Project.R

patel

2023-12-09

```r
#######################################################################
# Name: Kush Patel
# Project
# cohort: GDC TCGA Cervical Cancer (CESC) - HTSeq - Counts
# Exploring Grade Gene Expression (G1 vs G3) Analysis in GDC TCGA Cervical Cancer (CESC) Using RNA-Seq Data
# https://xenabrowser.net/datapages/?cohort=GDC%20TCGA%20Cervical%20Cancer%20(CESC)&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443
#######################################################################

# Libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```r
library(UCSCXenaTools) # needed to retrieve data
```

```
## Warning: package 'UCSCXenaTools' was built under R version 4.3.2
```

```
## =========================================================================================
## UCSCXenaTools version 1.4.8
## Project URL: https://github.com/ropensci/UCSCXenaTools
## Usages: https://cran.r-project.org/web/packages/UCSCXenaTools/vignettes/USCSXenaTools.html
##
## If you use it in published research, please cite:
## Wang et al., (2019). The UCSCXenaTools R package: a toolkit for accessing genomics data
##   from UCSC Xena platform, from cancer multi-omics to single-cell RNA-seq.
##   Journal of Open Source Software, 4(40), 1627, https://doi.org/10.21105/joss.01627
## =========================================================================================
##                               --Enjoy it--
```

```r
library(edgeR) # needed for processing, such as TMM
```

```
## Loading required package: limma
```

```r
library(limma) # needed to find DE probes

##############################################################################
# 1) Retrieve Data
##############################################################################

data(XenaData)

#GDC TCGA Cervical Cancer (CESC)

# limit to desired cohort
cesc <- XenaData %>% filter(XenaCohorts == 'GDC TCGA Cervical Cancer (CESC)')

# Get the phenotype / clinical data
cli_query = cesc %>%
  filter(Label == "Phenotype") %>%  # select clinical dataset
  XenaGenerate() %>%  # generate a XenaHub object
  XenaQuery() %>%     # generate the query
  XenaDownload()      # download the data
```

```
## This will check url status, please be patient.
```

```
## All downloaded files will under directory C:\Users\patel\AppData\Local\Temp\RtmpO4FBdT.
```

```
## The 'trans_slash' option is FALSE, keep same directory structure as Xena.
```

```
## Creating directories for datasets...
```

```
## Downloading TCGA-CESC.GDC_phenotype.tsv.gz
```

```r
# prepare (load) the data into R
cesc_pheno <- XenaPrepare(cli_query)

# Get the RNA-seq data, including the "probe map"
cli_query <- cesc %>% filter(Label == 'HTSeq - Counts') %>%
  XenaGenerate() %>%  # generate a XenaHub object
  XenaQuery() %>%
  XenaDownload(download_probeMap = TRUE)
```

```
## This will check url status, please be patient.
```

```
## Check ProbeMap urls of datasets.
```

```
## All downloaded files will under directory C:\Users\patel\AppData\Local\Temp\RtmpO4FBdT.
```

```
## The 'trans_slash' option is FALSE, keep same directory structure as Xena.
```

```
## Creating directories for datasets...
```

```
## Downloading TCGA-CESC.htseq_counts.tsv.gz
```

```
## Downloading gencode.v22.annotation.gene.probeMap
```

```r
# prepare (load) the data into R
cesc_counts <- XenaPrepare(cli_query)

##########################################################
# (2) Data pre-processing: we need to do a fair amount of
# filtering and re-arranging to work with the data, so
# that the expression and phenotype data are aligned
##########################################################

# First, let's use more manageable names
#   - X: expression data, with probes as row names
#   - probeMap: the probeMap
#   - Y: the pheno/clinical data

# for X, we need to set the rownames and remove the probe column
# from the data matrix
X <- data.frame(cesc_counts$TCGA.CESC.htseq_counts.tsv.gz)
rownames(X) <- X$Ensembl_ID
X <- X[,-1]  # remove the probe name column

# probeMap = probe names
probeMap <- cesc_counts$gencode.v22.annotation.gene.probeMap

# Y = pheno data
Y <- cesc_pheno
# The expression and clinical data need to match; currently, e.g.,
# The first column of the expression data does not correspond
# to the first row of the pheno data; the data is also not
# in a consistent format (one has '.' and the other has '-')

# compare sample names between X and Y; they do not match, and are
# not even in the same format!
print(colnames(X)[1])
```

```
## [1] "TCGA.DS.A7WI.01A"
```

```r
print(Y$submitter_id.samples[1])
```

```
## [1] "TCGA-EA-A97N-01A"
```

```r
# 'change '.' to '-' so sample ID format is consistent
colnames(X) <- gsub('\\.', '-', colnames(X))

# Note that the sample ID is a barcode that has a special meaning:
#   https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/
# In particular, the 4th section describes the 'Sample' which is
#   either tumor (01 - 09) or normal (10-19). For details see:
#   https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/sample-type-codes

# Keep only the '01A' tumor samples
g <- grep('01A$', colnames(X))
X <- X[,g]

# We still need to match the expression data with the clinical data
# Let's do that by first finding the samples that are common
# between the expression and clinical data. We can use
# intersect(a,b) to return a vector containing the elements common
# to vectors 'a' and 'b'

common_samples <- intersect(colnames(X), Y$submitter_id.samples)

# we then use match(x, t) to get a vector of indices. The value
# x[i] is the index of 't' containing the i^th value of 'x'

mx <- match(common_samples, colnames(X))
my <- match(common_samples, Y$submitter_id.samples)

X <- X[,mx]
Y <- Y[my,]

# Make sure that the samples match -- if they don't, this will produce an error
stopifnot(all(colnames(X) == Y$submitter_id.samples))

#Total number of samples and probes before processing
#"Total number of samples"
print(ncol(X))
```

```
## [1] 296
```

```r
#"Total number of probes"
print(nrow(X))
```

```
## [1] 60488
```

```r
##############################################
# Setup step 3: Process the expression data
##############################################

# convert from log2(count + 1) to count data
X <- round(2**X - 1)

# remove genes with low counts
dge <- DGEList(counts=X)
keep <- filterByExpr(dge,min.prop = .10 )
```

```
## Warning in filterByExpr.DGEList(dge, min.prop = 0.1): All samples appear to
## belong to the same group.
```

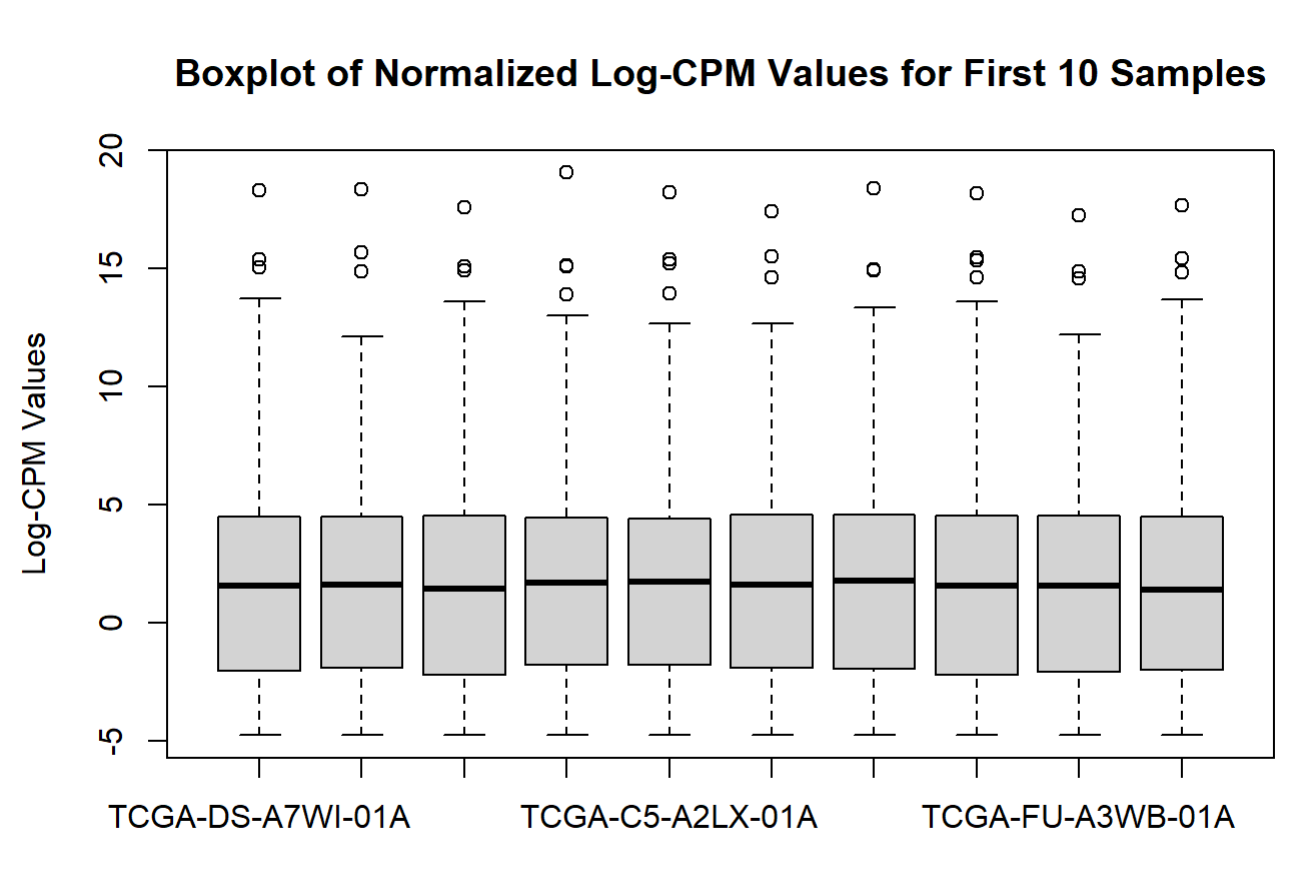
```r
dge <- dge[keep,,keep.lib.sizes=FALSE]

# apply TMM normalization, which computes the normalization
# factors. The actual normalization is done in a later step
dge <- calcNormFactors(dge, method = "TMM")

# Calculate the log CPM values, using the normalization factors;
# 3 counts are added to each observation to prevent log 0 values
logCPM <- cpm(dge, log = TRUE, prior.count = 3)

#Total number of samples and probes after processing
#"Total number of samples"
print(ncol(logCPM))
```

```
## [1] 296
```

```r
#Total number of probes"
print(nrow(logCPM))
```

```
## [1] 23996
```

```r
#print(colnames(X)[1])
#print(Y$submitter_id.samples[1])


##############################################
#5 Box plot for the first 10 samples
##############################################

boxplot(logCPM[, 1:10],
        main = "Boxplot of Normalized Log-CPM Values for First 10 Samples",
        ylab = "Log-CPM Values")
```




```r
##############################################
#6 Done
#7 Extract the column for the grade
##############################################

grade <- Y$neoplasm_histologic_grade
grade[is.na(grade)] <- "Unknown"
design <- model.matrix(~-1+grade)
colnames(design) <- c("G1", "G2", "G3", "G4", "GX", "Unknown")
#design
columns_to_keep <- colnames(design)[!(colnames(design) %in% c("G2", "G4", "GX", "Unknown"))]
design <- design[, columns_to_keep]
head(design)
```

```
##   G1 G3
## 1  0  0
## 2  1  0
## 3  0  0
## 4  0  1
## 5  0  0
## 6  0  0
```

```r
##############################################
#8 Use Limma to find probes across groups using fdr of 10%
##############################################

#fit the linear model to each row of the expression matrix
fit <- lmFit(logCPM, design)
#(G3 - G1)
contrast.matrix <- makeContrasts(G3 - G1, levels = design)
#fit model based on contrasts (e.g., G3 - G1)
fit <- contrasts.fit(fit, contrast.matrix)
#apply the 'eBayes' step to calculate moderated t statistics
fit.de <- eBayes(fit, trend = TRUE)
#FDR < %100
#FDR < %10 Resulted in 0 Values
tt.1 <- topTable(fit.de, sort.by = "p", p.value = 1, number = 30)
nrow(tt.1)
```

```
## [1] 30
```

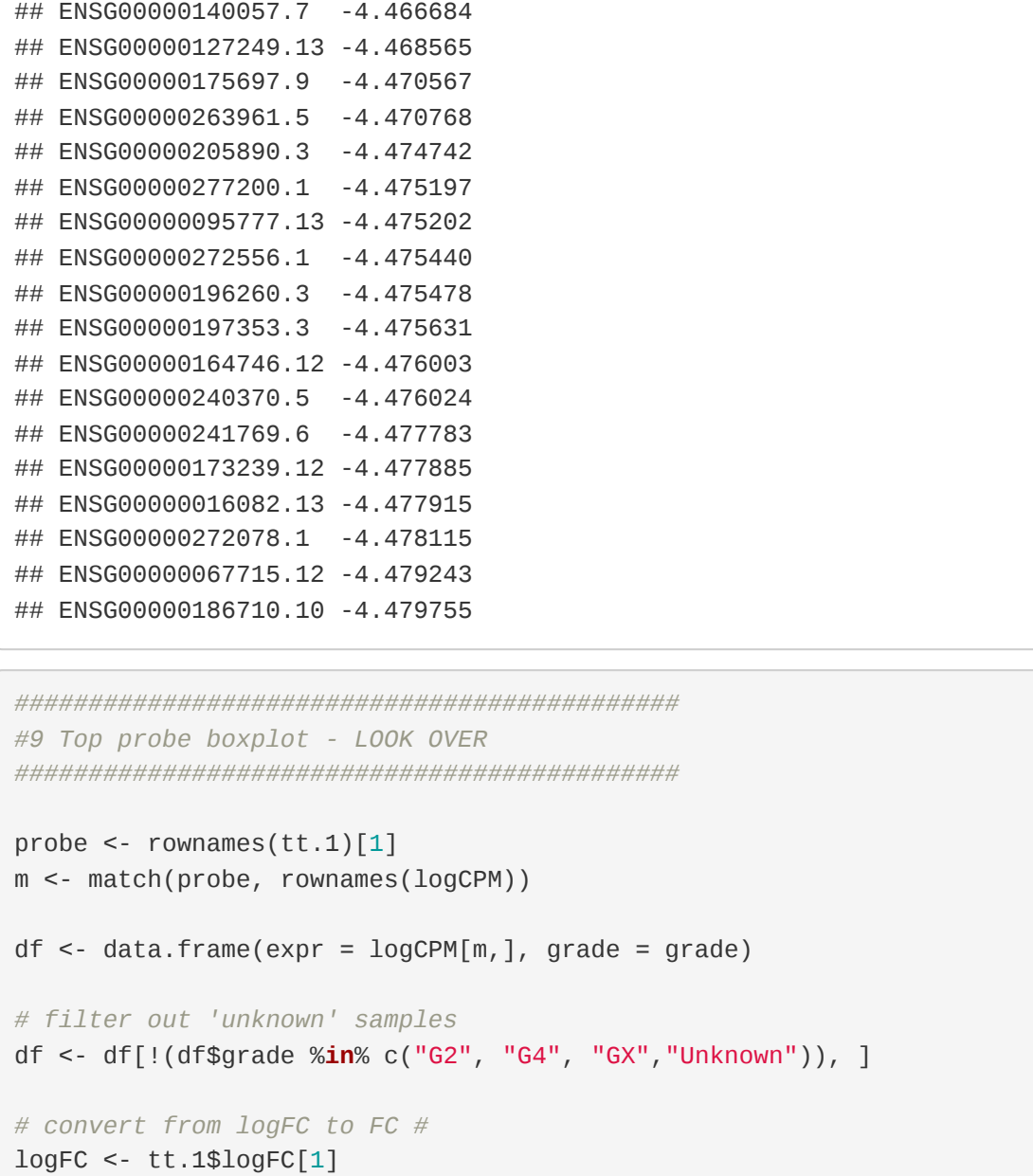
```r
#Top 30 (All of them)
tt.1
```

```
##                          logFC     AveExpr         t      P.Value adj.P.Val
## ENSG00000225439.2   1.4639931 -0.52648936  3.999947 7.928972e-05 0.6461721
## ENSG00000258818.3  -1.4749698 -0.74007813 -3.996740 8.031744e-05 0.6461721
## ENSG00000156413.12 -3.1950797  1.08862718 -3.995294 8.078497e-05 0.6461721
## ENSG00000125780.11 -3.0851285 -1.54226719 -3.822855 1.594564e-04 0.9565792
## ENSG00000166589.11 -3.0443380 -0.86844879 -3.648343 3.095801e-04 0.9999822
## ENSG00000144199.10  1.5644469  1.54307915  3.620460 3.434023e-04 0.9999822
## ENSG00000180861.8  -2.5096469 -0.25071249 -3.556783 4.341068e-04 0.9999822
## ENSG00000100078.3  -2.4274136  0.22361728 -3.537777 4.652605e-04 0.9999822
## ENSG00000060709.12 -2.2743976 -1.37718071 -3.437348 6.677174e-04 0.9999822
## ENSG00000007933.11 -2.4486036 -0.53373339 -3.372320 8.399259e-04 0.9999822
## ENSG00000162571.12 -1.5210566 -1.35574464 -3.359465 8.785347e-04 0.9999822
## ENSG00000175170.13 -1.6015619 -0.89059618 -3.332658 9.644262e-04 0.9999822
## ENSG00000140057.7  -1.4159020  0.16998708 -3.247565 1.291602e-03 0.9999822
## ENSG00000127249.13 -2.2661268 -0.11795747 -3.225349 1.392569e-03 0.9999822
## ENSG00000175697.9   1.6368251 -0.39786091  3.201568 1.508718e-03 0.9999822
## ENSG00000263961.5  -2.7358695  1.87739667 -3.199171 1.520907e-03 0.9999822
## ENSG00000205890.3  -1.2772494  0.42394454 -3.151459 1.783372e-03 0.9999822
## ENSG00000277200.1  -1.2166635 -0.31720894 -3.145957 1.816192e-03 0.9999822
## ENSG00000095777.13 -2.9051871  1.15872668 -3.145897 1.816555e-03 0.9999822
## ENSG00000272556.1   0.8772601 -0.64428598  3.143011 1.833994e-03 0.9999822
## ENSG00000196260.3  -2.5375017 -1.31566004 -3.142558 1.836744e-03 0.9999822
## ENSG00000197353.3  -2.9101791 -0.10418300 -3.140701 1.848061e-03 0.9999822
## ENSG00000164746.12 -2.0722942 -1.80976639 -3.136186 1.875850e-03 0.9999822
## ENSG00000240370.5  -0.7043624  0.11535959 -3.135931 1.877432e-03 0.9999822
## ENSG00000241769.6  -0.9190944  0.30858924 -3.114517 2.014669e-03 0.9999822
## ENSG00000173239.12 -1.1875510 -0.52780901 -3.113276 2.022905e-03 0.9999822
## ENSG00000016082.13 -2.1389327 -0.79961169 -3.112903 2.025383e-03 0.9999822
## ENSG00000272078.1  -0.6900014 -0.06771662 -3.110458 2.041707e-03 0.9999822
## ENSG00000067715.12  2.0569976 -0.93859151  3.096643 2.136245e-03 0.9999822
## ENSG00000186710.10 -1.4214327 -0.66946988 -3.090358 2.180575e-03 0.9999822
##                            B
## ENSG00000225439.2  -4.396548
## ENSG00000258818.3  -4.396873
## ENSG00000156413.12 -4.397019
## ENSG00000125780.11 -4.414146
## ENSG00000166589.11 -4.430845
## ENSG00000144199.10 -4.433452
## ENSG00000180861.8  -4.439344
## ENSG00000100078.3  -4.441086
## ENSG00000060709.12 -4.450157
## ENSG00000007933.11 -4.455911
## ENSG00000162571.12 -4.457037
## ENSG00000175170.13 -4.459374
## ENSG00000140057.7  -4.466684
## ENSG00000127249.13 -4.468565
## ENSG00000175697.9  -4.470567
## ENSG00000263961.5  -4.470768
## ENSG00000205890.3  -4.474742
## ENSG00000277200.1  -4.475197
## ENSG00000095777.13 -4.475202
## ENSG00000272556.1  -4.475440
## ENSG00000196260.3  -4.475478
## ENSG00000197353.3  -4.475631
## ENSG00000164746.12 -4.476003
## ENSG00000240370.5  -4.476024
## ENSG00000241769.6  -4.477783
## ENSG00000173239.12 -4.477885
## ENSG00000016082.13 -4.477915
## ENSG00000272078.1  -4.478115
## ENSG00000067715.12 -4.479243
## ENSG00000186710.10 -4.479755
```

```r
##############################################
#9 Top probe boxplot - LOOK OVER
##############################################

probe <- rownames(tt.1)[1]
m <- match(probe, rownames(logCPM))

df <- data.frame(expr = logCPM[m,], grade = grade)

# filter out 'unknown' samples
df <- df[!(df$grade %in% c("G2", "G4", "GX","Unknown")), ]

# convert from logFC to FC #
logFC <- tt.1$logFC[1]
2**logFC
```

```
## [1] 2.758709
```

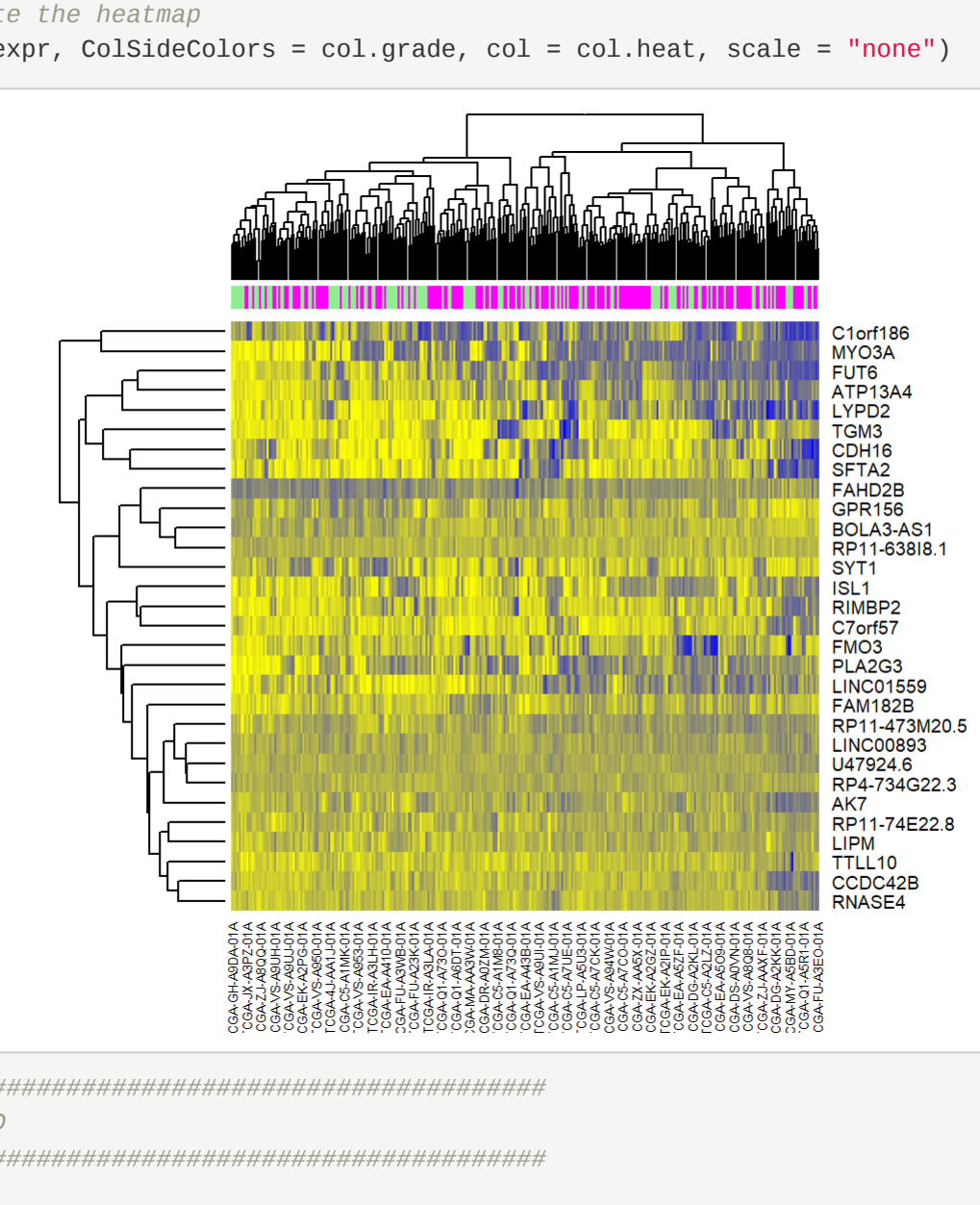
```r
## visualize ##
FC <- paste0("FC = ", round(2**logFC, 2))
main <- paste0("Expression of ", probe, ", ", FC, ", FDR = 100%")

ggplot(df, aes(x = grade, y = expr, fill = grade)) + geom_boxplot() +
  ylab("log2 expression") + ggtitle(main) +
  scale_fill_manual(values = c("pink", "lightblue")) +
  theme_classic() + theme(legend.position = "none")
```




```r
##############################################
#10 Df of all genes
##############################################
# gene names, probe names, logFC, and adjusted p-values
probe_names <- rownames(tt.1)
m <- match(probe_names, probeMap$id)
gene_names <- probeMap$gene[m]
logFc_Values <- tt.1$logFC
p_Values <- tt.1$adj.P.Val

#DF
result_df <- data.frame(
  GeneNames = gene_names,
  ProbeNames = probe_names,
  LogFC = logFc_Values,
  AdjPValues = p_Values
)

head(result_df, 5)
```

```
##   GeneNames         ProbeNames     LogFC AdjPValues
## 1 BOLA3-AS1  ENSG00000225439.2  1.463993  0.6461721
## 2    RNASE4  ENSG00000258818.3 -1.474970  0.6461721
## 3      FUT6 ENSG00000156413.12 -3.195080  0.6461721
## 4      TGM3 ENSG00000125780.11 -3.085128  0.9565792
## 5     CDH16 ENSG00000166589.11 -3.044338  0.9999822
```

```r
##############################################
#11 Heat Map - Recheck
##############################################

m <- match(result_df$ProbeNames, rownames(logCPM))
expr <- logCPM[m,]
rownames(expr) <- result_df$GeneNames
# create a color range consisting of 200 values between yellow and blue
col.heat <- colorRampPalette(c("yellow", "blue"))(200)

# set colors for grade
col.grade <- as.integer(as.factor(!(grade %in% c("G2", "G4", "GX","Unknown"))))
col.grade <- c("magenta", "lightgreen")[col.grade]

# Generate the heatmap
heatmap(expr, ColSideColors = col.grade, col = col.heat, scale = "none")
```




```r
##############################################
#12 DAVID
##############################################

#genes <- unique(result_df$GeneNames) # get unique set of genes (removes duplicates)
#genes
#write.table(genes, row.names = FALSE, quote = FALSE, file = "top_genes.txt")
```