BIOS 7659 Homework 2

Tim Vigers

27 September 2020

# Quality Control

## a) Read the 8 CEL files

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

kable(head(exprs(Data)),digits = 3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| High1 | High2 | High3 | High4 | Low1 | Low2 | Low3 | Low4 |
| 197 | 81 | 106 | 69 | 174 | 103 | 100 | 100 |
| 13571 | 9974 | 9267 | 6591 | 7231 | 4831 | 4724 | 4724 |
| 248 | 137 | 100 | 94 | 194 | 147 | 137 | 137 |
| 13810 | 9895 | 9550 | 6866 | 7407 | 4886 | 4919 | 4919 |
| 123 | 63 | 60 | 76 | 77 | 79 | 71 | 71 |
| 173 | 73 | 61 | 68 | 116 | 81 | 103 | 103 |

sampleNames(Data)

## [1] "High1" "High2" "High3" "High4" "Low1" "Low2" "Low3" "Low4"

head(probeNames(Data))

## [1] "1007\_s\_at" "1007\_s\_at" "1007\_s\_at" "1007\_s\_at" "1007\_s\_at" "1007\_s\_at"

kable(head(mm(Data)),digits = 3)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | High1 | High2 | High3 | High4 | Low1 | Low2 | Low3 | Low4 |
| 370871 | 338 | 123 | 115 | 113 | 131 | 152 | 145 | 145 |
| 564482 | 831 | 435 | 393 | 397 | 695 | 791 | 852 | 852 |
| 1050513 | 1190 | 520 | 345 | 583 | 685 | 688 | 1033 | 1033 |
| 239977 | 1288 | 691 | 592 | 564 | 837 | 846 | 891 | 891 |
| 1141565 | 2740 | 1161 | 1035 | 1477 | 1540 | 1938 | 2269 | 2269 |
| 1131946 | 786 | 376 | 335 | 355 | 591 | 631 | 791 | 791 |

kable(head(pm(Data)),digits = 3)

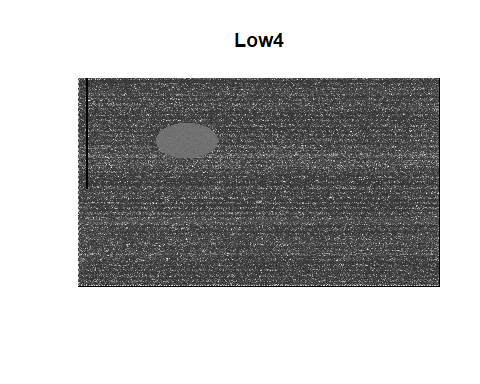
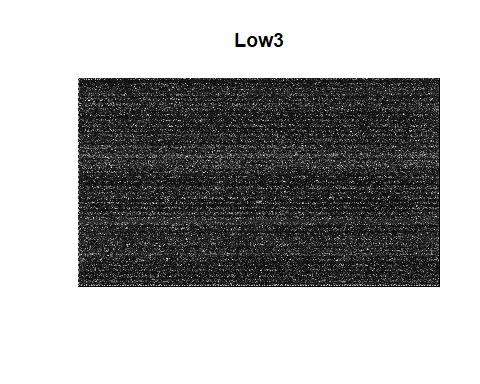
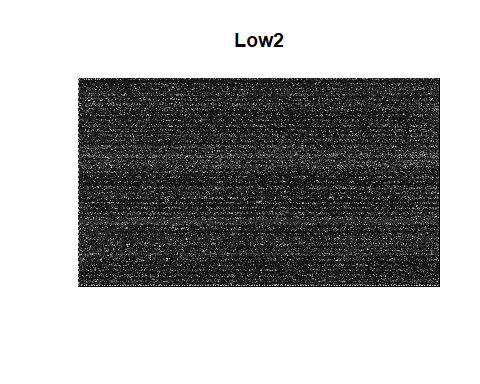
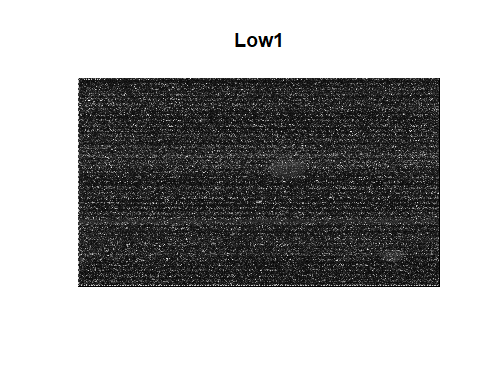
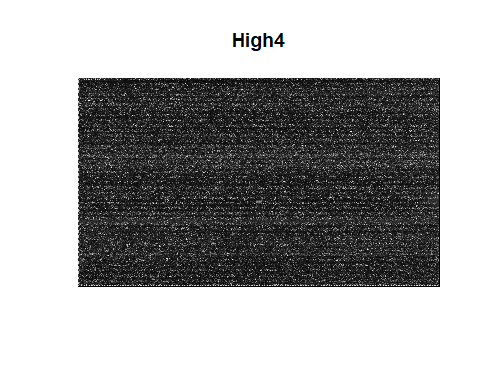
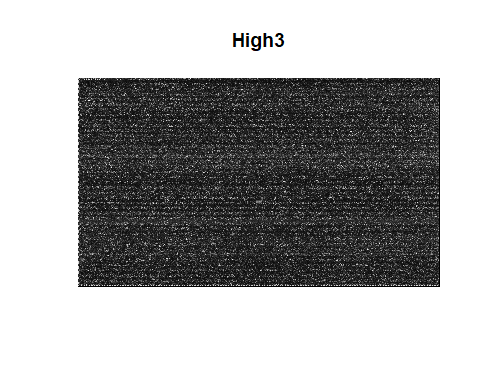
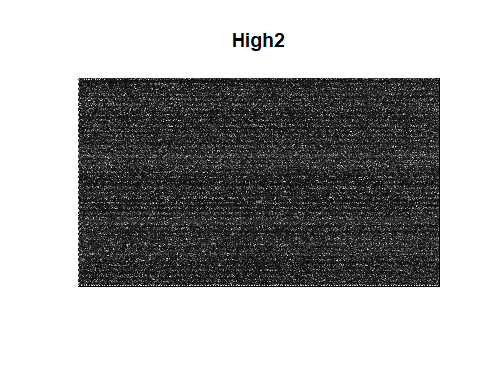
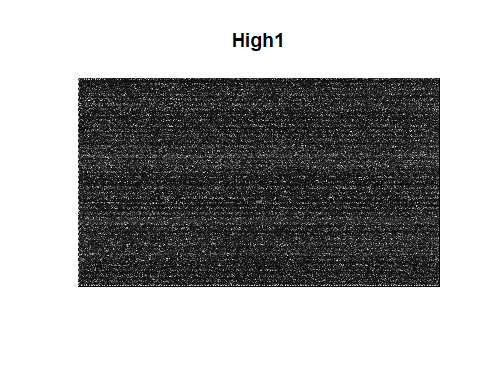
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | High1 | High2 | High3 | High4 | Low1 | Low2 | Low3 | Low4 |
| 369707 | 368 | 220 | 216 | 248 | 405 | 379 | 476 | 476 |
| 563318 | 1350 | 687 | 590 | 585 | 1455 | 1626 | 1765 | 1765 |
| 1049349 | 1414 | 674 | 599 | 586 | 1464 | 1422 | 1701 | 1701 |
| 238813 | 3838 | 1830 | 1472 | 1446 | 2980 | 3205 | 3701 | 3701 |
| 1140401 | 3018 | 1338 | 1165 | 1445 | 2208 | 2616 | 3151 | 3151 |
| 1130782 | 1652 | 783 | 668 | 700 | 1249 | 1520 | 1784 | 1784 |

kable(pData(Data),digits = 3)

|  |  |  |
| --- | --- | --- |
|  | FileName | Target |
| High1 | High\_1\_HG-U133\_Plus\_2.CEL | High |
| High2 | High\_2\_HG-U133\_Plus\_2.CEL | High |
| High3 | High\_3\_HG-U133\_Plus\_2.CEL | High |
| High4 | High\_4\_HG-U133\_Plus\_2.CEL | High |
| Low1 | Low\_1\_HG-U133\_Plus\_2.CEL | Low |
| Low2 | Low\_2\_HG-U133\_Plus\_2.CEL | Low |
| Low3 | Low\_3\_HG-U133\_Plus\_2.CEL | Low |
| Low4 | Low\_4\_HG-U133\_Plus\_2.CEL | Low |

## b) Plot the raw microarray images

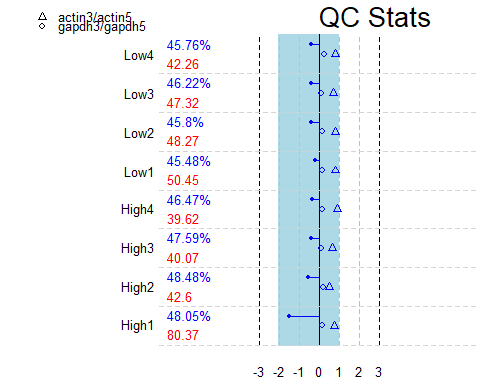
image(Data)



For the most part nothing in these plots stands out to me, except for “Low 4.” This array appears to be generally bright (washed out), with an oval patch of high fluorescence and a thin strip of probes with no fluorescence. It’s not obvious exactly what happened, but it appears that the processing for this plate went very wrong at some point.

## c) Plot quality control metrics

plot.qc.stats(qc.affy(Data))



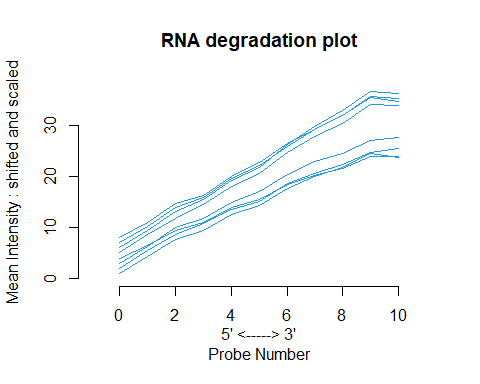
Interestingly, the quality control metrics appear to be pretty good based on this plot. The top number (a percentage) for each array is the present call, which is the percentage of genes reliably detected, and can be considered a measure of overall quality. In general these arrays appear to have similar percentages, which is a good sign. The bottom number represents average background level, and this appears to be pretty similar across arrays except for “High1” which has a much higher level than the rest. This suggests potential signal-to-noise problems with this array.

Also, it is a little bit surprising that the background for “Low4” is among the lower end of background values, because visual inspection suggested there was a lot of noise. The package automatically flags significantly different (by default defined as more than 10% difference for present call and 20 units for background) values in red. So in general these values are acceptable except for the background value of “High1.”

The GAPDH ratios (circles) and beta actin ratios (triangles) also look reasonable for all arrays. In general the triangles should be within 3-fold change and the circles within 1, and the simpleaffy package will again automatically flag values outside these ranges by coloring them red. Lastly, the horizontal line for each array represents the distance from 0-fold change to its scale factor. The ends of each line should be within the shaded blue region, so the scale factors for all of these plates are compatible.

## d) Plot the mean intensity from 3’ to 5’ end of the target mRNA

plotAffyRNAdeg(AffyRNAdeg(Data))



Probes in a probeset are ordered relative to the 5’ end. Intensities are averaged by location across all probesets, for each array. Because RNA tends to start degradation at the 5’ end, so if RNA degradation is a problem one would expect lower intensities at that end. This appears to be the case for this dataset, but it’s probably worth checking using summaryAffyRNAdeg, which provides a slope estimate and p value ot accompany this plot:

summaryAffyRNAdeg(AffyRNAdeg(Data))

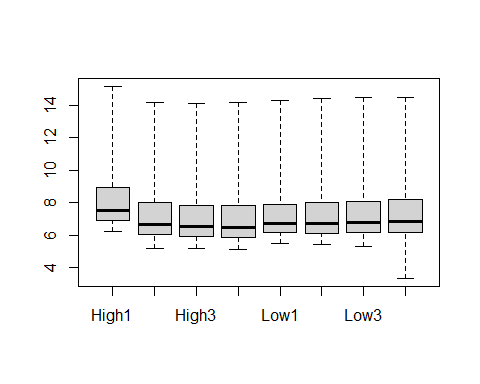
## High1 High2 High3 High4 Low1 Low2 Low3 Low4  
## slope 2.40e+00 2.37e+00 2.53e+00 2.09e+00 3.06e+00 3.23e+00 2.97e+00 2.91e+00  
## pvalue 3.32e-09 3.72e-10 3.90e-10 1.22e-09 7.00e-11 1.21e-10 3.71e-10 1.02e-10

This confirms that RNA degradation is likely a problem with these data (for all chips). On the positive side, it appears that the slopes are fairly similar across all arrays, which probably means that we can still make relative comparisons.

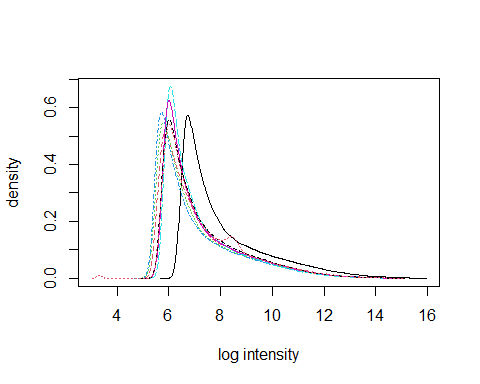
## e) Examine the distribution of intensity values for the perfect-match and mismatch probes

### Perfect match

boxplot(Data,which = "pm")

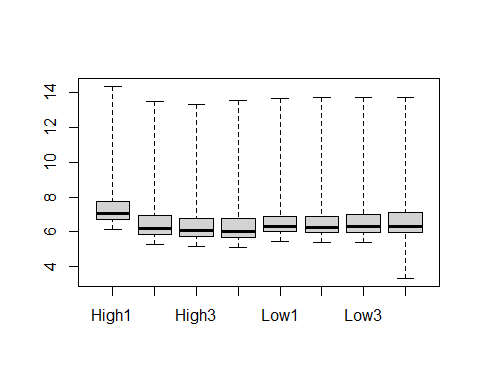


plotDensity.AffyBatch(Data,which = "pm")

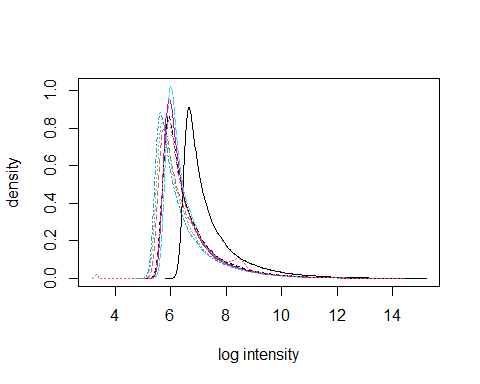


### Mismatch

boxplot(Data,which = "mm")



plotDensity.AffyBatch(Data,which = "mm")



For both sets of probes, the intensity tends to be higher on “High1” compared to the rest of the arrays (at least I’m assuming that the solid black line produced by plotDensity.AffyBatch() is the “High1” array, but unfortunately it isn’t obvious how to add a legend to this figure). The rest of the arrays appear to be in pretty close agreement with each other based on these plots, although “Low4” has a wider range than the others.

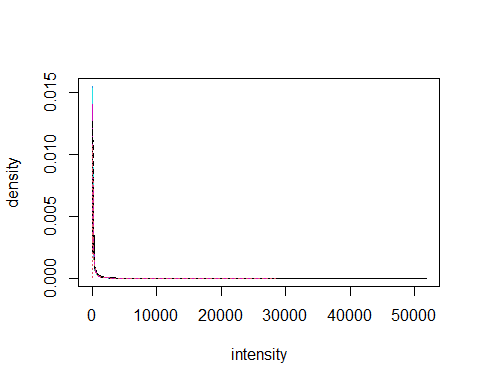
## f) Recommendations

Based on the QC plots and tables, I would recommend keeping an eye on “High1” at the very least. Its background level of intensity seems significantly different from the other arrays, but hopefully this will be fixed during the normalization step. It might also be worth looking into array “Low4” a little more as well, because the raw image looks awful. It’s surprising to me that “Low4” didn’t particularly stand out during QC. Finally, I’m not sure how much of a concern the RNA degradation is. It appears to be significant, but at a comparable level across all the arrays. So I probably wouldn’t exclude anything at this point, but “High1” and “Low4” are candidates for exclusion at later steps and warrant extra attention.

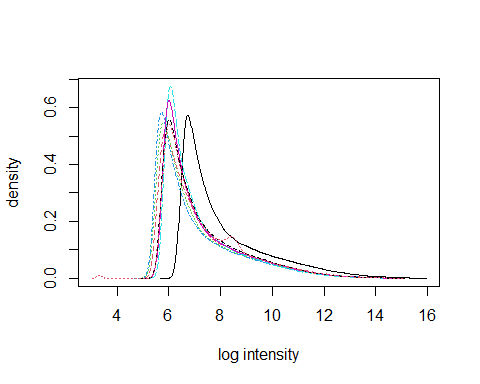
# Normalization

## a) Plot log-transformed and non-transformed data

plotDensity.AffyBatch(Data,log = F)



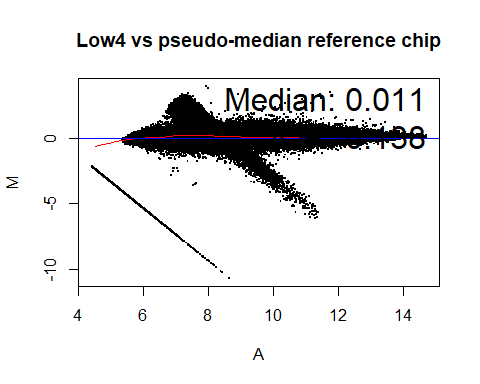
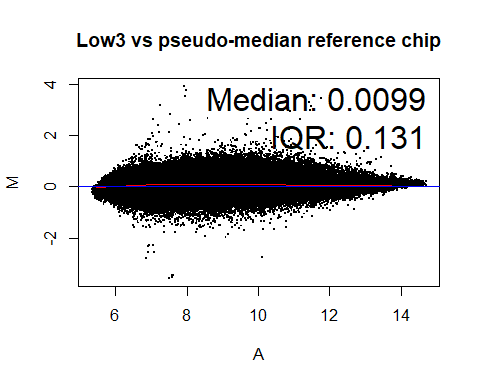
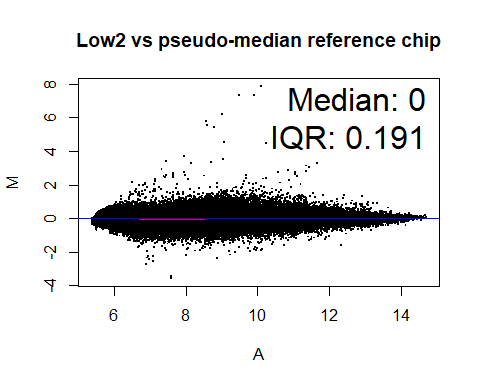
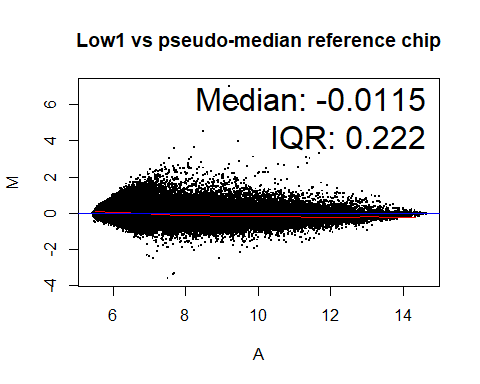
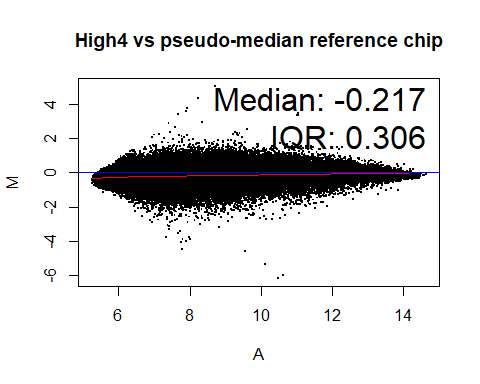
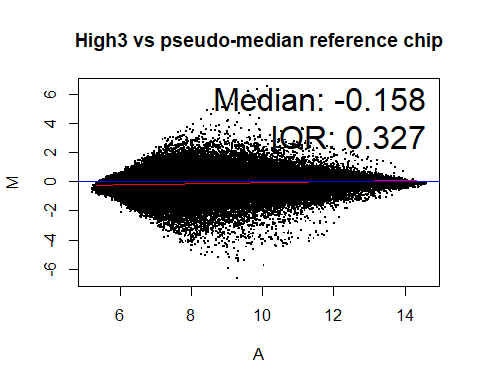
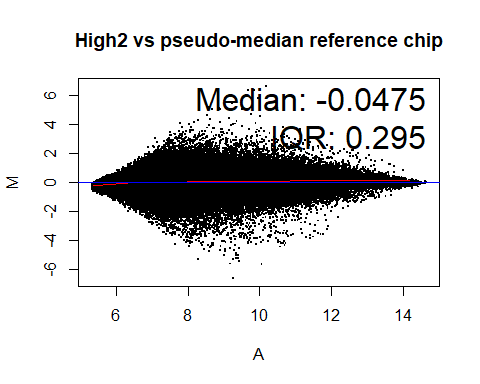
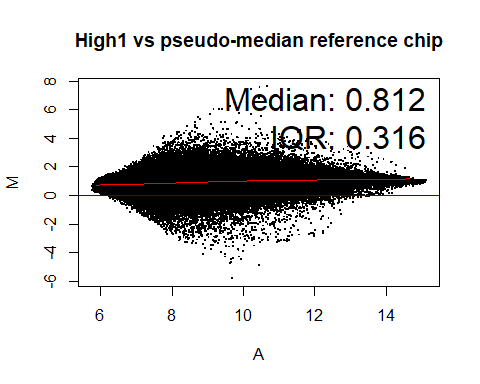
plotDensity.AffyBatch(Data)



The plot with non-transformed data is almost impossible to read because the data are so skewed and the range is so large. Most intensity values are close to 0, but some are as high as 50,000. Again, one of the arrays seems a little different from the others, although the distribution is approximately the same shape, so hopefully normalization helps this.

## b) MA plots

MAplot(Data)



The MA plots generally support the patterns observed in the QC plots, with most of the chips looking reasonably similar. However, “High1” is not clustered around the x-axis and the loess line is a little bit higher. This could be seen during QC as well, with expression for that array being generally higher than the others.

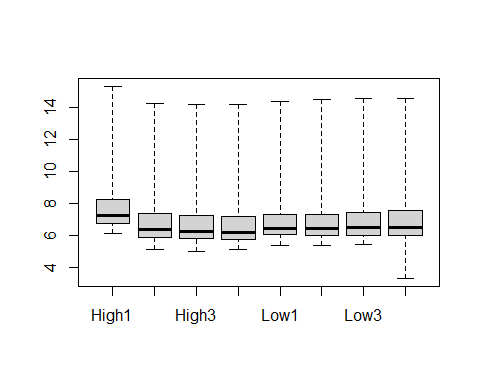
Also, the MA plots really show how bad the “Low4” array is. The loess line is actually pretty close to the x-axis (except maybe in the left tail), which explains why the QC intensity plots didn’t look too bad. However, there are points that form a thin straight line and another cloud, both at a pretty severe angle to the x-axis. You would have to assume that these are due to the obvious array defects that you can see in the raw image.

## c) Using expresso(), try different methods

### Changing the normalization method

#### Non-normalized

boxplot(Data)

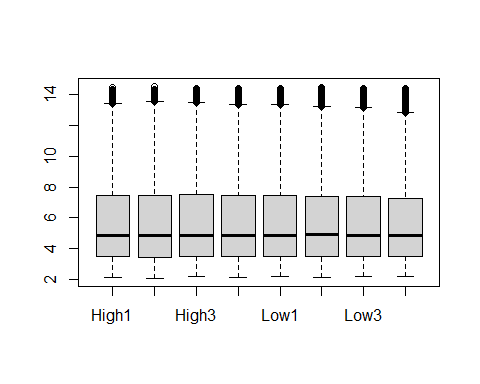


#### Quantiles

quantile\_data <- expresso(Data,normalize.method = "quantiles",  
 summary.method="avgdiff",  
 bgcorrect.method="rma",  
 pmcorrect.method="pmonly")

## background correction: rma   
## normalization: quantiles   
## PM/MM correction : pmonly   
## expression values: avgdiff   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(quantile\_data),base = 2))

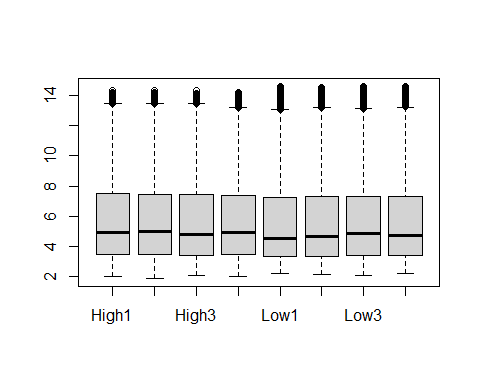


#### Loess

loess\_data <- expresso(Data,normalize.method = "loess",  
 summary.method="avgdiff",  
 bgcorrect.method="rma",  
 pmcorrect.method="pmonly")

## background correction: rma   
## normalization: loess   
## PM/MM correction : pmonly   
## expression values: avgdiff   
## background correcting...done.  
## normalizing...Done with 1 vs 2 in iteration 1   
## Done with 1 vs 3 in iteration 1   
## Done with 1 vs 4 in iteration 1   
## Done with 1 vs 5 in iteration 1   
## Done with 1 vs 6 in iteration 1   
## Done with 1 vs 7 in iteration 1   
## Done with 1 vs 8 in iteration 1   
## Done with 2 vs 3 in iteration 1   
## Done with 2 vs 4 in iteration 1   
## Done with 2 vs 5 in iteration 1   
## Done with 2 vs 6 in iteration 1   
## Done with 2 vs 7 in iteration 1   
## Done with 2 vs 8 in iteration 1   
## Done with 3 vs 4 in iteration 1   
## Done with 3 vs 5 in iteration 1   
## Done with 3 vs 6 in iteration 1   
## Done with 3 vs 7 in iteration 1   
## Done with 3 vs 8 in iteration 1   
## Done with 4 vs 5 in iteration 1   
## Done with 4 vs 6 in iteration 1   
## Done with 4 vs 7 in iteration 1   
## Done with 4 vs 8 in iteration 1   
## Done with 5 vs 6 in iteration 1   
## Done with 5 vs 7 in iteration 1   
## Done with 5 vs 8 in iteration 1   
## Done with 6 vs 7 in iteration 1   
## Done with 6 vs 8 in iteration 1   
## Done with 7 vs 8 in iteration 1   
## 1 0.6082201   
## done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(loess\_data),base = 2))

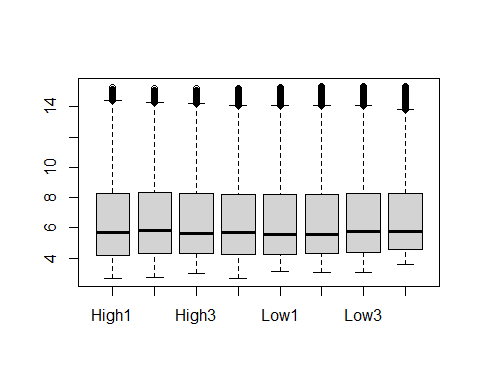


#### Constant

constant\_data <- expresso(Data,normalize.method = "constant",  
 summary.method="avgdiff",  
 bgcorrect.method="rma",  
 pmcorrect.method="pmonly")

## background correction: rma   
## normalization: constant   
## PM/MM correction : pmonly   
## expression values: avgdiff   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(constant\_data),base = 2))



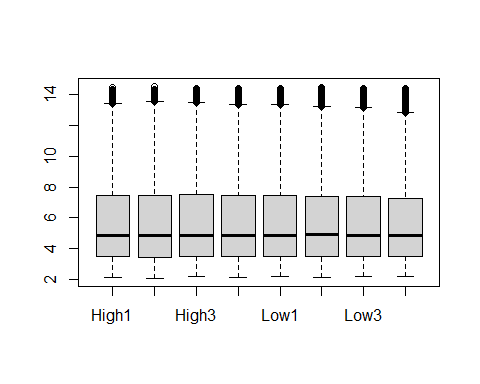
Based on visual inspection of these boxplots, I don’t think that one normalization method performs better than the others (at least not with summary.method="avgdiff", bgcorrect.method="rma", and pmcorrect.method="pmonly"). All of the arrays look similar to each other, and with all methods “High1” and “Low4” look much improved. However, the quantile method is faster than the others, so I would probably favor this method. The loess method is particularly slow and I would avoid it unless the other two wouldn’t work for some reason.

### Changing the summary method

The following plots are generated using the quantile normalization method, pmcorrect.method = "pmonly", and bgcorrect.method="rma" with different options for summary.method.

#### AvgDiff

boxplot(log(exprs(quantile\_data),base = 2))

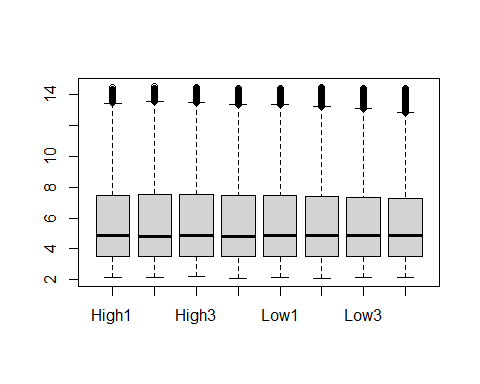


#### MAS

mas\_data <- expresso(Data,summary.method="mas",  
 normalize.method = "quantiles",  
 bgcorrect.method="rma",  
 pmcorrect.method="pmonly")

## background correction: rma   
## normalization: quantiles   
## PM/MM correction : pmonly   
## expression values: mas   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(mas\_data),base = 2))

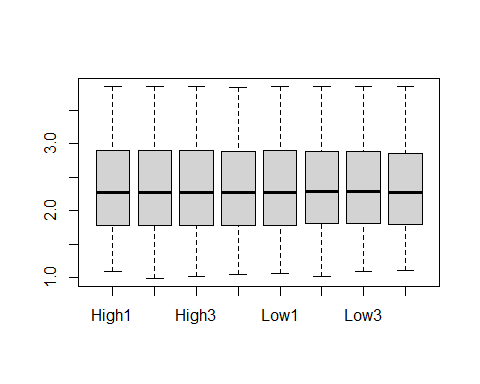


#### Median polish

median\_data <- expresso(Data,summary.method="medianpolish",  
 normalize.method = "quantiles",  
 bgcorrect.method="rma",  
 pmcorrect.method="pmonly")

## background correction: rma   
## normalization: quantiles   
## PM/MM correction : pmonly   
## expression values: medianpolish   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(median\_data),base = 2))



The MAS and AvgDiff boxplots look almost identical (in fact they are so similar that I was convinced I had accidentally messed up my code), but the median polish method does look different from the other two. It appears that the median polish method produces expression values that are more normally distributed than those produced by the other methods. I did not notice big differences in speed, so I think that median polish is the best of these options (at least with the normalization and background and PM correction methods I selected).

### Changing the PM correction method

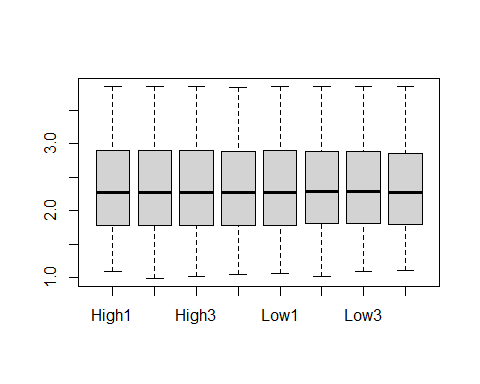
Using the quantile normalization method, summary.method = "medianpolish", and bgcorrect.method="rma".

#### PM Only

pmonly\_data <- expresso(Data,pmcorrect.method="pmonly",  
 summary.method="medianpolish",  
 normalize.method = "quantiles",  
 bgcorrect.method="rma")

## background correction: rma   
## normalization: quantiles   
## PM/MM correction : pmonly   
## expression values: medianpolish   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(pmonly\_data),base = 2))



#### MAS

mas\_data\_2 <- expresso(Data,pmcorrect.method="mas",  
 summary.method="medianpolish",  
 normalize.method = "quantiles",  
 bgcorrect.method="rma")

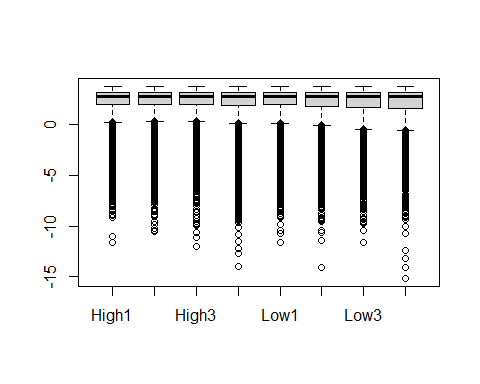
## background correction: rma   
## normalization: quantiles   
## PM/MM correction : mas   
## expression values: medianpolish   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |#############

## Warning in medpolish(log2(x), trace.iter = FALSE, ...): medpolish() did not  
## converge in 10 iterations

## #######|

boxplot(log(exprs(mas\_data\_2),base = 2))

## Warning in boxplot(log(exprs(mas\_data\_2), base = 2)): NaNs produced

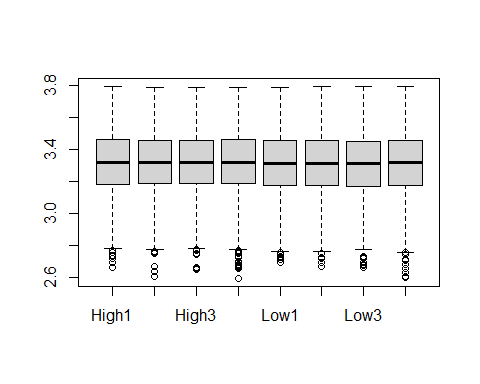


#### Subtract mismatch

subtractmm\_data <- expresso(Data,pmcorrect.method="subtractmm",  
 summary.method="medianpolish",  
 normalize.method = "quantiles",  
 bgcorrect.method="rma")

## background correction: rma   
## normalization: quantiles   
## PM/MM correction : subtractmm   
## expression values: medianpolish   
## background correcting...done.  
## normalizing...done.  
## 54675 ids to be processed  
## | |  
## |####################|

boxplot(log(exprs(subtractmm\_data),base = 2))



Unfortunately, neither the MAS or subtract MM approaches seem to work well with the median polish summary method (the subtract MM plot looks reasonable but it produces a lot of concerning “NaN” warnings that I suppressed in the output above). The PM only plot with median polish as the summary method looks excellent though, so this would be my choice moving forward.

## d) Get present and absent calls

# Presence/absence detection  
calls <- mas5calls(Data)

## Getting probe level data...  
## Computing p-values  
## Making P/M/A Calls

# Get p values  
calls\_sig <- as.data.frame(assayData(calls)$exprs)  
# Subset those with at least one significant p value in each group  
filtered <- calls\_sig %>%   
 filter(("P"==High1|"P"==High2|"P"==High3|"P"==High4)&  
 ("P"==Low1|"P"==Low2|"P"==Low3|"P"==Low4))

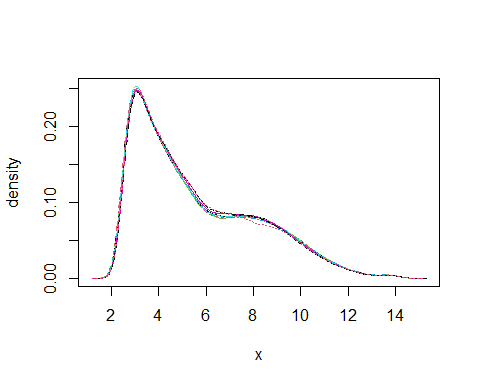
After filtering there are 25726 probesets left out of 54675 (47.1%).

## e) Problematic chips

Although most of the boxplots seem reasonable for “Low4,” the MA plot in comparison to the pseudochip suggests that we should exclude it. The bright oval and dark strip that you can see in the raw image are very obvious in this plot, and I worry that this chip will negatively affect normalization. Because I chose a normalization technique that uses information from all the chips, it’s important to exclude obvious problematic arrays before the normalization step.

However, the potential problems with “High1” appear to have been fixed by the normalization procedures:

plotDensity(exprs(pmonly\_data))



So the most appropriate approach seems to be to exclude “Low4” but keep “High1” in the dataset for analysis.

### Re-normalization

Import data without “Low4”:

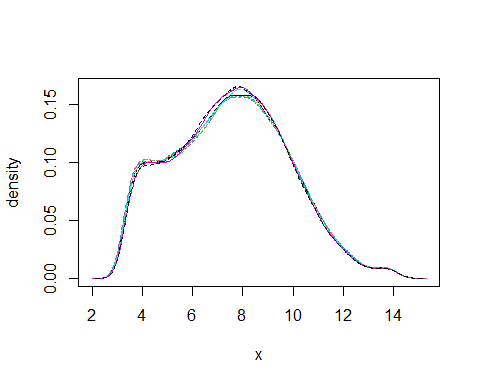
pd = read.AnnotatedDataFrame("./targets\_low4\_excluded.txt" ,  
 header=TRUE,row.names=1,as.is=TRUE)  
Data = ReadAffy(filenames=pData(pd)$FileName, phenoData=pd,  
 sampleNames=sampleNames(pd))

Normalize using the subset from the filtering step d) above, and generate new plots:

normalized <- rma(Data,subset = rownames(filtered))

## Background correcting  
## Normalizing  
## Calculating Expression

plotDensity(exprs(normalized))



Save files of the intensities and calls locally:

# Save normalized data  
save(normalized,file = "./normalized.R")  
# Get calls and save  
calls <- mas5calls(Data,ids=rownames(filtered))

## Getting probe level data...  
## Computing p-values  
## Making P/M/A Calls

save(calls,file = "./calls.R")

Show the top lines:

kable(head(log(exprs(normalized),base = 2)),digits = 3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | High1 | High2 | High3 | High4 | Low1 | Low2 | Low3 |
| 1007\_s\_at | 3.322 | 3.298 | 3.308 | 3.274 | 3.514 | 3.492 | 3.505 |
| 1053\_at | 3.215 | 3.218 | 3.212 | 3.287 | 3.193 | 3.193 | 3.203 |
| 117\_at | 2.666 | 2.663 | 2.635 | 2.476 | 2.499 | 2.518 | 2.480 |
| 121\_at | 2.849 | 2.817 | 2.822 | 2.851 | 2.847 | 2.841 | 2.889 |
| 1294\_at | 2.375 | 2.366 | 2.416 | 2.358 | 2.383 | 2.407 | 2.407 |
| 1316\_at | 2.032 | 2.027 | 2.043 | 2.032 | 2.014 | 2.069 | 2.058 |

kable(head(assayData(calls)$exprs),digits = 3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | High1 | High2 | High3 | High4 | Low1 | Low2 | Low3 |
| 1007\_s\_at | P | P | P | P | P | P | P |
| 1053\_at | P | P | P | P | P | P | P |
| 117\_at | P | P | P | P | P | P | P |
| 121\_at | P | P | P | P | P | P | P |
| 1294\_at | P | A | P | A | M | P | P |
| 1316\_at | P | P | P | A | A | P | M |