Analyzing An Illumina Microarray Dataset Using R and Bioconductor

Giovanni Coppola, MD

Program in Neurogenetics
Department of Neurology
University of California Los Angeles
gcoppola@ucla.edu

August 2010

Introduction

The goal of this study (Lange et al 2008, [ref]) is to identify the genes that are up- or downregulated in response to oxidative stress induced by glutathione depletion, in normal neurons and neurons missing the ATF4 transcription factor. We examined primary immature neurons from KO mice for ATF4, and compared them to neurons from WT mice. Neurons were treated with HCA (glutathione depletion) or control drug. In summary, our series includes 2 genotypes (ATF4 and WT), 2 treatments (HCA and Control), and 4 to 8 replicates per condition.

1 Reading the Data In

After installing `R`,

- open a folder on your computer called `ArrayAnalysis`
- copy in this folder the present file, and the `targets.txt` file.
- Download and unzip in the folder `ArrayAnalysis` the GEO dataset file `GSE10470_Microarray_raw_data.txt`.
- Launch `R`

First, we need to load the packages that `R` needs to run the analysis:

```
> library (Biobase)
> library (marray)
> library (limma)
> library (gplots)
```

Then, we can read the data in from the file `GSE10470_Microarray_raw_data.txt`, and store it in the `mydataRAW` object.

```
> mydataRAW<-read.delim("GSE10470_Microarray_raw_data.txt",
                        sep="\t",skip=7,header=TRUE)
```

Note that the `read.delim` command 1) reads from the current directory; 2) uses `\t` (tab) as a separator, and 3) skips the first 7 rows of the Illumina output. The `mydataRAW` object now contains all the raw data we need for our analysis. We have a first column with a probe name and then 8 columns per array, so the total number of columns is 193 (=24 samples * 8 columns +1 column for probe name). The total number of rows corresponds to the number of probes on the array. We will use the signal column for normalization and differential expression analysis, and the other columns for quality control. The command `dim` shows the dimensions of the `mydataRAW` object:

```
> dim(mydataRAW)
```

```
[1] 24048   193
```

We then need to read in the sample information. This is stored in the `targets.txt` file.

```
> targets<-read.delim(file="targets.txt", header=T)
```

This command uploads the sample list and stores it in the object named `targets` (Table 1). We use the commans `str` to check the structure of an obect: each row represents a column, with the column type (character, numeric etc).

```
> str(targets)
```

```
'data.frame':        24 obs. of  6 variables:
 $ Array    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Batch    : int  1 1 1 1 1 1 1 1 2 2 ...
 $ Genotype : Factor w/ 2 levels "ko","wt": 2 2 2 2 1 1 1 1 1 2 ...
 $ Replicate: int  1 2 3 4 1 2 3 4 5 1 ...
 $ Drug     : Factor w/ 2 levels "ctrl","hca": 1 1 1 1 1 1 1 1 1 2 ...
 $ ArrayCode: Factor w/ 24 levels "1412066061_A",..: 1 2 3 4 5 6 7 8 9 10 ...
```

2 Checking Data Quality

First, we can extract the signal values from the main data object, and store it in the `mydata` object.

```
> nsampl=24
> mydata<-mydataRAW[,seq(1,(nsampl*8),8)+2]
> names(mydata)<-paste(targets$Genotype,targets$Drug,targets$Array,sep=".")
> rownames(mydata)<-mydataRAW$TargetID
```

	Array	Batch	Genotype	Replicate	Drug	ArrayCode
1	1	1	wt	1	ctrl	1412066061_A
2	2	1	wt	2	ctrl	1412066061_B
3	3	1	wt	3	ctrl	1412066061_C
4	4	1	wt	4	ctrl	1412066061_D
5	5	1	ko	1	ctrl	1412066061_E
6	6	1	ko	2	ctrl	1412066061_F
7	7	1	ko	3	ctrl	1412066061_G
8	8	1	ko	4	ctrl	1412066061_H
9	9	2	ko	5	ctrl	1541554116_A
10	10	2	wt	1	hca	1541554116_B
11	11	2	ko	1	hca	1541554116_C
12	12	2	wt	5	ctrl	1541554116_D
13	13	2	ko	6	ctrl	1541554116_E
14	14	2	wt	2	hca	1541554116_F
15	15	2	ko	2	hca	1541554116_G
16	16	2	wt	6	ctrl	1541554116_H
17	17	3	wt	7	ctrl	1541554117_A
18	18	3	ko	3	hca	1541554117_B
19	19	3	wt	3	hca	1541554117_C
20	20	3	ko	7	ctrl	1541554117_D
21	21	3	wt	8	ctrl	1541554117_E
22	22	3	ko	4	hca	1541554117_F
23	23	3	wt	4	hca	1541554117_G
24	24	3	ko	8	ctrl	1541554117_H

Table 1: Sample List

Signal is usually log2 transformed, and this is easily done in R:

```
> mydata<-as.matrix(log2(mydata))
```

We now will take a look at the signal distribution of the raw (non-normalized) data, using the boxplot function, in Figure 1 (page 4).

```
> batch<-as.character(targets$Batch+1)
> boxplot(as.data.frame(mydata),main="Signal distribution, Not normalized",
 col=batch,names=1:nsampl,xlab="Array",ylab="Log2 Intensity",cex=0.8)
```

Similarly, we can extract data about the detection score (Figure 2) — a number produced by the BeadStudio software and indicating the likelihood that a gene is expressed, ranging between 0 (not expressed) and 1 (expressed) — and the average number of beads comprising each beadset in each array (Figure 3).

```
> detsco<-mydataRAW[,seq(1,(nsampl*8),8)+8]
> boxplot(as.data.frame(detsco),main="Detection scores",
 names=1:nsampl,col=batch,xlab="Array",ylab="Detection Score",cex=0.8)
```

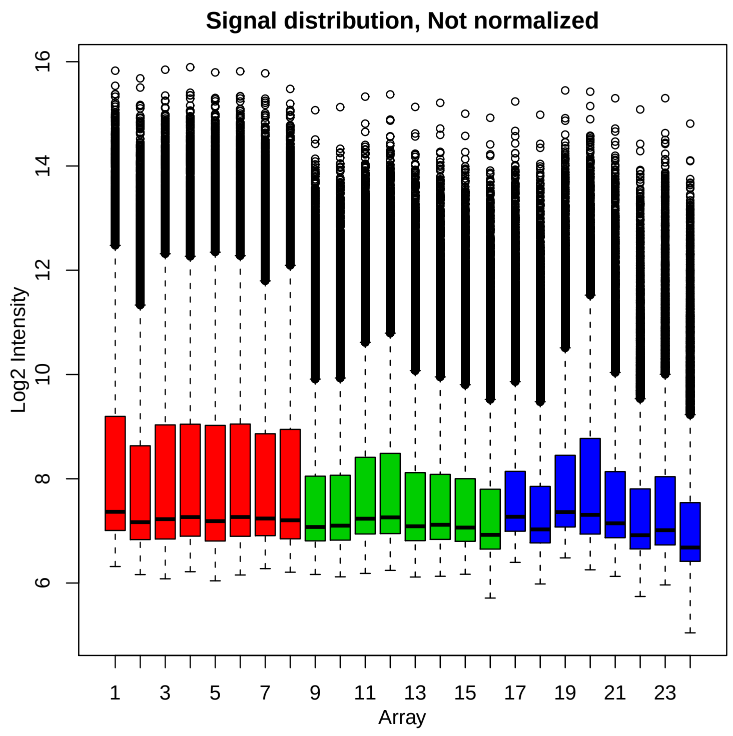

Figure 1: Boxplots representing the raw, log2-transformed signal in 24 arrays. Colors represent array batches.

```
> beadno<-mydataRAW[,seq(1,(nsampl*8),8)+7]
> boxplot(as.data.frame(beadno),main="Bead number per beadset",
 names=1:nsampl,col=batch,xlab="Array",ylab="Number of beads",cex=0.8)
```

3 Clustering samples based on Pearson correlation

A correlation matrix using all the probes on the array (Figure 4) can be used to cluster arrays and identify technical outliers.

```
> matcor<-cor(mydata)
> range(as.vector (matcor))
```

```
[1] 0.9569689 1.0000000
```

```
> geno<-as.character(as.numeric(targets$Genotype))
> treatment<-as.character(as.numeric(targets$Drug))
```

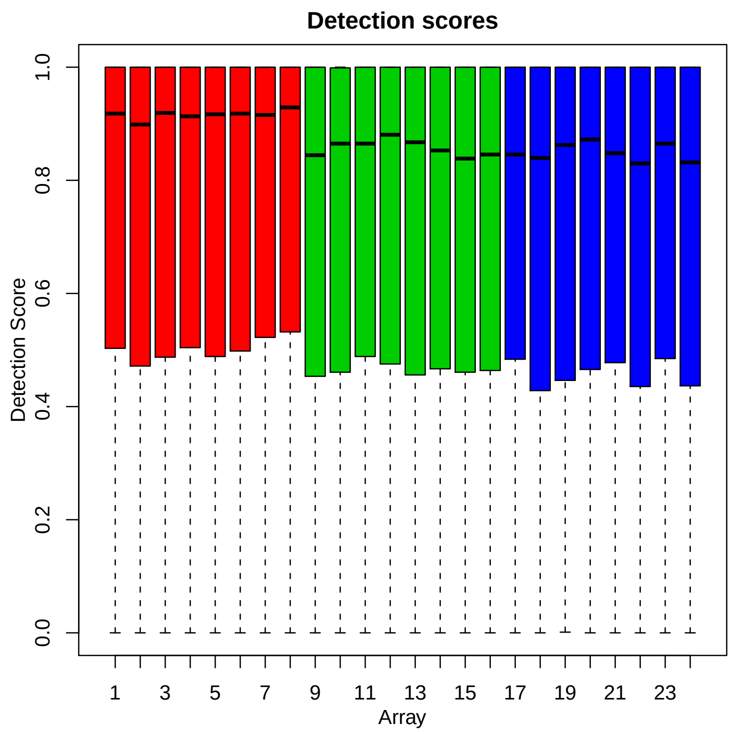

Figure 2: Boxplots representing the distribution of the Illumina detection score in 24 arrays.

```
> heatmap.2(matcor,trace="none",col=heat.colors(40),ColSideColors=geno,
            RowSideColors=treatment, cexCol=1,cexRow=1,labCol="")
```

4 Data Normalization

After quality check (and outlier exclusion if needed) inter-array normalization is performed using the quantile normalization (ref?). The function `normalizeBetweenArrays` from the package `limma` performs a number of normalization methods, specified in the parameter `method`. Boxplots of normalized values show normalized distribution (Figure 5).

```
> normalizeBetweenArrays(as.matrix(mydata),method="quantile")->mydataQuantile
> boxplot(as.data.frame(mydataQuantile),main="Quantile normalization",
        col= batch,names=1:nsampl)
```

After normalization, we can choose the top most variant genes using the median absolute deviation [ref.] and cluster them (Figure 6).

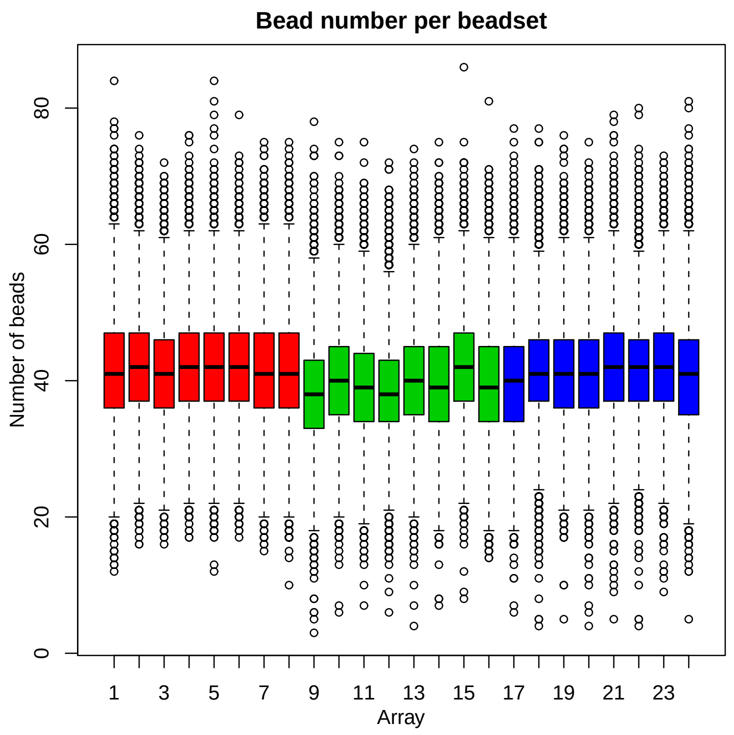

Figure 3: Boxplots representing the distribution of the number of beads comprising the beadset, for each array.

```
> labls<-paste (targets[,2], targets[,3], sep="_")
> standardize <- function(z) {
   rowmed <- apply(z, 1, median)
   rowmad <- apply(z, 1, mad)
   rv <- sweep(z, 1, rowmed)
   rv <- sweep(rv, 1, rowmad, "/")
   return(rv)
 }
> rowMads <- apply(mydata, 1, mad)
> ord <- order(rowMads,decreasing=TRUE)
> top1000 <- ord[1:1000]
> mydata1000<- mydata[top1000, ]
> mydata1000dist<- dist(t(standardize(mydata1000)))

> heatmap.2(as.matrix(mydata1000dist), col=rev(heat.colors(75)),
  distfun=function(x) as.dist(x),
  trace="none",ColSideColors=geno,cexRow=1,cexCol=1,
```

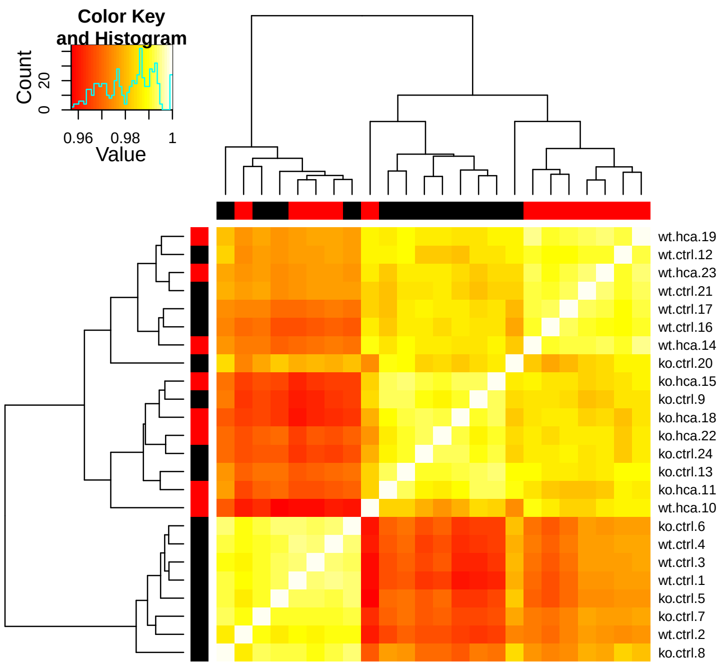

Figure 4: Heatmap and cluster dendrogram representing the correlation matrix of arrays with each other. Each box represents a pearson correlation value. Color coding (top bar) represents the genotype (top), and the treatment (side). Technical replicates tend to cluster together and no outliers are evident.

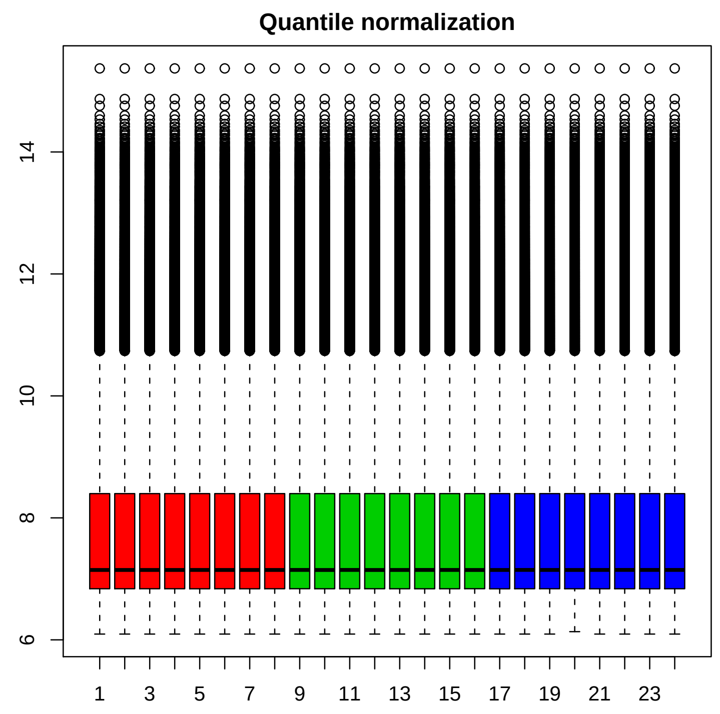

Figure 5: Boxplots representing the distribution of the Illumina detection score in 24 arrays after quantile normalization. Colors: slide batch

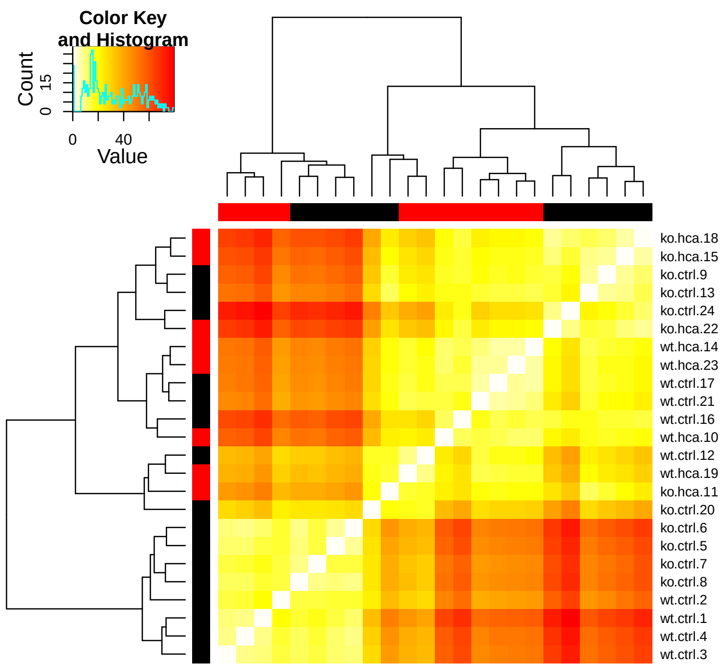

Figure 6: Clustering based on the top 1000 most variant genes. Color coding by genotype (top) and treatment (side bar).

```
  RowSideColors=treatment,labCol="")
```

The analysis of differential expression is aimed identifying the differentially expressed genes when comparing two or more conditions. First, we store all the possible conditions in the variable `conditions`

```
> conditions<- paste(targets$Genotype, targets$Drug,sep=".")
```

Then, we build a design matrix (Table 2, for details see the `limma` package documentation [ref]).

```
> conditions <- factor(conditions, levels=unique(conditions))
> design <- model.matrix(~0+ conditions)
> colnames(design) <- levels(conditions)
```

The command `lmFit` fits a linear model across the conditions

```
> fit <- lmFit(mydataQuantile, design)
```

	wt.ctrl	ko.ctrl	wt.hca	ko.hca
1	1.00	0.00	0.00	0.00
2	1.00	0.00	0.00	0.00
3	1.00	0.00	0.00	0.00
4	1.00	0.00	0.00	0.00
5	0.00	1.00	0.00	0.00
6	0.00	1.00	0.00	0.00
7	0.00	1.00	0.00	0.00
8	0.00	1.00	0.00	0.00
9	0.00	1.00	0.00	0.00
10	0.00	0.00	1.00	0.00
11	0.00	0.00	0.00	1.00
12	1.00	0.00	0.00	0.00
13	0.00	1.00	0.00	0.00
14	0.00	0.00	1.00	0.00
15	0.00	0.00	0.00	1.00
16	1.00	0.00	0.00	0.00
17	1.00	0.00	0.00	0.00
18	0.00	0.00	0.00	1.00
19	0.00	0.00	1.00	0.00
20	0.00	1.00	0.00	0.00
21	1.00	0.00	0.00	0.00
22	0.00	0.00	0.00	1.00
23	0.00	0.00	1.00	0.00
24	0.00	1.00	0.00	0.00

Table 2: Design matrix

After creating the design matrix and fitting the linear model, we create a contrast matrix (Table 3) in order to compare conditions. Then, the `eBayes` function assesses statistical significance using an empirical Bayes statistics.

```
> cont.matrix<- makeContrasts(
         KOvsWT= ko.ctrl- wt.ctrl,
         HCAwt= wt.hca- wt.ctrl,
         HCAko= ko.hca- ko.ctrl,
         HCAwtnotHCAko=(wt.hca- wt.ctrl)-( ko.hca- ko.ctrl),
         levels=design)
```

```
> fit.cont<- contrasts.fit(fit, cont.matrix)
> fit.cont<- eBayes(fit.cont)
```

We output an overview of the numbers of gene changes at different statistical thresholds (Figure 7)

	KOvsWT	HCAwt	HCAko	HCAwtnotHCAko
wt.ctrl	-1.00	-1.00	0.00	-1.00
ko.ctrl	1.00	0.00	-1.00	1.00
wt.hca	0.00	1.00	0.00	1.00
ko.hca	0.00	0.00	1.00	-1.00

Table 3: Contrast matrix

```
> decide <- matrix(c("fdr",0.05,
 "fdr",0.1, "none",0.005, "none", 0.01),nrow=4,ncol=2,byr=T)
> # initialize:
> mysum <- as.list(1:nrow(decide))
> mynum <- 0
> maxmax <- 0
> for (test in 1:nrow(decide)){
     results<-decideTests(fit.cont,
                          adjust.method=decide[test,1],
 p=as.numeric(decide[test,2]))
    summary(results) -> mysum[[test]]
    mynum[test] <-length(which(apply(results,1,function(x)any(x,na.rm=T))))
    maxmax <- max(c(maxmax, as.vector(mysum[[test]][c(1,3),])))
 }

> par(mfrow=c(1,nrow(decide)))
> for (test in 1:nrow(decide))
 {

   as.numeric(as.vector(mysum[[test]][3,]))->plotMe1
   as.numeric(as.vector(mysum[[test]][1,]))->plotMe2
   maxData = max(plotMe1)
   maxData2 = max(plotMe2)
   barplot(plotMe1,horiz=T,col="red",xlim=c(-maxmax,maxmax),
           main=paste("Gene Changes \np<",decide[test,2], ", " , decide[test,1],
             " (" ,mynum[test] ,")",sep=""))->yy
   barplot(-plotMe2,horiz=T,col="green",add=T)->yy

   xx<-vector("integer",ncol(mysum[[test]]))
   text(xx,yy,colnames(mysum[[test]]))
   text((plotMe1+10)*0 + .9*maxData,yy+0.1,format(plotMe1,digits=3))
   text((-plotMe2-10)*0 - .9*maxData2,yy+0.1,format(plotMe2,digits=3))
   }
```

We decide to apply a false discovery rate of 5% and use the corresponding parameters in the `decideTests` function from the `limma` package. This yields 281 genes across 4 contrasts.

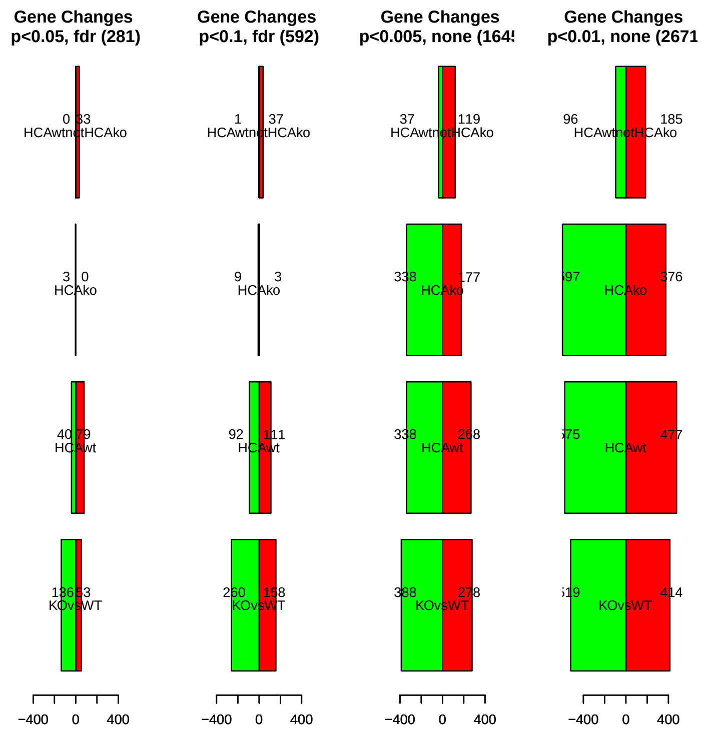

Figure 7: Number of differentially expressed genes (at each contrast) at different statistical thresholds. Red bars: upregulated genes, green bars: downregulated genes.

```
>   results<-decideTests(fit.cont,adjust.method="fdr",p=0.05)
> summary(results)

   KOvsWT HCAwt HCAko HCAwtnotHCAko
-1    136    40     3             0
0   23859 23929 24045         24015
1      53    79     0            33

> summary(results)->mysum05
> mysum05<-mysum05[,1:3]
> mysum05_no<-length(which(results[,1]!=0|results[,2]!=0|results[,3] !=0))
> maxmax<-max(as.vector(mysum05[c(1,3),]))
> mysum05<-mysum05[,1:3]
> as.numeric(as.vector(mysum05[3,]))->plotMe1
> as.numeric(as.vector(mysum05[1,]))->plotMe2
> maxData = max(plotMe1)
> maxData2 = max(plotMe2)
> barplot(plotMe1,horiz=T,col="red",xlim=c(-maxmax,maxmax),
         main=paste("Gene Changes @ FDR 5% (",mysum05_no,")",sep=""))->yy
> barplot(-plotMe2,horiz=T,col="green",add=T)->yy
> xx<-vector("integer",ncol(mysum05))
> text(xx,yy,colnames(mysum05))
> text((plotMe1+10)*0 + .9*maxmax,yy,format(plotMe1,digits=3))
> text((-plotMe2-10)*0 - .9*maxmax,yy,format(plotMe2,digits=3))
```

The resulting `fit` object hs columns for coefficients, t-test statistics, p-values, and a result code indicating whether the probe is differentially expressed.

```
>   write.fit(fit.cont,file="dummy.xls",adjust="fdr",results=results)
> read.table(file="dummy.xls",head=T)->fitObj
> str(fitObj)

'data.frame':        24048 obs. of  24 variables:
 $ A                      : num  9.65 7.07 13.56 13.44 10.82 ...
 $ Coef.KOvsWT            : num  0.005 -0.013 -0.082 -0.071 -0.242 0.091 -0.081 0.097 -0.0
 $ Coef.HCAwt             : num  -1.924 0.002 0.361 0.376 0.335 ...
 $ Coef.HCAko             : num  -1.529 0.041 0.144 0.598 -0.082 ...
 $ Coef.HCAwtnotHCAko     : num  -0.395 -0.04 0.217 -0.222 0.417 -0.041 -0.059 -0.071 -0.0
 $ t.KOvsWT               : num  0.01 -0.23 -0.78 -0.44 -5.09 1.25 -1.21 1.59 -0.16 -0.9 .
 $ t.HCAwt                : num  -2.23 0.02 2.8 1.88 5.77 1.67 -0.67 -0.26 -1.57 0.76 ...
 $ t.HCAko                : num  -1.77 0.59 1.11 2.99 -1.4 2.13 0.05 0.69 -1.15 -0.09 ...
 $ t.HCAwtnotHCAko        : num  -0.32 -0.4 1.19 -0.78 5.07 -0.32 -0.51 -0.67 -0.3 0.6 ...
 $ p.value.KOvsWT         : num  0.99407 0.82195 0.44522 0.66581 0.00003 ...
 $ p.value.HCAwt          : num  0.03495 0.98285 0.00976 0.07195 0.00001 ...
 $ p.value.HCAko          : num  0.08854 0.55758 0.27568 0.00619 0.17245 ...
 $ p.value.HCAwtnotHCAko  : num  0.74889 0.68895 0.24524 0.43983 0.00003 ...
 $ p.value.adj.KOvsWT     : num  0.99948 0.98415 0.90759 0.95854 0.00948 ...
```

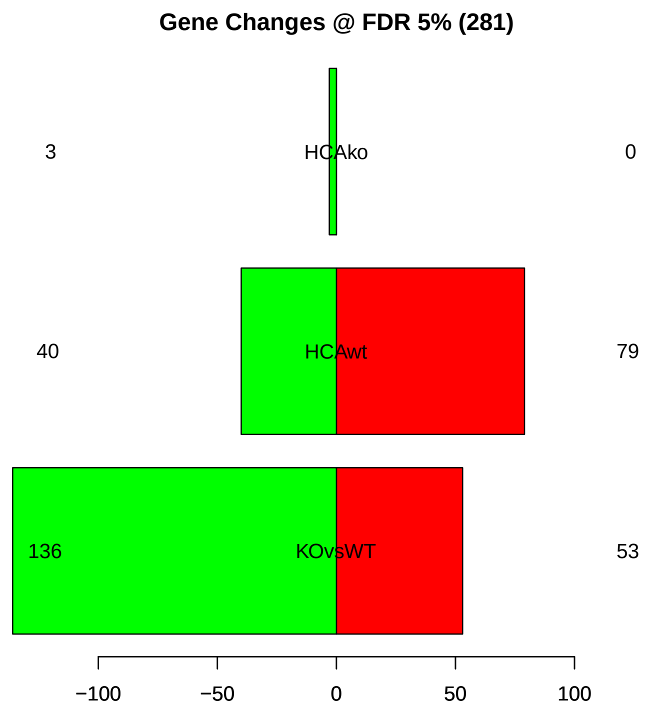

Figure 8: Number of differentially expressed genes at FDR 5%. Red bars: upregulated genes, green bars: downregulated genes.

```
 $ p.value.adj.HCAwt        : num  0.31045 0.99418 0.22671 0.38073 0.00414 ...
 $ p.value.adj.HCAko        : num  0.39 0.797 0.595 0.235 0.495 ...
 $ p.value.adj.HCAwtnotHCAko: num  0.9999 0.9999 0.9999 0.9999 0.0285 ...
 $ F                        : num  2.72 0.12 4.09 4.15 43.89 ...
 $ F.p.value                : num  0.0656 0.9487 0.0171 0.0161 0 ...
 $ Res.KOvsWT               : int  0 0 0 0 -1 0 0 0 0 0 ...
 $ Res.HCAwt                : int  0 0 0 0 1 0 0 0 0 0 ...
 $ Res.HCAko                : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Res.HCAwtnotHCAko        : int  0 0 0 0 1 0 0 0 0 0 ...
 $ ID                       : Factor w/ 24048 levels "18S_rRNA_X00686_523-S",..: 1 2 5 6 7 8
```

We can select only the differentially expressed genes at this threshold and save them in the fitObjDE object.

```
>   myNames<-names(fitObj)
> res.col<- which(regexpr("Res.",myNames)>0)
> DElist<- which(apply(fitObj[,res.col],1,function(x)any(x,na.rm=T)))
> length(DElist)

[1] 281

> fitObjDE <-fitObj[DElist,]
```

Finally, we need the annotation file for the array in order to annotate our results. This contains array-specific information, such as: array probe name, gene symbol, gene description, probe sequence etc. Array annotation files are usually available from the manufacturer's website.

```
> ill.array <- read.csv(file="Mouse_Ref-8_V1.csv", header=T)
> #excluding a few duplicates in the original illumina file:
> ill.array.unique <- ill.array[!duplicated(ill.array$Target),]
```

We saved the array info in the `ill.array` object. You can try the command `str(ill.array)` to check its structure.

We merge the list of differentially expressed genes with the illumina annotation file. The resulting object and be exported as a `csv` file.

```
> fitFinal <-merge(fitObjDE, ill.array.unique, by.x="ID",by.y="Target")
> colnames(fitFinal)[1]<-"Target"
> str(fitFinal,nchar.max=40,vec.len=2)

'data.frame':        280 obs. of  36 variables:
 $ Target                   : Factor w/ 24048 levels "18S_rRNA_X00686_523-S",..: 7 340 536 5
 $ A                        : num  10.82 7.73 ...
 $ Coef.KOvsWT              : num  -0.242 -0.458 0.108 0.192 -0.684 ...
 $ Coef.HCAwt               : num  0.335 0.902 0.34 0.522 -0.405 ...
 $ Coef.HCAko               : num  -0.082 0.072 0.024 0.302 -0.352 ...
 $ Coef.HCAwtnotHCAko       : num  0.417 0.83 0.316 0.22 -0.053 ...
 $ t.KOvsWT                 : num  -5.09 -4.75 1.92 1.94 -4.26 ...
 $ t.HCAwt                  : num  5.77 7.65 4.95 4.31 -2.06 ...
 $ t.HCAko                  : num  -1.4 0.61 0.35 2.49 -1.79 ...
 $ t.HCAwtnotHCAko          : num  5.07 4.98 3.26 1.29 -0.19 ...
 $ p.value.KOvsWT           : num  3e-05 7e-05 ...
 $ p.value.HCAwt            : num  1e-05 0e+00 ...
 $ p.value.HCAko            : num  0.172 0.548 ...
 $ p.value.HCAwtnotHCAko    : num  3e-05 4e-05 ...
 $ p.value.adj.KOvsWT       : num  0.00948 0.01761 ...
 $ p.value.adj.HCAwt        : num  0.004143 0.000078 ...
 $ p.value.adj.HCAko        : num  0.495 0.791 ...
 $ p.value.adj.HCAwtnotHCAko: num  0.0285 0.0349 ...
 $ F                        : num  43.9 48.7 ...
 $ F.p.value                : num  0 0 0.00056 0.00033 0.00008 ...
 $ Res.KOvsWT               : int  -1 -1 0 0 -1 ...
 $ Res.HCAwt                : int  1 1 1 1 0 ...
 $ Res.HCAko                : int  0 0 0 0 0 ...
 $ Res.HCAwtnotHCAko        : int  1 1 0 0 0 ...
 $ Search_key               : Factor w/ 23978 levels "18S_rRNA_X00686_523",..: 7 338 531 588
 $ ProbeId                  : int  6770524 110368 4670471 3060161 770594 ...
 $ Gid                      : Factor w/ 18595 levels "","GI_10048405",..: 4509 11458 11888 1
 $ Transcript               : Factor w/ 23978 levels "18S_rRNA_X00686_523",..: 7 338 531 588
 $ Accession                : Factor w/ 19076 levels "","AF102532.1",..: 1318 4381 2905 2380
 $ Symbol                   : Factor w/ 18197 levels "","0610005A07Rik",..: 11760 4516 4882 
 $ Type                     : Factor w/ 1 level "S": 1 1 1 1 1 ...
 $ Start                    : int  5385 277 4318 7389 278 ...
 $ Probe_Sequence           : Factor w/ 23981 levels "AAAAAAACACCTGGATCCCTGTCTTTCAATGGCTTCAA
 $ Definition               : Factor w/ 16583 levels "","beta-13-glucuronyltransferase 3 (gl
 $ Ontology                 : Factor w/ 7734 levels "","go_component: 26S proteasome [goid 0
 $ Synonym                  : Factor w/ 11544 levels "","0610005A07Rik;4930401L23Rik;4930505
```

```
> write.csv(fitFinal,file="final_GeneList.csv",quote=F)
```

5 Data Visualization

First, we need to compute the individual ratios

```
> all.samples<-as.data.frame(mydataQuantile)
> colnames(all.samples)<-paste(targets$Genotype,targets$Drug,sep=".")
> #1. Controls
> all.contr<-all.samples[,colnames(all.samples)=="wt.ctrl"]
> all.contrM<-rowMeans(all.contr)
> #2. Exp
> all.exp<-all.samples[,colnames(all.samples)=="ko.ctrl"]
> #3. Ratios
> all.coef<-all.exp-all.contrM
> colnames(all.coef)<- paste("ko.ctrl",1:8,sep=".")
> #this is to have the single ratios data
> mydata2<-cbind(rownames(all.coef),all.coef)
> colnames(mydata2)[1]<-"Target"
> #1. Controls HCA
> all.contr<-all.samples[,colnames(all.samples)=="wt.ctrl"]
> all.contrM<-rowMeans(all.contr)
> #2. Exp
> all.exp<-all.samples[,colnames(all.samples)=="wt.hca"]
> #3. Ratios
> all.coef<-all.exp-all.contrM
> colnames(all.coef)<- paste("wt.HCAvsCTRL",1:4,sep=".")
> #this is to have the single ratios data
> mydata2<-cbind(mydata2,all.coef)
> #1. KO HCA
> all.contr<-all.samples[,colnames(all.samples)=="ko.ctrl"]
> all.contrM<-rowMeans(all.contr)
> #2. Exp
> all.exp<-all.samples[,colnames(all.samples)=="ko.hca"]
> #3. Ratios
> all.coef<-all.exp-all.contrM
> colnames(all.coef)<- paste("ko.HCAvsCTRL",1:4,sep=".")
> #this is to have the single ratios data
> mydata2<-cbind(mydata2,all.coef)
> coef.toplot<-mydata2[DElist,2:17]
> colnames(coef.toplot)[9:12]<-"wtHCA"
> colnames(coef.toplot)[13:16]<-"koHCA"
> status<-as.character(c(rep(2,8),rep(3,4),rep(4,4)))

> heatmap.2(as.matrix(coef.toplot), col=rev(redgreen(52)),
 main= "FDR 5%",trace="none",cexCol=1 , cex.sub=1,
 cexRow=0.95, breaks=(c(-4,-3.5,-3,-2.5,-2,-1.5,seq(-1,1,0.05),1.5,2,2.5,3,3.5,4)),
 ColSideColors=status,sub="Color Coding: Group")


> KO<-paste("KO",summary(results)[1,1]+summary(results)[3,1],sep="-")
> HCAwt<-paste("HCAwt",summary(results)[1,2]+summary(results)[3,2],sep="-")
> HCAko<-paste("HCAko",summary(results)[1,3]+summary(results)[3,3],sep="-")
> vennDiagram(results[,1:3],names=c(KO,HCAwt,HCAko),main="Contrast analysis, FDR@5%")
```

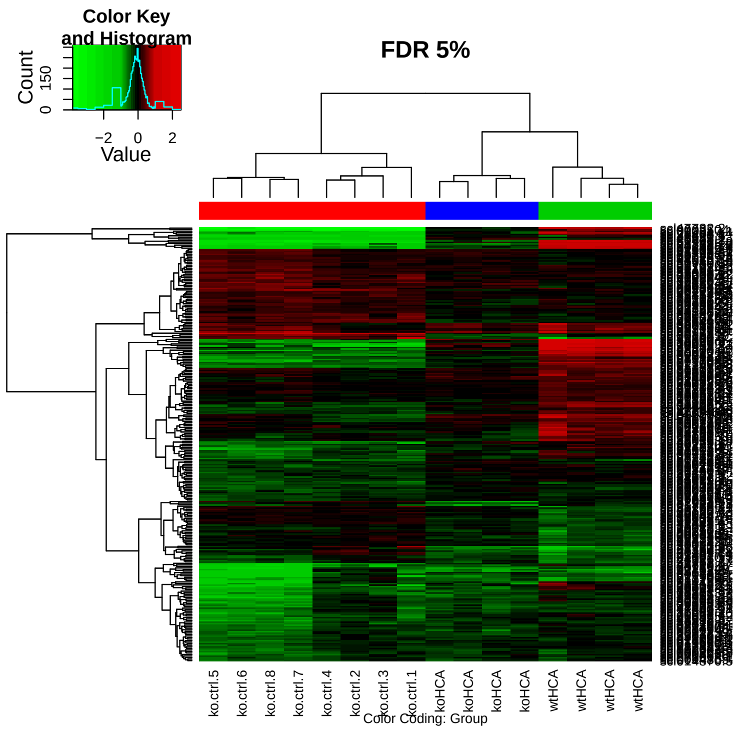

Figure 9: Heatmap of differentially expressed genes across all conditions. Genes are in rows and samples are in columns. Shades of red indicate upregulation compared to the control condition, shades of green indicate downregulation. Samples and genes are clustered by similarity (represented by the dendrogram).

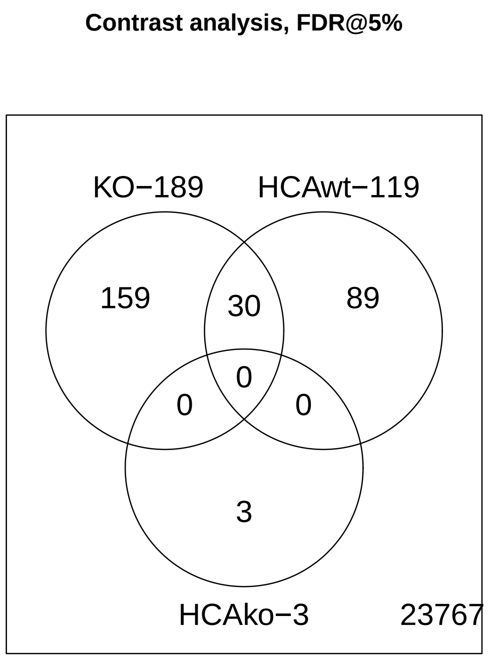

Figure 10: Venn Diagram