

```
#####
# L6.1. MULTIPLE COMPARISON
#####
rm(list=ls())
## define data size
nsamples = 6
nfeatures = 1000 ## "genes"

## create random data
D = matrix(rnorm(nsamples*nfeatures),
           nrow=nfeatures, ncol=nsamples)
## vector of p-values
pv = double(nfeatures)

## let's apply t.test to each feature (gene)
## comparing 1,2,3 columns to 4,5,6
for (i in 1:nfeatures) {
  pv[i] = t.test(D[i,1:3], D[i,4:6])$p.value
}
## count number of 'significant' p-values
sum(pv < 0.05)

par(mfcol=c(2,4))
for (i in which(pv < 0.01)) {
  plot(D[i,], col=c(2,2,2,4,4,4), pch=19, cex=2)
}
## make FDR adjustment...
sum(p.adjust(pv, method="fdr") < 0.05)

#####
# L6.2. SURVIVAL ANALYSIS
#####
rm(list=ls())
library(survival)
str(lung)

## create a survival object
## lung$status: 1-censored, 2-dead
sData = Surv(lung$time, event = lung$status == 2)
print(sData)

## Let's visualize it
fit = survfit(sData~1)
plot(fit)

## Let's visualize it for male/female
fit.sex = survfit(sData ~ lung$sex)
plot(fit.sex, col=c("blue", "red"), conf.int = F)
str(fit.sex)

## Rank test for survival data
dif.sex = survdiff(sData ~ lung$sex)
dif.sex
str(dif.sex)

## build Cox regression model
```

```

mod1 = coxph(sData ~ sex, data=lung)
summary(mod1)

## build Cox regression model
mod2 = coxph(sData ~ sex + age, data=lung)
summary(mod2)

#####
library(survival)
str(melanoma)
sData = Surv(melanoma$time, event = melanoma$status == 1)
print(sData)
fit = survfit(sData~1)
plot(fit)

fit.sex = survfit(sData ~ melanoma$sex)
plot(fit.sex, col=c("red", "blue"), conf.int=T)

melanoma$th = factor(c("low", "high")[as.integer(melanoma$thickness>median(melanoma$thickness
))+1])
fit.th = survfit(sData ~ melanoma$th)
plot(fit.th, col=c("red", "blue"), conf.int=T)

## build Cox regression model
mod1 = coxph(sData ~ thickness, data=melanoma)
summary(mod1)

## build Cox regression model
mod1 = coxph(sData ~ age, data=melanoma)
summary(mod1)

#####
# L6.3. MICROARRAY DATA ANALYSIS
#####
rm(list=ls())
Data = read.table("N:/DOCUMENTS/Pedagogics/_upload/lusc.txt",
header=T, sep="\t", as.is=T)
## data preparation -> transform to a matrix
## and remove repeating gene
X = as.matrix(Data[-16273,-1])
rownames(X) = Data$gene[-16273]
str(X)
group = sub("[.] .+", "", colnames(X))
color = group
color[color=="tumour"]="red"
color[color=="normal"]="blue"

## see data distribution
plot(density(X), lwd=2)
for(i in 1:ncol(X)) lines(density(X[,i]), col=color[i])
abline(v=5, col="green")

## how many genes are expressed?
idx.expr = apply(X, 1, max)>5
sum(idx.expr)

```

```

## let's filter out non-expressed
X = X[idx.expr, ]

## perform PCA
PC = prcomp(t(X))
str(PC)
plot(PC$x[,1], PC$x[,2], col=color, pch=19)

## DEA
library(limma)
design=matrix(~ -1 + as.factor(group))
str(design)
colnames(design)=c("normal", "tumour")
fit = lmFit(X, design=design)
contrast.matrix=makeContrasts(tumour-normal, levels=design)
fit2 = contrasts.fit(fit, contrast.matrix)
EB = eBayes(fit2)
TopGeneTable = topTable(EB, number=nrow(X), adjust="BH", sort.by="p")
str(TopGeneTable)
## top 100 genes
genes = rownames(TopGeneTable)[1:100]
# genes = TopGeneTable$ID[1:100]

XTop = X[genes, ]
heatmap(t(scale(t(XTop))),
        # ColSideColors=color,
        # scale="none",
        # col= colorRampPalette(c("blue", "wheat", "red"))(1000))

save(list=c("X", "TopGeneTable"), file = "MA.RData")

## or Alternatively

source("http://sablab.net/scripts/limmaEBS2Class.r")
res = limmaEBS2Class(data = X,
                    # anno = rownames(X),
                    # classes = group,
                    # plotTop=100,
                    # col=c("blue", "red"))
#####
rm(list=ls())
Meta = read.table("http://edu.sablab.net/data/gz/Affymetrix_miRNA2.txt", sep="\t", header=T,
as.is=T)
Data = read.table("http://edu.sablab.net/data/gz/rma-sketch.summary.txt", sep="\t", header=T,
as.is=T)
idx.hsa = grep("^hsa-", Data$probeset_id)
X = as.matrix(Data[idx.hsa, -1])
rownames(X) = Data[idx.hsa, 1]

group = factor(sub("[.] +", "", Meta$name))
color = rainbow(10)[as.integer(group)]

## see data distribution
plot(density(X), lwd=2)
for(i in 1:ncol(X)) lines(density(X[,i]), col=color[i])

```

```

abline(v=5, col="green")

## how many genes are expressed?
idx.expr = apply(X, 1, max) > 5
sum(idx.expr)
## let's filter out non-expressed
X = X[idx.expr, ]

## perform PCA
PC = prcomp(t(X))
str(PC)
plot(PC$x[, 1], PC$x[, 2], col=col, cex=2, pch=19)

## perform DEA
library(limma)
design=matrix(~ -1 + group)
str(design)
colnames(design)=sub("group", "", colnames(design))
fit = lmFit(X, design=design)
contrast.matrix=makeContrasts(T24-T000,
..... T48-T000,
->>>>>>> T72-T000,
..... levels=design)
fit2 = contrasts.fit(fit, contrast.matrix)
EB = eBayes(fit2)
TopGeneTable = topTable(EB, number=nrow(X), adjust="BH")
str(TopGeneTable)

##now lets see for each pair
Res24 = topTable(EB,
..... coef="T24 - T000",
..... number=nrow(X),
..... adjust="BH",
..... sort.by="none",
..... genelist=row.names(X))

idx = (TopGeneTable$adj.P.Val < 0.05)

heatmap(t(scale(t(X[idx, ]))),
->>Colv = NA,
..... ColSideColors=color,
..... scale="none",
..... col = colorRampPalette(c("blue", "wheat", "red"))(1000))
.
#####
## Functional enrichment
#####
idx = TopGeneTable$adj.P.Val < 0.0001 &
..... abs(TopGeneTable$logFC) > 2
sum(idx)
write.table(TopGeneTable[idx, ],
..... file="DE.txt", sep="\t")

#####
# L6.4. RNASEQ DATA ANALYSIS
#####

```

```
rm(list=ls())
load("LUSC60.RData")
source("http://sablabs.net/scripts/LibDEA.r")
library("DESeq2")
res.DESeq = DESeq(counts=LUSC$counts,
  ..., group=LUSC$meta$sample_type,
  >> key0="NT",
  >> key1="TP")

library("edgeR")
res.edgeR = DESeq(counts=LUSC$counts,
  ..., group=LUSC$meta$sample_type,
  >> key0="NT",
  >> key1="TP")
#####
## manual

dge = DGEList(counts=LUSC$counts,
  ..., group=as.factor(LUSC$meta$sample_type))
prior.df=10
dge = calcNormFactors(dge)
dge = estimateCommonDisp(dge)
dge = estimateTagwiseDisp(dge,prior.df=prior.df)
res = exactTest(dge)[[1]]
res$FDR=p.adjust(res$PValue,"fdr")
res = data.frame(id=rownames(LUSC$counts),res,stringsAsFactors=F)
```