# 36-669 HW5

```r
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomicRanges
## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
library(hexbin)
library(vsn)
library(pheatmap)
library(SummarizedExperiment)
library(genefilter)
```

```
##
## Attaching package: 'genefilter'

## The following objects are masked from 'package:MatrixGenerics':
##
##     rowSds, rowVars

## The following objects are masked from 'package:matrixStats':
##
##     rowSds, rowVars
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse()   masks IRanges::collapse()
## x dplyr::combine()    masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count()      masks matrixStats::count()
## x dplyr::desc()       masks IRanges::desc()
## x tidyr::expand()     masks S4Vectors::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks S4Vectors::first()
## x dplyr::lag()        masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()     masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()     masks S4Vectors::rename()
## x dplyr::slice()      masks IRanges::slice()
## x readr::spec()       masks genefilter::spec()
```

```
load(url("https://github.com/xuranw/469_public/raw/main/hw5/airway.RData"))
```

# Question 1

## 1.A

```
class(dds)
```

```
## [1] "DESeqDataSet"
## attr(,"package")
## [1] "DESeq2"
```

```r
SummarizedExperiment::colData(dds)
```

```
## DataFrame with 8 rows and 5 columns
##                 names    donor     condition     cell      dex
##              <factor> <factor>      <factor> <factor> <factor>
## SRR1039508 SRR1039508  N61311  Untreated      N61311     untrt
## SRR1039509 SRR1039509  N61311  Dexamethasone  N61311     trt
## SRR1039512 SRR1039512  N052611 Untreated      N052611    untrt
## SRR1039513 SRR1039513  N052611 Dexamethasone  N052611    trt
## SRR1039516 SRR1039516  N080611 Untreated      N080611    untrt
## SRR1039517 SRR1039517  N080611 Dexamethasone  N080611    trt
## SRR1039520 SRR1039520  N061011 Untreated      N061011    untrt
## SRR1039521 SRR1039521  N061011 Dexamethasone  N061011    trt
```

```r
count_dds <- SummarizedExperiment::assay(dds)
class(count_dds) # matrix/array
```

```
## [1] "matrix" "array"
```

```r
dim(count_dds) # 17213 rows of genes, 8 cols of samples
```

```
## [1] 17213     8
```

```r
colnames(count_dds) # names of the samples
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```r
head(rownames(count_dds)) # first 6 genes
```

```
## [1] "ENSG00000000003.14" "ENSG00000000419.12" "ENSG00000000457.13"
## [4] "ENSG00000000460.16" "ENSG00000000971.15" "ENSG00000001036.13"
```

```r
head(count_dds) # first 6 rows of the 8 samples
```

```
##                    SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003.14        708        468        901        424       1188
## ENSG00000000419.12        455        510        604        352        583
## ENSG00000000457.13        312        268        363        224        361
## ENSG00000000460.16         88         74         52         45        107
## ENSG00000000971.15       3228       3655       6066       4210       6636
## ENSG00000001036.13       2298       1630       2682       1357       2209
##                    SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003.14       1091        806        599
## ENSG00000000419.12        774        410        499
## ENSG00000000457.13        430        300        289
## ENSG00000000460.16        101         97         83
## ENSG00000000971.15      10855       5032       7771
## ENSG00000001036.13       2214       2115       1736
```

```r
colData(dds)
```

```
## DataFrame with 8 rows and 5 columns
##                 names    donor     condition     cell      dex
##              <factor> <factor>      <factor> <factor> <factor>
## SRR1039508 SRR1039508  N61311  Untreated      N61311     untrt
## SRR1039509 SRR1039509  N61311  Dexamethasone  N61311     trt
## SRR1039512 SRR1039512  N052611 Untreated      N052611    untrt
## SRR1039513 SRR1039513  N052611 Dexamethasone  N052611    trt
## SRR1039516 SRR1039516  N080611 Untreated      N080611    untrt
## SRR1039517 SRR1039517  N080611 Dexamethasone  N080611    trt
## SRR1039520 SRR1039520  N061011 Untreated      N061011    untrt
## SRR1039521 SRR1039521  N061011 Dexamethasone  N061011    trt
```

#### 1.A.1

The matrix stores the count of how often a specific gene expression appeared in a given sample.

#### 1.A.2

The output displays the sample name, the donor/cell (factor variables), and the condition that was applied to the cell. In each sample, a treated cell was treated with Dexamethasone or it did not receive treatment.

#### 1.A.3

The gene symbol for ENSG00000012048 is BRCA1, which is the gene used for determining the likelihood a women will develop early onset breast cancer.

## 1.B

```r
dds <- DESeq2::DESeq(dds)
```

```
## estimating size factors

## using 'avgTxLength' from assays(dds), correcting for library size

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

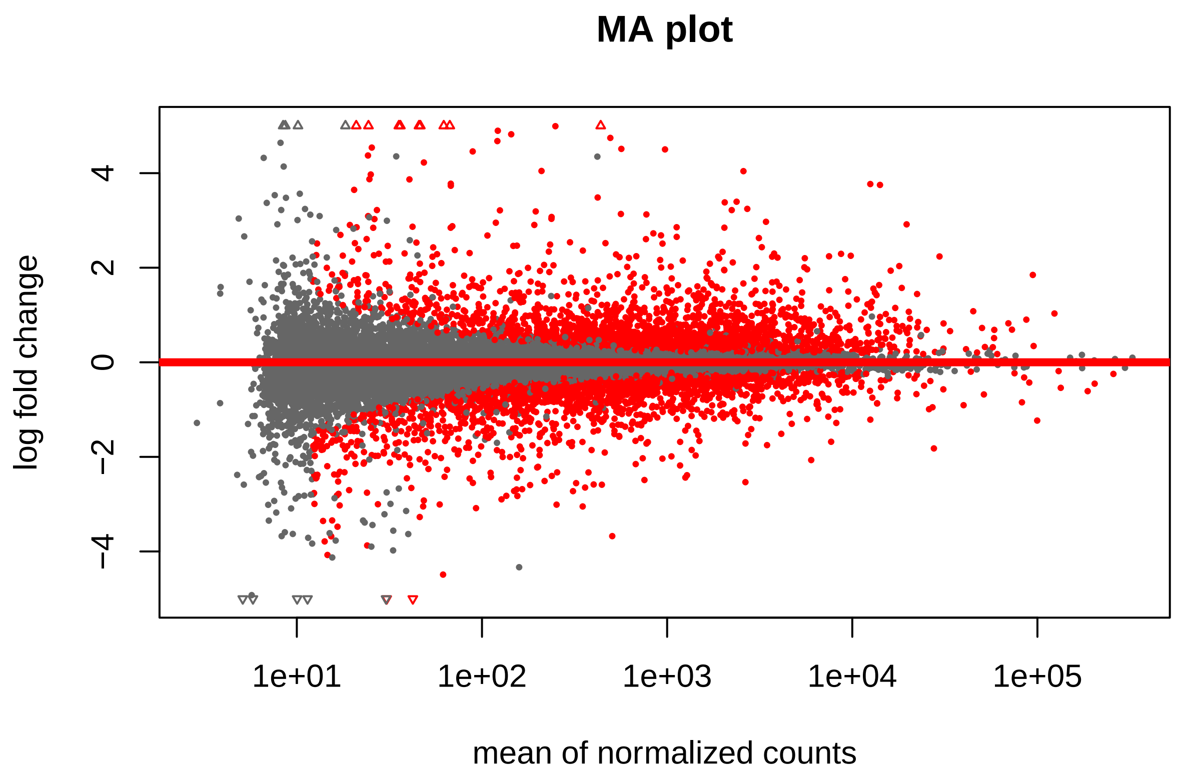
```r
res <- DESeq2::results(dds)
res
```

```
## log2 fold change (MLE): dex trt vs untrt
## Wald test p-value: dex trt vs untrt
## DataFrame with 17213 rows and 6 columns
##                     baseMean log2FoldChange     lfcSE      stat      pvalue
##                    <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003.14  740.0279     -0.3640119 0.1063885 -3.421534 6.22688e-04
## ENSG00000000419.12  511.7064      0.2035448 0.1274540  1.597006 1.10264e-01
## ENSG00000000457.13  314.1800      0.0350979 0.1556850  0.225442 8.21636e-01
## ENSG00000000460.16   79.8027     -0.1188394 0.3091822 -0.384367 7.00707e-01
## ENSG00000000971.15 5715.4351      0.4443182 0.0891291  4.985108 6.19271e-07
## ...                      ...            ...       ...       ...         ...
## ENSG00000285953.1    29.5674      -1.926449  0.662795 -2.906553  0.00365435
## ENSG00000285967.1   181.1566      -0.324601  0.179797 -1.805377  0.07101578
## ENSG00000285976.1   875.4883       0.263452  0.143474  1.836232  0.06632334
## ENSG00000285979.1    38.3528       0.339672  0.350635  0.968735  0.33267736
## ENSG00000285991.1    11.2795      -0.115622  0.731278 -0.158110  0.87437017
##                           padj
##                      <numeric>
## ENSG00000000003.14 4.34023e-03
## ENSG00000000419.12 2.80439e-01
## ENSG00000000457.13 9.17629e-01
## ENSG00000000460.16 8.51654e-01
## ENSG00000000971.15 9.10444e-06
## ...                        ...
## ENSG00000285953.1     0.019530
## ENSG00000285967.1     0.204421
## ENSG00000285976.1     0.194274
## ENSG00000285979.1     0.570883
## ENSG00000285991.1           NA
```

DESeq uses the generalized linear model with a negative binomial distribution to conduct differential expression analysis on the sequencing data. The first step of the analysis is estimation of the size factors for each gene, which is important because the size factors are utilized to calculated the fitted mean for a given sample, where the mean is a parameter in the binomial distribution. The second step of the analysis is an estimation of dispersion, which is important for the analysis because it is one of the parameters for the negative binomial distribution that is used to calculate the variance and account for additional variability in the data as the mean changes. The third step of the analysis is fitting the Negative Binomial GLM and Wald statistics, which is performed from the results obtained in the first two steps of the analysis. This allows us to draw inferential conclusions based on the beta coefficients in the model to evaluate the log fold change between populations accounting for additional variability in the data.

## 1.C

```r
DESeq2::plotMA(res, ylim = c(-5, 5), main = "MA plot", colSig = "red", colNonSig = "gray40", colLine =
```



### 1.C.1

Each point represents the count of a specific gene, and the triangles represent points classified as outliers. The color represents whether or not a point is statistically significant at the 10% level by using MA to estimate their per gene variance and account for smaller counts of genes by shrinking them toward 0. Points that are highlighted gray have p-values greater than or equal to 0.1 based on their log fold change, and red points have a p-value less than 0.1.

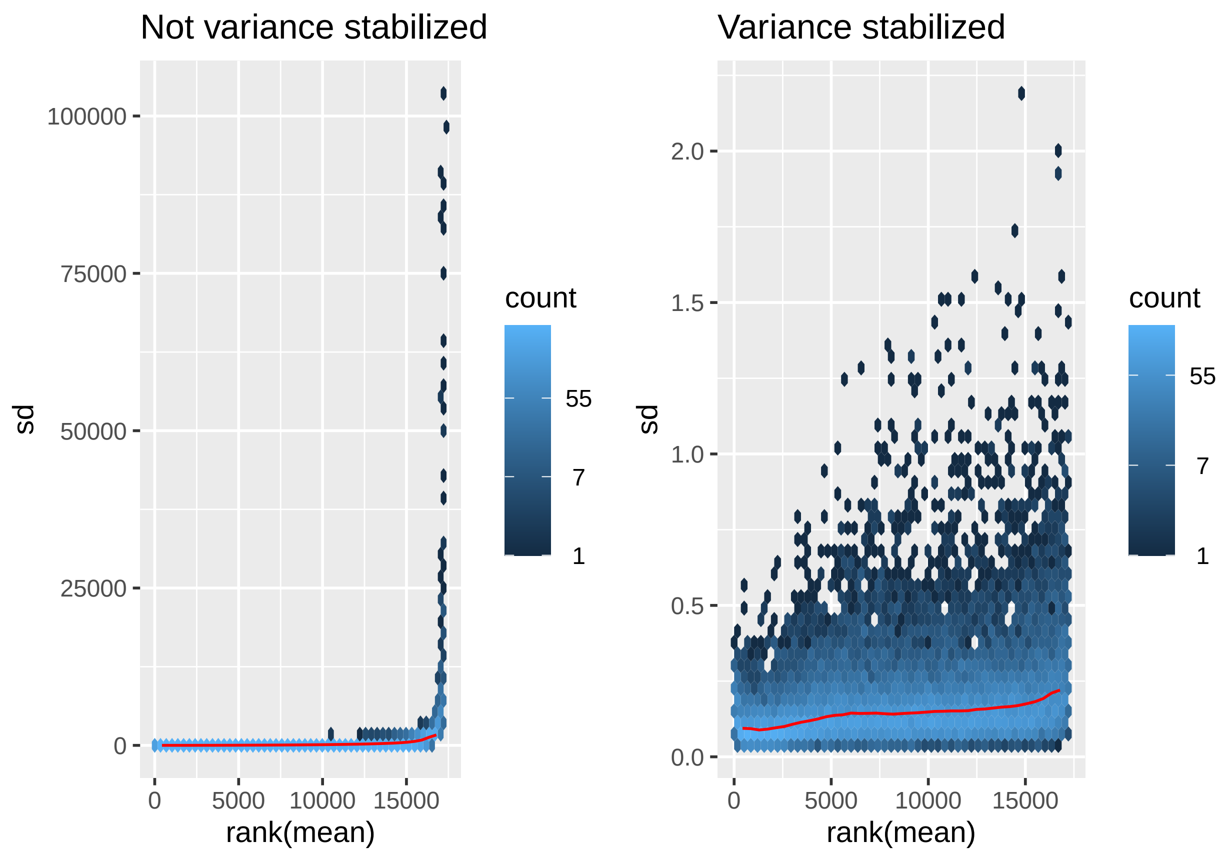
### 1.C.2

The X-axis represents the mean expression of the normalized counts in the case-control comparison for each gene in the analysis, and the Y-axis represents the log fold change of these counts when comparing treated to untreated. Larger fold changes have a greater distance from 0 and are more likely to be statistically significant.

## 1.D

```r
vsd <- DESeq2::vst(dds, blind = FALSE)
count_vsd <- SummarizedExperiment::assay(vsd)
dds_plot <- vsn::meanSdPlot(count_dds, plot = F)
dds_final <- dds_plot$gg + labs(title = "Not variance stabilized")
vsd_plot <- vsn::meanSdPlot(count_vsd, plot = F)
```

```r
vsd_final <- vsd_plot$gg + labs(title = "Variance stabilized")
gridExtra::grid.arrange(dds_final, vsd_final, nrow = 1)
```



### 1.D.1

It would be less desirable because the unstabilized variance are extremely skewed data with large standard deviation values (violating the assumptions of the residuals need for regression modeling) and since hypotheses testing requires us to divide by standard deviation, inflated or unstable standard deviations would potentially mask statistically significant results. Only the genes with the largest changes changes could still be statistically significant, and we may not detect other significant results that would have been captured if the variance were stabilized.

### 1.D.2

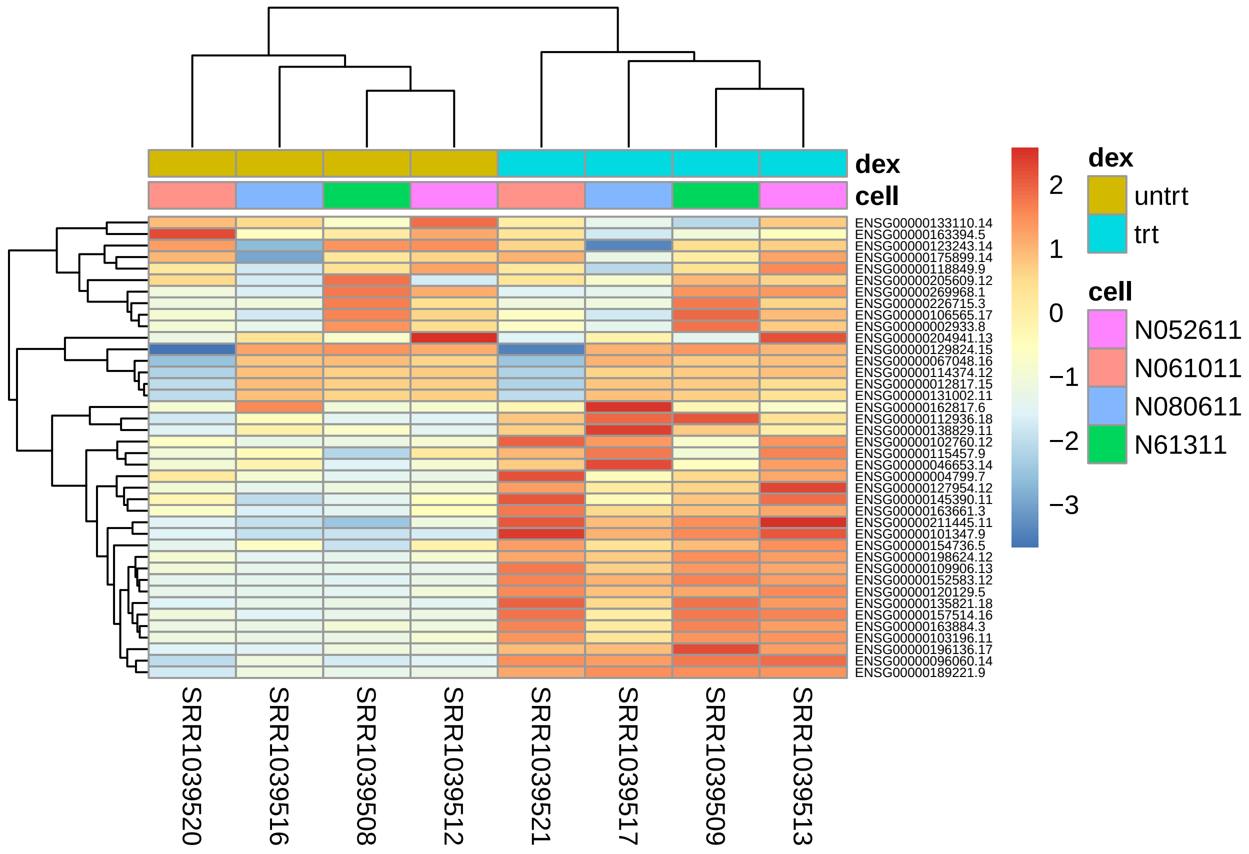
The function creates a visualization of a scatter plot with the row-wise mean on the X-axis and standard deviation on the Y-axis of the matrix inputted into the function for the gene expressions, allowing us to visualize if there is a dependence between the mean and variance. The plane of graphic is divided into regular hexagons, and the number of cases are counted within those hexagons. How many observations lie within a given hexagon determine its count, where lighter shades of blue imply more observations.

### 1.D.3

The right plot looks more variance-stabilized since the range of values for the standard deviations is much smaller (0, 2.5) than the range of values for the unstabilized variances (0, 10,000). The spread of the errors is more uniform and more closely appears to be random.

## 1.E

```r
topVarGenes <- head(order(genefilter::rowVars(count_vsd),
decreasing = TRUE), 40)
dat <- count_vsd[topVarGenes,]
dat <- dat - rowMeans(dat)
anno <- as.data.frame(SummarizedExperiment::colData(vsd)[, c("cell","dex")])
pheatmap::pheatmap(dat, annotation_col = anno, fontsize_row = 5)
```



### 1.E.1

The top x genes (i.e. top 40) are selected by taking highest variances for the genes in descending order, which is calculated by taking the variance of a given row.

### 1.E.2

The columns in the figure correspond to whether a given sample received treatment or not, and the columns are grouped by treated and untreated and then by the cell of the donor, with the sample number listed at the bottom. The rows correspond to a gene in a given experiment, and color of a cell is based on how different the variance of the gene in a given study is away from the mean of the expression of the gene across all samples. Clustering of the genes is based on how similarly they expression is in each sample, and samples are clustered together based how similarly their genes are expressed.

### 1.E.3

Examining the graphic, we see that the cells clustered together for top half of the rows (ENSG00000133110.14 - ENSG00000131002.11) have roughly similar values regardless of whether they are in the treatment or control group for a given study. However, the bottom half of the rows (ENSG00000162817.6 and lower) have different (redder) colors, and therefore more differential gene expression, for the treated compared to the control groups for a given study, suggesting that these genes are more affected by the treatment.

## Question 2

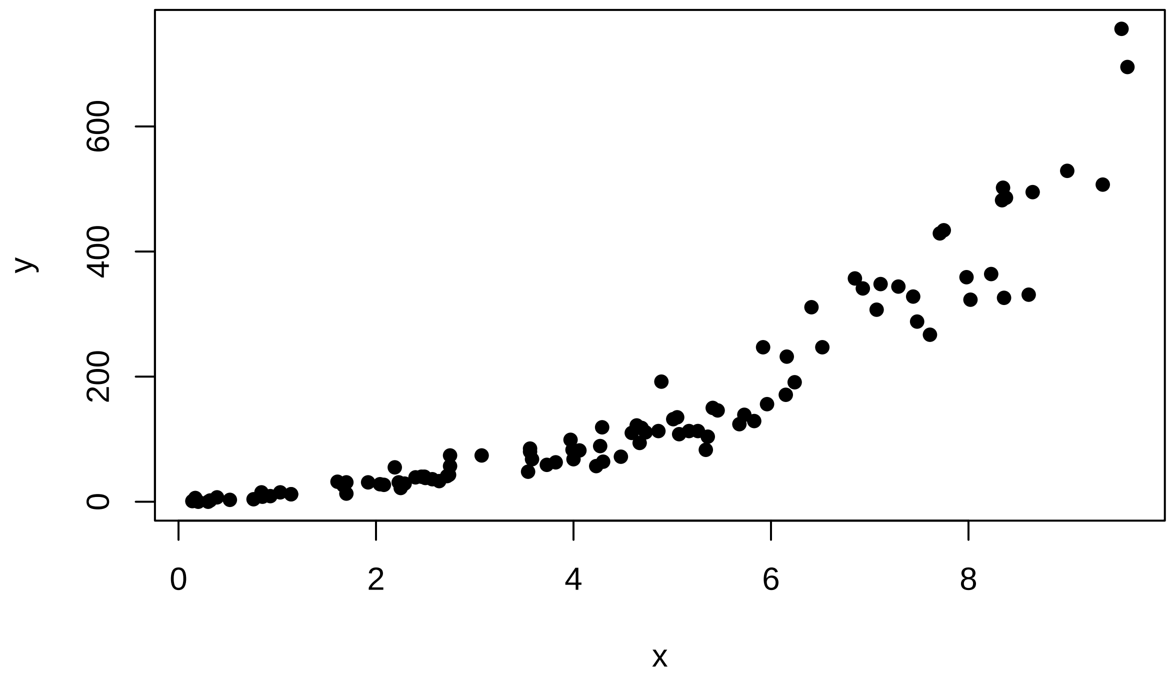
```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## The following object is masked from 'package:genefilter':
##
##     area
```

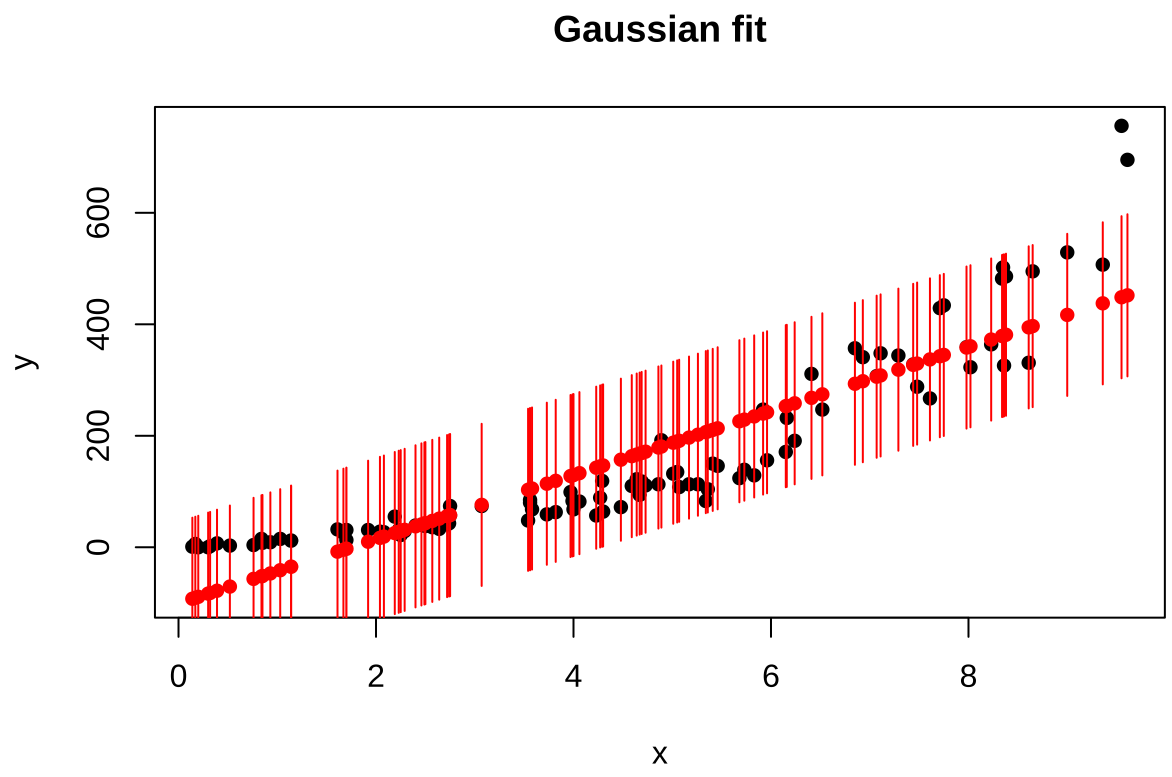
```
dat <- read.csv("https://raw.githubusercontent.com/xuranw/469_public/master/hw5/synthetic.csv")
plot(dat$x, dat$y, pch = 16, xlab = "x", ylab = "y")
```



**2.A**

```
fit_lm <- stats::lm(y ~ ., data = dat)
pred_lm <- stats::predict(fit_lm, newdata = dat, type = "response")
est_sd <- summary(fit_lm)$sigma

plot(dat$x, dat$y, pch = 16, ylim = range(c(pred_lm, dat$y)),
main = "Gaussian fit", xlab = "x", ylab = "y")
points(dat$x, pred_lm, col = "red", pch = 16, xlab = "x", ylab =
"y")
for(i in 1:length(dat$x)){
lines(rep(dat$x[i], 2), c(pred_lm[i]-2*est_sd, pred_lm[i]+2*est_sd), col = "red")
}
```

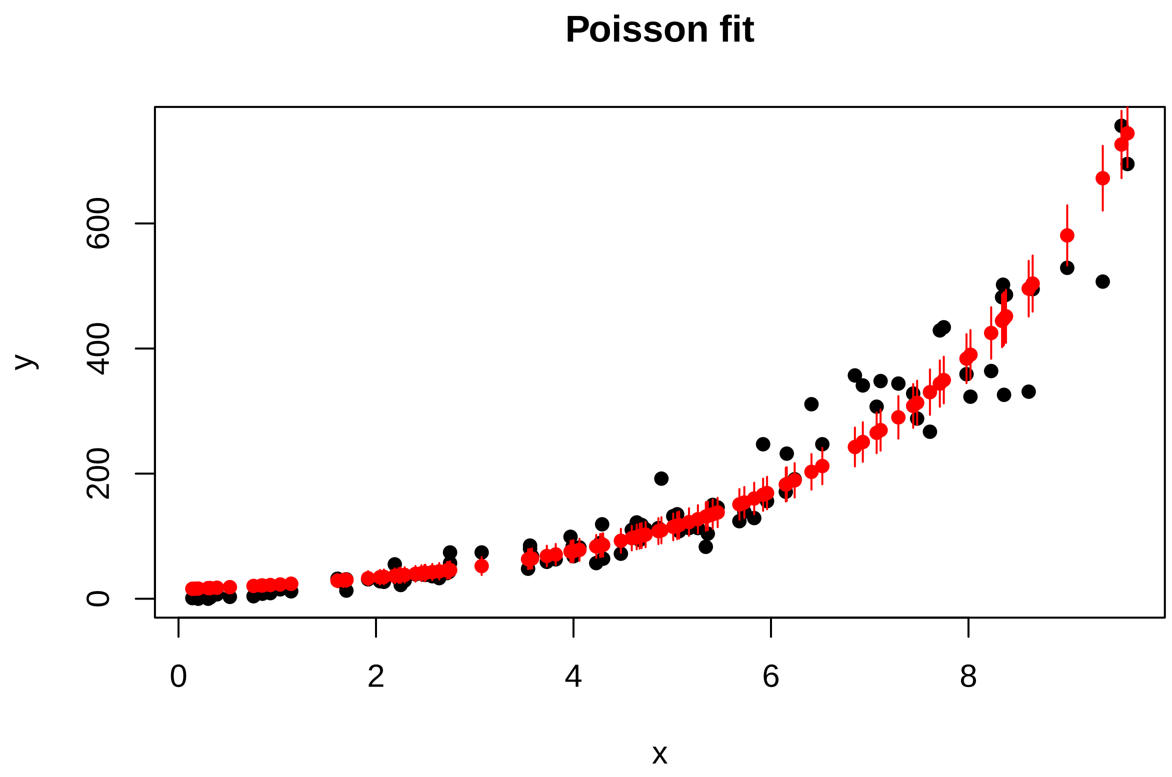


The linear regression fit would not be appropriate because there is not a linear relationship between x and y. There is clearly a pattern in the residuals based on the figure, which violates the constant variance assumption for a linear regression model. That is, we would expect not to see a pattern in the residuals and the majority of the values would like within 2 standard deviations of the mean of the residuals, which is 0.

## 2.B

```
fit_poisson <- stats::glm(y ~ ., data = dat, family = "poisson")
pred_poisson <- stats::predict(fit_poisson, newdata = dat, type = "response")
est_sd_poisson <- sqrt(pred_poisson)

plot(dat$x, dat$y, pch = 16, ylim = range(c(pred_poisson, dat$y)), main = "Poisson fit", xlab = "x", yla
points(dat$x, pred_poisson, col = "red", pch = 16, xlab = "x", ylab = "y")
for (i in 1:length(dat$x)){
lines(rep(dat$x[i], 2), c(pred_poisson[i] - 2 * est_sd_poisson[i], pred_poisson[i] + 2 * est_sd_poisson
}
```

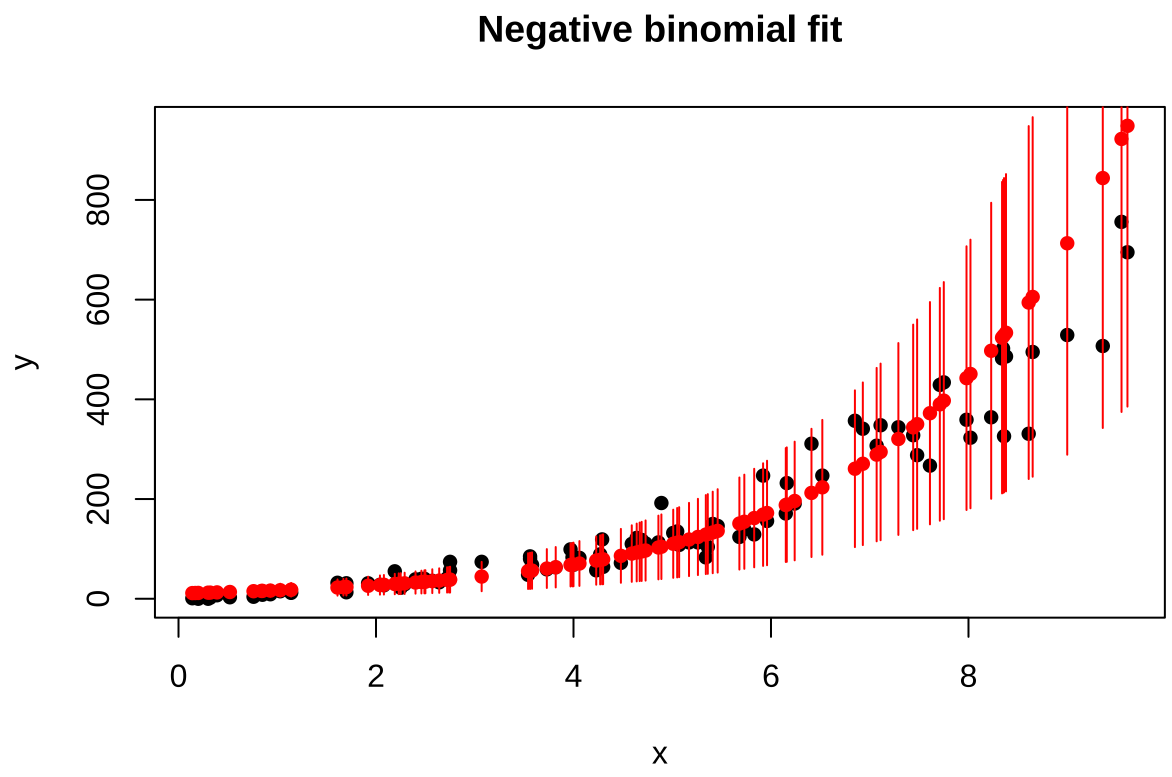


The Poisson regression fit is more appropriate for the data since y appears to be exponentially related to x instead of linearly related. In the Poisson regression model, the mean and the variance are assumed to be equal and so they change together for a given value of x rather than assuming a constant change in y for a given value of x. Note that while the mean of the Poisson regression fit appears to be at the center of the data, most of the observations lie outside the ±2 standard deviation interval from the mean, suggesting that the data are overdispersed.

## 2.C

```
fit_nb <- MASS::glm.nb(y ~ ., data = dat)
pred_nb <- stats::predict(fit_nb, newdata = dat, type = "response")
est_sd_nb <- sqrt(pred_nb + I(pred_nb^2)/summary(fit_nb)$theta)

plot(dat$x, dat$y, pch = 16, ylim = range(c(pred_nb, dat$y)), main = "Negative binomial fit", xlab = "x
points(dat$x, pred_nb, col = "red", pch = 16, xlab = "x", ylab = "y")
for (i in 1:length(dat$x)){
lines(rep(dat$x[i], 2), c(pred_nb[i] - 2 * est_sd_nb[i], pred_nb[i] + 2 * est_sd_nb[i]), col = "red")
}
```



The Negative Binomial regression fit is more appropriate for the data than the Poisson fit because it better accounts for the overdispersion in the data. While the mean for both models is approximately equal, the variance in the Negative Binomial model accounts for overdispersion since the variance is a function of the mean plus the squared mean divided by an overdispersion parameter. Therefore, the the $\pm 2$ standard deviation interval from the mean will better account for the data in the model and overcome overdispersion. This is important because accounting for overdispersion will result in more larger and accurate estimates for the standard errors for the coefficient estimates and help us to avoid drawing inappropriate inferences from the data based on the statistical significance of these estimates.