Lab #4
Jaimee Beckett

In this lab, we will be working with a few data sets, each run on a different platform. The first data set is an R object generated from a 2-channel cDNA array that is called swirl. This data set is an experiment that was run on a zebrafish to study the early development. The data is named such because “swirl is a point mutant in the BMP2 gene that affects the dorsal/ventral body axis.” The objective of the experiment was to evaluate the transcript differences between wildtype zebrafish and those with this mutation. As I mentioned above, swirl is an R object, so the format and structure of this binary file has to be accessed through various R functions. If you type “swirl”, you will immediately see that there are attributes that make up this file (e.g. @maInfo) beyond the typical channel information. Included is metadata information that makes up the experimental parameters, in addition to the raw intensity data.

The second 2 data sets are raw intensity files – one from an Agilent platform and the other from an Affymetrix platform. Both of these files are on the course website. These are not R objects, rather the Agilent files are raw text files generated from the Agilent software and the Affymetrix files are raw binary files generated from the Affymetrix software.

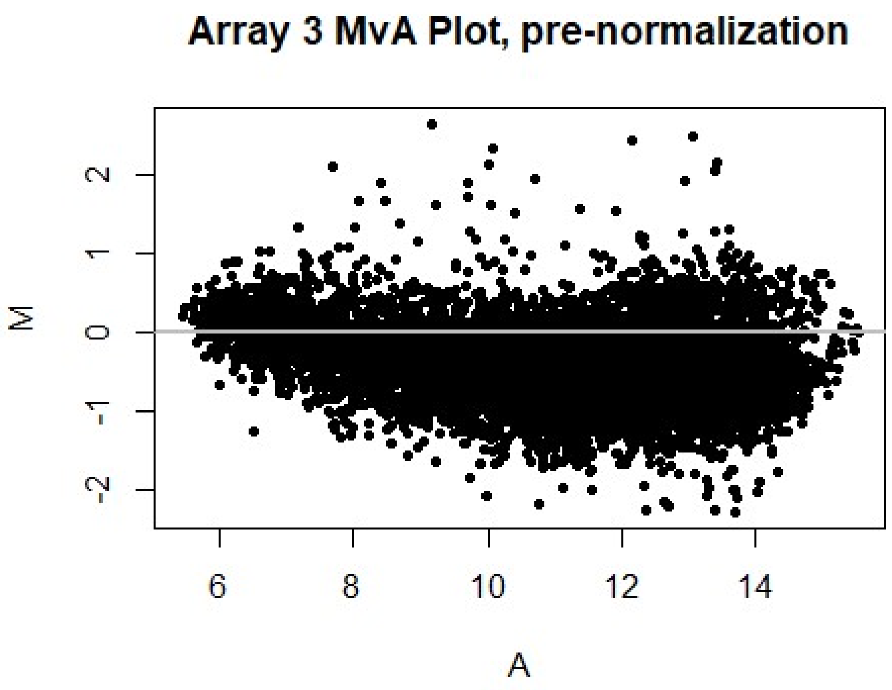
Since both R objects and raw data files are typically what an analyst is given when asked to analyze an experiment, this lab will give you experience processing raw intensity files and normalizing them appropriately. This is typically the first step in conducting any microarray analysis, so it is important to make sure that the data is normalized appropriately before beginning any subsequent steps.

1.) Load the marray library and the swirl data set. This data set is an R metadata object, so there are multiple pieces of information (e.g., red/green background and foreground intensities, chip layout design, etc.) that are stored in this R data object.

```
library(marray)
data(swirl)
```

2.) Plot an MvA plot of array 3 without any stratified lines.

```
maPlot(swirl[,3], legend.func=NULL, lines.func=NULL,
       main="Array 3 MvA Plot, pre-normalization")
```

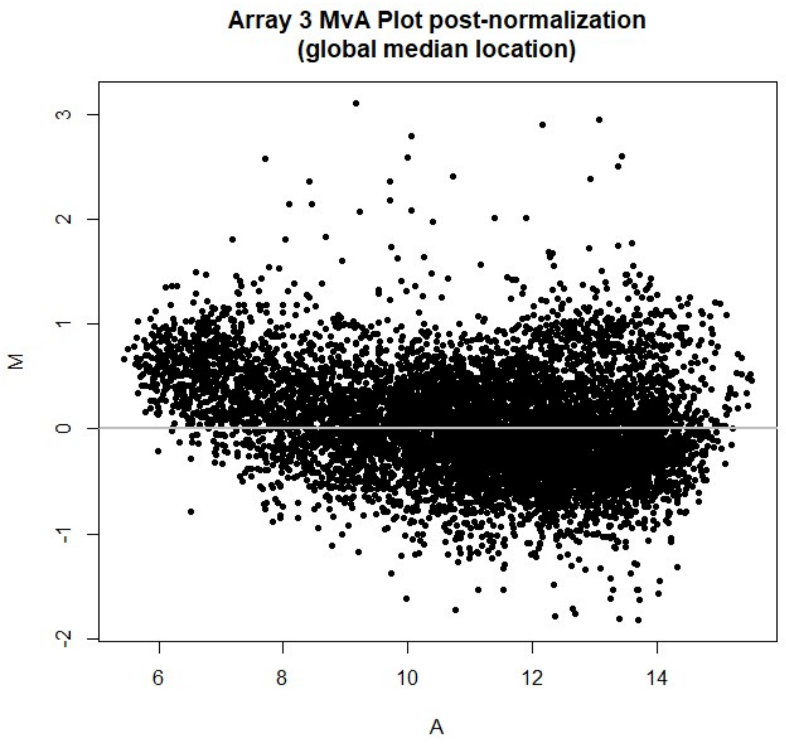


3.) Normalize array 3 by global median location normalization.

```
mnorm <- maNorm(swirl[,3], norm="median")
```

4.) Plot an MvA plot of the normalized array without the stratified lines or legend.

```
maPlot(mnorm, legend.func=NULL, lines.func=NULL,
       main="Array 3 MvA Plot post-normalization (global median location)")
```

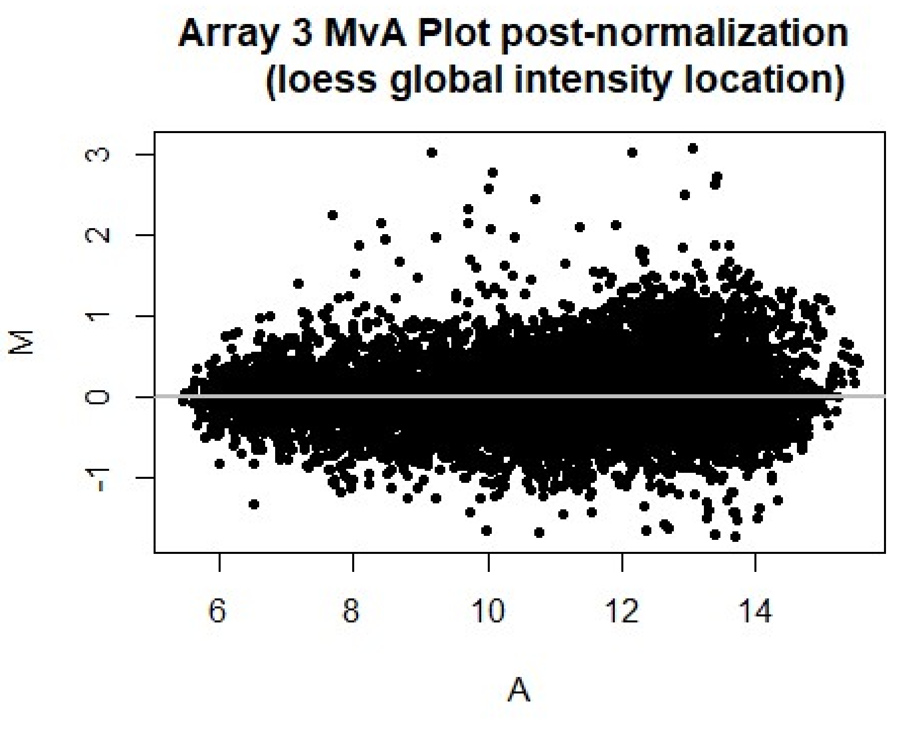


5.) What is different between the normalized data and the non-normalized data?

Non-normalized data is raw data, but normalized data is scaled with a numeric function. Normalization removes background noise and systematic effects so that you can make comparisons with your data

6.) Repeat #3 and #4 applying loess global intensity normalization.

```
mnorm <- maNorm(swirl[,3], norm="loess", span=0.45)
maPlot(mnorm, legend.func=NULL, lines.func=NULL,
       main="Array 3 MvA Plot post-normalization
       (loess global intensity location)")
```



7.) Which of the two normalizations appears to be better for this array?

The loess global intensity normalization appears to be better for this array since there is more symmetry at M=0 than the global median location normalization.

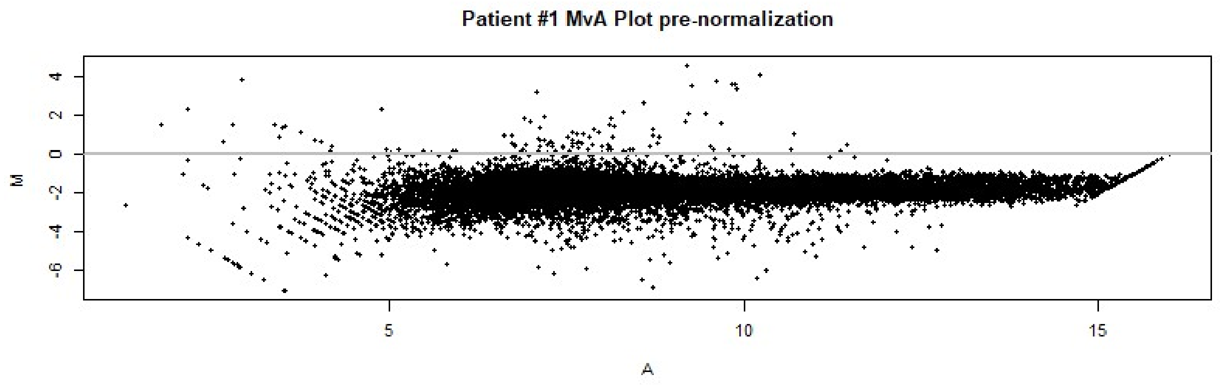
8.) Now we would like to read in raw GenePix files for 2 cDNA arrays that represent 2 patient samples. Go to the course website and retrieve the compressed file called 'GenePix files'. Open it up and put the contents in a directory. Now using the sample code below, read in the 2 array files.

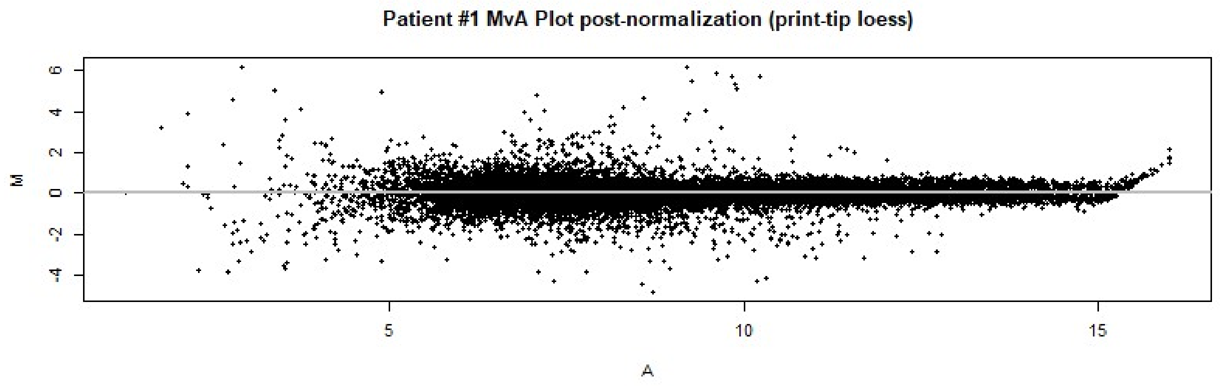
```
a.dcna <- read.GenePix(name.Gf="F532 Median", name.Gb="B532 Median", name.RF=
                                 "F635 Median",name.Rb="B635 Median", name.W="Flags")
```

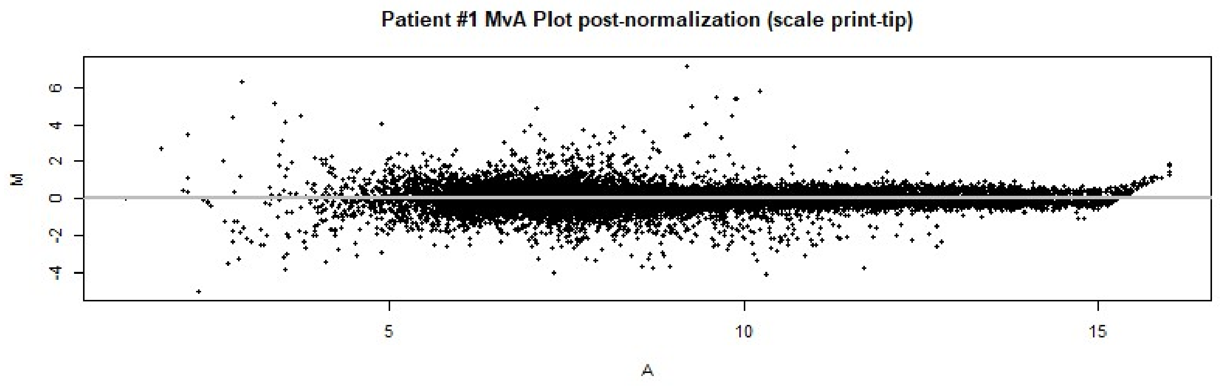
9.) Using the a.cdna object, which is analogous to the swirl metadata object, normalize both arrays and provide MvA plots for each array normalized by the following 3 methods: no normalization, print-tip loess normalization, and scale print-tip normalization using the MAD. Hint: use the par(mfrow=c(3,1)) function to put the 3 plots for a single patient array on the same page.

```
mnorm.loess <- maNorm(a.dcna, norm="p", span=0.45)
mnorm.scale <- maNorm(a.dcna, norm="s")

par(mfrow=c(3,1))
#patient 1
#no normalization
maPlot(a.dcna[,1], legend.func=NULL, lines.func=NULL,
       main="Patient #1 MvA Plot pre-normalization")
```

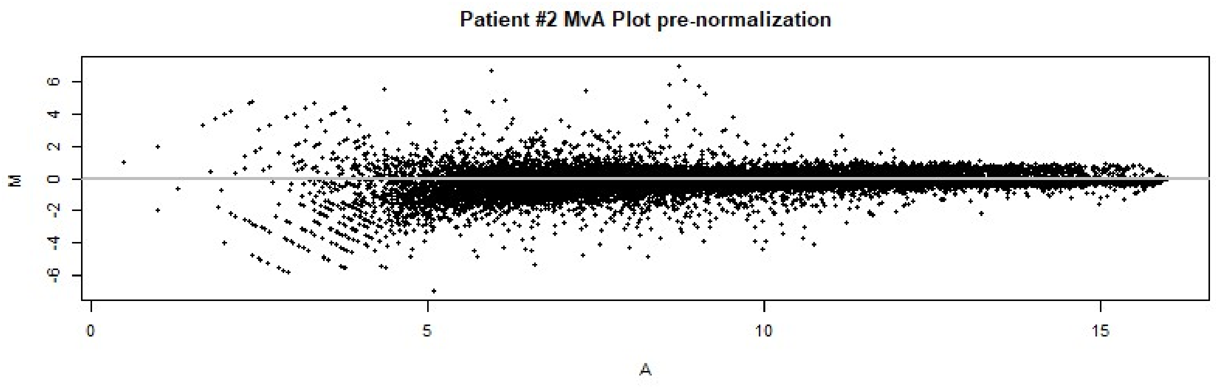
```r
#print-tip loess normalization
maPlot(mnorm.loess[,1], legend.func=NULL, lines.func=NULL,
       main="Patient #1 MvA Plot post-normalization (print-tip loess)")

#scale print-tip normalization
maPlot(mnorm.scale[,1], legend.func=NULL, lines.func=NULL,
       main="Patient #1 MvA Plot post-normalization (scale print-tip)")
```

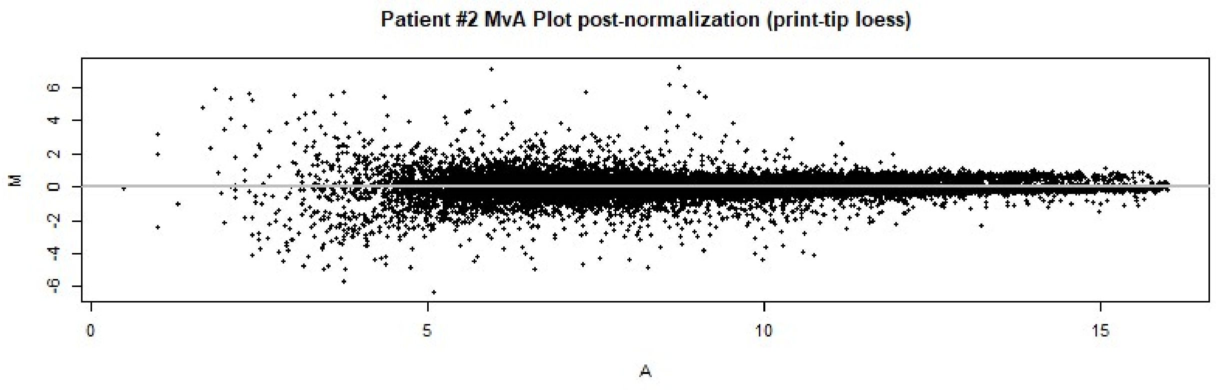


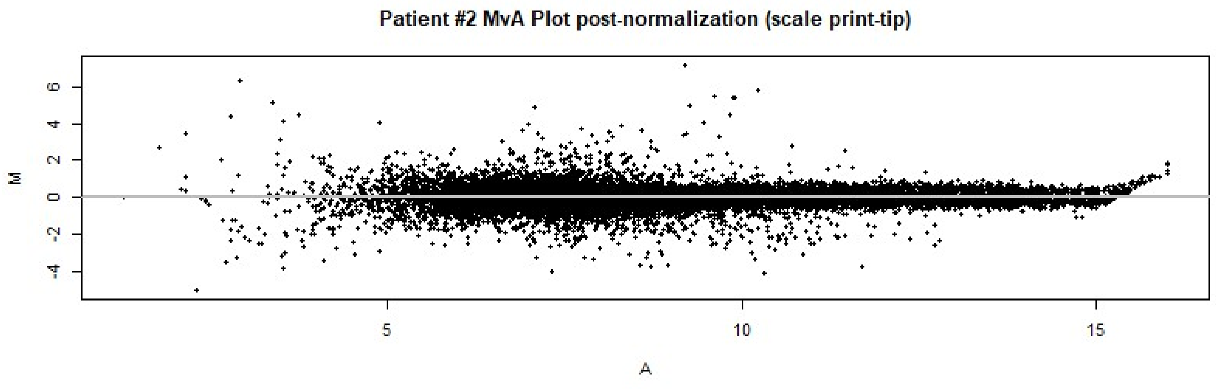




```r
#patient 2
#no normalization
maPlot(a.dcna[,2], legend.func=NULL, lines.func=NULL,
          main="Patient #2 MvA Plot pre-normalization")

#print-tip loess normalization
maPlot(mnorm.loess[,2], legend.func=NULL, lines.func=NULL,
          main="Patient #2 MvA Plot post-normalization (print-tip loess)")

#scale print-tip normalization
maPlot(mnorm.scale, legend.func=NULL, lines.func=NULL,
          main="Patient #2 MvA Plot post-normalization (scale print-tip)")
```







10.) Finally, we would like to create a data matrix that can be written out to a file with 19,200 rows and 2 columns (i.e. each patient array). Using the functions maM(), maGnames(), and maLabels(), figure out how to create the data matrix, get the probe IDs, and assign the probe IDs to the row names. Do this for the 2 normalized metadata objects that you created in #9 above (don't worry about the un-normalized data matrix).

```
m.loess <- maM(mnorm.loess)
m.loess.labels <- maLabels(maGnames(mnorm.loess))
rownames(m.loess) <- m.loess.labels
dim(m.loess)

## [1] 19200     2

m.scale.pt <- maM(mnorm.scale)
m.scale.pt.labels <- maLabels(maGnames(mnorm.scale))
rownames(m.scale.pt) <- m.scale.pt.labels
dim(m.scale.pt)

## [1] 19200     2
```

11.) Load the following libraries: affy, limma, simpleaffy, affyPLM, and fpc.

```
library(affy)
library(limma)
library(simpleaffy)
library(affyPLM)
library(fpc)
```

12.) Now we would like to read in 3 raw Affymetrix .CEL files and normalize them with 2 different algorithms. These 3 arrays represent 3 normal healthy subjects that should have similar expression profiles. They are on the course website in the compressed file called Affymetrix .CEL files. Use the following code below to read in a metadata object for the 3 arrays (dir.path should be the same as above).

```
fns <- sort(list.celfiles(full.names=TRUE))
data.affy <- ReadAffy(filenames=fns, phenoData = NULL)
```

13.) Using the function: expresso in addition to exprs(), create the normalized data matrices with 54,675 rows and 3 columns for the 2 different normalization algorithms.

Be sure to use normalize.method="quantiles", summary.method="medianpolish", and for RMA: pmcorrect.method="pmonly" ###MAS: pmcorrect.method="mas"

```
#RMA
eset <-expresso(data.affy,bgcorrect.method="rma", normalize.method =
               "quantiles", summary.method="medianpolish",pmcorrect.method =
               "pmonly")

norm.rma <- exprs(eset)
dim(norm.rma)

## [1] 54675     3
```

```
#MAS
eset <- expresso(data.affy,bgcorrect.method="mas",normalize.method="quantiles
",
                 summary.method="medianpolish",pmcorrect.method="mas")

norm.mas <- exprs(eset)
dim(norm.mas)

## [1] 54675     3
```

14.) Now use the cor() function to calculate the correlation between the 3 arrays for both normalized data matrices. Since these 3 subjects are all healthy normal individuals, we would expect to see somewhat good correlation structure between them all when looking across all genes on the array. Which normalization method has a higher overall correlation structure for these 3 normal healthy subjects? Show how you arrived at this answer.

```
norm.rma.cor <- cor(norm.rma)
norm.rma.cor

##             normal1.CEL normal2.CEL normal3.CEL
## normal1.CEL   1.0000000   0.9766785   0.9758377
## normal2.CEL   0.9766785   1.0000000   0.9913585
## normal3.CEL   0.9758377   0.9913585   1.0000000

norm.mas.cor <- cor(norm.mas)
norm.mas.cor

##             normal1.CEL normal2.CEL normal3.CEL
## normal1.CEL   1.0000000   0.8959518   0.9003087
## normal2.CEL   0.8959518   1.0000000   0.9486851
## normal3.CEL   0.9003087   0.9486851   1.0000000
```

RMA normalization method has a higher overall correlation structure than MAS. When you take the average of each column for both correlation matrices, the average for each column in RMA's correlation matrix is closer to 1 than any column average in the MAS correlation matrix:

```
colMeans(norm.rma.cor)

## normal1.CEL normal2.CEL normal3.CEL
##   0.9841721   0.9893457   0.9890654

colMeans(norm.mas.cor)

## normal1.CEL normal2.CEL normal3.CEL
##   0.9320869   0.9482123   0.9496646
```