Julie Garcia
October 3, 2017
Lab #4
Normalization and Bioconductor

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.

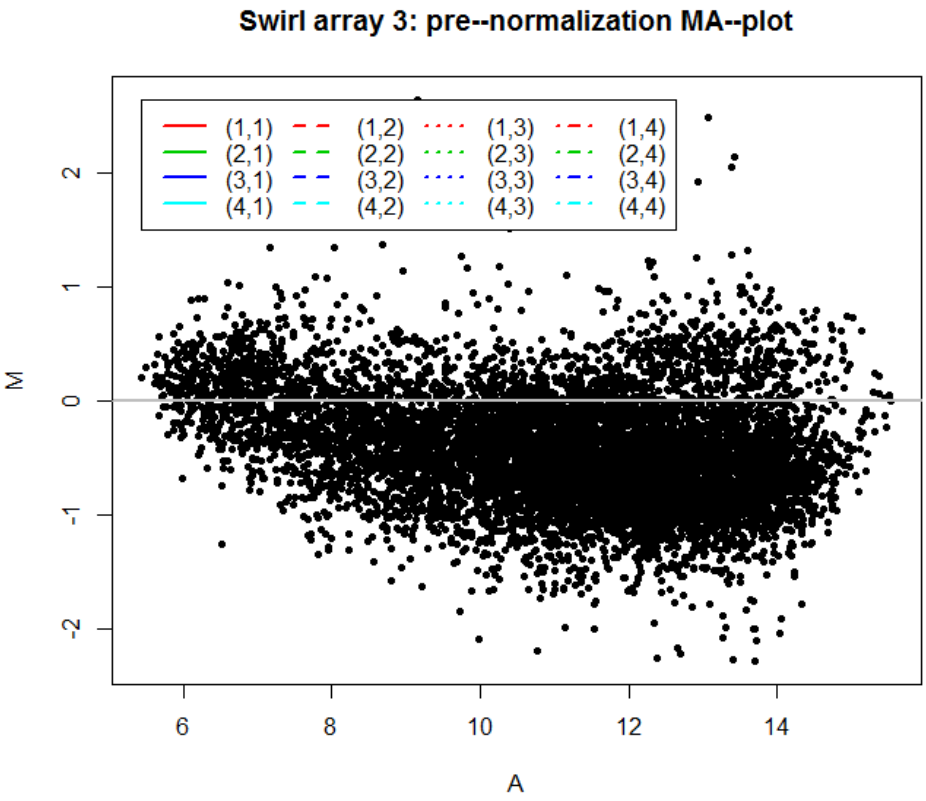
1.) Load the marray library and the swirl data set. This data set is an R metadata object, so there are multiple pieces of information (e.g., red/green background and foreground intensities, chip layout design, etc.) that are stored in this R data object.

```
source("https://bioconductor.org/biocLite.R")
biocLite("marray")
library(marray)
data(swirl)
```

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

```
swirl3 <- swirl[,3]
plot(swirl3,lines.func=NULL,main="Swirl array 3: pre-
normalization MA-
```

```
plot")
```

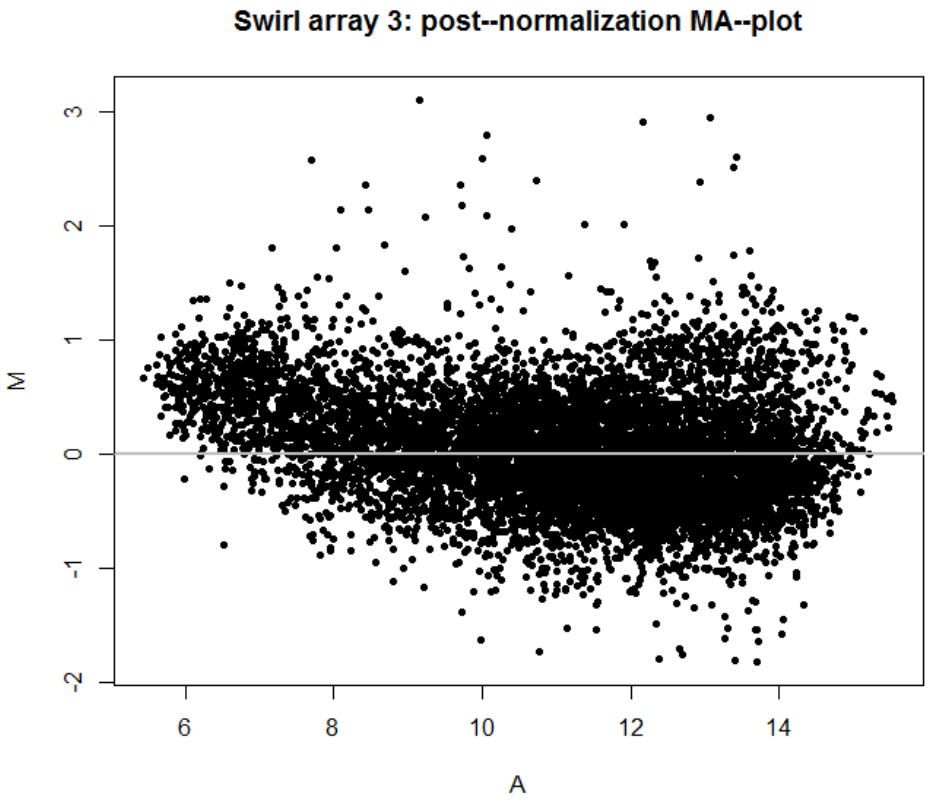


3.) Normalize array 3 by global median location normalization.

```
swirl3.norm <- maNorm(swirl3, norm="median")
```

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

```
plot(swirl3.norm, lines.func=NULL, legend=NULL,
     main="Swirl array 3: post--normalization MA--plot")
```

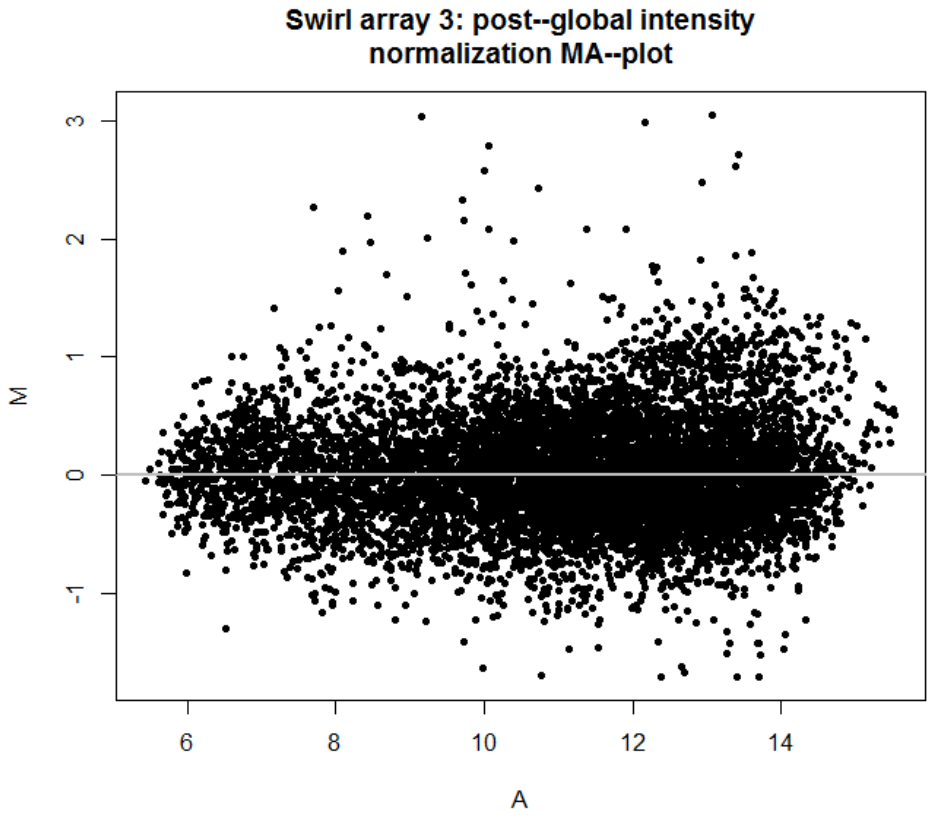


5.) What is different between the normalized data and the non-normalized data?

```
The data is more centralized across zero on the y-axis.
```

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

```
swirl3.norm.gi <- maNorm(swirl3, norm="loess")
plot(swirl3.norm.gi, lines.func=NULL, legend=NULL,
          main="Swirl array 3: post–global
          intensity\nnormalization MA--plot")
```



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

Global intensity normalization seemed to work better for this array. The data appears more normalized around the center line.

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.

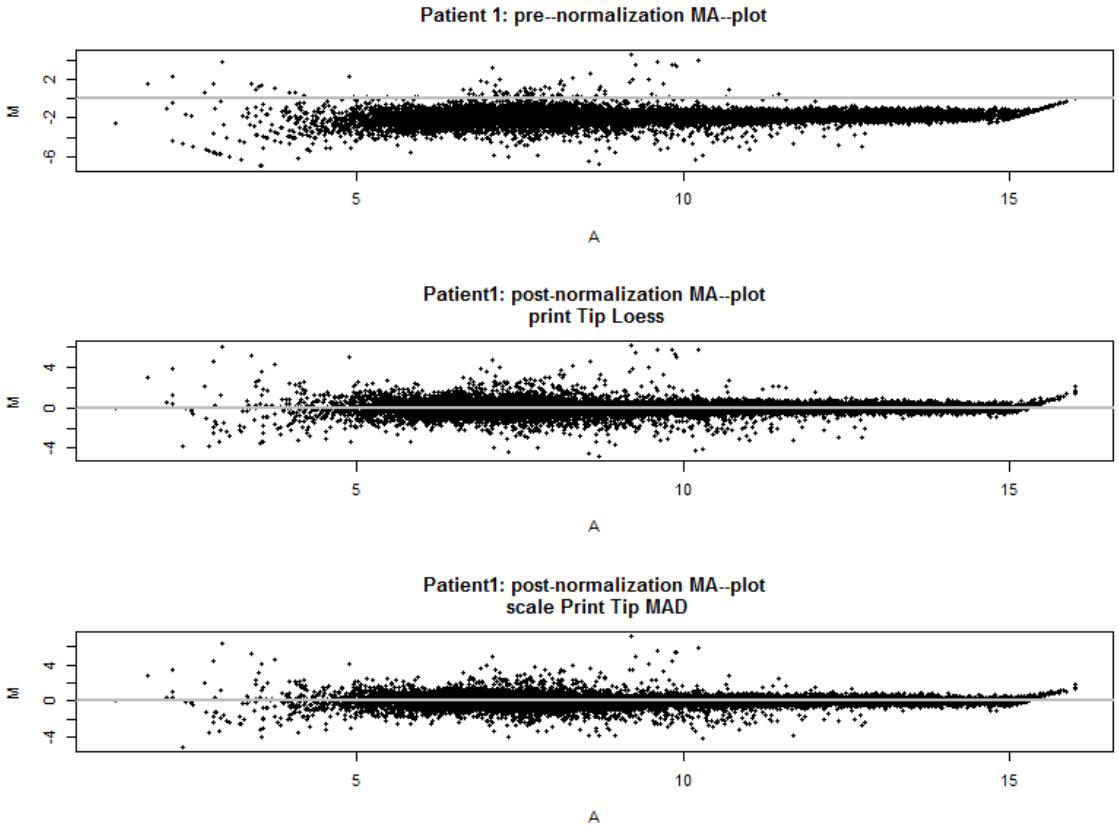
> dir.path <- "C:\\Documents and Settings\\higgsb\\Desktop\\"
> a.cdna <- read.GenePix(path=dir.path,name.Gf = "F532 Median",name.Gb ="B532 Median", name.Rf = "F635 Median", name.Rb = "B635 Median",name.W ="Flags")

```
dir.path <- "C:\\temp\\"
a.cdna <- read.GenePix(path=dir.path,name.Gf = "F532
Median",name.Gb ="B532 Median", name.Rf = "F635 Median", name.Rb
= "B635 Median",name.W ="Flags")
```

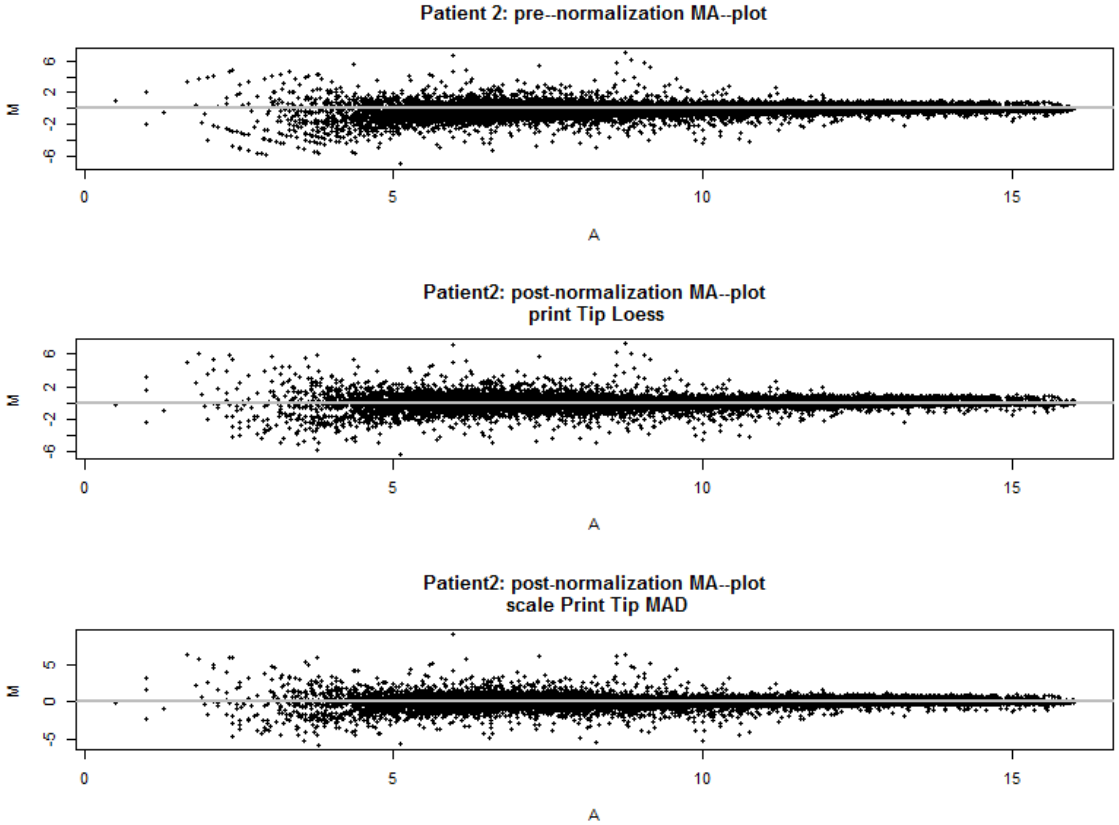
9.) Using the a.cdna object, which is analogous to the swirl metadata object, normalize both arrays and provide MvA plots for each array normalized by the following 3 methods: no normalization, print-tip loess normalization, and scale print-tip

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.

```
a.cdna.patient1 <- a.cdna[,1]
a.cdna.patient2 <- a.cdna[,2]
par(mfrow=c(3,1))
plot(a.cdna.patient1,lines.func=NULL, legend=NULL,
     main="Patient 1: pre--normalization MA--plot")

a.cdna.patient1.printeTipLoess <- maNorm(a.cdna.patient1,
norm="printTipLoess")
plot(a.cdna.patient1.printeTipLoess, lines.func=NULL,
legend=NULL,
     main="Patient1: post-normalization MA--plot\n print Tip
Loess")

a.cdna.patient1.scalePrintTipMAD <- maNorm(a.cdna.patient1,
norm="scalePrintTipMAD")
plot(a.cdna.patient1. scalePrintTipMAD, lines.func=NULL,
legend=NULL,
     main="Patient1: post-normalization MA--plot\n scale Print
Tip MAD")
```



```
par(mfrow=c(3,1))
plot(a.cdna.patient2,lines.func=NULL, legend=NULL,
     main="Patient 2: pre--normalization MA--plot")

a.cdna.patient2.printeTipLoess <- maNorm(a.cdna.patient2,
norm="printTipLoess")
plot(a.cdna.patient2.printeTipLoess, lines.func=NULL,
legend=NULL,
     main="Patient2: post-normalization MA--plot\n print Tip
Loess")

a.cdna.patient2.scalePrintTipMAD <- maNorm(a.cdna.patient2,
norm="scalePrintTipMAD")
plot(a.cdna.patient2. scalePrintTipMAD, lines.func=NULL,
legend=NULL,
     main="Patient2: post-normalization MA--plot\n scale Print
Tip MAD")
```



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).

```
p1.print <- maM(a.cdna.patient1.printeTipLoess)
p2.print <- maM(a.cdna.patient2.printeTipLoess)
probesets <- maLabels(maGnames(a.cdna))
df.print <- cbind(p1.print, p2.print)
dimnames(df.print)[[1]] <- probesets
df.print

p1.scale <- maM(a.cdna.patient1.scalePrintTipMAD)
p2.scale <- maM(a.cdna.patient2.scalePrintTipMAD)
df.scale <- cbind(p1.scale, p2.scale)
dimnames(df.scale)[[1]] <- probesets
df.scale
```

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

```
source("https://bioconductor.org/biocLite.R")
biocLite("affy")
biocLite("limma")
biocLite("simpleaffy")
biocLite("affyPLM")
biocLite("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(path=dir.path,full.names=TRUE))
> data.affy <- ReadAffy(filenames=fns,phenoData=NULL)
```

```
library(affy)
fns <- sort(list.celfiles(path="c:/temp/",full.names=TRUE))
data.affy <- ReadAffy(filenames=fns,phenoData=NULL)
```

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

```
library(simpleaffy)
data.affy.mas <- justMAS(data.affy)
data.affy.mas
mas <- exprs(data.affy.mas)
dim(mas)
data.affy.rma <- rma(data.affy)
data.affy.rma
rma <- exprs(data.affy.rma)
dim(rma)
```

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.

```
mas.cor <- cor(mas)
mas.cor
             normal1.CEL normal2.CEL normal3.CEL
normal1.CEL   1.0000000   0.7922069   0.7964508
normal2.CEL   0.7922069   1.0000000   0.8918153
normal3.CEL   0.7964508   0.8918153   1.0000000


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

The rma normalization method appears to create a better correlation between the three arrays. The numbers in this correlation matrix are closer to 1, than in the other matrix, therefore they are better correlated.