Homework #1

Suruchi Ahuja, Abbas Rizvi, Hoi Lam Tai, Jingchen Zhang

February 15, 2016

Problem 1

The dataset from Lec04.R was loaded. The following code was provided to us by Dr. Gaile.

```
require(knitr)
require(GEOquery)
load("gse19439.RData")
#so I am just going to load it now
load(file="gse19439.RData")
gse19439
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 48791 features, 42 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM484448 GSM484449 ... GSM484489 (42 total)
##   varLabels: title geo_accession ... data_row_count (46 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: ILMN_1343291 ILMN_1343295 ... ILMN_2416019 (48791
##     total)
##   fvarLabels: ID nuID ... GB_ACC (30 total)
##   fvarMetadata: Column Description labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: GPL6947
```

Dr. Gaile creates a factor object `FTB` for the groups (control, latent TB, and positive TB) that corresponds to the samples (in the same order).

```
# make a factor object for Control, latent TB and Positive TB
tmp <- as.character(pData(phenoData(gse19439))[,1])
J <- length(tmp) # J=number of samples
TBgroup <- rep("",J)
for(j in 1:J) TBgroup[j]=substring(tmp[j],1,3)
# make a factor for TBgroup
FTB <- factor(TBgroup,levels=c("CON","LTB","PTB"))
X <- exprs(gse19439)
# do a quick kruskal-wallis scan
myKrusk <- function(i){
        cat(i,"...",fill=F)
        kruskal.test(x=X[i,], g=FTB)$p.value
}
```

The p-values that were calculated from the Kruskal-Wallis test were recorded into `myPvals.RData`. The best 4 p-values (`myPvals`) were subsequently sorted in ascending order (the lowest value at the beginning of the vector). This order was assigned to groups in STA 525.

```
load("myPvals.RData")
GroupLabels <- c("Group I","Group II","Group III","Group IV")
# pick the best 4 p-values and assign them to the students.
best4 <- order(myPvals)[1:4]
# print out list of best 4
print(best4)
```

```
## [1]  6874 10685 26058 47526
```

```
for(i in 1:length(GroupLabels)){
        cat("Group Label:", GroupLabels[i], "\t\t row assignment:", best4[i], fill=T)
}
```

```
## Group Label: Group I          row assignment: 6874
## Group Label: Group II         row assignment: 10685
## Group Label: Group III        row assignment: 26058
## Group Label: Group IV         row assignment: 47526
```

```
myrow <- gse19439[10685,]
dim(myrow)
```

```
## Features  Samples
##        1       42
```

Since we are group II, our row assignment was determined to be 10685 (as shown above). We were asked whether or not the assigned row corresponded to a specific gene and if so, what is known about the gene. Since each row does correspond to a gene, we subsetted `ExpressionSet` just to our assigned row and pulled out the `featureData`.

```
featureNames(myrow)
```

```
## [1] "ILMN_1703335"
```

```
gene.info <- pData(featureData(myrow))
gene.info
```

```
##                         ID               nuID      Species Source
## ILMN_1703335 ILMN_1703335 iMeiqS6uUsu15619eA Homo sapiens RefSeq
##              Search_Key Transcript ILMN_Gene Source_Reference_ID
## ILMN_1703335 ILMN_24565 ILMN_24565     LACTB         NM_032857.2
##                RefSeq_ID Unigene_ID Entrez_Gene_ID       GI   Accession
## ILMN_1703335 NM_032857.2                    114294 26051230 NM_032857.2
##              Symbol Protein_Product Array_Address_Id Probe_Type
## ILMN_1703335  LACTB     NP_116246.2          1570669          I
##              Probe_Start
## ILMN_1703335        1493
##                                                        SEQUENCE Chromosome
## ILMN_1703335 ATACTGGAGGGGCAGTGGGTGCCAGTAGTGTCCTGCTGGTCCTTCCTGAA         15
##              Probe_Chr_Orientation Probe_Coordinates Cytoband
## ILMN_1703335                     + 40256913-40256962 15q22.2b
```

```
##
## ILMN_1703335 Homo sapiens lactamase, beta (LACTB), nuclear gene encoding mitochondrial protein, trans
##                                                                                 Ontology_Component
## ILMN_1703335 membrane [goid 16020] [evidence IEA]; integral to membrane [goid 16021] [evidence IEA]
##
## ILMN_1703335 beta-lactam antibiotic catabolism [goid 30655] [evidence IEA]; response to antibiotic [g
##                                                                                                 Ontolog
## ILMN_1703335 beta-lactamase activity [goid 8800] [evidence IEA]; hydrolase activity [goid 16787] [evi
##                            Synonyms     Obsolete_Probe_Id      GB_ACC
## ILMN_1703335 FLJ14902; G24; MRPL56 FLJ14902; G24; MRPL56 NM_032857.2
```

Problem 2

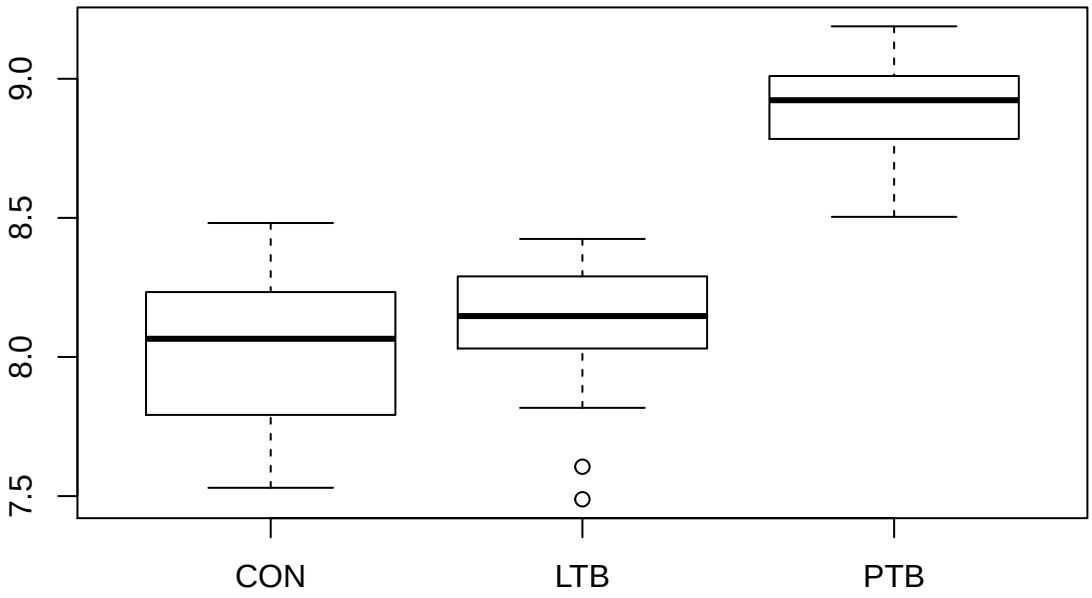
2.1 Boxplot

The data was log2 transformed for scability.

```
sampleNames(myrow)
```

```
##  [1] "GSM484448" "GSM484449" "GSM484450" "GSM484451" "GSM484452"
##  [6] "GSM484453" "GSM484454" "GSM484455" "GSM484456" "GSM484457"
## [11] "GSM484458" "GSM484459" "GSM484460" "GSM484461" "GSM484462"
## [16] "GSM484463" "GSM484464" "GSM484465" "GSM484466" "GSM484467"
## [21] "GSM484468" "GSM484469" "GSM484470" "GSM484471" "GSM484472"
## [26] "GSM484473" "GSM484474" "GSM484475" "GSM484476" "GSM484477"
## [31] "GSM484478" "GSM484479" "GSM484480" "GSM484481" "GSM484482"
## [36] "GSM484483" "GSM484484" "GSM484485" "GSM484486" "GSM484487"
## [41] "GSM484488" "GSM484489"
```

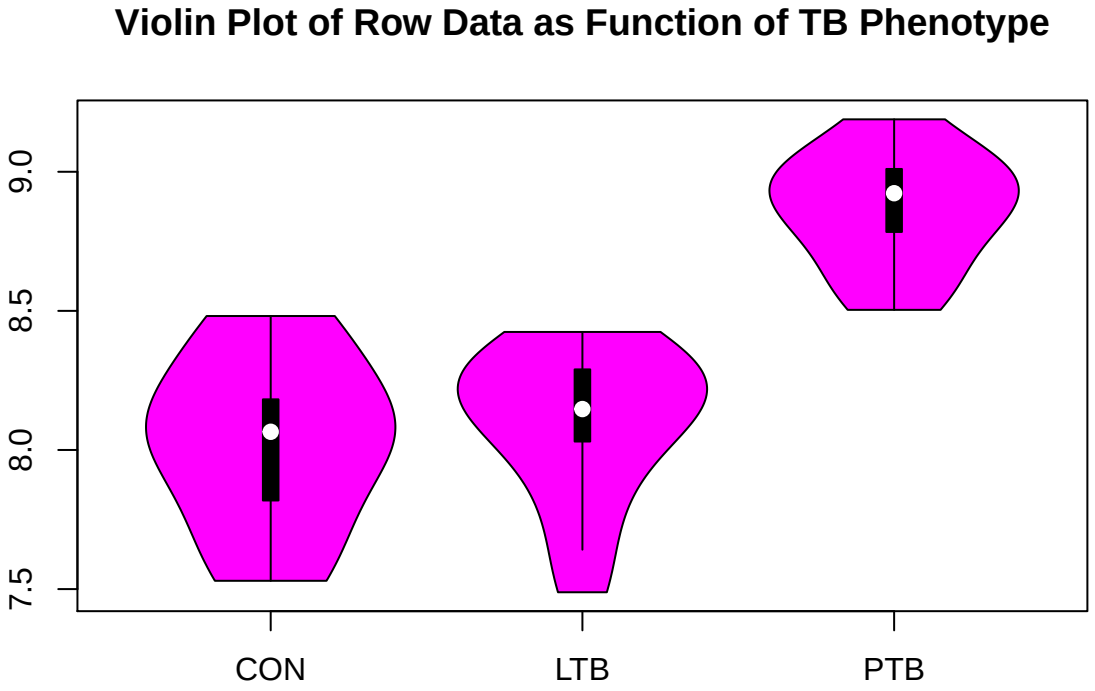
```
boxplot(log2(X[10685,sampleNames(myrow)[FTB == 'CON']]),
        log2(X[10685,sampleNames(myrow)[FTB == 'LTB']]),
        log2(X[10685,sampleNames(myrow)[FTB == 'PTB']]),
        main = "Boxplot of Row Data as Function of TB Phenotype",
        names = c("CON", "LTB", "PTB"))
```




```
#boxplots disguise multimodal data
```

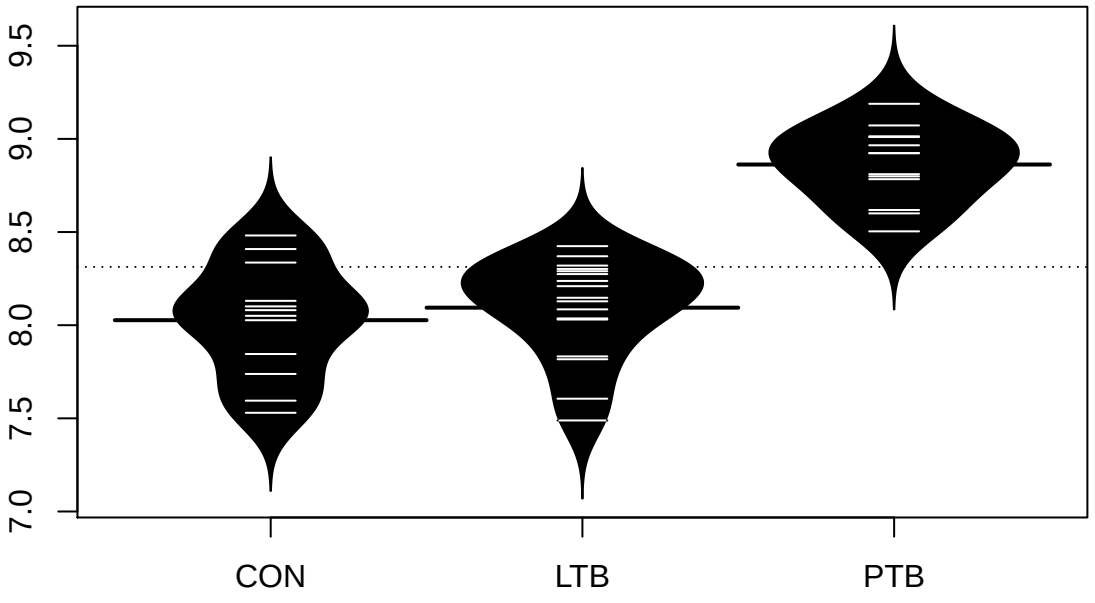
2.2 Violin Plot

```
require(vioplot)
vioplot(log2(X[10685,sampleNames(myrow)[FTB == 'CON']]),
        log2(X[10685,sampleNames(myrow)[FTB == 'LTB']]),
        log2(X[10685,sampleNames(myrow)[FTB == 'PTB']]),
        names = c("CON", "LTB", "PTB"))
title("Violin Plot of Row Data as Function of TB Phenotype")
```



2.3 Bean Plot

```
require(beanplot)
beanplot(log2(X[10685,sampleNames(myrow)[FTB == 'CON']]),
        log2(X[10685,sampleNames(myrow)[FTB == 'LTB']]),
        log2(X[10685,sampleNames(myrow)[FTB == 'PTB']]),
        names = c("CON", "LTB", "PTB"))
title("Bean Plot of Row Data as Function of TB Phenotype")
```

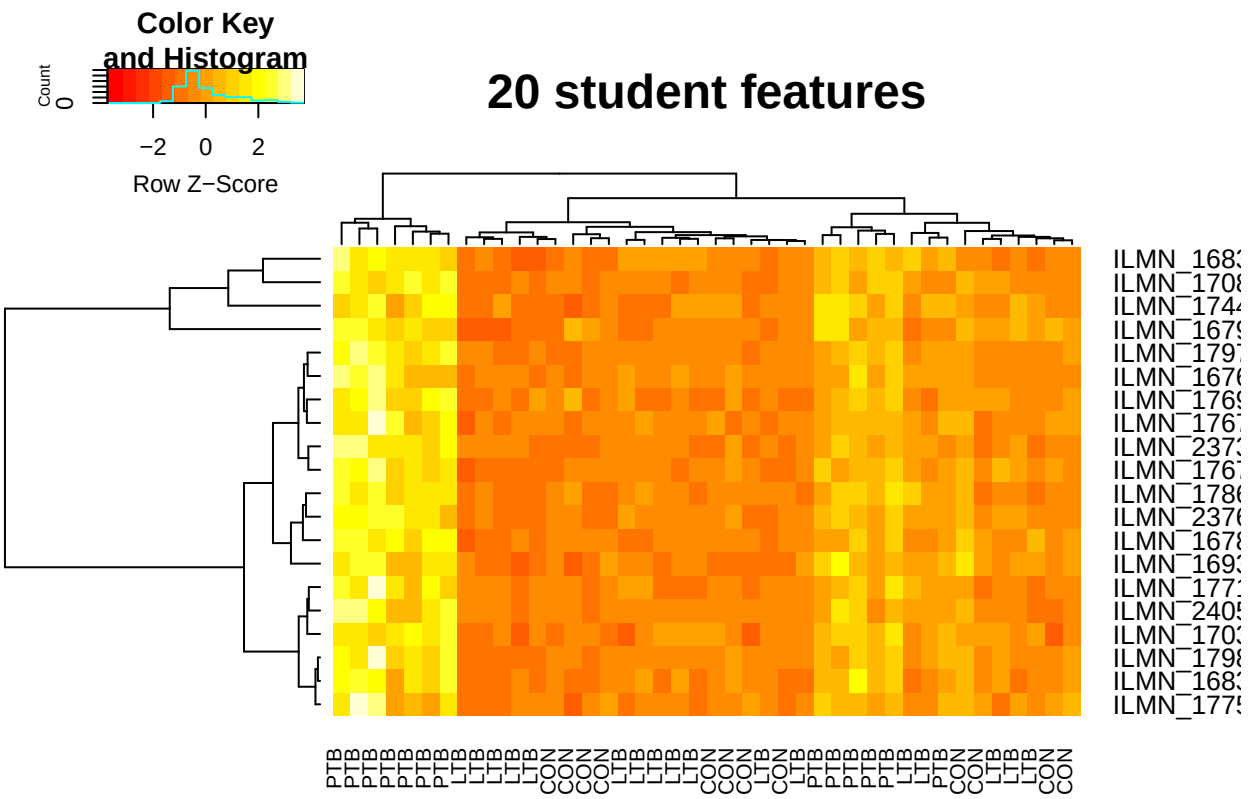


Problem 3

3.1 Heatmaps

The best 20 student features were clustered and plotted in a heatmap using `heatmap.2`. The clustering method worked reasonably well in determined what group the samples belonged to (see column labels of heatmap).

```
require(gplots)
best20 <- order(myPvals)[1:20]
x.best <- X[best20,]
heatmap.2(X[best20,], main="20 student features",
          scale = "row", trace="none", labCol = FTB)
```



Problem 4

4.1 Pre-processing

We removed NAs and zeroes.

```
spikeDF <- read.table(file="AffyProbeSpikeValues.csv",sep="\t")
levels(spikeDF[,2])
```

```
##  [1] "0"                "0.25"             "0.285714285714286"
##  [4] "0.4"              "0.666666666666667" "0.833333333333333"
```

```
##  [7] "1"                 "1.5"               "1.7"
## [10] "2"                 "3"                 "3.5"
## [13] "MC"                "MF"
```

```
summary(spikeDF[,2])
```

```
##                  0               0.25 0.285714285714286               0.4
##              13337                192                174                163
## 0.666666666666667 0.833333333333333                  1                1.5
##                189                166               3426                167
##                1.7                  2                  3                3.5
##                166                184                 98                445
##                 MC                 MF
##                231                 14
```

```
#grab Spike fold changes for all entries as numeric ...
#...so anything that was not a number is now an NA
SpikeFC <- as.numeric(levels(spikeDF[,2])[spikeDF[,2]])
```

```
## Warning: NAs introduced by coercion
```

```
#grab IDs
names(SpikeFC) <- spikeDF$V1

#remove NAs
nonZeroDX <- which(!is.na(SpikeFC) & (SpikeFC != 0))
spikeFC.clean <- SpikeFC[nonZeroDX]

#check how many genes are left in dataset
length(spikeFC.clean)
```

```
## [1] 5370
```

4.2 Normalization of 8 routes with expresso() and subset into ExpressionSet

```
require(affy)
bgcorrect.mtd <- c("rma", "mas", "rma", "mas",
                   "rma", "none", "mas", "mas")
normalized.mtd <- c("constant", "quantiles", "quantiles", "loess",
                    "loess", "constant", "qspline", "qspline")
pmcorrect.mtd <- c("pmonly", "pmonly", "subtractmm", "mas",
                   "mas", "pmonly", "subtractmm", "mas")
summary.mtd <- c("mas", "mas", "avgdiff", "medianpolish",
                 "medianpolish", "avgdiff", "mas", "mas")

route.expsets <- list()
for (i in 1:length(bgcorrect.mtd)){
        routes <- expresso(affydata,
                           bgcorrect.method = bgcorrect.mtd[i],
                           normalize.method = normalized.mtd[i],
```

```
                            pmcorrect.method = pmcorrect.mtd[i],
                            summary.method = summary.mtd[i])
        route.expsets[[i]] <- exprs(routes)[nonZeroDX,]
}
```

4.3 Multiple hypothesis testing

```
#create labels for multitesting class labels -- 18 samples, 9 control, 9 experimental
labels <- factor(c(rep(0, 9), rep(1, 9))) #0 is control, 1 is experimental

#source("http://bioconductor.org/biocLite.R")
#biocLite("multtest")
library(multtest)
stats <- list()
for (i in 1:length(route.expsets)){
        #conduct multiple t test for 10000 permutations
        testing.routes <- mt.maxT(route.expsets[[i]],
                                  classlabel = labels, B = 10000)
        stats[[i]] <- testing.routes$teststat[order(testing.routes$index)]
}

nrow <- nrow(route.expsets[[1]])
myresponse <- rep(NA, nrow)
#if the spike value is 1 ... assign as 0s in matrix ... 1s are the control
myresponse[which(spikeFC.clean == 1)] = 1
#if spike value is not 1 ... assign as 1s in the matrix ... 0s are exp.
myresponse[which(spikeFC.clean != 1)] = 0
```

4.4 ROC Curves

```
#apply roc function on input
roc.fnct <- function(x){roc(response = myresponse,
                            predictor = abs(x), plot=TRUE, print.auc=TRUE)}
#apply roc function on all the ordered test statistics in the list stats
roc <- lapply(stats, roc.fnct)
```

```
load("question4.RData")
rainbow <-  palette(rainbow(length(route.expsets)))
plot(roc[[1]], main = "ROC curves for different normalization routes using expresso()",
     col=rainbow[1])
```

```
##
## Call:
## roc.default(response = myresponse, predictor = abs(x), plot = TRUE,     print.auc = TRUE)
##
## Data: abs(x) in 1944 controls (myresponse 0) > 3426 cases (myresponse 1).
## Area under the curve: 0.9307
```

```
legendText <- c()
for(i in 1:length(route.expsets)){
        plot(roc[[i]], add=TRUE, col=rainbow[i])
        legendText[i] <- paste(bgcorrect.mtd[i],
                               "/",normalized.mtd[i],"/",
                               pmcorrect.mtd[i],
                               "/",summary.mtd[i],
                               "    AUC: ",
                               round(as.numeric(roc[[i]]$auc),3), sep="")
}
legend("bottomright",
       legendText,
       lty=c(rep(1,length(route.expsets))),
       lwd=c(rep(3,length(route.expsets))),
       col=rainbow)
```