Differential Gene Expression Analysis with Machine Learning

Yogindra Raghav, Nityam Rathi, Matt Eckelmeyer, Kyle Coleman

Supervisor: Dr. Junshu Bao

November 19, 2018

Goals

1. Identify which genes are differentially expressed after dexamethasone treatment.

   NOTE: The threshold for classifying a gene as "differentially expressed" will be determined based on the values we obtain after tidying the data. The subset tabulation that includes differentially expressed genes will be the first result of our statistical analysis. The term "signal intensity" refers to how strongly a gene is being expressed at a given time.

2. Construct biplot and volcano plot that will allow us to more easily visualize the genes that are classified as differentially expressed at each of the two time points.

3. Construct statistical prediction model (linear regression) using a random subset of the data. We aim to give the model quantitative information to see how accurate predictions from the model are. Residuals will be plotted and the summary statistics of our regression modeling will be the second result of our statistical analysis.

Analysis

Loading packages:

```
library(mdsr) # for data tidying and analysis
library(readr) # for reading in files
```

Loading datasets into variables:

```
hour_0_replicate_1 = read_delim("/Users/MECKELMEYER/Desktop/GSM154262.txt",de
lim = "\t", skip = 9) # load in tab delimited file and skip first 9 lines sin
ce they contain statistical information about Agilent Microarray machine that
is irrelevant for our analysis.

hour_0_replicate_2 = read_delim("/Users/MECKELMEYER/Desktop/GSM154263.txt",de
lim = "\t", skip = 9)

hour_0_replicate_3 = read_delim("/Users/MECKELMEYER/Desktop/GSM154264.txt",de
lim = "\t", skip = 9)

hour_6_replicate_1 = read_delim("/Users/MECKELMEYER/Desktop/GSM154277.txt",de
lim = "\t", skip = 9)

hour_6_replicate_2 = read_delim("/Users/MECKELMEYER/Desktop/GSM154278.txt",de
lim = "\t", skip = 9)

hour_6_replicate_3 = read_delim("/Users/MECKELMEYER/Desktop/GSM154279.txt",de
lim = "\t", skip = 9)
```

Selecting for relevant columns:

```
# GeneName will be used for joining tables of biological replicates.
# gNetSignal gives signal difference between signal intensity and background
intensity, therefore giving true normalized measure of intensity.
# gIsWellAboveBG is for genes whose intensity signal is 2.6 standard deviatio
ns from average signal intensity.

hour_0_replicate_1 = hour_0_replicate_1 %>% select(ProbeName,GeneName, gNetSi
gnal, gIsWellAboveBG)
hour_0_replicate_2 = hour_0_replicate_2 %>% select(ProbeName,GeneName, gNetSi
gnal, gIsWellAboveBG)
hour_0_replicate_3 = hour_0_replicate_3 %>% select(ProbeName,GeneName, gNetSi
gnal, gIsWellAboveBG)
hour_6_replicate_1 = hour_6_replicate_1 %>% select(ProbeName,GeneName, gNetSi
gnal, gIsWellAboveBG)
hour_6_replicate_2 = hour_6_replicate_2 %>% select(ProbeName,GeneName, gNetSi
gnal, gIsWellAboveBG)
hour_6_replicate_3 = hour_6_replicate_3 %>% select(ProbeName,GeneName, gNetSi
gnal, gIsWellAboveBG)
```

Getting rid of “placeholder” gene names:

```
# Certain signal intensities observed by the instrument could not be associat
ed with any known gene. For these genes, the gene name column was given a pla
ceholder name of the molecular probe ID (ProbeName) that was used to target t
he gene.

# First we must figure out what "placeholder" values could be possible for ge
nes by looking at the ProbeName column. ProbeName entries are standardized an
d will have the same possibilities between all data files. For this reason, w
e will go ahead and use the hour_0_replicate_1 table to find the prefixes of
all possible ProbeNames that could be put into the GeneName column.

ProbeName = hour_0_replicate_1$ProbeName

substr(ProbeName, start = 1, stop =2) %>% unique()

## [1] "Br" "(-" "A_" "(+"

# These are possible substrings. Upon further analysis of the "Br" prefix, we
realized the "Br" prefix in ProbeName refers to a control sample called "Brig
htCorner" which we need to get rid of from our data set (assuming it is conta
ined at some point in the GeneName column). Another control GeneName by conve
ntion is "NegativeControl" which we will also filter out.

# Let's filter out control entries:

hour_0_replicate_1 = hour_0_replicate_1 %>% filter(GeneName!="BrightCorner"&
GeneName!="NegativeControl")
hour_0_replicate_2 = hour_0_replicate_2 %>% filter(GeneName!="BrightCorner"&
GeneName!="NegativeControl")
hour_0_replicate_3 = hour_0_replicate_3 %>% filter(GeneName!="BrightCorner"&
GeneName!="NegativeControl")
hour_6_replicate_1 = hour_6_replicate_1 %>% filter(GeneName!="BrightCorner" &
GeneName!="NegativeControl")
hour_6_replicate_2 = hour_6_replicate_2 %>% filter(GeneName!="BrightCorner" &
GeneName!="NegativeControl")
hour_6_replicate_3 = hour_6_replicate_3 %>% filter(GeneName!="BrightCorner" &
GeneName!="NegativeControl")

# Let's filter out possible probe entries that are being used as placeholder
gene names:

hour_0_replicate_1 = hour_0_replicate_1 %>% filter(substr(GeneName, 1,2) != "
(-")
hour_0_replicate_1 = hour_0_replicate_1 %>% filter(substr(GeneName, 1,2) != "
(+")
hour_0_replicate_1 = hour_0_replicate_1 %>% filter(substr(GeneName, 1,2) != "
A_")


hour_0_replicate_2 = hour_0_replicate_2 %>% filter(substr(GeneName, 1,2) != "
(-")
hour_0_replicate_2 = hour_0_replicate_2 %>% filter(substr(GeneName, 1,2) != "
(+")
hour_0_replicate_2 = hour_0_replicate_2 %>% filter(substr(GeneName, 1,2) != "
A_")


hour_0_replicate_3 = hour_0_replicate_3 %>% filter(substr(GeneName, 1,2) != "
(-")
hour_0_replicate_3 = hour_0_replicate_3 %>% filter(substr(GeneName, 1,2) != "
(+")
hour_0_replicate_3 = hour_0_replicate_3 %>% filter(substr(GeneName, 1,2) != "
A_")

hour_6_replicate_1 = hour_6_replicate_1 %>% filter(substr(GeneName, 1,2) != "
(-")
hour_6_replicate_1 = hour_6_replicate_1 %>% filter(substr(GeneName, 1,2) != "
(+")
hour_6_replicate_1 = hour_6_replicate_1 %>% filter(substr(GeneName, 1,2) != "
A_")

hour_6_replicate_2 = hour_6_replicate_2 %>% filter(substr(GeneName, 1,2) != "
(-")
hour_6_replicate_2 = hour_6_replicate_2 %>% filter(substr(GeneName, 1,2) != "
(+")
hour_6_replicate_2 = hour_6_replicate_2 %>% filter(substr(GeneName, 1,2) != "
A_")

hour_6_replicate_3 = hour_6_replicate_3 %>% filter(substr(GeneName, 1,2) != "
(-")
hour_6_replicate_3 = hour_6_replicate_3 %>% filter(substr(GeneName, 1,2) != "
(+")
hour_6_replicate_3 = hour_6_replicate_3 %>% filter(substr(GeneName, 1,2) != "
A_")
```

Averaging Identical Gene Names:

```
# There are rows that contain identical GeneName entries. It is important for
us to average the gNetSignal found for these genes so that we do not consider
the same gene as a different data point.

hour_0_replicate_1 = hour_0_replicate_1 %>% group_by(GeneName) %>% summarise(
gNetSignal = mean(gNetSignal), gIsWellAboveBG = mean(gIsWellAboveBG)) # for e
very gene name this averages gNetSignal and gIsWellAboveBG

nrow(hour_0_replicate_1)

## [1] 27916

hour_0_replicate_2 = hour_0_replicate_2 %>% group_by(GeneName) %>% summarise(
gNetSignal = mean(gNetSignal), gIsWellAboveBG = mean(gIsWellAboveBG))

nrow(hour_0_replicate_2)

## [1] 27916

hour_0_replicate_3 = hour_0_replicate_3 %>% group_by(GeneName) %>% summarise(
gNetSignal = mean(gNetSignal), gIsWellAboveBG = mean(gIsWellAboveBG))

nrow(hour_0_replicate_3)

## [1] 27916

hour_6_replicate_1 = hour_6_replicate_1 %>% group_by(GeneName) %>% summarise(
gNetSignal = mean(gNetSignal), gIsWellAboveBG = mean(gIsWellAboveBG))

nrow(hour_6_replicate_1)

## [1] 27916
```

```
hour_6_replicate_2 = hour_6_replicate_2 %>% group_by(GeneName) %>% summarise(
gNetSignal = mean(gNetSignal), gIsWellAboveBG = mean(gIsWellAboveBG))

nrow(hour_6_replicate_2)
```

```
## [1] 27916
```

```
hour_6_replicate_3 = hour_6_replicate_3 %>% group_by(GeneName) %>% summarise(
gNetSignal = mean(gNetSignal), gIsWellAboveBG = mean(gIsWellAboveBG))

nrow(hour_6_replicate_3)
```

```
## [1] 27916
```

```
# First 5 GeneName entries after summarise are alpha-numeric strings that are
placeholders. As you can see, this data really is messy.

hour_0_replicate_1 = tail(hour_0_replicate_1, -5) # get rid of first five ent
ries.

hour_0_replicate_2 = tail(hour_0_replicate_2, -5)

hour_0_replicate_3 = tail(hour_0_replicate_3, -5)

hour_6_replicate_1 = tail(hour_6_replicate_1, -5)

hour_6_replicate_2 = tail(hour_6_replicate_2, -5)

hour_6_replicate_3 = tail(hour_6_replicate_3, -5)
```

Joining biological replicates:

```
# gNetSignal is normalized value that takes into account changes in machine b
ackground intensity for microarrays. Based on this, we can compare gNetSignal
values between different runs on the same machine. These were done with biolo
gical replicates. For this reason, we are going to average gNetSignal values
across the three replicates for each gene.

hour_0_isoform_a = left_join(hour_0_replicate_1, hour_0_replicate_2, by = "Ge
neName") # we can left_join since we have confirmed with anti_join (not inclu
ded) and nrow() that the tables have the same genes.

hour_0_isoform_a = hour_0_isoform_a %>% left_join(., hour_0_replicate_3, by =
"GeneName") # join it once more

head(hour_0_isoform_a)
```

```
## # A tibble: 6 x 7
##   GeneName gNetSignal.x gIsWellAboveBG.x gNetSignal.y gIsWellAboveBG.y
##   <chr>           <dbl>            <dbl>        <dbl>            <dbl>
## 1 A1BG              599.              1          553.              0.5
## 2 A2BP1              73.7             0.5         78.5             0.5
## 3 A2M               117.              1          164.              1
## 4 A2ML1              94.7             1          111.              1
## 5 A4GALT            130.              1          148.              1
## 6 A4GNT              50.5             0           51.5             0
## # ... with 2 more variables: gNetSignal <dbl>, gIsWellAboveBG <dbl>
```

```
hour_0_isoform_a = hour_0_isoform_a %>% group_by(GeneName) %>% summarise(gNet
Signal = (gNetSignal.x+gNetSignal.y+gNetSignal)/3, gIsWellAboveBG = (gIsWellA
boveBG.x+ gIsWellAboveBG.y + gIsWellAboveBG)/3) #  Average the gNetSignal and
gIsWellAboveBg values between the 3 biological replicates

head(hour_0_isoform_a)
```

```
## # A tibble: 6 x 3
##   GeneName gNetSignal gIsWellAboveBG
##   <chr>         <dbl>          <dbl>
## 1 A1BG          483.           0.667
## 2 A2BP1          73.4          0.5
## 3 A2M           135.           1
## 4 A2ML1         100.           1
## 5 A4GALT        129.           1
## 6 A4GNT          48.7          0
```

```
hour_6_isoform_a = left_join(hour_6_replicate_1, hour_6_replicate_2, by = "Ge
neName")

hour_6_isoform_a = hour_6_isoform_a %>% left_join(., hour_6_replicate_3, by =
"GeneName")

hour_6_isoform_a = hour_6_isoform_a %>% group_by(GeneName) %>% summarise(gNet
Signal = (gNetSignal.x+gNetSignal.y+gNetSignal)/3, gIsWellAboveBG = (gIsWellA
boveBG.x+ gIsWellAboveBG.y + gIsWellAboveBG)/3)

head(hour_6_isoform_a)
```

```
## # A tibble: 6 x 3
##   GeneName gNetSignal gIsWellAboveBG
##   <chr>         <dbl>          <dbl>
## 1 A1BG          476.           0.5
## 2 A2BP1         105.           0.667
## 3 A2M           275.           1
## 4 A2ML1         149.           1
## 5 A4GALT        137.           1
## 6 A4GNT          64.5          0
```

Merging both time points into one table and renaming columns:

```
hour_0_hour_6_isoform_a_combined = hour_0_isoform_a %>% left_join(hour_6_isof
orm_a, by = "GeneName") # join the two tables using "GeneName"
```

```
head(hour_0_hour_6_isoform_a_combined)

## # A tibble: 6 x 5
##   GeneName gNetSignal.x gIsWellAboveBG.x gNetSignal.y gIsWellAboveBG.y
##   <chr>           <dbl>            <dbl>        <dbl>            <dbl>
## 1 A1BG            483.             0.667        476.             0.5
## 2 A2BP1            73.4            0.5          105.             0.667
## 3 A2M             135.             1            275.             1
## 4 A2ML1           100.             1            149.             1
## 5 A4GALT          129.             1            137.             1
## 6 A4GNT            48.7            0             64.5            0

colnames(hour_0_hour_6_isoform_a_combined)[colnames(hour_0_hour_6_isoform_a_c
ombined) == "gNetSignal.x"] <- "Hour_0_gNetSignal"  # change column names

colnames(hour_0_hour_6_isoform_a_combined)[colnames(hour_0_hour_6_isoform_a_c
ombined) == "gNetSignal.y"] <- "Hour_6_gNetSignal"

colnames(hour_0_hour_6_isoform_a_combined)[colnames(hour_0_hour_6_isoform_a_c
ombined) == "gIsWellAboveBG.x"] <- "Hour_0_gIsWellAboveBG"

colnames(hour_0_hour_6_isoform_a_combined)[colnames(hour_0_hour_6_isoform_a_c
ombined) == "gIsWellAboveBG.y"] <- "Hour_6_gIsWellAboveBG"

head(hour_0_hour_6_isoform_a_combined) # show new column names

## # A tibble: 6 x 5
##   GeneName Hour_0_gNetSign… Hour_0_gIsWellA… Hour_6_gNetSign…
##   <chr>               <dbl>            <dbl>            <dbl>
## 1 A1BG                483.             0.667            476.
## 2 A2BP1                73.4            0.5              105.
## 3 A2M                 135.             1                275.
## 4 A2ML1               100.             1                149.
## 5 A4GALT              129.             1                137.
## 6 A4GNT                48.7            0                 64.5
## # ... with 1 more variable: Hour_6_gIsWellAboveBG <dbl>
```

Log Transform gNetSignal values:

```
max(hour_0_hour_6_isoform_a_combined$Hour_0_gNetSignal)

## [1] 209024

min(hour_0_hour_6_isoform_a_combined$Hour_0_gNetSignal)

## [1] 31.70782

max(hour_0_hour_6_isoform_a_combined$Hour_6_gNetSignal)

## [1] 220744.3
```

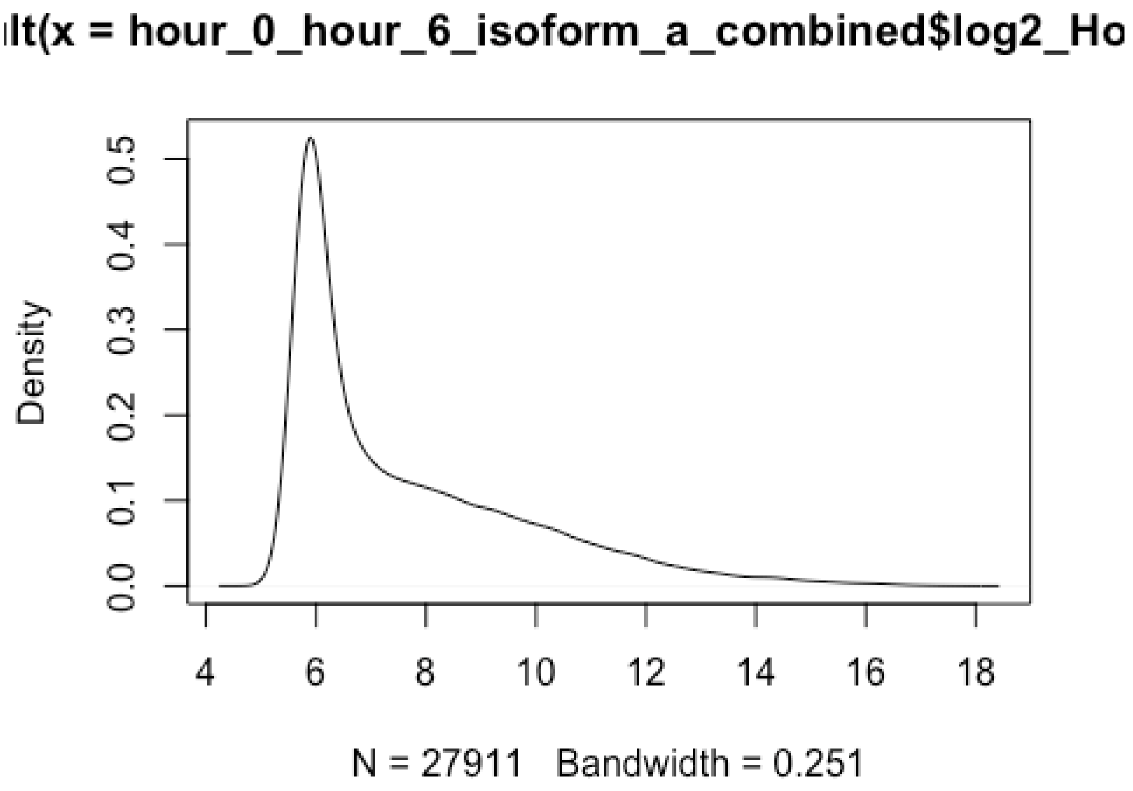
```
min(hour_0_hour_6_isoform_a_combined$Hour_6_gNetSignal)

## [1] 46.26027

# since spread of values is really great for gNetSignal variable, we are goin
g to log transform these columns.

hour_0_hour_6_isoform_a_combined= hour_0_hour_6_isoform_a_combined %>% mutate
(log2_Hour_0_gNetSignal = log2(Hour_0_gNetSignal), log2_Hour_6_gNetSignal = l
og2(Hour_6_gNetSignal)) # log transform these values

head(hour_0_hour_6_isoform_a_combined)

## # A tibble: 6 x 7
##   GeneName Hour_0_gNetSign… Hour_0_gIsWellA… Hour_6_gNetSign…
##   <chr>               <dbl>            <dbl>            <dbl>
## 1 A1BG                483.             0.667            476.
## 2 A2BP1                73.4            0.5              105.
## 3 A2M                 135.             1                275.
## 4 A2ML1               100.             1                149.
## 5 A4GALT              129.             1                137.
## 6 A4GNT                48.7            0                 64.5
## # ... with 3 more variables: Hour_6_gIsWellAboveBG <dbl>,
## #   log2_Hour_0_gNetSignal <dbl>, log2_Hour_6_gNetSignal <dbl>
```

Calculate Signal Difference:

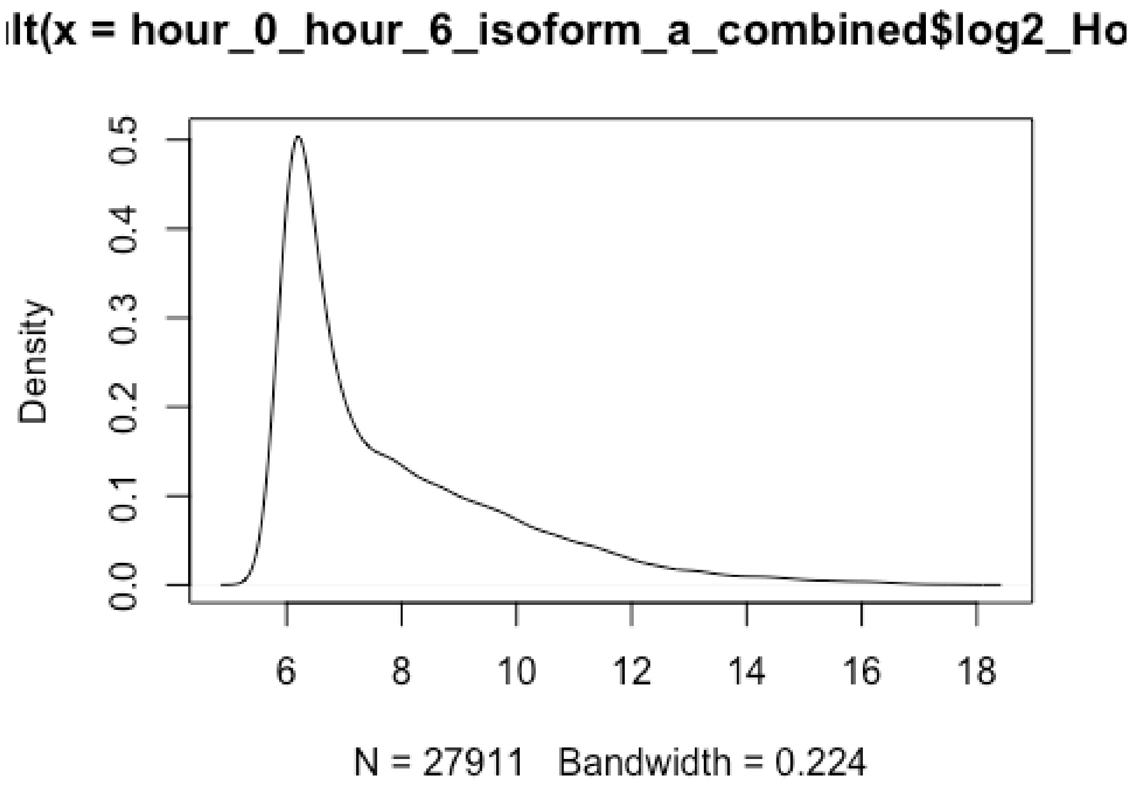
```
# Making a column that contains the difference between the log2 values. We ar
e going to subtract the values from the log normalized values at each time po
int since subtracting logs is equivalent to the fold change. For example, if
log2(later time point) - log2(earlier time point) = 1, then this corresponds
to a 2-fold increase in gene expression between the early and late stages.

hour_0_hour_6_isoform_a_combined= hour_0_hour_6_isoform_a_combined %>% mutate
(difference = log2_Hour_6_gNetSignal - log2_Hour_0_gNetSignal)

head(hour_0_hour_6_isoform_a_combined)

## # A tibble: 6 x 8
##   GeneName Hour_0_gNetSign… Hour_0_gIsWellA… Hour_6_gNetSign…
##   <chr>               <dbl>            <dbl>            <dbl>
## 1 A1BG                483.             0.667            476.
## 2 A2BP1                73.4            0.5              105.
## 3 A2M                 135.             1                275.
## 4 A2ML1               100.             1                149.
## 5 A4GALT              129.             1                137.
## 6 A4GNT                48.7            0                 64.5
## # ... with 4 more variables: Hour_6_gIsWellAboveBG <dbl>,
## #   log2_Hour_0_gNetSignal <dbl>, log2_Hour_6_gNetSignal <dbl>,
## #   difference <dbl>
```

Plotting Density:

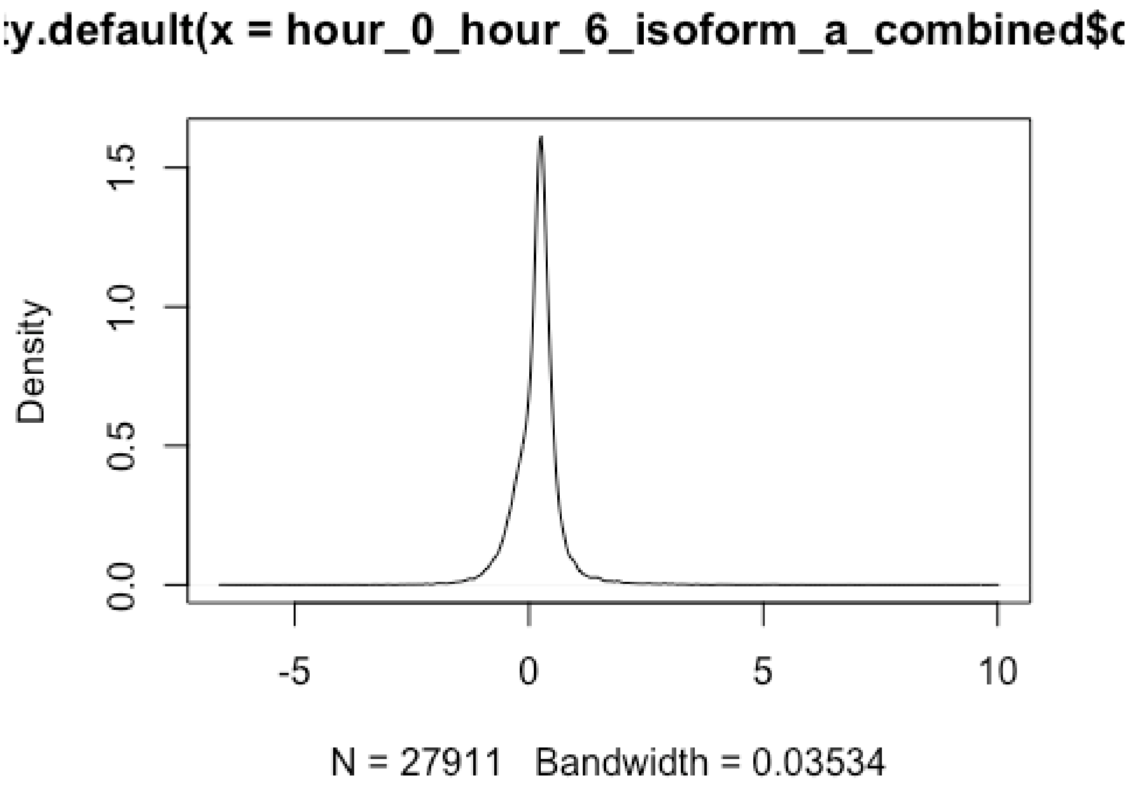
```
# Plotting density to show spread of log transformed gNetSignals as well as d
ifferences.

plot(density(hour_0_hour_6_isoform_a_combined$log2_Hour_0_gNetSignal))
```



```
plot(density(hour_0_hour_6_isoform_a_combined$log2_Hour_6_gNetSignal))
```

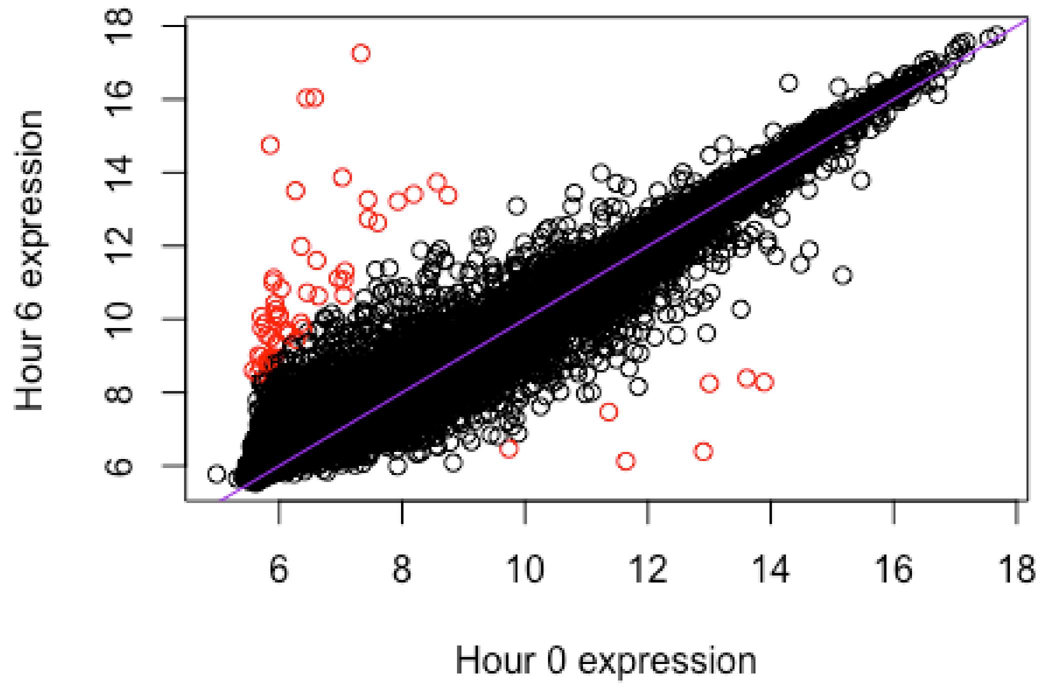


```
plot(density(hour_0_hour_6_isoform_a_combined$difference)) # As seen in this
density plot containing data for both hour 0 and hour 6, the data appears mor
e normalized, indicating that our normalization was successful to at least a
reasonable extent.
```



BiPlot for hour 0 and hour 6 data expression levels:

```
#Plot of hour 0 vs. hour 6 gNetSignals. All points that show a 1.5 fold incre
ase between samples are highlighted in red. We deem these points differential
ly expressed.

plot(hour_0_hour_6_isoform_a_combined$log2_Hour_0_gNetSignal, hour_0_hour_6_i
soform_a_combined$log2_Hour_6_gNetSignal, xlab="Hour 0 expression", ylab="Hou
r 6 expression", col=ifelse(hour_0_hour_6_isoform_a_combined$log2_Hour_0_gNet
Signal/hour_0_hour_6_isoform_a_combined$log2_Hour_6_gNetSignal >= 1.5, "red",
ifelse(hour_0_hour_6_isoform_a_combined$log2_Hour_6_gNetSignal/hour_0_hour_6_
isoform_a_combined$log2_Hour_0_gNetSignal >= 1.5, "red", "black")))
abline(a=0, b=1, col= "purple")
```



This plot shows hour 0 vs hour 6 expression levels for all genes in our data set. We have highlighted "differentially expressed" genes in red. The criteria for being differentially expressed is a fold-difference of 1.5 or more between hour 0 and hour 6 expression levels. Based on this criteria, we ended up with two distinct clusters of differentially expressed genes, those that have significantly higher expression in hour 6 and those that have significantly higher expression in hour 0.

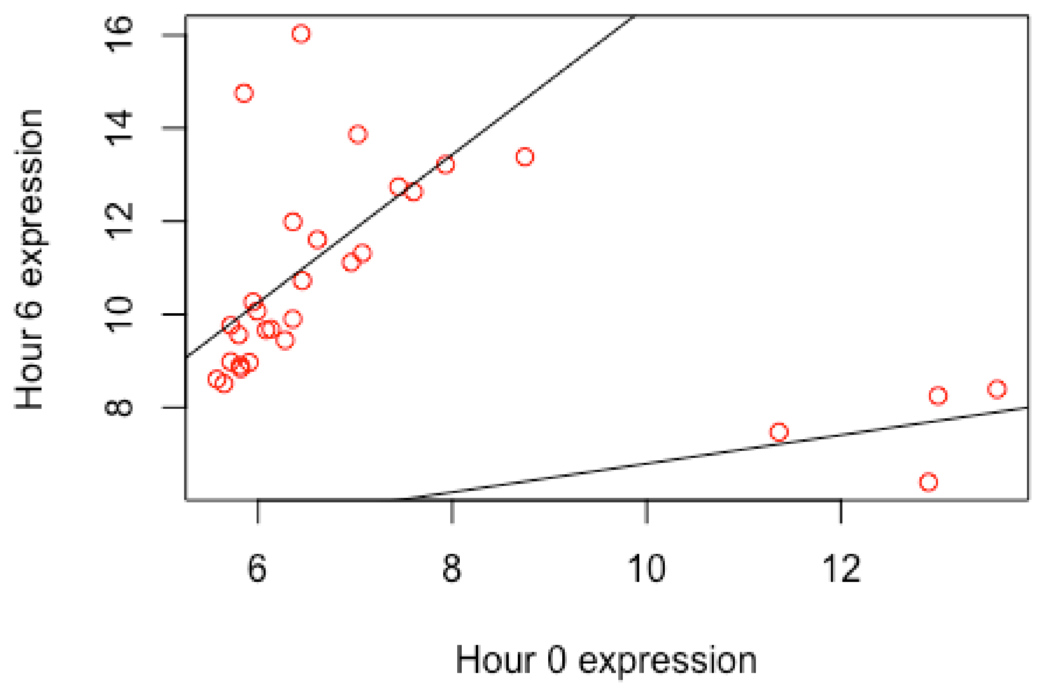
```
#Here we subsetted the original data set to obtain a subset of only different
ially expressed genes (genes that show a 1.5 fold increase between samples).

hour_0_hour_6_isoform_a_combined_diff_expressed = hour_0_hour_6_isoform_a_com
bined[ which (hour_0_hour_6_isoform_a_combined$log2_Hour_0_gNetSignal/hour_0_
hour_6_isoform_a_combined$log2_Hour_6_gNetSignal >= 1.5 | hour_0_hour_6_isofo
rm_a_combined$log2_Hour_6_gNetSignal/hour_0_hour_6_isoform_a_combined$log2_Ho
ur_0_gNetSignal >= 1.5), ]
hour_0_hour_6_isoform_a_combined_diff_expressed

## # A tibble: 60 x 8
##    GeneName Hour_0_gNetSign… Hour_0_gIsWellA… Hour_6_gNetSign…
##    <chr>               <dbl>            <dbl>            <dbl>
##  1 ACSBG1               65.5            0                1819.
##  2 ACTL8                60.2            0                2230.
##  3 ADHFE1               70.3            0.333             822.
##  4 AK027091            174.             1                 6848.
##  5 AK124344            133.             1                 1615.
##  6 AMIGO2             2633.             1                  177.
##  7 ARC                 430.             1                10688.
##  8 ATP4A                63.6            0                 1075.
##  9 BC071773             97.8            1                 3116.
## 10 C1orf172             62.3            0                  646.
## # ... with 50 more rows, and 4 more variables:
## #   Hour_6_gIsWellAboveBG <dbl>, log2_Hour_0_gNetSignal <dbl>,
## #   log2_Hour_6_gNetSignal <dbl>, difference <dbl>
```

```
#This creates a subset of differentially expressed genes which show at least
a 1.5 fold increase in expression at hour 6 compared to hour 0 (positively di
fferentially expressed).

hour_0_hour_6_isoform_a_combined_pos_diff_expressed = hour_0_hour_6_isoform_a
_combined_diff_expressed[ which (hour_0_hour_6_isoform_a_combined_diff_expres
sed$log2_Hour_6_gNetSignal/hour_0_hour_6_isoform_a_combined_diff_expressed$lo
g2_Hour_0_gNetSignal >= 1.5), ]

#This creates a subset of differentially expressed genes which show at least
a 1.5 fold increase in expression at hour 0 compared to hour 6 (negatively di
fferentially expressed).

hour_0_hour_6_isoform_a_combined_neg_diff_expressed = hour_0_hour_6_isoform_a
_combined_diff_expressed[ which (hour_0_hour_6_isoform_a_combined_diff_expres
sed$log2_Hour_0_gNetSignal/hour_0_hour_6_isoform_a_combined_diff_expressed$lo
g2_Hour_6_gNetSignal >= 1.5), ]
```

Statistical Prediction Model:

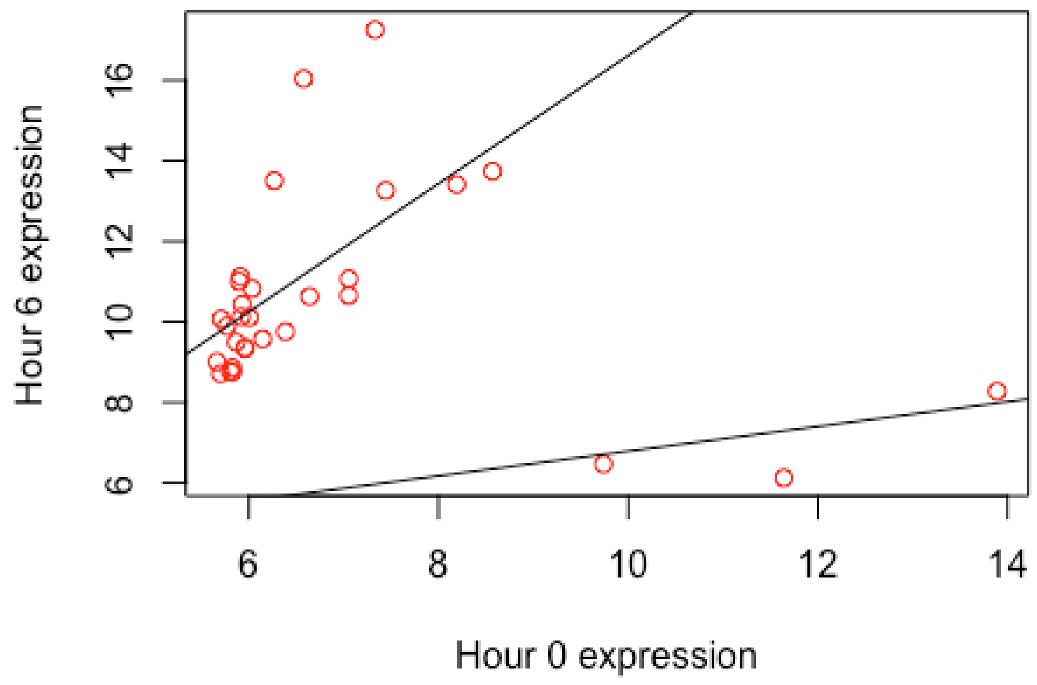
```
#Creating the training and testing data sets for all differentially expressed
genes.

set.seed(99)
train3 <- hour_0_hour_6_isoform_a_combined_diff_expressed %>% sample_frac(siz
e = 0.5)
test3 <- hour_0_hour_6_isoform_a_combined_diff_expressed %>% setdiff(train3)

#Creating the training and testing data sets for positively differentially ex
pressed genes.

set.seed(99)
train1 <- hour_0_hour_6_isoform_a_combined_pos_diff_expressed %>% sample_frac
(size = 0.5)
test1 <- hour_0_hour_6_isoform_a_combined_pos_diff_expressed %>% setdiff(trai
n1)

#Creating the training and testing data sets for negatively differentially ex
pressed genes.
```

```
set.seed(99)
train2 <- hour_0_hour_6_isoform_a_combined_neg_diff_expressed %>% sample_frac
(size = 0.5)
test2 <- hour_0_hour_6_isoform_a_combined_neg_diff_expressed %>% setdiff(trai
n2)

#Combinding both training sets and testing sets for positive and negative dif
ferentially expressed genes.

train_total = rbind(train1, train2)
test_total = rbind(test1, test2)

#Plotting the combined training data for all differentially expressed genes u
sing hour 0 and hour 6 expression levels. We also plotted two regression line
s based on the training data, one for the positive differentially expressed g
enes and one for the negative differentially expressed genes.

plot(train_total$log2_Hour_0_gNetSignal, train_total$log2_Hour_6_gNetSignal,
xlab="Hour 0 expression", ylab="Hour 6 expression", col= "red")
abline(lm(train1$log2_Hour_6_gNetSignal~train1$log2_Hour_0_gNetSignal))
abline(lm(train2$log2_Hour_6_gNetSignal~train2$log2_Hour_0_gNetSignal))
```



This plot shows the combined training data for all differentially expressed genes using hour 0 and hour 6 expression levels. The two plotted regression lines are based on the correlation between hour 6 and hour 0 expression levels from the training data. One of the regression lines is for the positive differentially expressed genes and the other is for the negative differentially expressed genes.

```
#Plotting the combined testing data for all differentially expressed genes us
ing hour 0 and hour 6 expression levels. We used the same regression lines fr
om our training data analysis to show how closely these training set regressi
on lines fit our testing data sets.

plot(test_total$log2_Hour_0_gNetSignal, test_total$log2_Hour_6_gNetSignal, xl
ab="Hour 0 expression", ylab="Hour 6 expression", col= "red")
abline(lm(train1$log2_Hour_6_gNetSignal~train1$log2_Hour_0_gNetSignal))
abline(lm(train2$log2_Hour_6_gNetSignal~train2$log2_Hour_0_gNetSignal))
```



This plot shows the combined testing data for all differentially expressed genes using hour 0 and hour 6 expression levels. The two plotted regression lines are the same lines established from out training data set. Even though these regression lines are not from our testing data set, they fit the testing data very well and appear to have low residuals. This is an indication that our regression lines from our training data are a good fit for positively and negatively differentially expressed genes.

```r
#Getting a summary of the linear model for the positively differentially expr
essed training data set to obtain the regression line coefficents.

lin_mod_1 = lm(train1$log2_Hour_6_gNetSignal~train1$log2_Hour_0_gNetSignal)
lin_mod_1_summary = summary(lin_mod_1)
lin_mod_1_summary$coefficients

##                                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)                   0.7052012  2.7098969 0.2602317 0.7969046790
## train1$log2_Hour_0_gNetSignal 1.5909547  0.4180225 3.8059066 0.0008592068

#Calculating the residuals for the positively differentially expressed test d
ata set using the coefficients from the training data linear model.

test1$residual = test1$log2_Hour_6_gNetSignal - (1.5909547*test1$log2_Hour_0_
gNetSignal + 0.7052012)
test1$residual

##  [1]  0.523021714  1.012960229 -1.267049252 -0.852986963  0.003101034
##  [6]  0.289306939 -0.147785865 -0.903502974 -1.218462016 -0.817464555
## [11]  4.869305895  4.886496429 -0.323510010  2.830078184  0.915570902
## [16]  0.715686701 -1.161789304 -1.107235188 -1.107058026 -0.859712241
## [21] -0.636548342 -0.602070179  0.301870193 -0.703131996  0.016538738
## [26] -0.540547117 -1.057568485

#Getting a summary of the linear model for the negatively differentially expr
essed training data set to obtain the regression line coefficents.

lin_mod_2 = lm(train2$log2_Hour_6_gNetSignal~train2$log2_Hour_0_gNetSignal)
lin_mod_2_summary = summary(lin_mod_2)
lin_mod_2_summary$coefficients

##                                Estimate Std. Error   t value  Pr(>|t|)
## (Intercept)                   3.7224380  8.1896170 0.4545314 0.6940136
## train2$log2_Hour_0_gNetSignal 0.3069249  0.6425664 0.4776548 0.6800062

#Calculating the residuals for the negatively differentially expressed test d
ata set using the coefficients from the training data linear model.

test2$residual = test2$log2_Hour_6_gNetSignal - (0.3069249*test2$log2_Hour_0_
gNetSignal + 3.7224380)
test2$residual

## [1] -0.2430844  0.2946504 -1.1679804

#Calculating SSE for the residuals from the combined differentially expressed
genes to determine how well our test data fits our training linear model.

test_total = rbind(test1, test2)
SSE = sum((test_total$residual)^2)
SSE
```

```
## [1] 72.57448
```

Appendix

Group Member's Contribution:

All group members were greatly involved in devising and planning this project, which involved: reading the scientific literature related to the data sets utilized in this study, identifying key points of analysis, and formulating a plan.

Nityam Rathi and Yogi Raghav initially mined through the raw datasets (GSE6711) and determined which data sets to use to fulfill our goal. Nityam and Yogi then read pertinent literature on microarray data to identify which variables in the massive data sets could be used to calculate differential gene expression and for subsequent statistical sets. After cutting the data sets to include only our variables of interest (gNetSignal and gIsWellAboveBG), Nityam and Yogi performed data wrangling and appropriate normalization techniques to create a final data set upon which the density plots are shown. This abridged dataset was then used for the differential gene expression analysis and statistal learning/regression models by Kyle and Matt. Nityam and Yogi provided necessary instruction and guidance for subsequent data visualization and statistical learning tests.

Matt and Kyle subsetted the data into positively, negatively, and overall differentialluy expressed genes. Matt and Kyle created the expression biplot for hour 0 and hour 6 expression levels to visualize differentially expressed genes. Matt and Kyle also split the data into training and testing data sets for development of a statistical learning model. From this, Matt and Kyle developed a plot and two regression lines for the differentially expressed genes from the training data. They then plotted the testing data with the training data regression lines to see how well these lines fit the testing data. Matt and Kyle then calculated the residuals and SSE for the testing observations relative to the respective training regression lines and found a relatively low SSE, indicating that the model was a good fit.

Files Used for Group Project

GSM154262.txt

GSM154263.txt

GSM154264.txt

GSM154277.txt

GSM154278.txt

GSM154279.txt