Analysis of Affymetrix Microarray Data with R and Bioconductor

Instructors: Jorge Andrade, Ph.D. & Chunling Zhang M.S.

Introduction:

In this tutorial we are going to show you how to use **R** and **BioConductor** packages to perform the analysis of Affymetrix microarray data. We will start by downloading a dataset from the **Gene Expression Omnibus (GEO)** a public functional genomics data repository supporting MIAME-compliant data submissions. We will show you how to normalize the data, perform quality control checks and how to identify differentially expressed genes.

The data:

- For this tutorial we are going to be working with the dataset GSE20986, generated by Dr. Andrew Browning. Human Umbilical Vein Endothelial Cells (HUVECs) are often used in research, in this experiment iris, retina and choroidal microvascular endothelial cells were isolated from donor eyes, and compared to HUVECs in order to establish whether the HUVEC line is a suitable surrogate for studying ocular disorders.
- This experiment was performed using Affymetrix Human Genome U133 Plus 2.0 Array, 12 arrays: iris, retina, choroid and human umbilical vein were included in triplicates.
- The raw data for this study is available at: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE20986

Install BioConductor packages:

We will start by installing BioConductor packages and load libraries for our analysis:

```
> # download the BioC installation routines
> source("http://bioconductor.org/biocLite.R")

> # install the core packages
> biocLite()

> # install the GEO libraries
> biocLite("GEOquery")
> library(GEOquery)
```

https://bioconductor.org/install/

R version 3.5 or later,

BiocManager:: install()


```
> biocLite("simpleaffy")
> library(simpleaffy)

> # load color libraries
> biocLite("RColorBrewer")
> library(RColorBrewer)

> biocLite("affyPLM")
> library(affyPLM)

> # install rgl package for PCA mapping
> biocLite("rgl")
> library(rgl)

> # install the Linear Models for Microarray Data (limma) package
> biocLite("limma")
> library(limma)

> # install annotation packages
> biocLite("hgu133plus2.db")
> library(hgu133plus2.db)
> library(annotate)
```

Now we will define our working directory:

First, using the OS, create the folder HUVEC under /Documents

```
> setwd("/Users/jorgeandrade/Documents/HUVEC")      ## set up your work space
> getwd()
[1] "/Users/jorgeandrade/Documents/HUVEC"
```

Getting the data:

We are now going to get the raw data directly from GEO, to do this we need to load the GEOquery library, and then provide the ID of the experiment we want to analyze. This download is approximately 53MB.

```
> #library(GEOquery)                    ## you may use local data directly
> getGEOSuppFiles("GSE20986")
```

If you open a file browser you will see this creates a GSE20986 directory under your working directory.

The GSE20986_RAW.tar file is a compressed archive of the CEL files (the Affymetrix native file format). Before we can work on them we need to unpack them:

```
> untar('GSE20986/GSE20986_RAW.tar', exdir='data')
```

This will create the directory data under your working directory.

NOTE: R on OS X 10.9 (Mavericks) seems to set a wrong TAR environment variable, you can fix this by executing:

```
> Sys.setenv(TAR = '/usr/bin/tar')
```

The CEL files subsequently need to be decompressed themselves to the directory in which we are working:

```
> cels <- list.files("data/", pattern = "[gz]")
> sapply(paste("data", cels, sep="/"), gunzip)
> cels
```

Describing the experiment:

We are now ready to start analysis, however in order to do this we need to capture the experimental information. This is just a text file that describes the chip names, and the source of the biological samples hybridized to them. We are now going to create that description file.

Open a new terminal window using your **operative system** (NOT INSIDE R) and type:

```
$ cd /Users/jorgeandrade/Documents/HUVEC/data
$ ls *.CEL > phenodata.txt
```

Open the newly created **`phenodata.txt`** file using Excel (**Important**: save it as 'tab-delimited text') and create 3 tab-delimited columns named 'Name', 'FileName' and 'Target'. The final result should look like:

```
Name FileName Target
GSM524662.CEL GSM524662.CEL iris
GSM524663.CEL GSM524663.CEL retina
GSM524664.CEL GSM524664.CEL retina
GSM524665.CEL GSM524665.CEL iris
GSM524666.CEL GSM524666.CEL retina
GSM524667.CEL GSM524667.CEL iris
GSM524668.CEL GSM524668.CEL choroid
GSM524669.CEL GSM524669.CEL choroid
GSM524670.CEL GSM524670.CEL choroid
GSM524671.CEL GSM524671.CEL huvec
GSM524672.CEL GSM524672.CEL huvec
GSM524673.CEL GSM524673.CEL huvec
```

You can also copy the text from this tutorial and paste it on: **`phenodata.txt`** file, and make sure that there are tabs between the columns and not spaces!

Loading and normalizing the data:

We are now going to load the data into an R object called 'celfiles', we can do that using the ***simpleaffy*** package, this package provides routines for handling CEL files including normalization and loading data with sample information.

```
> #library(simpleaffy)
> celfiles <- read.affy(covdesc="phenodata.txt", path="data")
```

You can confirm this has worked, and download the chip annotations at the same time (if this is your first run) by typing:

```
> celfiles
```

You should get the following output:

```
AffyBatch object
size of arrays=1164x1164 features (21 kb)
cdf=HG-U133_Plus_2 (54675 affyids)
number of samples=12
number of genes=54675
annotation=hgu133plus2
notes=
```

Now we are going to normalize the data using one of the available normalization algorithms (MAS5, RMA, GC-RMA and Li-Wong). For this tutorial we are going to use GeneChip RMA (GC-RMA), it is an improved version of RMA that is able to use sequence-specific probe affinities of the GeneChip probes to attain more accurate gene expression values (If this is the first time you have run it, it will download additional files during the process).

Robust Multi-array Average (RMA) —Irizarry et al., Biostatistics, 2003

Assumes all chips have same background, distribution of values

```
> celfiles.gcrma <- gcrma(celfiles)
Adjusting for optical effect.....Done.
Computing affinities.Done.
Adjusting for non-specific binding.....Done.
Normalizing
Calculating Expression
```

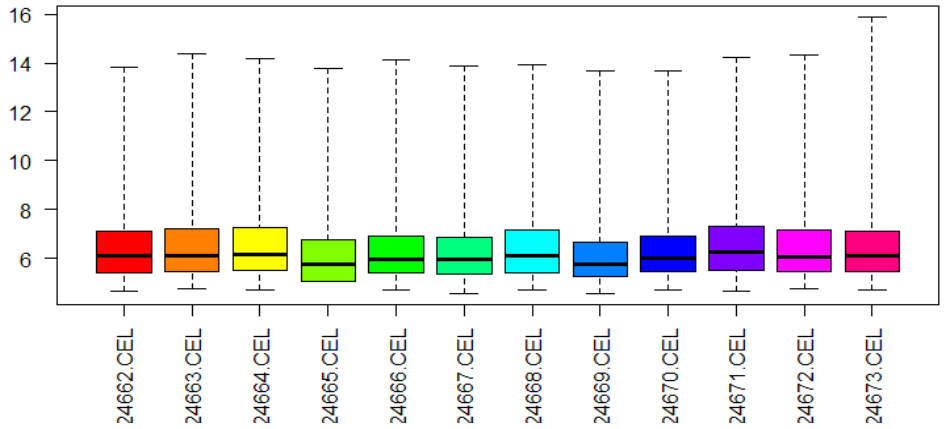
To see the data associated with the normalized object use:

```
> celfiles.gcrma
ExpressionSet (storageMode: lockedEnvironment)
assayData: 54675 features, 12 samples
  element names: exprs
protocolData
  sampleNames: GSM524662.CEL GSM524663.CEL ... GSM524673.CEL (12 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: GSM524662.CEL GSM524663.CEL ... GSM524673.CEL (12 total)
  varLabels: sample FileName Target
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu133plus2
```

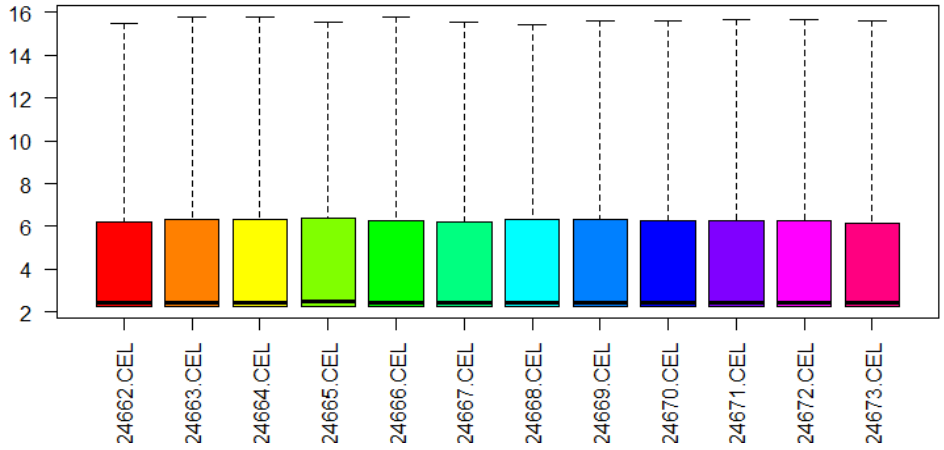
Quality control checks:

To ensure that there are no quality issues with the dataset, we will now perform some quality control checks. We will start by checking the effects of the normalization, by plotting a boxplot of probe intensities *before* and *after* normalization.

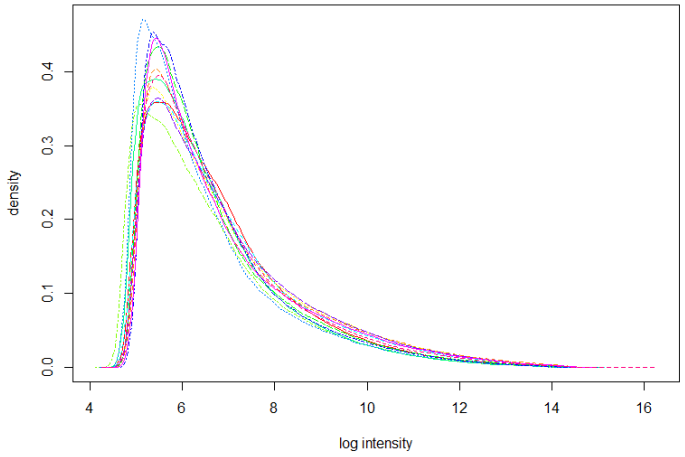
```
> # library(RColorBrewer)
> # plot a boxplot of unnormalized intensity values
> cols <- rainbow(12)
> boxplot(celfiles, col=cols)
```



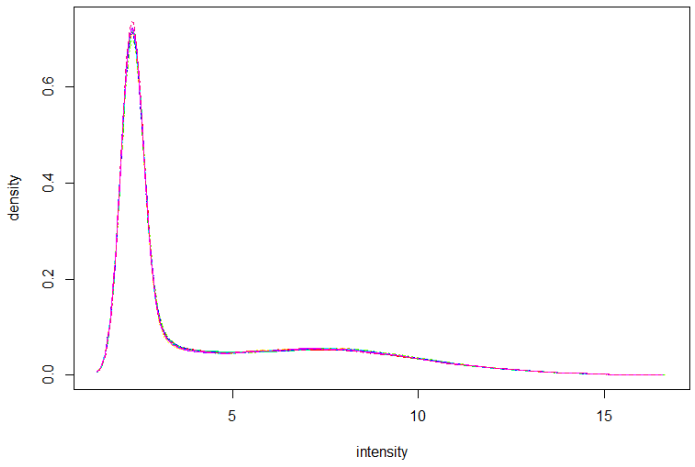

```
> # plot a boxplot of normalized intensity values
> # library(affyPLM)
> boxplot(celfiles.gcrma, col=cols)
```



```
> # Plot a density vs. log intensity histogram for the unnormalized
data
> hist(celfiles, col=cols)
```



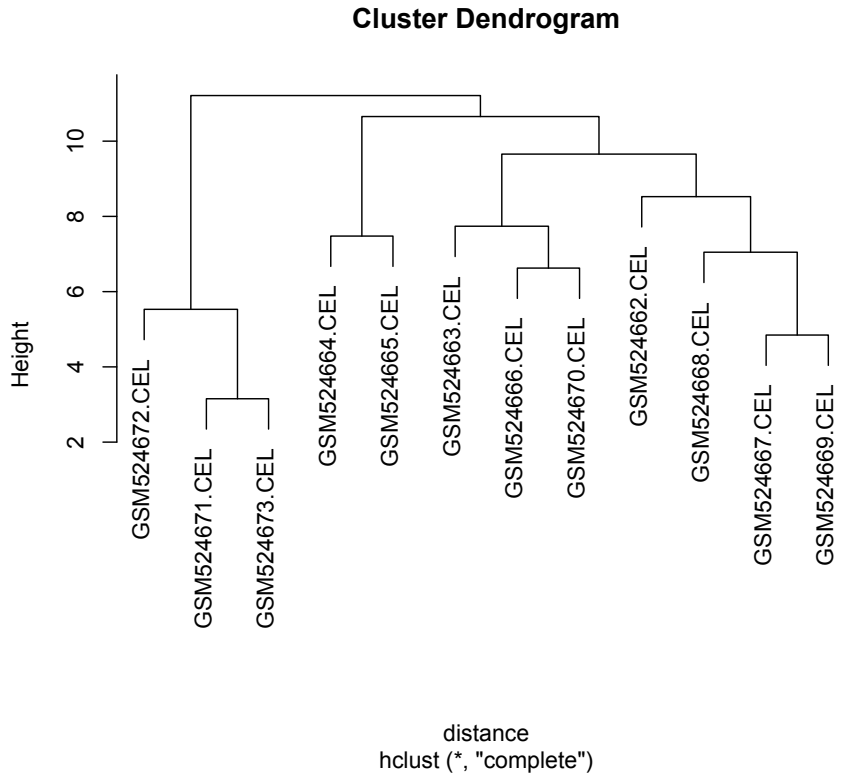

```
> # Plot a density vs log intensity histogram for the normalised data
> hist(celfiles.gcrma, col=cols)
```



From these plots we can conclude that there are no major deviations amongst the 12 chips, and normalization has brought the intensities from all of the chips into distributions with similar characteristics.

We can also look at the relationships between the samples using hierarchical clustering:

```
> eset <- exprs(celfiles.gcrma)
> distance <- dist(t(eset),method="maximum")
> clusters <- hclust(distance)
> plot(clusters)
```



The cluster shows that HUVEC samples (GSM524671.CEL, GSM524672.CEL, and GSM524673.CEL) are a separate group compared to the eye tissues, which show some evidence of clustering by tissue type.

Filtering data:

We will now filter out uninformative data such as control probesets and other internal controls as well as removing genes with low variance, which will be unlikely to pass statistical tests for differential expression, or are expressed uniformly close to background detection levels.

```
> celfiles.filtered <- nsFilter(celfiles.gcrma, require.entrez=FALSE,
remove.dupEntrez=FALSE)
> # What got removed and why?
> celfiles.filtered$filter.log
$numLowVar
[1] 27307

$feature.exclude
[1] 62

> filterEset <- exprs(celfiles.filtered$eset)
> dim(filterEset)
[1] 27306    12
```

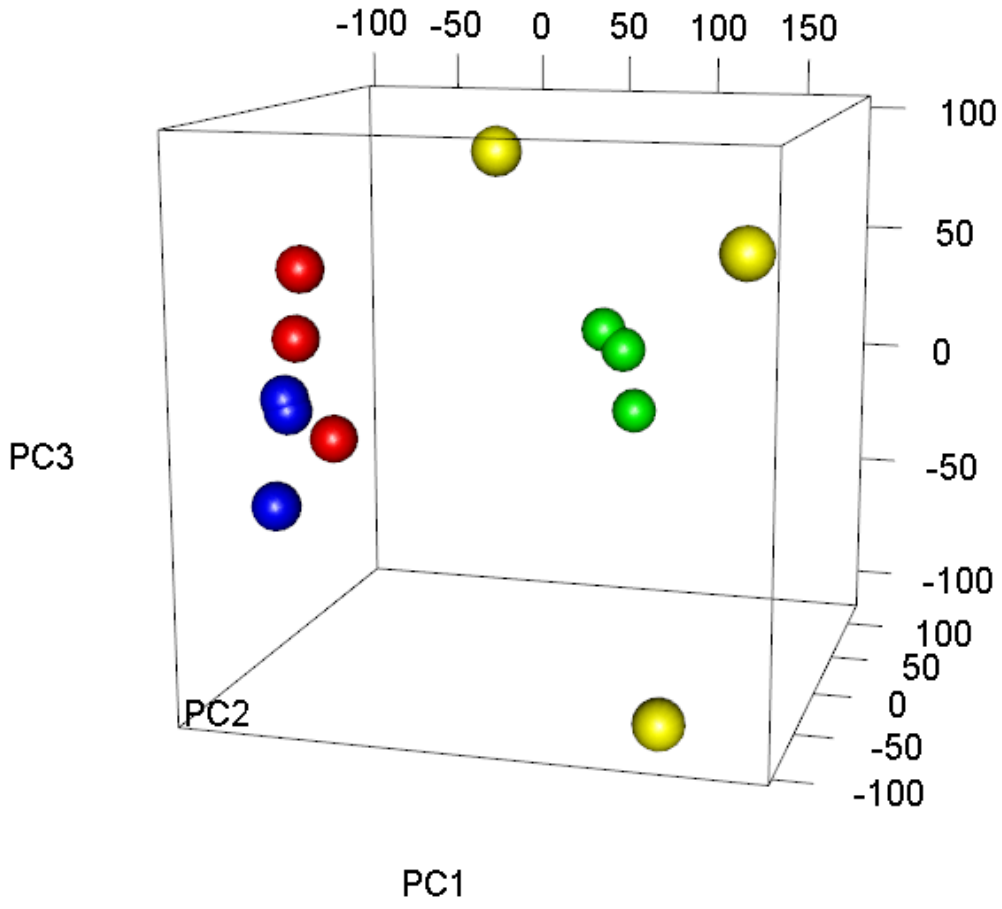
Exploratory analysis by PCA mapping:

Principal component analysis (PCA) is a dimension reduction and data visualization method. The axes in the PCA map represent 3 major components identified. It shows us the overall gene expression patterns. Here we use PCA to visualize samples relationship and to detect outliers.

```
> # library(rgl)
> pca <- prcomp(t(filterEset), scale=TRUE)
> summary(pca)
Importance of components:
                          PC1       PC2        PC3        PC4        PC5
PC6      PC7      PC8      PC9
Standard deviation         90.9472  72.6076  51.51000  49.30948  44.44554
38.95288 37.38655 35.38313 32.71420
Proportion of Variance    0.3029   0.1931   0.09717   0.08904   0.07234
0.05557  0.05119  0.04585  0.03919
Cumulative Proportion     0.3029   0.4960   0.59315   0.68219   0.75454
0.81010  0.86129  0.90714  0.94634
                          PC10     PC11      PC12
Standard deviation     29.53034 24.35828 2.516e-13
Proportion of Variance  0.03194  0.02173 0.000e+00
Cumulative Proportion   0.97827  1.00000 1.000e+00
```

```
> # choroid is red, huvec is green, iris is blue, retina is yellow
> myColors <- c("Blue", "yellow", "yellow", "Blue", "yellow", "Blue",
              rep("Red", 3), rep("Green", 3))

> plot3d(pca$x[, 1:3], col=myColors, xlab="PC1", ylab = "PC2", zlab =
"PC3", type = "s")
```

For OSX, maybe you need install XQuartz first, https://www.xquartz.org/




```
> # Save the above image into a file
> rgl.postscript("PCA.pdf", fmt="pdf", drawText=TRUE)
```

Finding differentially expressed probesets:

We will now use the Linear Models for Microarray Data (***limma***) package for the differential gene expression analysis. First of all we need to extract information about the samples:

```
> samples <- celfiles.gcrma$Target
> samples
 [1] "iris"    "retina"  "retina"  "iris"    "retina"  "iris"
"choroid" "choroid" "choroid" "huvec"
[11] "huvec"   "huvec"

> # convert into factors
> samples <- as.factor(samples)
> samples
 [1] iris    retina  retina  iris    retina  iris    choroid choroid
choroid huvec   huvec   huvec
Levels: choroid huvec iris retina

> # set up the experimental design
> design <- model.matrix(~0 + samples)
> colnames(design) <- c("choroid", "huvec", "iris", "retina")
> # inspect the experiment design
> design
   choroid huvec iris retina
1         0     0    1      0
2         0     0    0      1
3         0     0    0      1
4         0     0    1      0
5         0     0    0      1
6         0     0    1      0
7         1     0    0      0
8         1     0    0      0
9         1     0    0      0
10        0     1    0      0
11        0     1    0      0
12        0     1    0      0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$samples
[1] "contr.treatment"
```

At this point we have normalized, filtered the data, and added a description of the data and experimental design. This will be consumed by the ***limma*** packages for the DEG analysis.

```
> # library(limma)
```


```
> # fit the linear model to the filtered expression set
> fit <- lmFit(filterEset, design)
> contrast.matrix <- makeContrasts(huvec_choroid = huvec - choroid,
huvec_retina = huvec - retina, huvec_iris <- huvec - iris,
levels=design)

> # Now the contrast matrix is combined with the per-probeset linear model fit
> huvec_fits <- contrasts.fit(fit, contrast.matrix)
```

Now we are going to calculate the differential expression by empirical Bayes shrinkage of the standard errors towards a common value, by computing the moderated t-statistics, moderated F-statistic, and log-odds:

```
> huvec_ebFit <- eBayes(huvec_fits)

> # return the top 10 results for any given contrast
> # coef=1 is huvec_choroid, coef=2 is huvec_retina, coef=3 is huvec_iris

> topTable(huvec_ebFit, number=10, coef=1)
               logFC  AveExpr         t      P.Value    adj.P.Val        B
204779_s_at  7.367790 4.171707  72.77347 3.284937e-15 8.969850e-11 20.25762
207016_s_at  6.936667 4.027733  57.39252 3.694641e-14 5.044293e-10 19.44987
209631_s_at  5.192949 4.003992  51.24892 1.170273e-13 1.065182e-09 18.96660
242809_at    6.433238 4.168870  48.51842 2.043082e-13 1.394710e-09 18.70852
205893_at    4.480331 3.543714  40.56477 1.261400e-12 6.888757e-09 17.75050
227377_at    3.672710 3.209739  36.08942 4.132286e-12 1.880603e-08 17.03075
204882_at   -5.353547 6.513690 -34.70646 6.141288e-12 2.395629e-08 16.77396
38149_at    -5.054267 6.483784 -31.45817 1.661577e-11 5.282891e-08 16.09358
205576_at    6.586393 4.139624  31.31294 1.741230e-11 5.282891e-08 16.06036
205453_at    3.624587 3.210563  30.74481 2.095602e-11 5.722251e-08 15.92787
```

You can now create a top ***n*** table with the differentially expressed probes for any contrast, bellow the top 20 for huvec vs. iris contrast:

```
> topTable(huvec_ebFit, number=20, coef=3)
               logFC   AveExpr         t      P.Value    adj.P.Val        B
204779_s_at  7.364539  4.171707  72.74137 3.299754e-15 9.010308e-11 20.42224
207016_s_at  6.922940  4.027733  57.27894 3.769954e-14 5.147118e-10 19.58180
209631_s_at  5.192003  4.003992  51.23958 1.172446e-13 1.067161e-09 19.09205
242809_at    6.476779  4.168870  48.84679 1.907556e-13 1.302193e-09 18.86149
205893_at    4.336396  3.543714  39.26159 1.757424e-12 9.597642e-09 17.64938
204882_at   -5.724308  6.513690 -37.11007 3.114005e-12 1.417184e-08 17.29454
227377_at    3.672521  3.209739  36.08757 4.134444e-12 1.612787e-08 17.11237
38149_at    -5.410650  6.483784 -33.67632 8.334440e-12 2.844753e-08 16.64406
209990_s_at  6.659281  4.228214  31.34054 1.725776e-11 5.236005e-08 16.13175
205453_at    3.621710  3.210563  30.72041 2.112502e-11 5.768398e-08 15.98486
205576_at    6.286535  4.139624  29.88736 2.789557e-11 6.924694e-08 15.77972
211828_s_at  4.579372  3.714309  29.45904 3.227764e-11 7.344778e-08 15.67061
228109_at   -4.919429  7.359528 -28.81802 4.031276e-11 8.467539e-08 15.50249
207526_s_at  8.184859  5.342672  27.84578 5.700960e-11 1.040221e-07 15.23594
224650_at    3.907701  3.257685  27.83936 5.714247e-11 1.040221e-07 15.23413
205975_s_at  3.390947  3.139171  27.58014 6.280038e-11 1.071767e-07 15.16058
243154_at    4.081803  3.476199  27.36222 6.803396e-11 1.092785e-07 15.09791
223514_at    3.764746  3.285218  25.39924 1.440821e-10 2.185725e-07 14.49718
205366_s_at  3.409272  3.211259  25.08436 1.633632e-10 2.347788e-07 14.39436
```


```
203373_at   -4.242839 10.441247 -24.74428 1.874195e-10 2.558838e-07 14.28116
```

To impose a fold change cutoff, and see how many genes are returned you can use the ***lfc*** modifier for topTable, here we show the results for absolute value of fold changes > 5 using the first contrast huvec vs. choroid

```
> topTable(huvec_ebFit, coef=1, number=10000, lfc=5)
```

The table will show 88 probes, you can count them:

```
> nrow(topTable(huvec_ebFit, coef=1, number=10000, lfc=5))
[1] 88
```

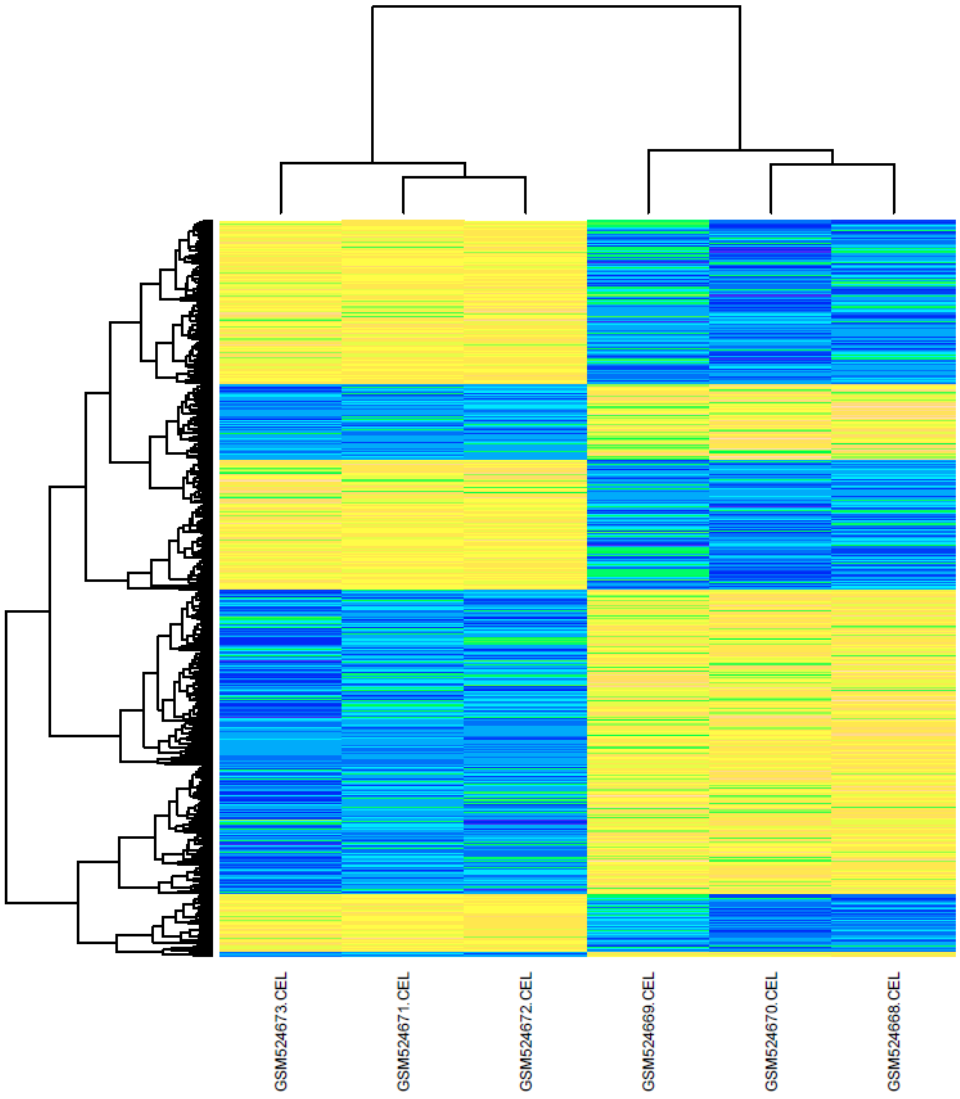
You can now create a list with probes filtered by fold change > ***n*** and p-value < ***m***, as follow:

```
> probeset.list <- topTable(huvec_ebFit, coef=1, p.value=0.05, lfc=5)
> probeset.list
               logFC  AveExpr         t      P.Value    adj.P.Val        B
204779_s_at  7.367790 4.171707  72.77347 3.284937e-15 8.969850e-11 20.25762
207016_s_at  6.936667 4.027733  57.39252 3.694641e-14 5.044293e-10 19.44987
209631_s_at  5.192949 4.003992  51.24892 1.170273e-13 1.065182e-09 18.96660
242809_at    6.433238 4.168870  48.51842 2.043082e-13 1.394710e-09 18.70852
204882_at   -5.353547 6.513690 -34.70646 6.141288e-12 2.395629e-08 16.77396
38149_at    -5.054267 6.483784 -31.45817 1.661577e-11 5.282891e-08 16.09358
205576_at    6.586393 4.139624  31.31294 1.741230e-11 5.282891e-08 16.06036
209990_s_at  6.448368 4.228214  30.34792 2.389911e-11 5.932628e-08 15.83288
207526_s_at  7.838572 5.342672  26.66767 8.817608e-11 1.416315e-07 14.84551
231780_at    5.261775 5.405038  26.05644 1.113924e-10 1.684530e-07 14.66070
```

Then we will create list of Differentially Expressed probes with adj.P.Val ≤ 0.05 and fold change ≥ 2 for the first contrast huvec vs. choroid, then we will make a heat map of the expression.

```
> selSamples <- subset(celfiles.gcrma$FileName, (celfiles.gcrma$Target
== "choroid") | (celfiles.gcrma$Target == "huvec"))
> selSamples
[1] "GSM524668.CEL" "GSM524669.CEL" "GSM524670.CEL" "GSM524671.CEL"
"GSM524672.CEL" "GSM524673.CEL"


# all probe data without filtering
> probeset.list1 <- topTable(huvec_ebFit, coef=1,
number=nrow(filterEset), lfc=0)


> #probe list filtered by adj.P.Val ≤ 0.05 and absolute fold change ≥ 2
```

```
> probeset.list2 <- probeset.list1[(probeset.list1$adj.P.Val <= 0.05) &
(abs(probeset.list1$logFC) >= 1), ]
> dim(probeset.list2)
[1] 2768    6

> # extract intensity data for this probe list
> selData <- filterEset[rownames(filterEset) %in%
rownames(probeset.list2), colnames(filterEset) %in% selSamples]


> # generate a heat map for the probe list and save as a pdf file
> pdf(file="Heatmap.pdf", width=8, height=6)
> heatmap(selData, labRow=c(""), col=topo.colors(16), cexCol=0.6)
> graphics.off()
```



Annotating the results with associated gene symbols:

In order to annotate the probesets into gene symbols we need to install and load the associated database package and the annotate package, then we can extract the probeset ID's from the topTable results, and match the symbols.

```
> # library(hgu133plus2.db)
> # library(annotate)
> gene.symbols <- getSYMBOL(rownames(probeset.list), "hgu133plus2")
> results <- cbind(probeset.list, gene.symbols)
> write.table(results, "results.txt", sep="\t", quote=FALSE)
```

Finally we are going to save our R workspace to a .RData file. If you don't specify a path, the current working directory is assumed:

```
> save.image()
```

Print version information about R, the OS and attached or loaded packages.

```
> sessionInfo()
```