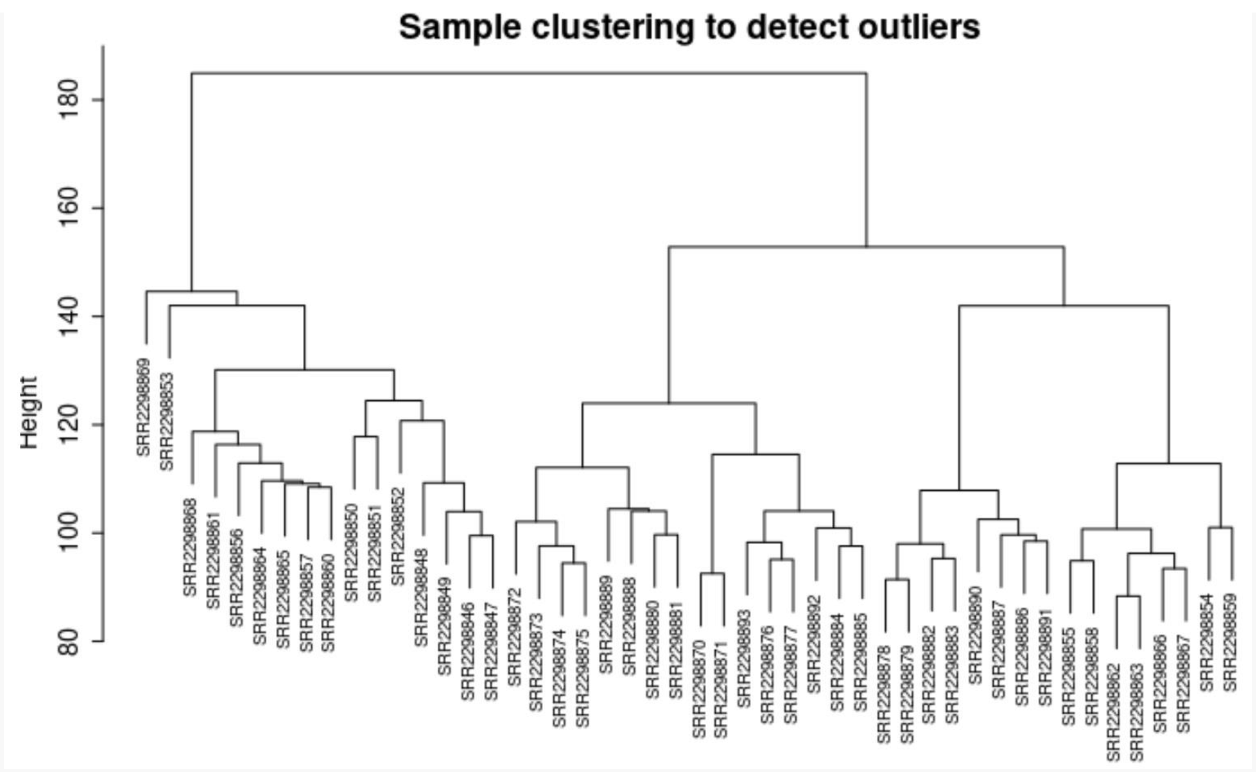
```r
# subset the data to only include lncRNAs with >100 interactions
subs_l <- corr1 %>% group_by(lncRNA) %>% filter(n()>100)

# only 3 lncRNAs have >100 interactions
#length(table(subs_l$lncRNA))

# heatmap for subset data
ggplot(data = subs_l, aes(x=lncRNA, y=mRNA, fill=corr_coef)) +
  geom_tile() +
  theme(axis.ticks.y = element_blank(),
        axis.text.y = element_blank())+
  scale_fill_viridis_c()+
  ggtitle("lncRNAs with >100 interactions")
```



11) WGCNA

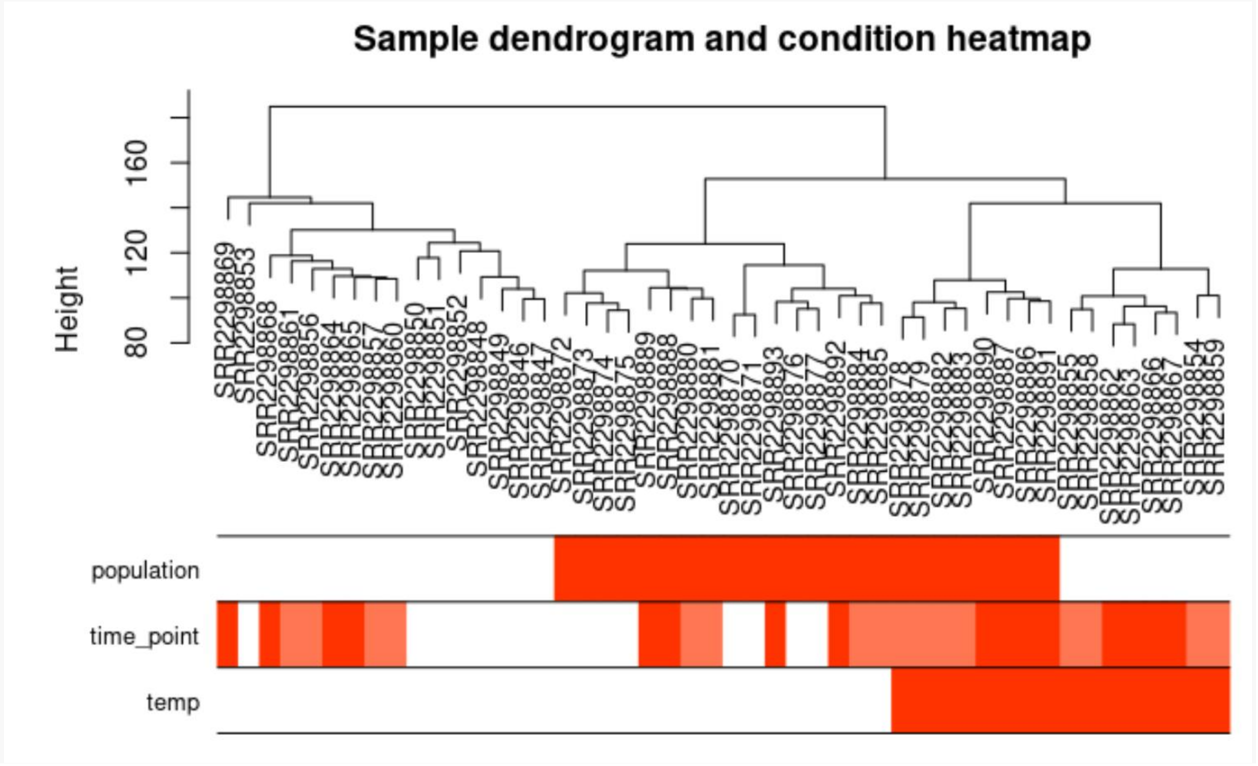
11-a) Setup/prepare data

```
# linear modeling
vfit2 <- lmFit(v, design)

# all possible pairwise contrasts
condition_pairs <- t(combn(levels(gt), 2))
comparisons <- list()
for (i in 1:nrow(condition_pairs)) {
  comparisons[[i]] <- as.character(condition_pairs[i,])
}

# vector to store DE genes
DEGs <- c()

# iterate over the contrasts and perform a differential expression test for
each pair
for (c in comparisons) {
    contrast_formula <- paste(c, collapse=' - ')
    contrast_mat <- makeContrasts(contrasts=contrast_formula, levels=design)
    contrast_fit <- contrasts.fit(vfit2, contrast_mat)
    eb <- eBayes(contrast_fit)
    # select highly ranked genes with a p-value cutoff of 0.05
    DEGs <- union(DEGs, rownames(topTable(eb, n=Inf, p.value=0.05)))
}

# filter out genes which are not differentially expressed for any contrast
v2 <- v$E[rownames(v) %in% DEGs,]
#dim(v2)
#dim(v$E)
# sanity check: 32095 (started with) - 1764 (removed) = 30331 (left)

# transpose the expression data for further analyses
datExpr0 <- as.data.frame(t(v2))

# check data for excessive missing values
gsg <- goodSamplesGenes(datExpr0, verbose = 3)

##  Flagging genes and samples with too many missing values...
##   ..step 1

gsg$allOK # returns TRUE - all genes have passed the cuts

## [1] TRUE

# cluster the samples to see if there are any obvious outliers
sampleTree <- hclust(dist(datExpr0), method = "average")

# plot the sample tree
# open a graphic output window of size 12 by 9 inches
sizeGrWindow(12,9)
par(cex = 0.6)
par(mar = c(0,4,2,0))
plot(sampleTree, main = "Sample clustering to detect outliers", sub="",
xlab="", cex.lab = 1.5, cex.axis = 1.5, cex.main = 2)
# there are no crazy outliers
```



```
# determine cluster under the line
clust <- cutreeStatic(sampleTree, minSize = 10)
#table(clust)

# define numbers of genes and samples
nGenes <- ncol(datExpr0)
nSamples <- nrow(datExpr0)

# conditions/trait data
expSamples <- rownames(datExpr0)
traitRows <- match(expSamples, samples$sample_id)

# remove the column holding the sample names
datTraits <- samples[traitRows, -1]

# remove "tolerance" column since it is the same as "population" column
datTraits <- datTraits[,-4]

# change values of the conditions to numeric ones
# set SM to 0 and MI to 1
datTraits <- datTraits %>% mutate(population = ifelse(population == "South
Molle (SM)",0,1))
# remove "day" from the time_point column so it is all numbers
datTraits$time_point <- as.double(str_sub(datTraits$time_point, 5,6))
# removing the "C" from temp column
```

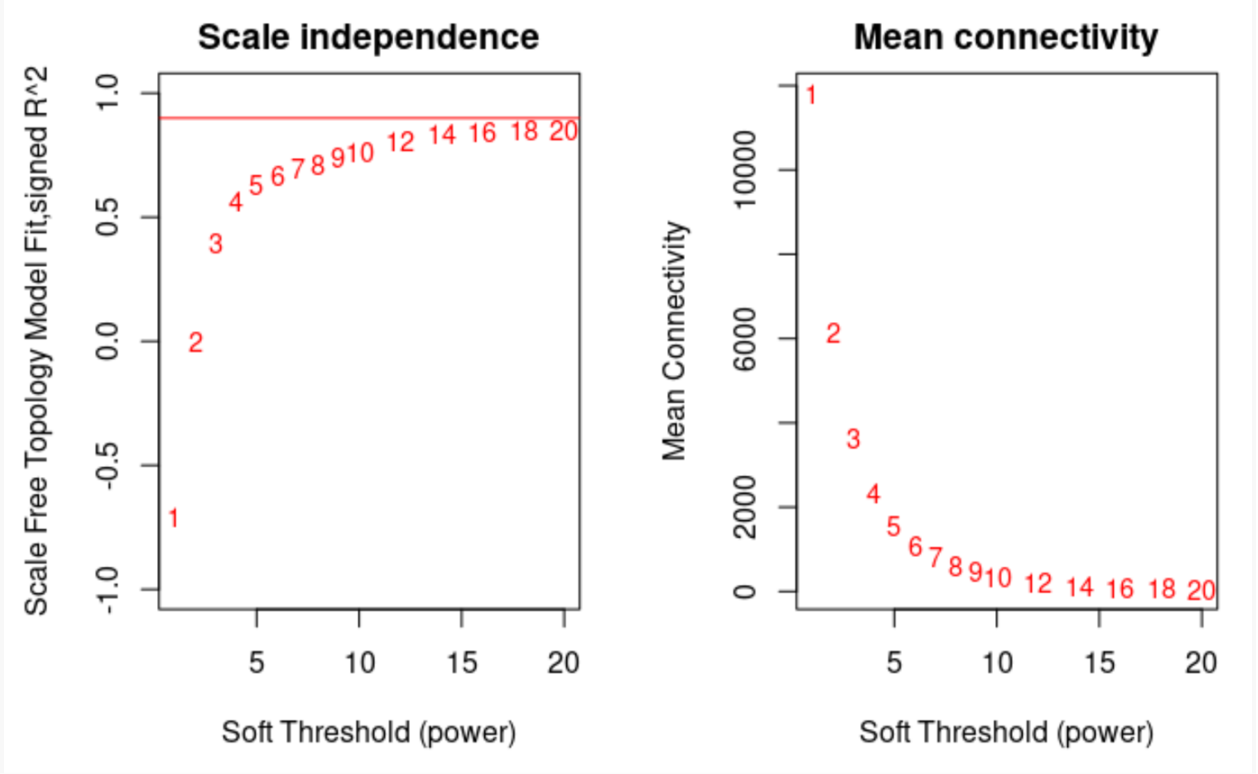
```
datTraits$temp <- as.double(str_sub(datTraits$temp, 1,2))

# set sample names as row names
rownames(datTraits) <- samples[traitRows, 1]

# re-cluster samples to see how the conditions/traits relate to the sample
dendrogram
sampleTree2 <- hclust(dist(datExpr0), method = "average")
# convert conditions to a color representation: white = low, red = high, grey
= missing entry
traitColors <- numbers2colors(datTraits, signed = FALSE)
# plot the sample dendrogram and the colors underneath
plotDendroAndColors(sampleTree2, traitColors, groupLabels = names(datTraits),
main = "Sample dendrogram and condition heatmap")
```



11-b) Choosing the soft-thresholding power - analysis of network topology

```
options(stringsAsFactors = FALSE)

# 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 (3-4 min)
sft <- pickSoftThreshold(datExpr0, powerVector = powers, verbose = 5)
```

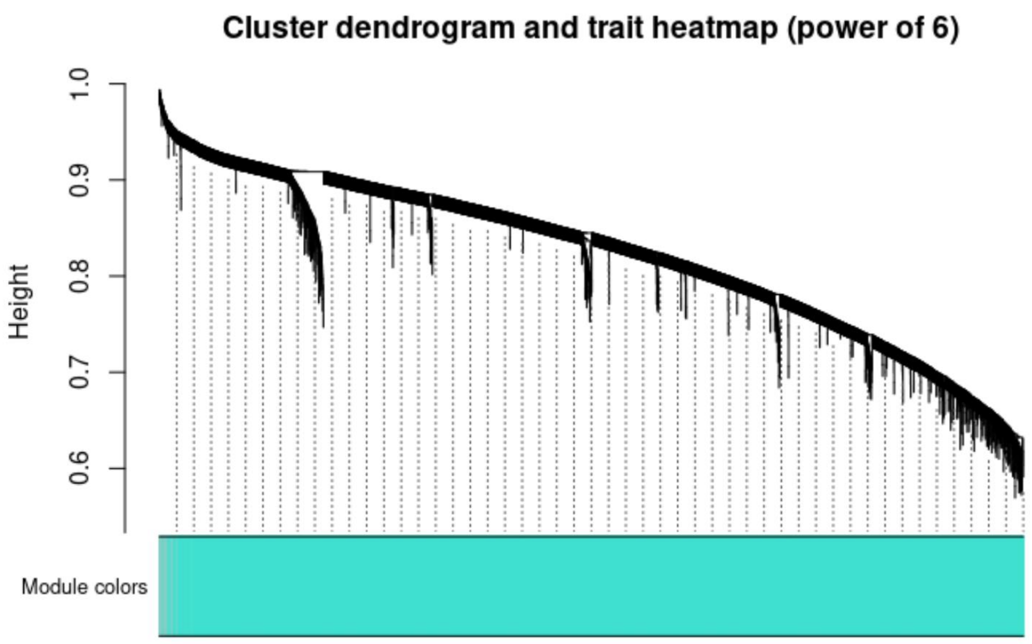
```
# plot the results
sizeGrWindow(9, 5)
par(mfrow = c(1,2))
cex1 = 0.9

# scale-free topology fit index as a function of the soft-thresholding power
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed
R^2",type="n", main = paste("Scale independence"), ylim = c(-1, 1))
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
labels=powers,cex=cex1,col="red")
# use an R^2 cut-off of h = 0.9
abline(h=0.90,col="red")

# mean connectivity as a function of the soft-thresholding power
plot(sft$fitIndices[,1], sft$fitIndices[,5], xlab="Soft Threshold
(power)",ylab="Mean Connectivity", type="n", main = paste("Mean
connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers,
cex=cex1,col="red")
```



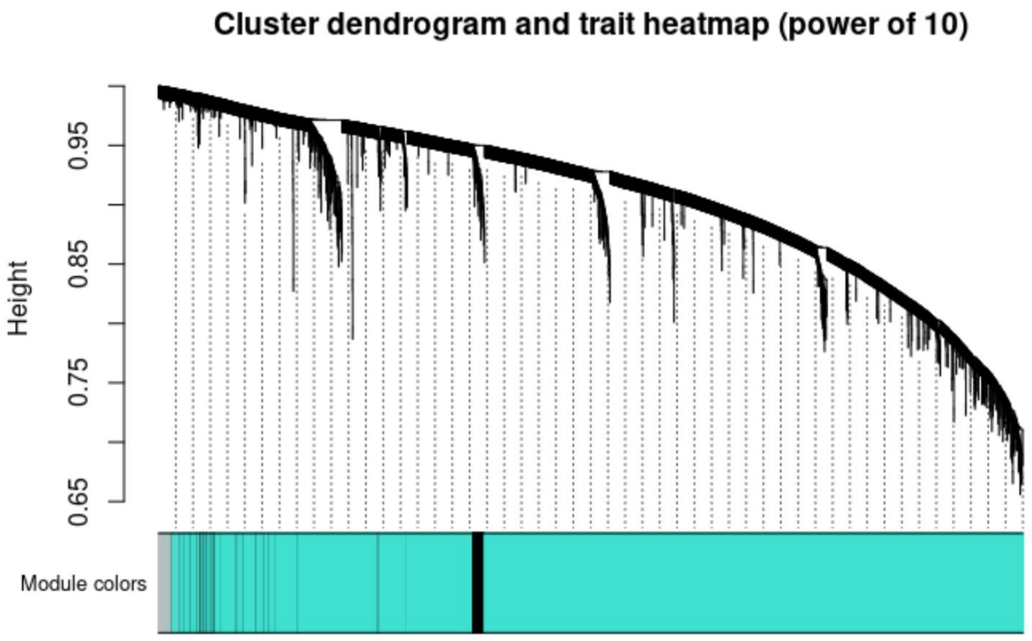
Chose powers of 6 and 10 for the following analyses

11-c) One-step network construction and module detection

```
# power of 6 (took 16 min to run)
net6 <- blockwiseModules(datExpr0, power = 6,
                        TOMType = "unsigned", minModuleSize = 30,
                        reassignThreshold = 0, mergeCutHeight = 0.25,
                        numericLabels = TRUE, pamRespectsDendro = FALSE,
verbose = 3)

# power of 10 (took 16 min to run)
net10 <- blockwiseModules(datExpr0, power = 10,
                        TOMType = "unsigned", minModuleSize = 30,
                        reassignThreshold = 0, mergeCutHeight = 0.25,
                        numericLabels = TRUE, pamRespectsDendro = FALSE,
verbose = 3)

# modules with their number of gene

# there are 11 modules, labeled 1 through 11 in order of descending size,
with sizes ranging from 9890 to 34 genes
# the label 0 is reserved for genes outside of all modules
table(net6$colors)

##
##    0    1    2    3    4    5    6    7    8    9   10   11
##  288 9890 7777 6080 2291 1803 1088  484  254  246   96   34

## there are 10 modules, labeled 1 through 10 in order of descending size,
with sizes ranging from 12760 to 118
table(net10$colors)

##
##     0     1     2     3     4     5     6     7     8     9    10
##  1369 12760  7703  3859  1944   832   765   479   370   132   118

# plots

# power of 6
# convert labels to colors for plotting
mergedColors6 <- labels2colors(net6$colors)
# plot the dendrogram and the module colors underneath
plotDendroAndColors(net6$dendrograms[[1]],
mergedColors6[net6$blockGenes[[1]]], "Module colors", dendroLabels = FALSE,
hang = 0.03,addGuide = TRUE, guideHang = 0.05, main = "Cluster dendrogram and
trait heatmap (power of 6)")
```

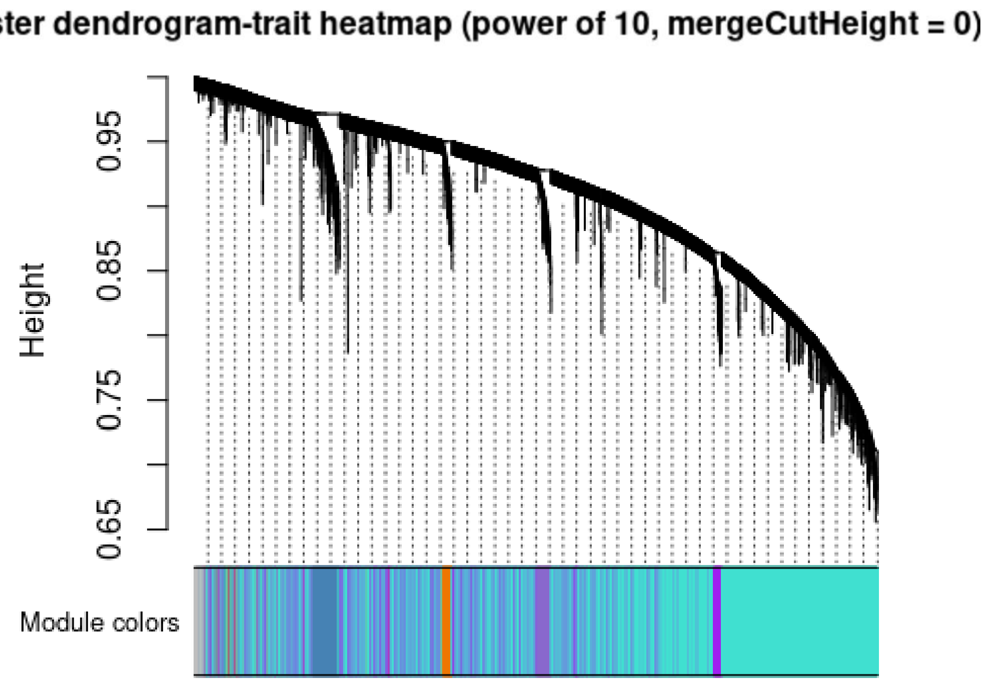


```
# power of 10
mergedColors10 <- labels2colors(net10$colors)
plotDendroAndColors(net10$dendrograms[[1]],
mergedColors10[net10$blockGenes[[1]]], "Module colors", dendroLabels = FALSE,
hang = 0.03,addGuide = TRUE, guideHang = 0.05, main = "Cluster dendrogram and
trait heatmap (power of 10)")
```



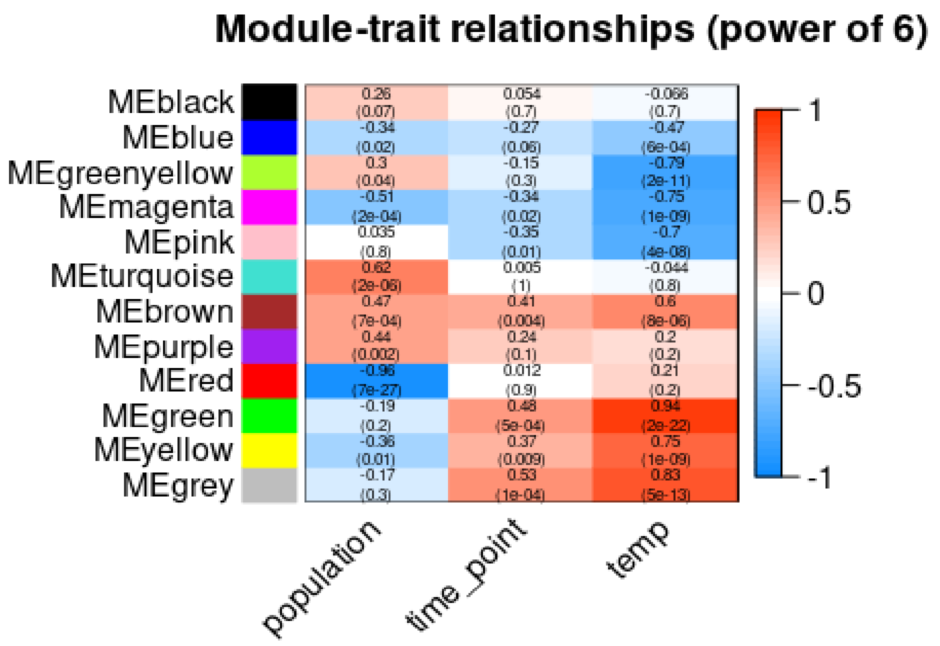
```
# since the cluster dendrogram and trait heatmap looked unusual it was
recommended to set the "TOMType" option of "blockwiseModules" to "signed"
which did not affect anything. However, when the "mergeCutHeight" option was
set to 0, the output had a lot more modules and the heatmap underneath the
```

```
dendrogram had more colors for power of 10. The command to run that is as
follows and the rest of the analyses can be run on it if desired.

trialnet10 <- blockwiseModules(datExpr0, power = 10,
                        TOMType = "unsigned", minModuleSize = 30,
                        reassignThreshold = 0, mergeCutHeight = 0,
                        numericLabels = TRUE, pamRespectsDendro = FALSE,
verbose = 3)

table(trialnet10$colors)

##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14
## 1369 3598 2567 2127 1750 1417 1207 1203 1056  894  833  832  812  764  751
##   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29
##  652  555  493  467  431  370  359  355  340  288  280  279  266  257  234
##   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44
##  229  226  212  211  208  187  179  165  151  148  141  137  132  130  129
##   45   46   47   48   49   50   51   52   53   54   55
##  124  118  115   96   88   81   80   71   64   53   50

trialmergedColors10 <- labels2colors(trialnet10$colors)
plotDendroAndColors(trialnet10$dendrograms[[1]],
trialmergedColors10[trialnet10$blockGenes[[1]]], "Module colors",
dendroLabels = FALSE, hang = 0.03,addGuide = TRUE, guideHang = 0.05, main =
"Cluster dendrogram and trait heatmap (power of 10 with mergeCutHeight = 0)")
```

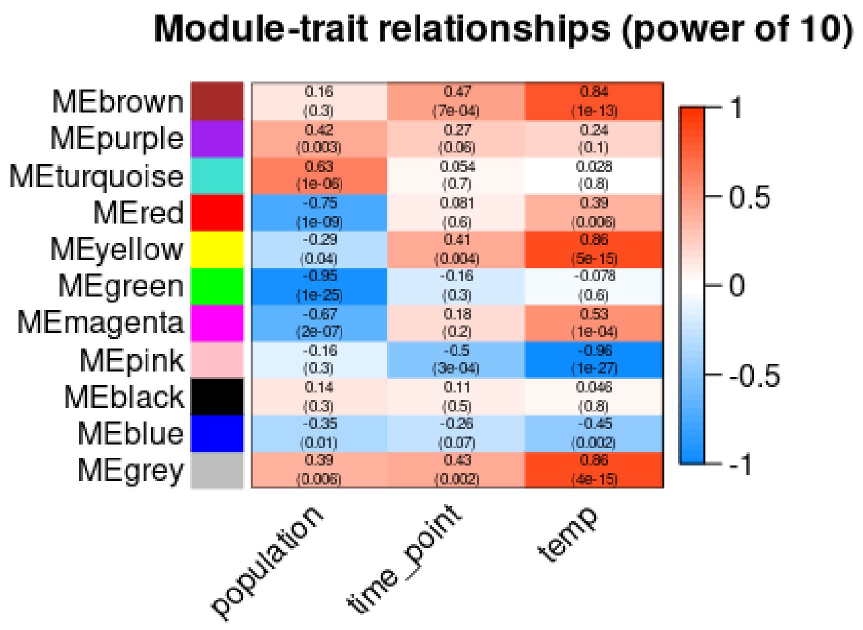


11-d-1) Quantifying module–trait associations - without separating the traits/conditions

```r
# recalculate MEs with color labels

# power of 6
moduleLabels <- net6$colors
moduleColors <- labels2colors(net6$colors)
MEs0 <- moduleEigengenes(datExpr0, moduleColors)$eigengenes
MEs <- orderMEs(MEs0)
moduleTraitCor <- cor(MEs, datTraits, use = "p")
moduleTraitPvalue <- corPvalueStudent(moduleTraitCor, nSamples)

# power of 10
moduleLabels10 <- net10$colors
moduleColors10 <- labels2colors(net10$colors)
MEs0_10 <- moduleEigengenes(datExpr0, moduleColors10)$eigengenes
MEs_10 <- orderMEs(MEs0_10)
moduleTraitCor10 <- cor(MEs_10, datTraits, use = "p")
moduleTraitPvalue10 <- corPvalueStudent(moduleTraitCor10, nSamples)

# heatmap - display correlations and their p-values (each association is
color coded by the correlation value)

# power of 6
textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
signif(moduleTraitPvalue, 1), ")", sep = "")
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(6, 8.5, 3, 3))
labeledHeatmap(Matrix = moduleTraitCor, xLabels = names(datTraits), yLabels =
names(MEs),
ySymbols = names(MEs), colorLabels = FALSE, colors = blueWhiteRed(50),
textMatrix = textMatrix, setStdMargins = FALSE, cex.text = 0.5, zlim = c(-
1,1), main = paste("Module-trait relationships (power of 6)"))
```



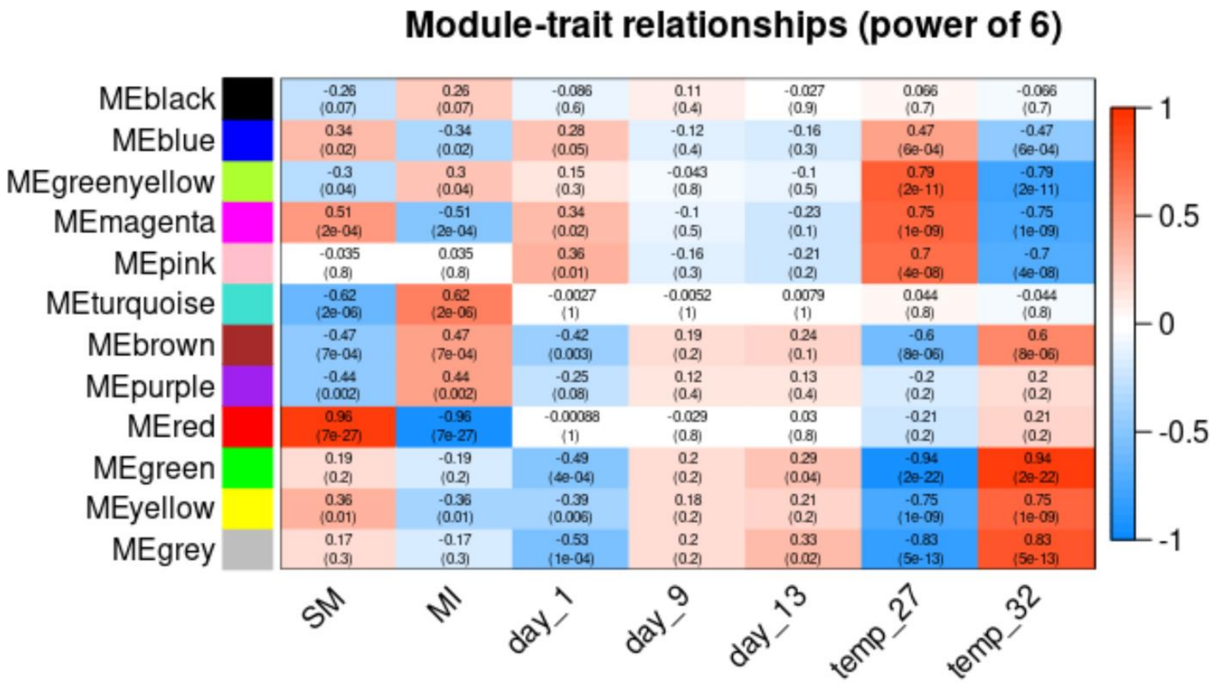
```
# power of 10
textMatrix10 = paste(signif(moduleTraitCor10, 2), "\n(",
signif(moduleTraitPvalue10, 1), ")", sep = "")
dim(textMatrix10) = dim(moduleTraitCor10)
par(mar = c(6, 8.5, 3, 3))
labeledHeatmap(Matrix = moduleTraitCor10, xLabels = names(datTraits), yLabels
= names(MEs_10), ySymbols = names(MEs_10), colorLabels = FALSE, colors =
blueWhiteRed(50), textMatrix = textMatrix10, setStdMargins = FALSE, cex.text
= 0.5, zlim = c(-1,1), main = paste("Module-trait relationships (power of
10)"))
```



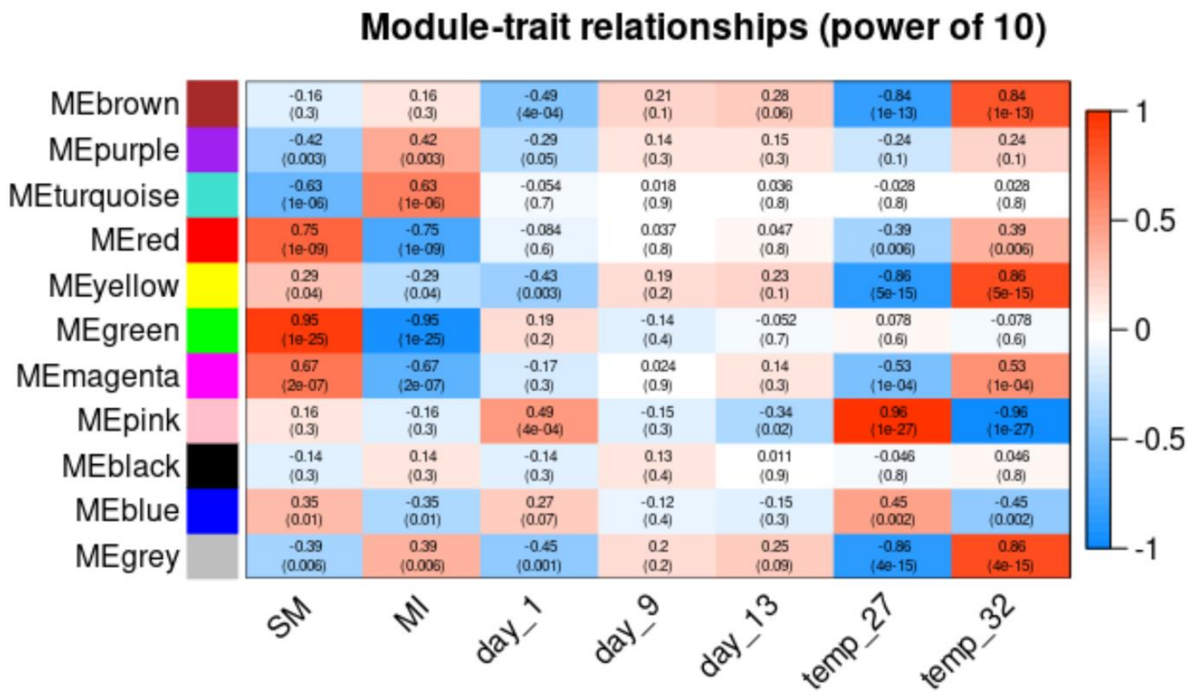
11-d-2) Quantifying module–trait associations and sample dendrogram and condition heatmap - with the traits/conditions separated into multiple columns and set as either 0 or 1

```r
# setup trait/condition table
sepTraits <- datTraits # make a copy

# population columns
sepTraits$SM <- NA
sepTraits <- sepTraits %>% mutate(SM = ifelse(population == 0,1,0))
colnames(sepTraits)[1] <- "MI"

# temperature columns
sepTraits$temp_27 <- NA
sepTraits <- sepTraits %>% mutate(temp_27 = ifelse(temp == 27,1,0))
sepTraits <- sepTraits %>% mutate(temp = ifelse(temp == 32,1,0))
colnames(sepTraits)[3] <- "temp_32"

# time point/day columns
sepTraits$day_1 <- NA
sepTraits$day_9 <- NA
sepTraits <- sepTraits %>% mutate(day_1 = ifelse(time_point == 1,1,0))
sepTraits <- sepTraits %>% mutate(day_9 = ifelse(time_point == 9,1,0))
sepTraits <- sepTraits %>% mutate(time_point = ifelse(time_point == 13,1,0))
colnames(sepTraits)[2] <- "day_13"

# reorder the columns
sepTraits <- sepTraits[c("SM", "MI", "day_1", "day_9", "day_13", "temp_27",
"temp_32")]

# add rownames
rownames(sepTraits) <- rownames(datTraits)

# module-trait associations
# power of 6
moduleTraitCor_sep <- cor(MEs, sepTraits, use = "p")
moduleTraitPvalue_sep <- corPvalueStudent(moduleTraitCor_sep, nSamples)

# power of 10
moduleTraitCor10_sep <- cor(MEs_10, sepTraits, use = "p")
moduleTraitPvalue10_sep <- corPvalueStudent(moduleTraitCor10_sep, nSamples)

# heatmap - display correlations and their p-values (each association is
color coded by the correlation value)

# power of 6
textMatrix_sep = paste(signif(moduleTraitCor_sep, 2), "\n(",
signif(moduleTraitPvalue_sep, 1), ")", sep = "")
```

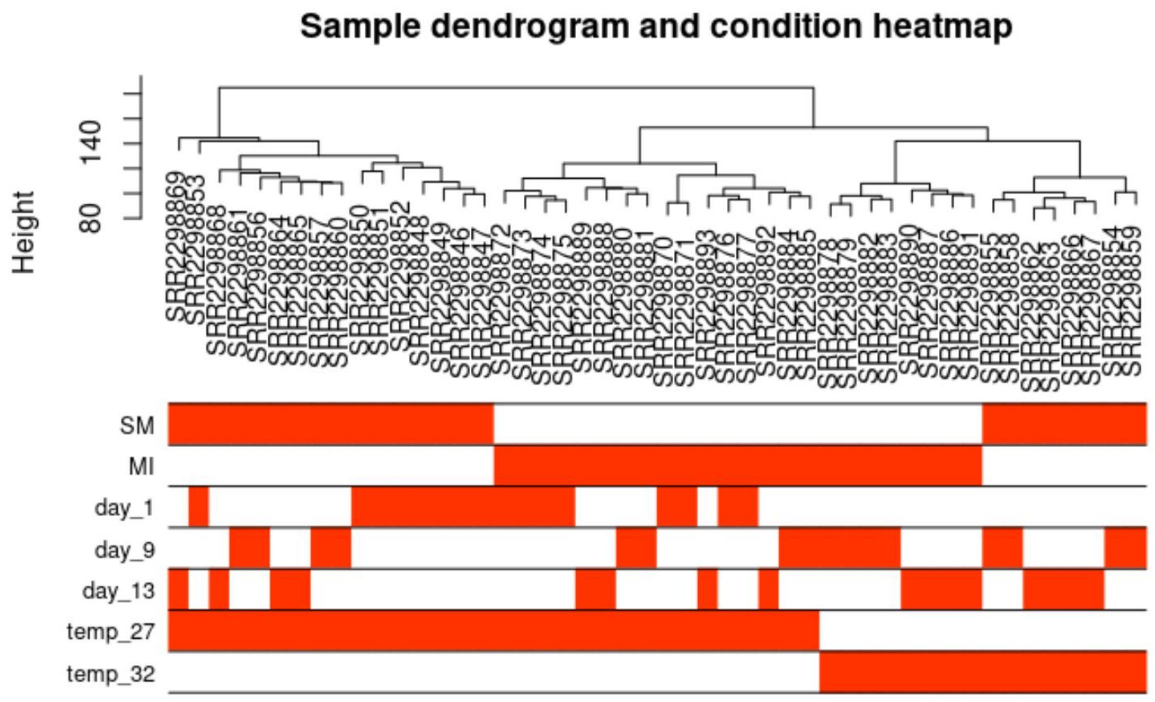
```
dim(textMatrix_sep) = dim(moduleTraitCor_sep)
par(mar = c(6, 8.5, 3, 3))
labeledHeatmap(Matrix = moduleTraitCor_sep, xLabels = names(sepTraits),
yLabels = names(MEs),
ySymbols = names(MEs), colorLabels = FALSE, colors = blueWhiteRed(50),
textMatrix = textMatrix_sep, setStdMargins = FALSE, cex.text = 0.5, zlim =
c(-1,1), main = paste("Module-trait relationships (power of 6)"))
```



```
# power of 10
textMatrix10_sep = paste(signif(moduleTraitCor10_sep, 2), "\n(",
signif(moduleTraitPvalue10_sep, 1), ")", sep = "")
dim(textMatrix10_sep) = dim(moduleTraitCor10_sep)
par(mar = c(6, 8.5, 3, 3))
labeledHeatmap(Matrix = moduleTraitCor10_sep, xLabels = names(sepTraits),
yLabels = names(MEs_10), ySymbols = names(MEs_10), colorLabels = FALSE,
colors = blueWhiteRed(50), textMatrix = textMatrix10_sep, setStdMargins =
FALSE, cex.text = 0.5, zlim = c(-1,1), main = paste("Module-trait
relationships (power of 10)"))
```



```
# sample dendrogram and condition heatmap
# convert conditions to a color representation: white = low, red = high, grey
= missing entry
traitColors_sep <- numbers2colors(sepTraits, signed = FALSE)
# plot the sample dendrogram and the colors underneath
plotDendroAndColors(sampleTree2, traitColors_sep, groupLabels =
names(sepTraits), main = "Sample dendrogram and condition heatmap")
```



11-e-1) lncRNAs and mRNAs found in each module (power of 6)

```
# genes found in each module
red <- names(datExpr0)[moduleColors=="red"]
blu <- names(datExpr0)[moduleColors=="blue"]
tur <- names(datExpr0)[moduleColors=="turquoise"]
yelw <- names(datExpr0)[moduleColors=="yellow"]
gry <- names(datExpr0)[moduleColors=="grey"]
brwn <- names(datExpr0)[moduleColors=="brown"]
blk <- names(datExpr0)[moduleColors=="black"]
grn <- names(datExpr0)[moduleColors=="green"]
man <- names(datExpr0)[moduleColors=="magenta"]
pink <- names(datExpr0)[moduleColors=="pink"]
pur <- names(datExpr0)[moduleColors=="purple"]
gyellow <- names(datExpr0)[moduleColors=="greenyellow"]

# lncNRAs
# number of lncRNAs in each module
length(grep("^MSTRG", red))
## [1] 159
length(grep("^MSTRG", blu))
## [1] 2980
length(grep("^MSTRG", tur))
## [1] 1762
length(grep("^MSTRG", yelw))
## [1] 395
length(grep("^MSTRG", gry))
## [1] 91
length(grep("^MSTRG", brwn))
## [1] 1460
length(grep("^MSTRG", blk))
## [1] 99
length(grep("^MSTRG", grn))
## [1] 272
length(grep("^MSTRG", man))
## [1] 40
length(grep("^MSTRG", pink))
## [1] 48
length(grep("^MSTRG", pur))
## [1] 29
length(grep("^MSTRG", gyellow))
## [1] 4

## merge the lncRNAs from each module into one big file
lncRNA_allmodules <- c(blk[grep("^MSTRG", blk)], blu[grep("^MSTRG", blu)],
gyellow[grep("^MSTRG", gyellow)], man[grep("^MSTRG", man)],
pink[grep("^MSTRG", pink)], tur[grep("^MSTRG", tur)], brwn[grep("^MSTRG",
brwn)], pur[grep("^MSTRG", pur)], red[grep("^MSTRG", red)],
grn[grep("^MSTRG", grn)], yelw[grep("^MSTRG", yelw)], gry[grep("^MSTRG",
gry)])
```

```
#length(lncRNA_allmodules) # 7339

## remove any repeated genes
uniq_lncRNA_allmodules <- unique(lncRNA_allmodules)
length(uniq_lncRNA_allmodules)

## [1] 7339

# 7339 lncRNAs - no overlapping lncRNAs among modules

# mRNAs
# number of mRNAs in each module
length(grep("^mRNA", red))
## [1] 929
length(grep("^mRNA", blu))
## [1] 4797
length(grep("^mRNA", tur))
## [1] 8128
length(grep("^mRNA", yelw))
## [1] 1896
length(grep("^mRNA", gry))
## [1] 197
length(grep("^mRNA", brwn))
## [1] 4620
length(grep("^mRNA", blk))
## [1] 385
length(grep("^mRNA", grn))
## [1] 1531
length(grep("^mRNA", man))
## [1] 206
length(grep("^mRNA", pink))
## [1] 206
length(grep("^mRNA", pur))
## [1] 67
length(grep("^mRNA", gyellow))
## [1] 30

## merge the mRNAs from each module into one big file
mRNA_allmodules <- c(blk[grep("^mRNA", blk)], blu[grep("^mRNA", blu)],
gyellow[grep("^mRNA", gyellow)], man[grep("^mRNA", man)], pink[grep("^mRNA",
pink)], tur[grep("^mRNA", tur)], brwn[grep("^mRNA", brwn)], pur[grep("^mRNA",
pur)], red[grep("^mRNA", red)], grn[grep("^mRNA", grn)], yelw[grep("^mRNA",
yelw)], gry[grep("^mRNA", gry)])

## remove any repeated genes
uniq_mRNA_allmodules <- unique(mRNA_allmodules)
length(uniq_mRNA_allmodules)

## [1] 22992
```

```
# 22992 mRNAs - no overlapping mRNAs among modules

# all of the lncRNAs and mRNAs fell in one of the modules and the results
were different from the Pearson correlation results possibly because in the
Pearson correlation, we had the threshold cutoffs for the coefficient and p-
value which could have lowered the number of lncRNAs while in WGCNA all of
the genes are assigned to a module without any cutoff
```

11-e-2) lncRNAs and mRNAs found in each module (power of 10)

```
# genes found in each module
gry10 <- names(datExpr0)[moduleColors10=="grey"]
blu10 <- names(datExpr0)[moduleColors10=="blue"]
tur10 <- names(datExpr0)[moduleColors10=="turquoise"]
yelw10 <- names(datExpr0)[moduleColors10=="yellow"]
brwn10 <- names(datExpr0)[moduleColors10=="brown"]
blk10 <- names(datExpr0)[moduleColors10=="black"]
grn10 <- names(datExpr0)[moduleColors10=="green"]
red10 <- names(datExpr0)[moduleColors10=="red"]
pink10 <- names(datExpr0)[moduleColors10=="pink"]
man10 <- names(datExpr0)[moduleColors10=="magenta"]
pur10 <- names(datExpr0)[moduleColors10=="purple"]

# lncNRAs
# number of lncRNAs in each module
length(grep("^MSTRG", gry10))
## [1] 396
length(grep("^MSTRG", blu10))
## [1] 2931
length(grep("^MSTRG", tur10))
## [1] 2542
length(grep("^MSTRG", yelw10))
## [1] 288
length(grep("^MSTRG", brwn10))
## [1] 775
length(grep("^MSTRG", blk10))
## [1] 89
length(grep("^MSTRG", grn10))
## [1] 119
length(grep("^MSTRG", red10))
## [1] 69
length(grep("^MSTRG", pink10))
## [1] 79
length(grep("^MSTRG", man10))
## [1] 19
length(grep("^MSTRG", pur10))
## [1] 32
```

```r
## merge the lncRNAs from each module into one big file
lncRNA_allmodules10 <- c(blk10[grep("^MSTRG", blk10)], blu10[grep("^MSTRG",
blu10)], man10[grep("^MSTRG", man10)], pink10[grep("^MSTRG", pink10)],
tur10[grep("^MSTRG", tur10)], brwn10[grep("^MSTRG", brwn10)],
pur10[grep("^MSTRG", pur10)], red10[grep("^MSTRG", red10)],
grn10[grep("^MSTRG", grn10)], yelw10[grep("^MSTRG", yelw10)],
gry10[grep("^MSTRG", gry10)])

## remove any repeated genes
uniq_lncRNA_allmodules10 <- unique(lncRNA_allmodules10)
length(uniq_lncRNA_allmodules10)

## [1] 7339

# 7339 lncRNAs - no overlapping lncRNAs among modules

# mRNAs
# number of mRNAs in each module
length(grep("^mRNA", gry10))
## [1] 973
length(grep("^mRNA", blu10))
## [1] 4772
length(grep("^mRNA", tur10))
## [1] 10218
length(grep("^mRNA", yelw10))
## [1] 1656
length(grep("^mRNA", brwn10))
## [1] 3084
length(grep("^mRNA", blk10))
## [1] 390
length(grep("^mRNA", grn10))
## [1] 713
length(grep("^mRNA", red10))
## [1] 696
length(grep("^mRNA", pink10))
## [1] 291
length(grep("^mRNA", man10))
## [1] 113
length(grep("^mRNA", pur10))
## [1] 86

## merge the mRNAs from each module into one big file
mRNA_allmodules10 <- c(blk10[grep("^mRNA", blk10)], blu10[grep("^mRNA",
blu10)], man10[grep("^mRNA", man10)], pink10[grep("^mRNA", pink10)],
tur10[grep("^mRNA", tur10)], brwn10[grep("^mRNA", brwn10)],
pur10[grep("^mRNA", pur10)], red10[grep("^mRNA", red10)], grn10[grep("^mRNA",
grn10)], yelw10[grep("^mRNA", yelw10)], gry10[grep("^mRNA", gry10)])

## remove any repeated genes
```

```
uniq_mRNA_allmodules10 <- unique(mRNA_allmodules10)
length(uniq_mRNA_allmodules10)

## [1] 22992

# 22992 mRNAs - no overlapping mRNAs among modules
# the results were the same as power of 6
```

11-f) Quantify associations of individual genes with the trait of interest

This can be done for any of the other traits/conditions found in either “datTraits” or “sepTraits”. As an example, temperature from datTraits is the condition chosen for the following analyses

11-f-1) Power of 6

```
# temperature column of datTraits
tmp <- as.data.frame(datTraits$temp)
names(tmp) <- "temp"

# names (colors) of the modules and remove the first two letters from each
name
modNames <- substring(names(MEs), 3)

geneModuleMembership <- as.data.frame(cor(datExpr0, MEs, use = "p"))
MMPvalue <- as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership),
nSamples))
# add "MM" before each column name - MM = module membership
names(geneModuleMembership) <- paste("MM", modNames, sep="")
names(MMPvalue) <- paste("p.MM", modNames, sep="")

geneTraitSignificance <- as.data.frame(cor(datExpr0, tmp, use = "p"))
GSPvalue <- as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance),
nSamples))
# add "GS." before each column name - GS = Gene Significance
names(geneTraitSignificance) <- paste("GS.", names(tmp), sep="")
names(GSPvalue) <- paste("p.GS.", names(tmp), sep="")

# create dataframe containing each gene, their significance for temperature,
and module membership & p-values in all modules
geneInfo0 <- data.frame(genes = names(datExpr0), moduleColor = moduleColors,
geneTraitSignificance, GSPvalue)

# order the modules in the dataframe by their significance for temperature
modOrder <- order(-abs(cor(MEs, tmp, use = "p")))

# add module membership information in the chosen order
for (mod in 1:ncol(geneModuleMembership)){
  oldNames <- names(geneInfo0)
  geneInfo0 <- data.frame(geneInfo0, geneModuleMembership[, modOrder[mod]],
                         MMPvalue[, modOrder[mod]])
  names(geneInfo0) <- c(oldNames, paste("MM.", modNames[modOrder[mod]],
sep=""),
                       paste("p.MM.", modNames[modOrder[mod]], sep=""))
}
# order the genes in the geneInfo dataframe first by module color then by
geneTraitSignificance
geneOrder <- order(geneInfo0$moduleColor, -abs(geneInfo0$GS.temp))
geneInfo <- geneInfo0[geneOrder, ]
# head(geneInfo,5)
```

11-f-2) Power of 10

```
# names (colors) of the modules and remove the first two letters from each
name
modNames10 <- substring(names(MEs_10), 3)

geneModuleMembership10 <- as.data.frame(cor(datExpr0, MEs_10, use = "p"))
MMPvalue10 <-
as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership10), nSamples))
# add "MM" before each column name - MM = module membership
names(geneModuleMembership10) <- paste("MM", modNames10, sep="")
names(MMPvalue10) <- paste("p.MM", modNames10, sep="")

# geneTraitSignificance and GSPvalue values were taken from step 11-f-1
# create dataframe containing each gene, their significance for temperature,
and module membership & p-values in all modules
geneInfo0_10 <- data.frame(genes = names(datExpr0), moduleColor =
moduleColors10, geneTraitSignificance, GSPvalue)

# order the modules in the dataframe by their significance for temperature
modOrder10 <- order(-abs(cor(MEs_10, tmp, use = "p")))

# add module membership information in the chosen order
for (mod in 1:ncol(geneModuleMembership10)){
  oldNames10 <- names(geneInfo0_10)
  geneInfo0_10 <- data.frame(geneInfo0_10, geneModuleMembership10[,
modOrder10[mod]],
                         MMPvalue10[, modOrder10[mod]])
  names(geneInfo0_10) <- c(oldNames10, paste("MM.",
modNames10[modOrder10[mod]], sep=""),
                       paste("p.MM.", modNames10[modOrder10[mod]], sep=""))
}
# order the genes in the geneInfo10 dataframe first by module color then by
geneTraitSignificance
geneOrder10 <- order(geneInfo0_10$moduleColor , -abs(geneInfo0_10$GS.temp))
geneInfo10 <- geneInfo0_10[geneOrder10, ]
# head(geneInfo10,5)
```