Homework 2 Solutions
BIOS 7659/CPBS 7659

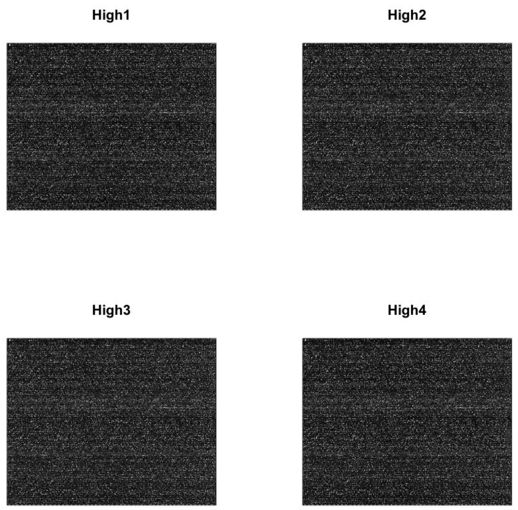
1) Quality Control

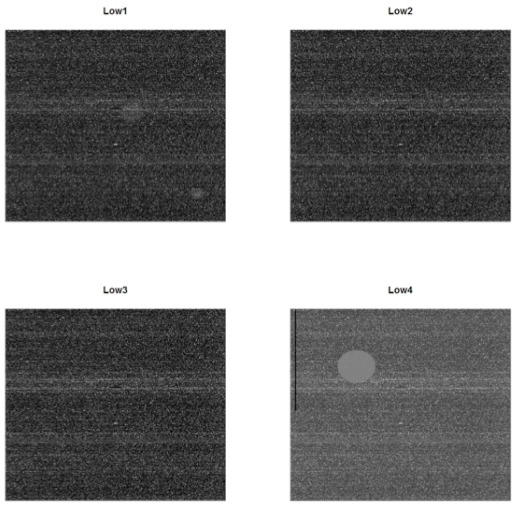
Microarrays provide high-throughput measurements of gene expression simultaneously for thousand of genes. It is important to assess the quality of the microarray expression data before moving forward to data processing and statistical analyses. Several quality control metrics and visualizations are provided in the affy and simleaffy R packages, which we will explore below on an example data set of human cell lines subject to two treatments ("low" or "high"), with n=4 per treatment.

a) Practice with Affy package

```
Code:
library(affy)
library(simpleaffy)
##Read in data
pd <- read.AnnotatedDataFrame("targets.txt",header=TRUE,row.names=1,as.is=TRUE)
Data <- ReadAffy(filenames=pData(pd)$FileName,phenoData=pd, sampleNames=
       sampleNames(pd))

## Practice extracting information from Data and the sample file "pd"
sampleNames(Data)
probeNames(Data)
mm(Data)
pm(Data)
pData(pd)
exprs(Data)
```

b) If the number of arrays is small, visualizing the microarray images is an important first step to assess data quality and to identify any technical artefacts. The images for the "High" samples generally look similar (data not shown). However, among the "Low" samples, the array for "Low4" has several defects including a larger circular artifact on the top left quadrant, a rectangular strip on the top left side and an overall lighter image. The chip for "Low1" appears to have some smudges as well but less pronounced than "Low4". We will keep an eye on "Low4" to see how it performs in other QC diagnostics.

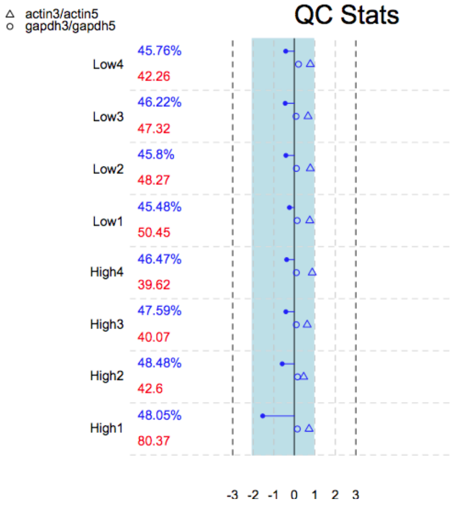




```
Code:
par(mfrow = c(2,2))
for(i in 1:8) {image(Data[,i])}
```

c) The quality control metrics from the `simpleaffy` package are plotted below for each array. On the left, the two numbers for each array are the percent of probes sets called present (top) and the average background intensity level (bottom). Probesets are considered present if the intensity level of perfect match probes is generally higher than the mismatch probes. The average background level is usually between 20-100. The blue line with a dot on the right hand side represents the scale factor, which is returned by Affymetrix MAS 5.0 and is used to equalize their mean intensities across the arrays. The scale factors should be within the blue region. The open circle and triangle represent the GAPDH ratio and beta actin control gene expression levels. These values should not be too high and within the blue region.

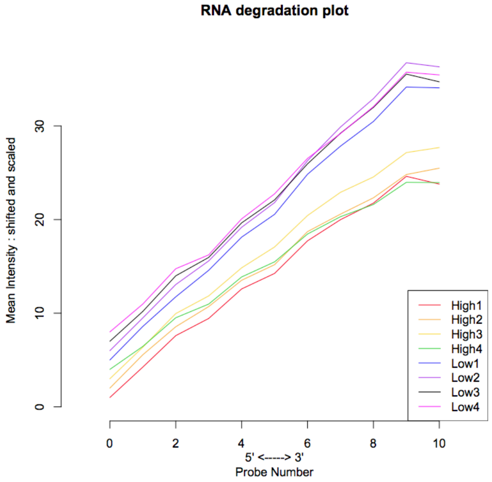
In general for all QC metrics, the values should be consistent across arrays, otherwise an array with very different values should be flagged. The QC plot for these arrays do not indicate any quality control issues, except that for "High1", the background level is high and the scale factor is relatively low compared to the other ararys. However, both values are within the standard ranges.



Code:

```
qcstats = qc.affy(Data)
plot.qc.stats(qcstats)
```

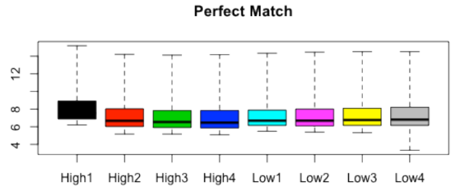
d) The RNA degradation plot shows the average intensity across probes within a probeset ordered from the 5' end to the 3' end. Intensity is expected to increase since RNA degradation starts from the 5' end of the molecule. The RNA degradation plot for the 8 arrays show a fairly consistent slope and an increase in intensity across all arrays, which is expected. However, arrays from the "High" group tend to have lower values than the "Low" group, which could either reflect biological differences or systematic technical differences.

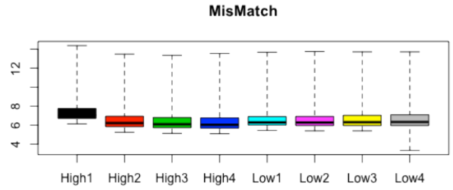


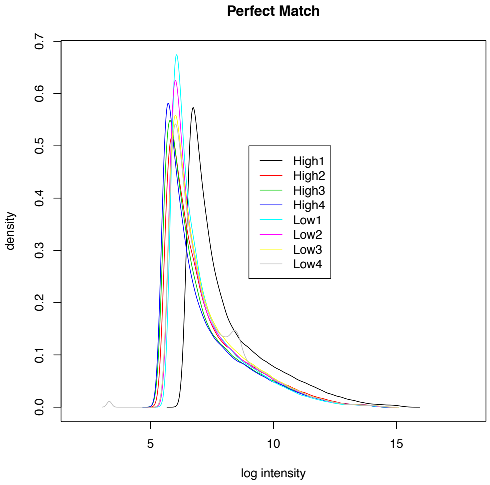
Code:

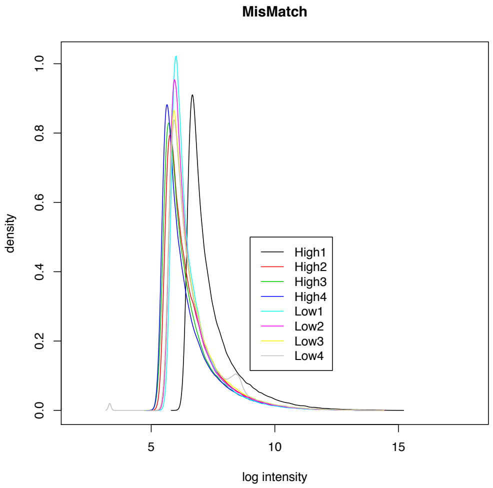
```
ard = AffyRNAdeg(Data)
plotAffyRNAdeg(ard, col = 1:8)
legend(8,15, legend= ard$sample.names, col = 1:8, lty = 1)
```

e) The boxplots and density plots for the perfect match (PM) and mismatch (MM) probes highlight that the “High1” array tends to have higher intensity values than the other arrays and that the “Low4” array has a larger spread of values and two additional “bumps” in the density below log 5 intensity and around 9 log intensity, which may be reflecting the artifacts observed in the array image plot (part b). Intensity values for PM are higher and more variable than MM, which is not unexpected since MM should be capturing background and nonspecific hybridization.









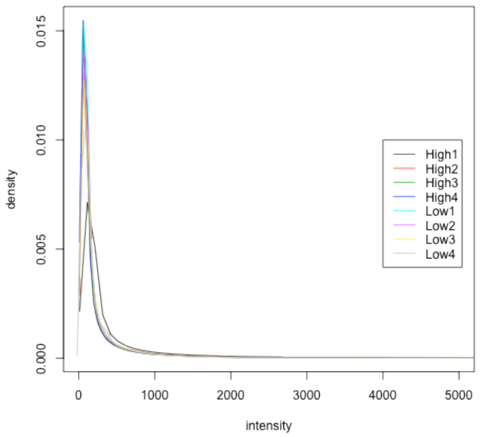
```
Code:
par(mfrow = c(1,2))
boxplot(Data, col = 1:8, which = "pm", main = "Perfect Match")
boxplot(Data, col = 1:8, which = "mm", main = "MisMatch")


plotDensity.AffyBatch(Data, col = 1:8, lty =1, which = "pm", xlim = c(2,18),
       main = "Perfect Match", ylim= c(0,1))
legend(9,.5, legend= ard$sample.names, col = 1:8, lty = 1)
plotDensity.AffyBatch(Data, col = 1:8, lty =1, which = "mm", xlim = c(2,18),
       main = "MisMatch")
legend(9,.5, legend= ard$sample.names, col = 1:8, lty = 1)
```

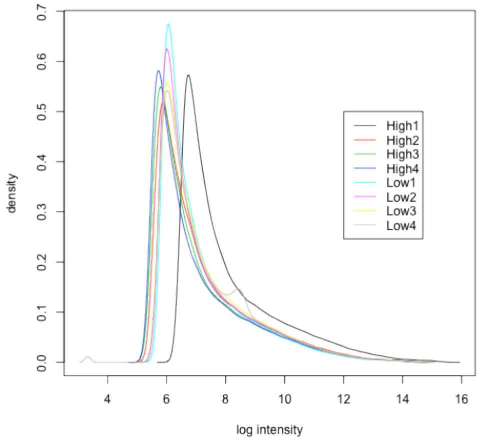
f) The different quality control metrics and plots indicate that “Low4” may have serious artefacts that affect the quality of the data and would be a candidate array to be removed from the analysis. There is also evidence that “High1” also has some technical variation that is different than the other arrays. These concerns do not warrant removal of the arrays at this point. We will wait to see if normalization addresses these concerns.

2) Normalization

In this report, we will compare options for pre-processing steps in a microarray data analysis (normalization, probeset summarization and PM correction methods). *Normalization* for microarray expression data is critical for removing some sources of variation so that they we can compare expression levels across arrays. For the Affymetrix array, *summarization* refers to how Perfect Match (PM) probes are summarized into one probeset expression value. *PM correction* refers to how PM probe values are adjusted by using the corresponding Mis Match (MM) values. We will also derive the Present/Absent calls for each probeset and filter the data based on these calls.

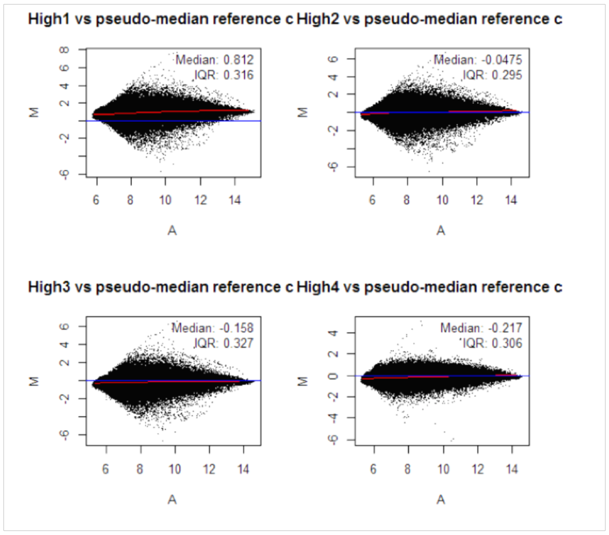
a) The density plots before log transformation show that the data are extremely skewed. After log transformation, the data look more normal although there is still a relatively long right tail.

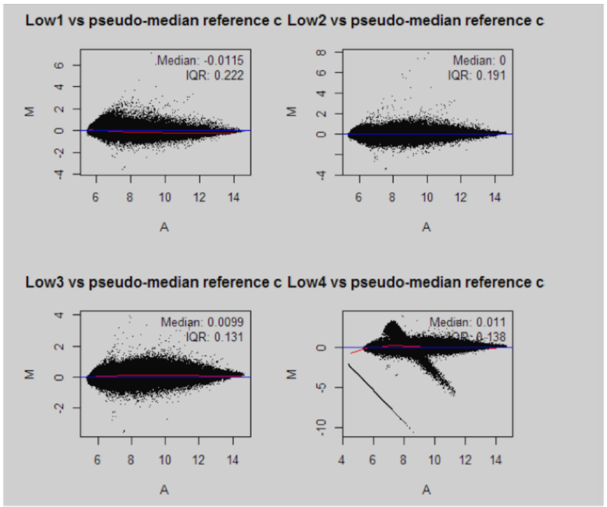




```
Code:
plotDensity.AffyBatch(Data, col = 1:8, lty =1, log = FALSE)
legend(20000,.01, legend= sampleNames(Data), col = 1:8, lty = 1)
plotDensity.AffyBatch(Data, col = 1:8, lty =1, log = TRUE)
legend(9,.5, legend= sampleNames(Data), col = 1:8, lty = 1)
```

b) The MA plots display the log ratio intensity for each chip compared to a (pseudo-median) reference chip against the average log ratio intensity for each chip compared to the reference chip. The points should be centered around the horizontal line around zero. The plots indicate that “High1” may have shifted values as the MA plot is not centered around 0, and that “Low4” has odd patterns, probably reflecting the artifacts identified in the QC plots.

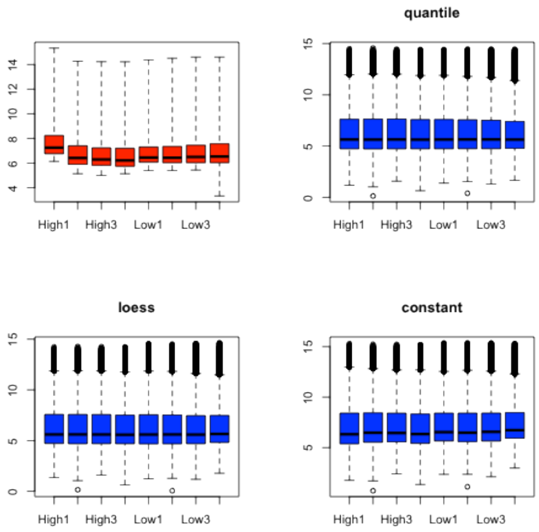




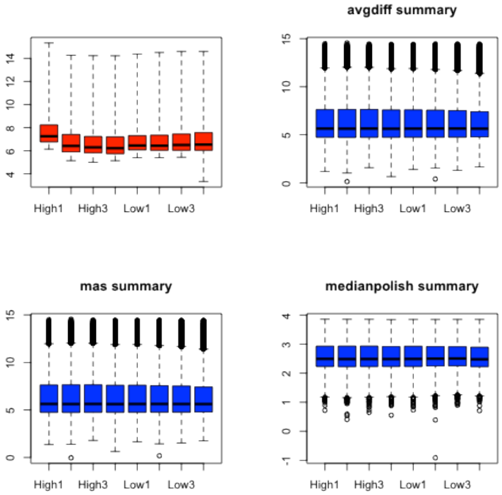
Code:

```
MAplot(Data)
```

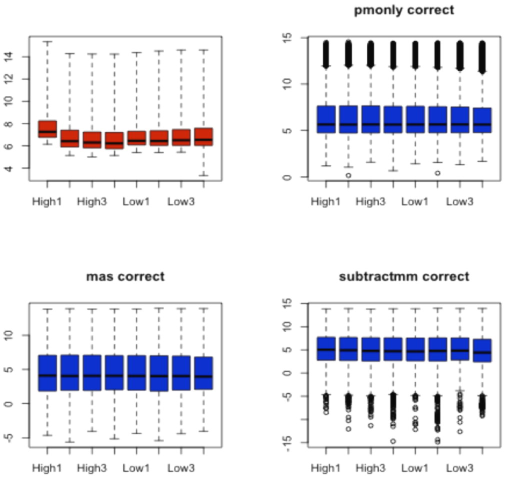
c) Three normalization methods were applied setting the summary method to “avgdiff” and the PMcorrect method to “pmonly”: 1) Quantile normalization, which gives each array the same distribution by requiring all quantiles to be the same 2) LOESS normalization, which is based on applying a loess curve to the MA plot and 3) Constant normalization, which rescales each array to have the same mean. Compared to the un-normalized data, the boxplots below indicate that all methods make the distributions more consistent across arrays, but the constant normalization method shows more variation than the other two.



Three summarization methods were applied setting the normalization method to “quantile” and the PMcorrect method to “pmonly”: 1) avgdiff, which computes the average difference between the PM and MM values within a probeset 2) MAS, which uses a robust average (Tukey biweight) and 3) median polish, which uses an iterative approach for taking the medians across rows and columns. The boxplots below indicate that all methods make the distributions more consistent across arrays, however median polish changes the scale and introduces negative values, which may not be desirable. The avgdiff and mas summary are the most similar.



Three PM correction methods were applied setting the normalization method to “quantile” and the summary method to “avgdiff”: 1) pmonly, which makes no adjustment 2) subtractmm, which subtracts MM from the corresponding PM and 3) MAS, which only subtracts MM when possible (it is less than PM). The boxplots below indicate that all methods make the distributions more consistent across arrays, however the subtractmm option introduces more variability. The MAS method and subtractmm option also introduce negative values.



Code:

```
par(mfrow = c(2,2))
boxplot(Data, col = "red") #original data
eset.q = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "pmonly", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.q))), col = "blue", main = "quantile")
eset.l = expresso(Data, normalize.method = "loess", bgcorrect.method = "mas",
       pmcorrect.method = "pmonly", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.l))), col = "blue", main = "loess")
eset.c = expresso(Data, normalize.method = "constant", bgcorrect.method =
       "mas", pmcorrect.method = "pmonly", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.c))), col = "blue", main = "constant")

par(mfrow = c(2,2))
boxplot(Data, col = "red") #original data
#eset.q = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "pmonly", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.q))), col = "blue", main = "avgdiff
       summary")
eset.qm = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "pmonly", summary.method = "mas")
boxplot(data.frame(log2(exprs(eset.qm))), col = "blue", main = "mas summary")
eset.qp = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "pmonly", summary.method = "medianpolish")
boxplot(data.frame(log2(exprs(eset.qp))), col = "blue", main = "medianpolish
       summary")

par(mfrow = c(2,2))
boxplot(Data, col = "red") #original data
#eset.q = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "pmonly", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.q))), col = "blue", main = "pmonly correct")
eset.qmas = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "mas", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.qmas))), col = "blue", main = "mas correct")
eset.qs = expresso(Data, normalize.method = "quantiles", bgcorrect.method =
       "mas", pmcorrect.method = "subtractmm", summary.method = "avgdiff")
boxplot(data.frame(log2(exprs(eset.qs))), col = "blue", main = "subtractmm
       correct")
```

d) The Affymetrix MAS 5.0 software provides detection calls for each probeset. The calls are based on a non-parametric statistical test (Wilcoxon signed rank test) for each probeset that tests whether the perfect match probes show more hybridization signal than their corresponding mismatch probes. After applying this method, we found that across the two groups, 25,726 probesets have at least one present call in each group.

Code:

```
calls = mas5calls(Data); callsdata = exprs(calls)
keephigh = apply(callsdata[,1:4], 1, function(x) sum(x == "P")>0)
keeplow = apply(callsdata[,5:8], 1, function(x) sum(x == "P")>0)
keepresent = keephigh & keeplow
sum(keepresent)
```

e) Based on the QC analysis in problem #4 and the MA plot, the array for Low 4 appears to be problematic and will be removed from subsequent analyses. The data will be renormalized without that array.

Code:

```
#Remove “Low4” array & normalize
cData = Data[,1:7]
eset = rma(cData)
calls = mas5calls(Data);
callsdata = exprs(calls)

#Filtering based on part d)
keephigh = apply(callsdata[,1:4], 1, function(x) sum(x == "P")>0)
keeplow = apply(callsdata[,5:8], 1, function(x) sum(x == "P")>0)
keepresent = keephigh & keeplow

#Output data
final = exprs(eset)[keepresent,]
write.table(final, file="arraydata.txt", quote =F, row.names = T, col.names=T)
write.table(callsdata[keepresent,], file="callsdata.txt", quote =F,
       row.names = T, col.names = T)
```

```
              High1    High2    High3    High4      Low1      Low2      Low3
1007_s_at 10.064231 9.921267 9.951345 9.755027 11.466151 11.267235 11.337803
1053_at    9.309349 9.330554 9.298448 9.779300  9.175758  9.146989  9.203968
117_at     6.599882 6.555252 6.415823 5.758104  5.838012  5.909986  5.724287
121_at     7.300462 7.183799 7.220009 7.321080  7.271990  7.223007  7.402417
1294_at    5.517051 5.442065 5.662856 5.422198  5.517051  5.573565  5.517051
1316_at    4.167586 4.159955 4.258703 4.167586  4.080589  4.239694  4.270392
```