HW1

Harpreet Sumal - 1002353099 - sumalhar

05/02/2020

Contents

Q1. Conceptual Challenges (30 Points)

(1) Let $\{X_i\}_{i=1}^n \overset{i.i.d}{\sim} p_\theta(x)$ be n random samples (Let X be their population variable). We denote their realizations (or outcomes) to be $\{x_i\}_{i=1}^n$. Select all the WRONG statements:

(2) Unbiasedness and Consistency. Select all the wrong statements:

(3) Law of Large Numbers (LLN) and Central Limit Theorem (CLT). Select all the WRONG statements:

(4) Linear Regression and Ordinary Least Squares (OLS). Select all the WRONG statements:

(5) Assuming the true distribution of the data follows a linear model $Y = \beta_0 + \beta_1 X + \epsilon$. We fit an ordinary least squares regression using this true model on the data, as the number of data points goes to infinite, your estimator will have:

Answers:

(1) = A, C

(2) = A

(3) = B

(4) = A

(5) = D, E

(6) = E

Q2. Short Question (15 Points)

Prove or Give Counter Examples:

1. If $X_n \xrightarrow{P} X$ and $Y_n \xrightarrow{P} Y$, then $X_n + Y_n \xrightarrow{P} X + Y$
2. If $X_n \xrightarrow{D} X$ and $Y_n \xrightarrow{D} Y$, then $X_n + Y_n \xrightarrow{D} X + Y$
3. If $X_n \xrightarrow{D} C$ for some constant C, then $X_n \xrightarrow{P} C$

Answers:

1. $|(X_n + Y_n) - (X + Y)| \leq |X_n - X| + |Y_n - Y|$

   If $|(X_n + Y_n) - (X + Y)| \geq \epsilon$ then, one of $|X_n - X|$ or $|Y_n - Y|$ must be $\geq \frac{\epsilon}{2}$

   $P(|(X_n + Y_n) - (X + Y)| \geq \epsilon)$

   $\leq P(|X_n - X| \geq \frac{\epsilon}{2}) + P(|Y_n - Y| \geq \frac{\epsilon}{2})$,

   Remark: '+' sign signifies logical or, as we stated above, only one of these terms needs to hold true.

   $= 0$ as $X_n, Y_n \overset{P}{\to} X, Y$ respectively.

   Therefore, $|(X_n + Y_n) - (X + Y)| \geq \epsilon \implies 0 \geq 0 \implies$ our triangle inequality, and hence the property holds. ■

2. Let X, Y be independent $N(0,1)$ random variables with $X_n \sim N(0,1)$, $Y_n \sim X_n$

   This implies $X_n \overset{D}{\to} X, Y_n \overset{D}{\to} Y$ but $(X_n, Y_n) \overset{D}{\to} (X, X_n)$ ■

3. $X_n \overset{D}{\to} X$ iff $\lim_{n\to\infty} F_{X_n}(x) = F(x)$

   We have: $X_n \overset{D}{\to} C$ which is true iff $\lim_{n\to\infty} F_{X_n}(x) = \delta(x > C)$ at all $x \neq C$ where $\delta_C(t) = 0$ if $x < C$ and $\delta_C(t) = 1$ if $x > C$ then $P(|X_n - C|) > \epsilon$

   Remark: This is from the Convergence of Probability definition

   $= 1 - P(|X_n - C| \leq \epsilon)$

   $= 1 - P(C - \epsilon \leq X_n \leq C + \epsilon) = 1 - (F_{X_n}(C + \epsilon) - F_{X_n}(C - \epsilon))$

   $\overset{n\to\infty}{\longrightarrow} 1 - (1 - 0)$

   $= 0$

   Therefore X_n converges in probability to C when it converges to C in distribution.■

Q3. Maximum Likelihood Estimator (MLE) and Asymptotic Normality (20 Points)

Part 1 Answer:

We know $\hat{\theta}_n \overset{P}{\to} \theta$ by the definition given in Part 1.

Furthermore, by definition, $\theta \in C_n$ and so, let $C_n = I(\theta)$

Finally, $\sqrt{n}(\hat{\theta}_n - \theta) \overset{D}{\to} N(0, \frac{1}{I(\theta)})$

Definition of Fisher Information function implies that:

$E_\theta(I'_n(\theta)) = 0$ and $Var_\theta(I'_n(\theta)) = I_n(\theta)$

Using Slutsky's Theorem we get:

$\hat{\theta}_n \overset{P}{\to} \theta$ and,

$\sqrt{n}(\hat{\theta}_n - \theta) \overset{D}{\to} N(0, \frac{1}{I(\theta)})$

as the two pieces for Slutsky's Theorem.

Using Slutsky's a second time we get the following end result:

$= \sqrt{nI(\hat{\theta})}\,(\hat{\theta}_n - \theta) \;\overset{D}{\to}\; I(\theta)\;\; N(0, \frac{1}{I(\theta)})$

$\Leftrightarrow \sqrt{nI(\hat{\theta})}\,(\hat{\theta}_n - \theta) \xrightarrow{D} N(0,1)$ as wanted ■

Part 2 Answers:

A.

$L(x) = \prod_{i=1}^{n} p(x_i)$ is the likelihood function we need to get in order to find the $MLE = max(L(x))$

Therefore let us find the $log\, L(x)$ for $p(x) = (\theta - 1)x^{-\theta}$

$L(\theta) = \sum_{i=1}^{n} ln(\theta - 1)x^{-\theta}$

$L(\theta) = -\ \sum_{i=1}^{n} ln(\theta) + \sum_{i=1}^{n} ln(\theta - 1)x_i + \sum_{i=1}^{n} ln(x_i)$

$L(\theta) = -nln(\theta) + \sum_{i=1}^{n} ln(\theta - 1)x_i + \sum_{i=1}^{n} ln(x_i)$

Taking the derivative with respect to θ we get

$L'(\theta) = -\frac{n}{\theta} + \sum_{i=1}^{n} ln(x_i)$

$\hat{\theta} = -\frac{n}{\sum_{i=1}^{n} x_i}$

B.

The Fisher Information, $I(\hat{\theta}_n)$ is the asymptotic variance

Therefore we must calculate:

$I(\hat{\theta}_n) = -\int (L''(\theta))\, L(\theta)dx$

$= -\int \frac{n}{\theta^2} dx$

Therefore asymptotic variance $= \frac{n}{\theta^2}$

C.

Using the same confidence interval formula as given in the question we get:

$C_n = \left[\hat{\theta}_n - \frac{z_{0.05/2}}{\sqrt{n\frac{n}{\theta^2}}}, \hat{\theta}_n + \frac{z_{0.05/2}}{\sqrt{n\frac{n}{\theta^2}}}\right]$

D.

Not sure how to setup the simulations even with the hints.

Hopefully solutions will go into detail, if not go to office hours.

R-Package imports

```
# Please import (and/or) these packages before running any other code, thank you!
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
library("tidyverse")
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2

## -- Attaching packages --------------------------------------------------------------------------- ti

## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3

## Warning: package 'tibble' was built under R version 3.6.2
## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'readr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

## Warning: package 'forcats' was built under R version 3.6.2

## -- Conflicts ------------------------------------------------------------------------------ tidyvers
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library("bigmemory")
```

```
## Warning: package 'bigmemory' was built under R version 3.6.2
```

```r
library("margins")
```

```
## Warning: package 'margins' was built under R version 3.6.2
```

Q4. Law of Large Numbers and Central Limit Theorem (20 Points)

Answers:

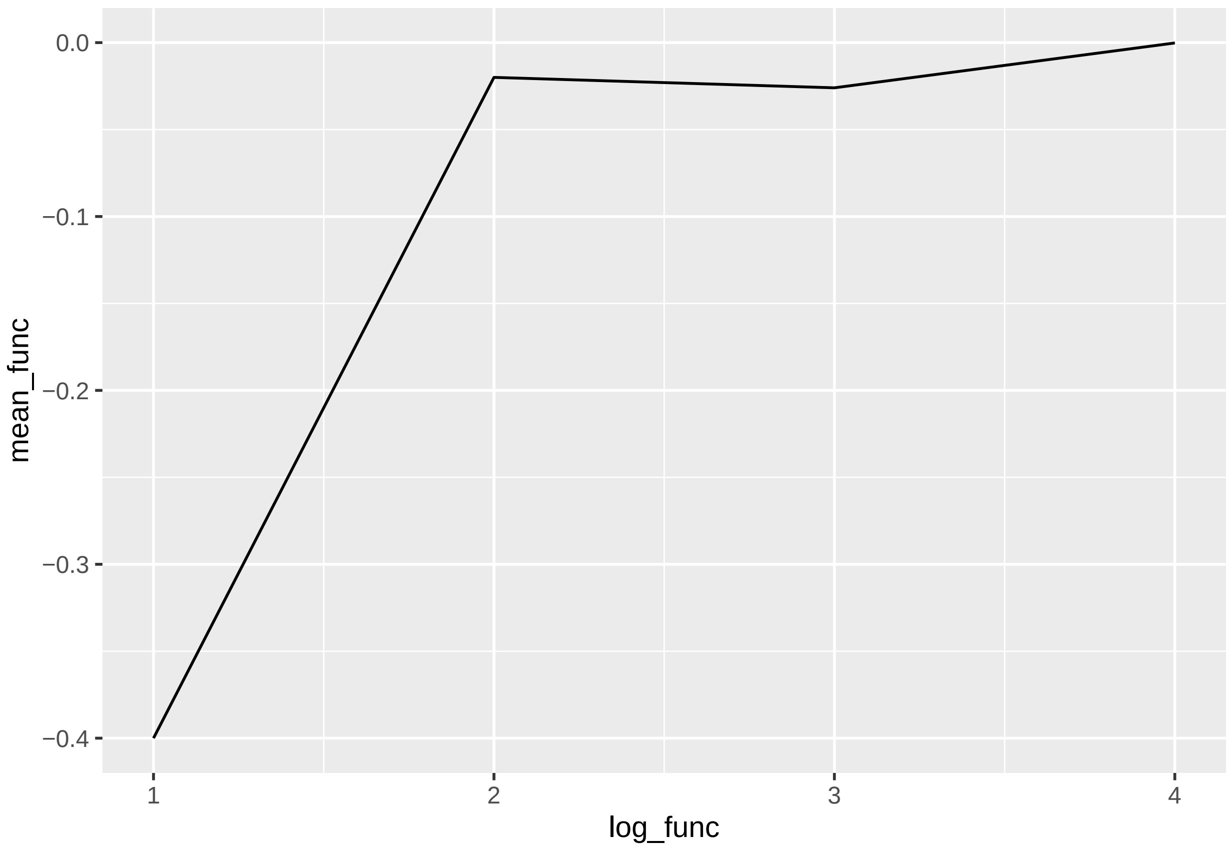
First let us generate the 10,000 the n-sized datasets for $\bar{X}_n^{(i)}$

```r
gen_sample <- function(n){
  # each data point size will have 10000 datasets
  x_bar_final = c(10000)
  # loop 10000 time for each dataset
  for(i in 1:10000){
    # generate data with n data points
    xn = runif(n, 0, 1)
    # create a vector of size n
    x_bar_n = c(n)
    # loop through each data point
    for(j in 1:n){
      # depending on condition, set values appropraitely
      if(xn[j] < 0.5){
        x_bar_n[j] = -1
      }else{
        x_bar_n[j] = 1
      }
    }
    # add mean value to the returning vector
    x_bar_final[i] = mean(x_bar_n)
  }
  return(x_bar_final)
}

# Generate samples for each datapoint (n) size
x_bar_10 = gen_sample(10)
x_bar_100 = gen_sample(100)
x_bar_1000 = gen_sample(1000)
x_bar_10000 = gen_sample(10000)

x_bar_final = matrix(nrow = 4, ncol = 10000)
```

```
# Each row is for a specific n value, i.e., row 1 is for n=10, row 2 for n=100 etc.
x_bar_final[1, ] = x_bar_10
x_bar_final[2, ] = x_bar_100
x_bar_final[3, ] = x_bar_1000
x_bar_final[4, ] = x_bar_10000


# Try using sum(runif(10) > 0.5)
# This returns the number (out of 10) of datapoints that were > 0.5... a.k.a the number of points to as
# Hence the number of points to assign -1 is 1 - the above value
```

A. $log_{10}(n)$ **v.s.** $\bar{X}_n^{(1)} - \mu$

```
library(ggplot2)
# Our log function will need to have points at each of the following n-values
n = c(10,100,1000,10000)

# Log function
log_func = log10(n)

# Using the same x_bar_final from the code block above
mean_func = x_bar_final[, 1] - 0
ggplot() + geom_line(aes(x=log_func, y=mean_func))
```



By plotting $\bar{X}_n^{(1)} - \mu$ v.s. $log_{10}(n)$ we can see that as n gets larger, ($log_{10}(n)$ increases), the value of $\bar{X}_n^{(1)} - \mu$ gets closer and closer to 0, hence convergence. Logically this makes sense, for example, if we were flipping coins, the average should get closer and closer with each toss. Thus, average 10 tosses < average 100 tosses < average 1000 tosses < average 10000 tosses. This similar train of thought can be applied here. Our samples of the uniform distribution will be closer to the true mean of the uniform distribution, the more samples we test.

B. $log_{10}(n)$ **v.s.** $\frac{1}{N}\sum_{i=1}^{N} I\{|\bar{X}_n^{(i)} - \mu| \geq \epsilon\}$ **for** $\epsilon = 0.5$**,** $\epsilon = 0.1$**, and** $\epsilon = 0.05$**.**

```
epsilon_0.5 = 0.5
epsilon_0.1 = 0.1
epsilon_0.05 = 0.05

# Using the same log_func and x_bar_final from the code blocks above

# Calculating number of values the various epsilon ranges for n=10
ten_0.5 = 1/10000*sum(abs(x_bar_final[1,]) > epsilon_0.5)
ten_0.5
```

```
## [1] 0.113
```

```
ten_0.1 = 1/10000*sum(abs(x_bar_final[1,]) > epsilon_0.1)
ten_0.1
```

```
## [1] 0.7562
```

```
ten_0.05 = 1/10000*sum(abs(x_bar_final[1,]) > epsilon_0.05)
ten_0.05
```

```
## [1] 0.7562
```

```
# Calculating number of values the various epsilon ranges for n=100
hundred_0.5 = 1/10000*sum(abs(x_bar_final[2,]) > epsilon_0.5)
hundred_0.5
```

```
## [1] 0
```

```
hundred_0.1 = 1/10000*sum(abs(x_bar_final[2,]) > epsilon_0.1)
hundred_0.1
```

```
## [1] 0.2669
```

```
hundred_0.05 = 1/10000*sum(abs(x_bar_final[2,]) > epsilon_0.05)
hundred_0.05
```

```
## [1] 0.6144
```

```
# Calculating number of values the various epsilon ranges for n=1000
thousand_0.5 = 1/10000*sum(abs(x_bar_final[3,]) > epsilon_0.5)
thousand_0.5
```

```
## [1] 0
```

```
thousand_0.1 = 1/10000*sum(abs(x_bar_final[3,]) > epsilon_0.1)
thousand_0.1
```

```
## [1] 0.0016
```

```
thousand_0.05 = 1/10000*sum(abs(x_bar_final[3,]) > epsilon_0.05)
thousand_0.05
```

```
## [1] 0.1077
```

```r
# Calculating number of values the various epsilon ranges for n=10000
hunthousand_0.5 = 1/10000*sum(abs(x_bar_final[4,]) > epsilon_0.5)
hunthousand_0.5
```

```
## [1] 0
```

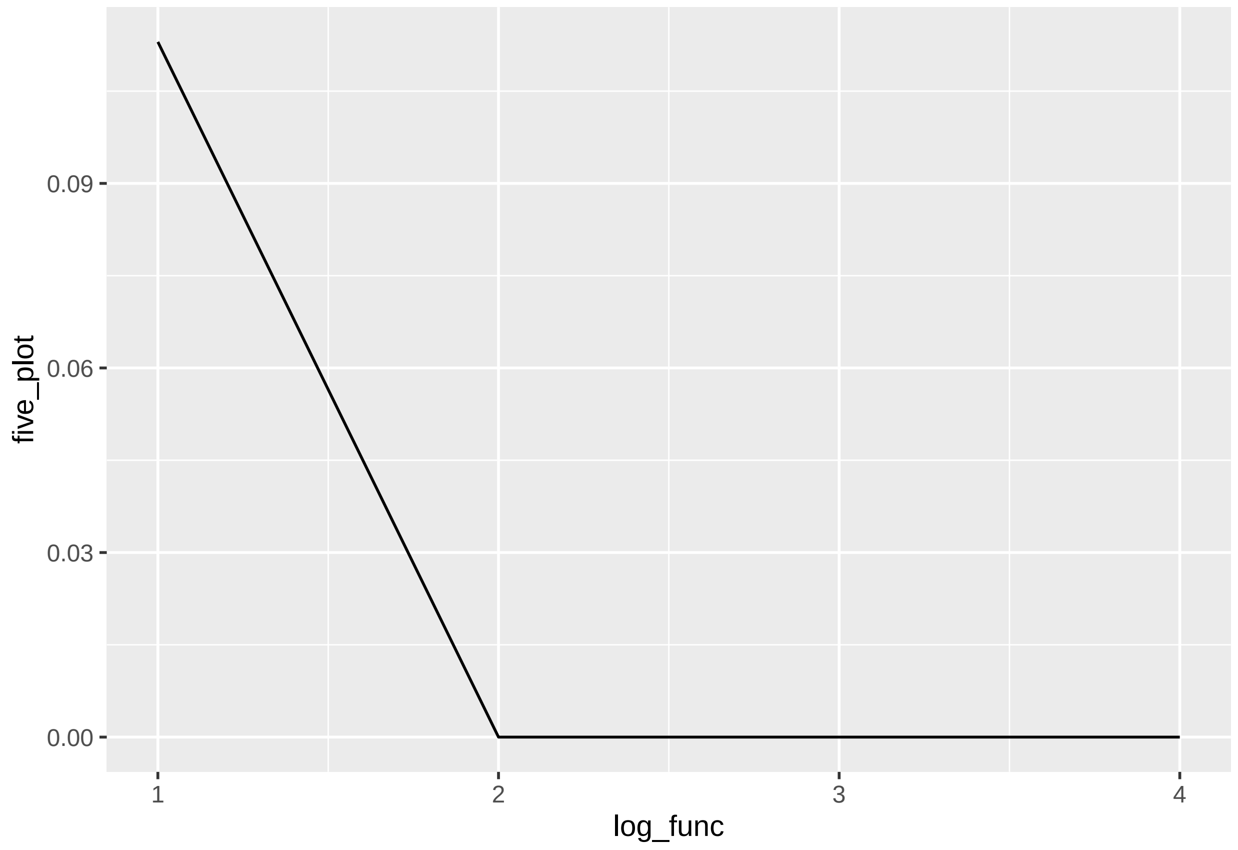
```r
hunthousand_0.1 = 1/10000*sum(abs(x_bar_final[4,]) > epsilon_0.1)
hunthousand_0.1
```

```
## [1] 0
```

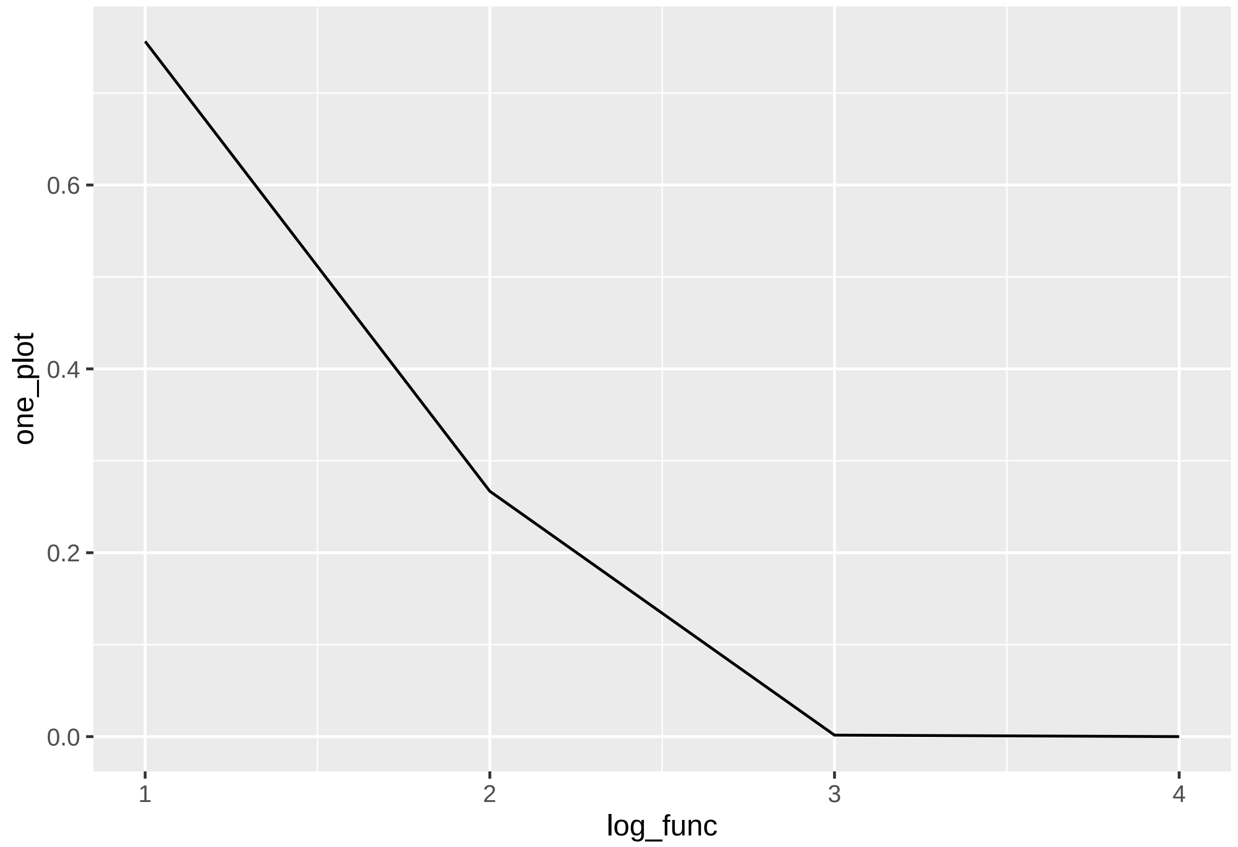
```r
hunthousand_0.05 = 1/10000*sum(abs(x_bar_final[4,]) > epsilon_0.05)
hunthousand_0.05
```

```
## [1] 0
```

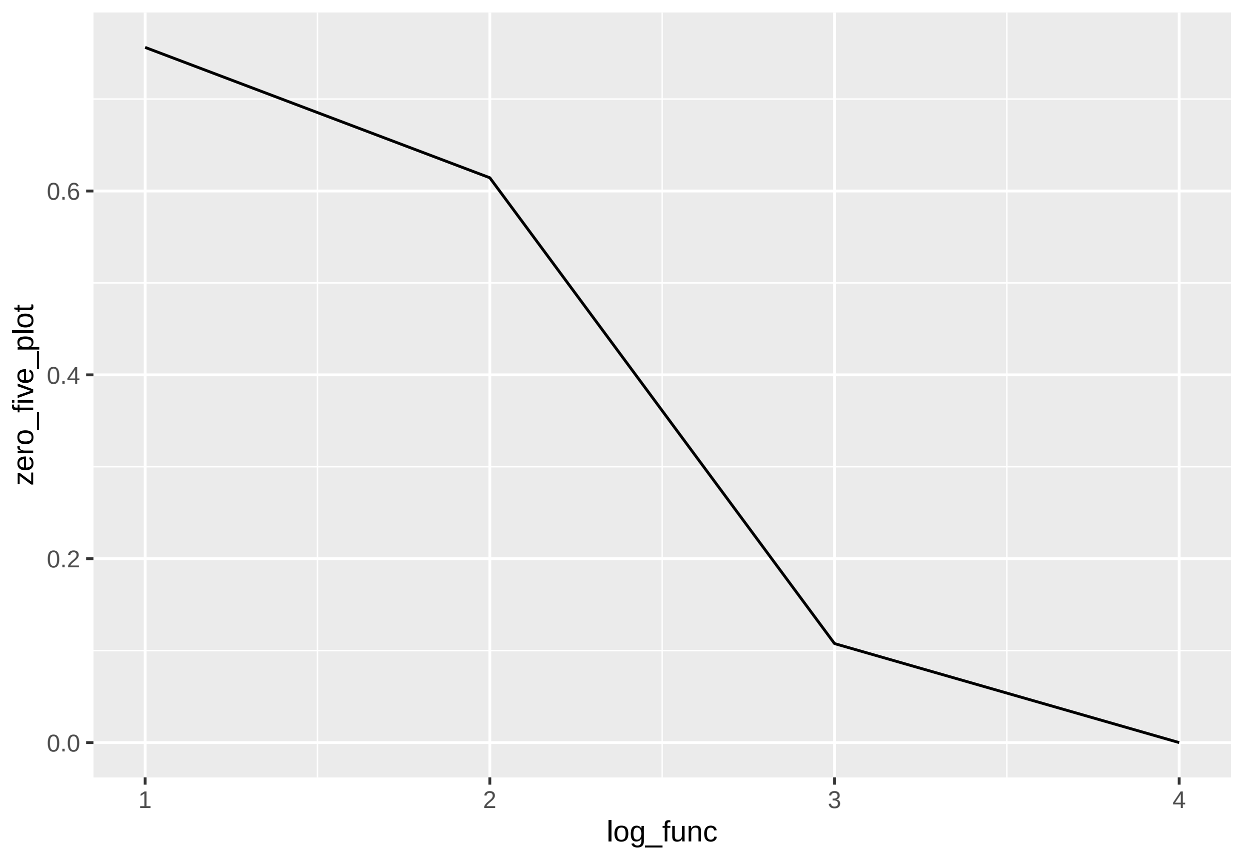
```r
# Create vectors to store the appropriate data for each plot
five_plot = c(ten_0.5, hundred_0.5, thousand_0.5, hunthousand_0.5)
one_plot = c(ten_0.1, hundred_0.1, thousand_0.1, hunthousand_0.1)
zero_five_plot = c(ten_0.05, hundred_0.05, thousand_0.05, hunthousand_0.05)

# Plot
ggplot() + geom_line(aes(x=log_func, y=five_plot))
```



```r
ggplot() + geom_line(aes(x=log_func, y=one_plot))
```



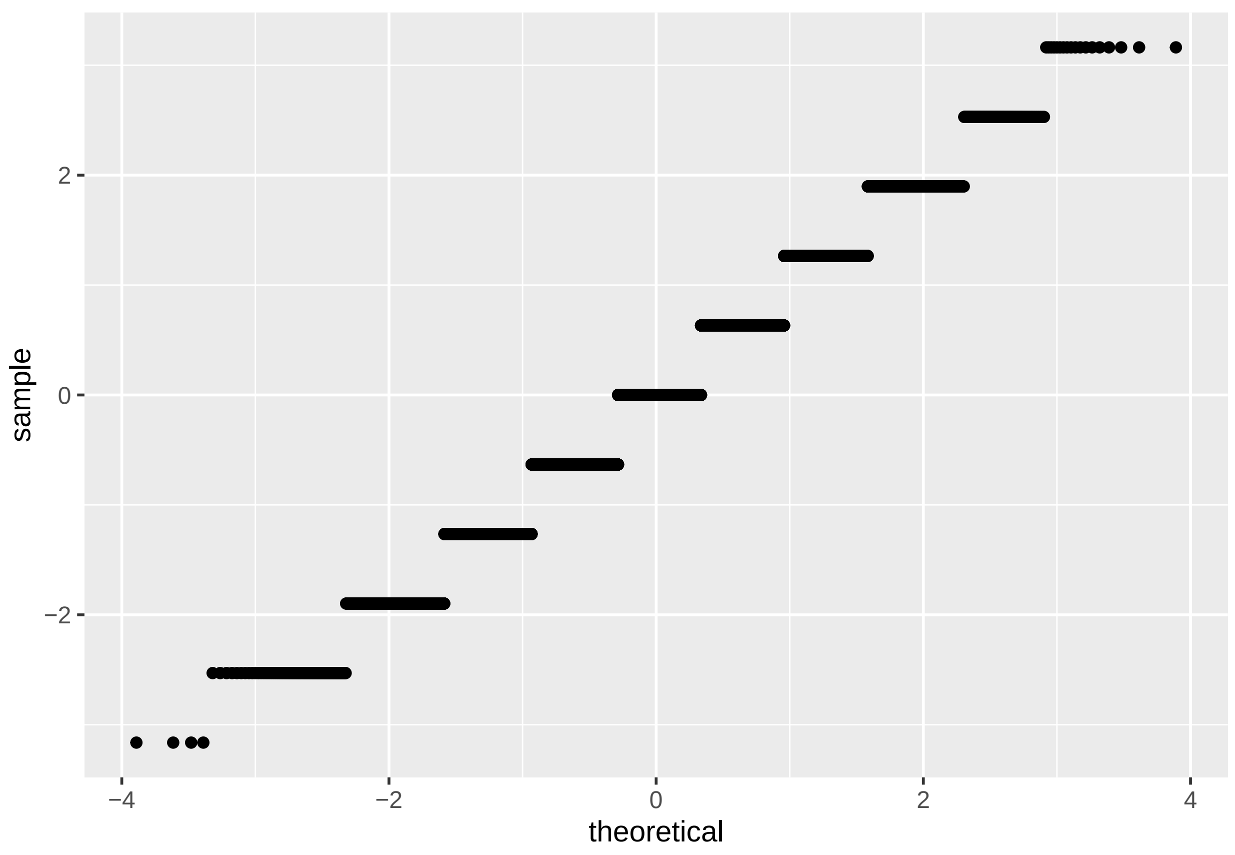
```r
ggplot() + geom_line(aes(x=log_func, y=zero_five_plot))
```



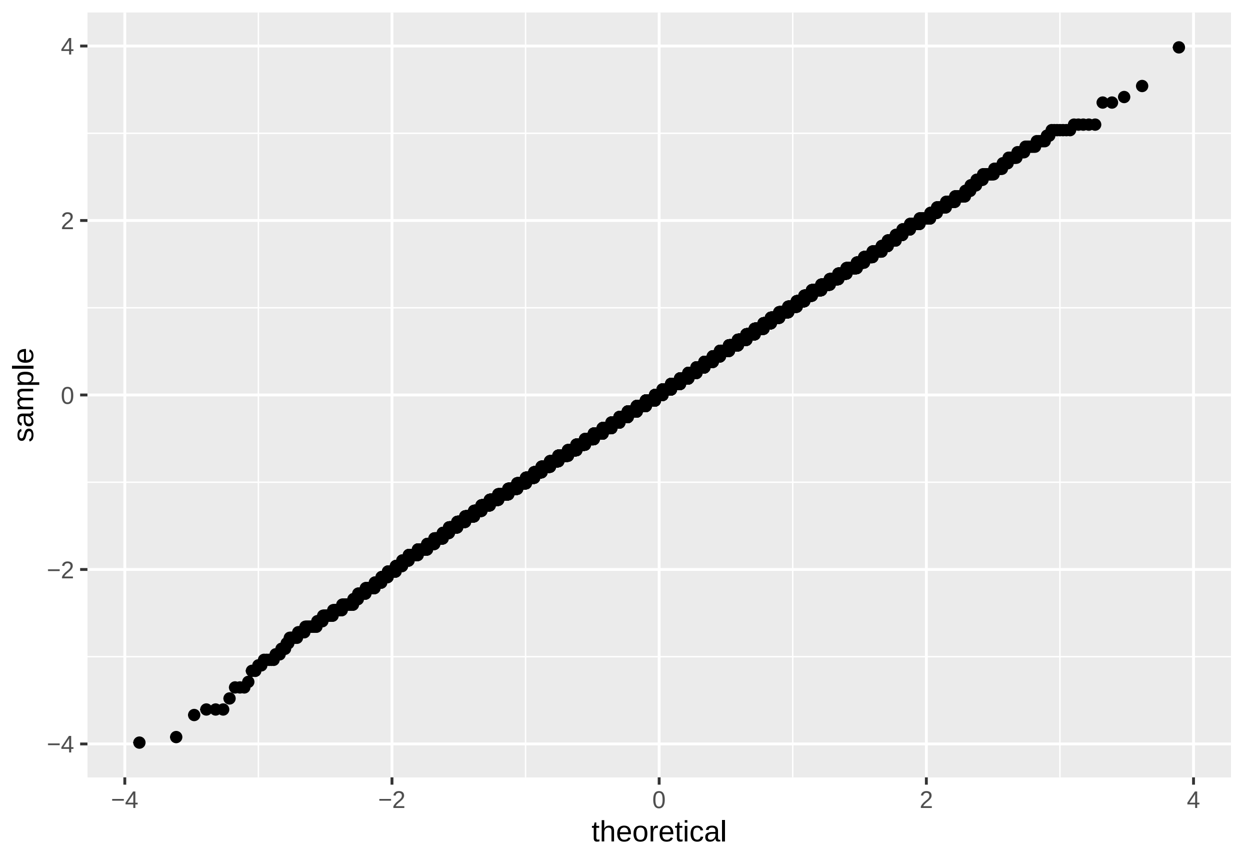
The above plots show the accuracy of our data, that as we loosen the restriction on epsilon (make it smaller), the smaller data point sets stray from the true mean of 0. This showcases the Law of Large Numbers because even when we loosen the restriction on epsilon, our largest data set with the most data points $(n = 10000, N = 10000)$ is always closest to the true mean.

C. $\sqrt{n}\frac{(\bar{X}_n^{(i)} - \mu)}{\sigma}$

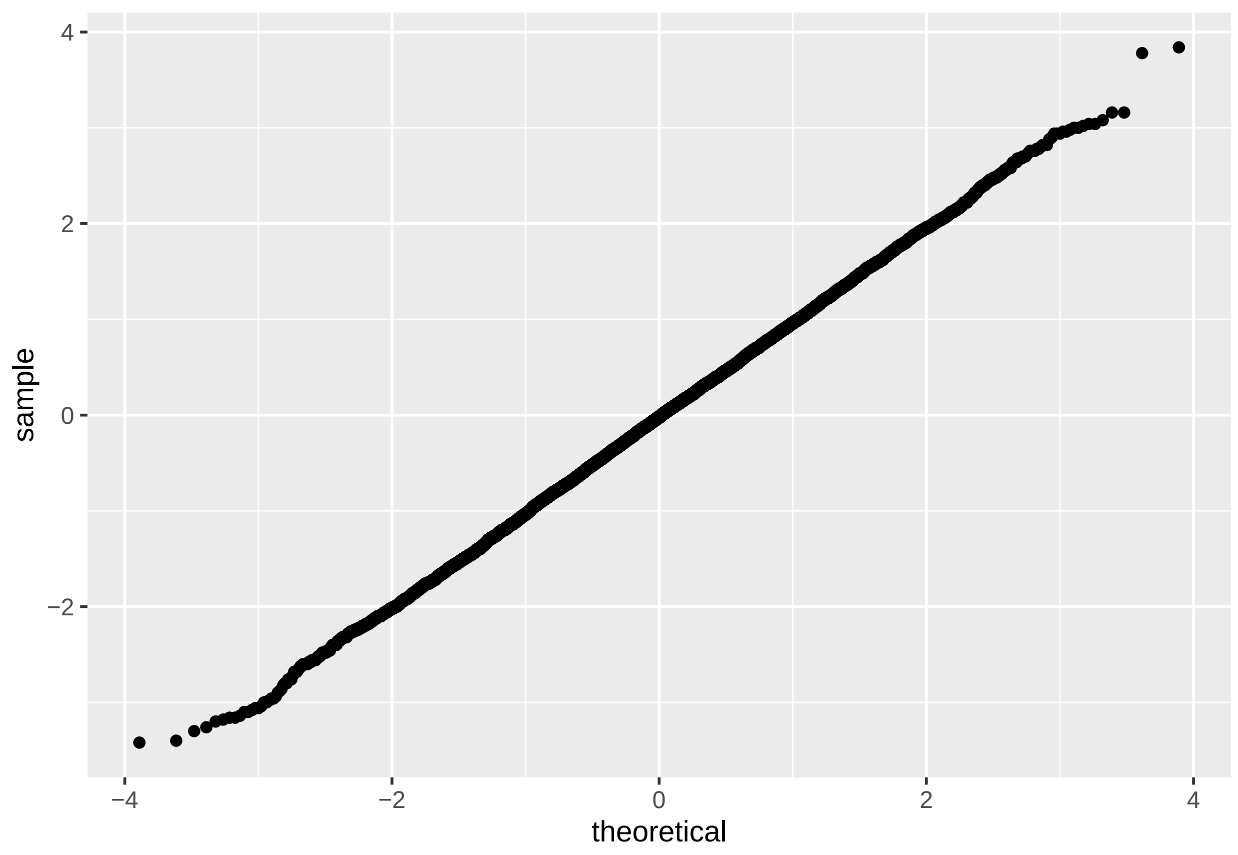
```r
# Variance for when expecation is 0 = 1

# n=10 data vectors
ten_vector = x_bar_final[1,]/1
ten_vector = sqrt(10)*ten_vector

# n=1000
thousand_vector = x_bar_final[3,]/1
thousand_vector = sqrt(1000)*thousand_vector

# n=10000
ten_thou_vector = x_bar_final[4,]/1
ten_thou_vector = sqrt(10000)*ten_thou_vector

# Q-Q plots
ggplot(data = NULL, df, mapping = aes(sample=ten_vector)) + stat_qq()
```



```r
ggplot(data = NULL, df, mapping = aes(sample=thousand_vector)) + stat_qq()
```

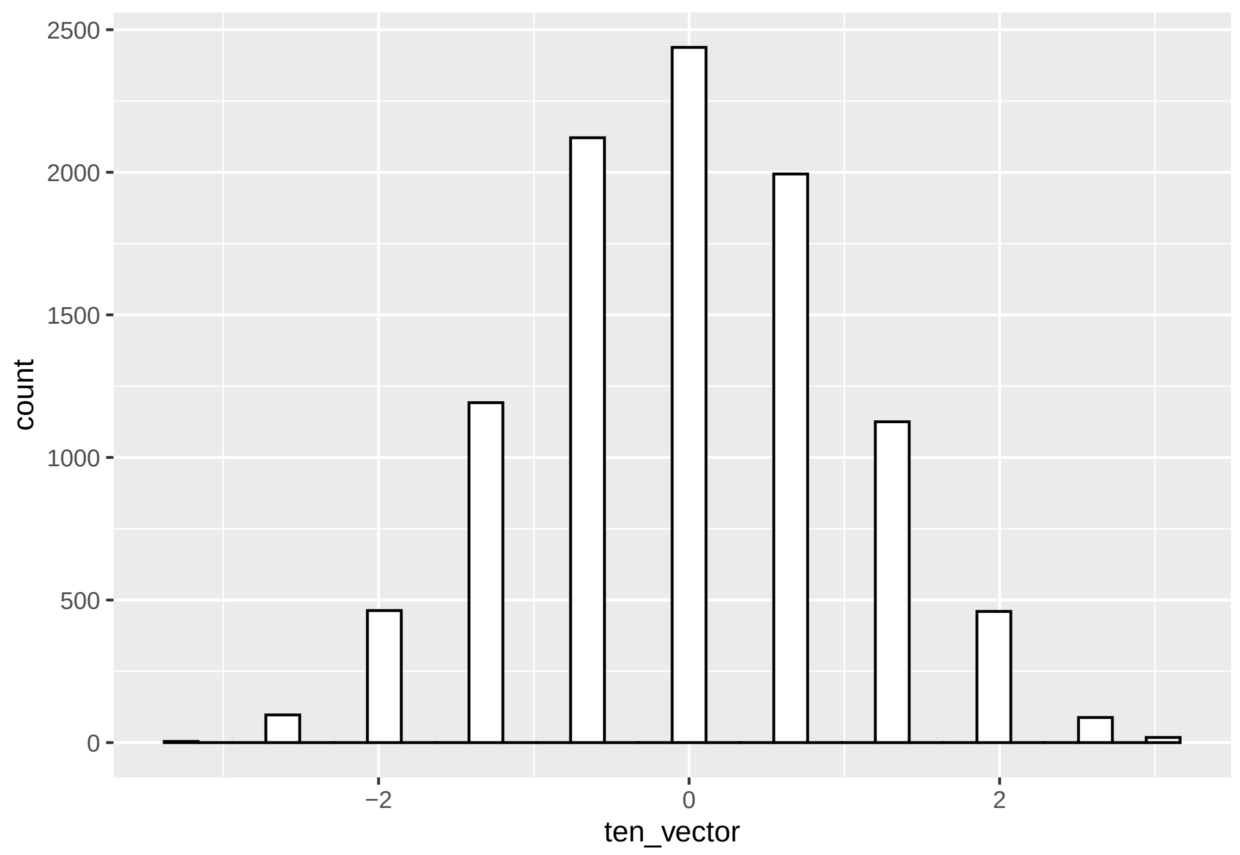


```r
ggplot(data = NULL, df, mapping = aes(sample=ten_thou_vector)) + stat_qq()
```

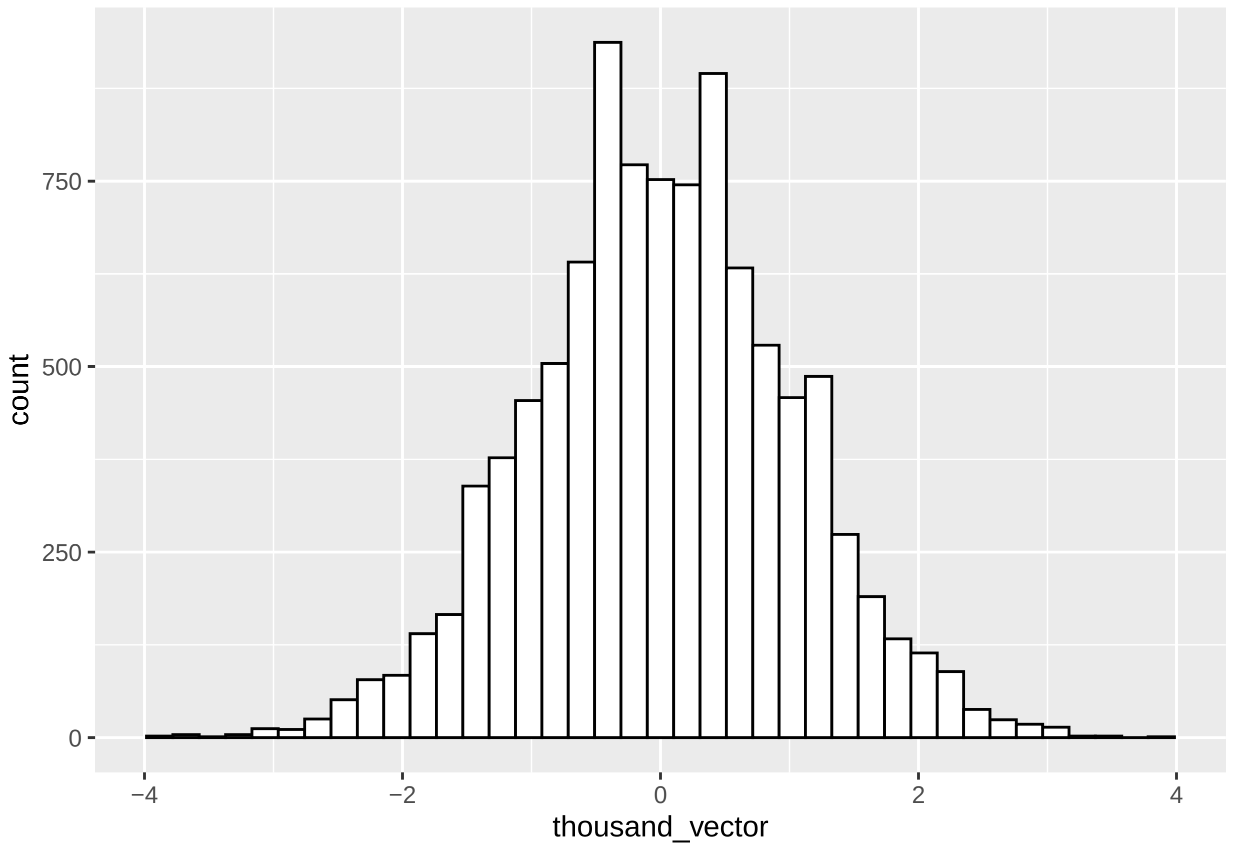


```
# Histogram plots
ggplot(data = NULL, df, mapping = aes(x=ten_vector)) + geom_histogram(fill="white",color="black")
```

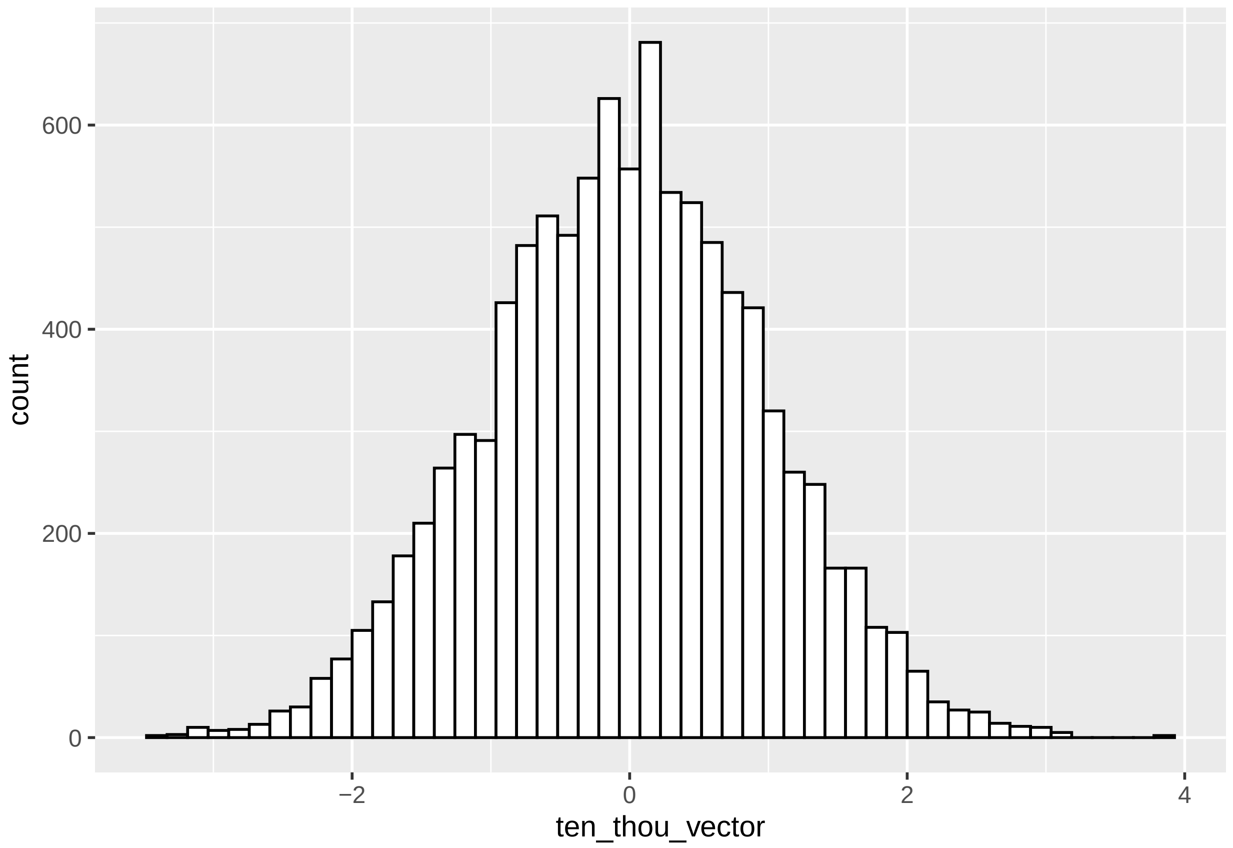
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
ggplot(data = NULL, df, mapping = aes(x=thousand_vector)) + geom_histogram(fill="white",color="black", 
```



```
ggplot(data = NULL, df, mapping = aes(x=ten_thou_vector)) + geom_histogram(fill="white",color="black", 
```



First looking at the Q-Q plots, we can see that the "steps" become smoothed as we increase the datapoint size there curve shape becomes more and more akin to the Normal Curve. Similarly with the Histograms, we notice that in the ten data point set, many of the bars are missing (even though the general histogram appears to be normal), when we go to the thousand data point set, we see that all the bars are filled and it begins to take the bell shape form, but not completely, some bars need to be swapped around. Finally with the final plot, there are still some mistakes, and there isn't a clear middle maximum point to the curve, but it is by far the most normal of the three data point sets.

D. $log_{10}(n)$ **v.s.** $\frac{1}{N}\sum_{i=1}^{N} I\{|\frac{\sqrt{n}(\bar{X}_n^{(i)}-\mu)}{\sigma-Y_i}| > \epsilon\}$ **for** $\epsilon = 0.001$**.**

```
# Generate the N=10000 random i.i.d normal RVs
y_n = rnorm(1*10000)

# Calculate the sum of the indicator values
sum10 = 1/10000*sum(abs((sqrt(10)*(x_bar_final[1,]))/(1 - y_n)) > 0.001)
sum10
```

```
## [1] 0.7562
```

```
sum100 = 1/10000*sum(abs((sqrt(100)*(x_bar_final[2,]))/(1 - y_n)) > 0.001)
sum100
```

```
## [1] 0.9247
```

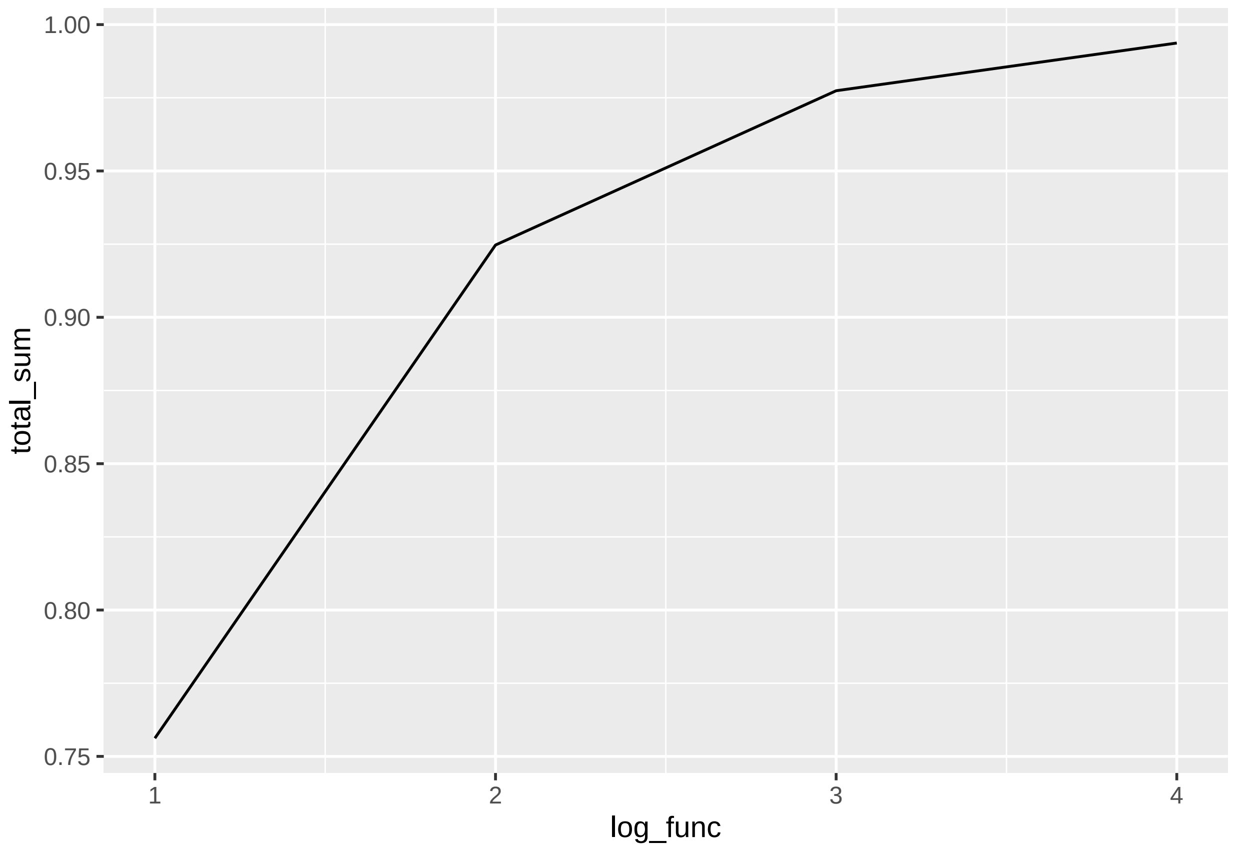
```
sum1000 = 1/10000*sum(abs((sqrt(1000)*(x_bar_final[3,]))/(1 - y_n)) > 0.001)
sum1000
```

```
## [1] 0.9774
```

```r
sum10000 = 1/10000*sum(abs((sqrt(10000)*(x_bar_final[4,]))/(1 - y_n)) > 0.001)
sum10000
```

```
## [1] 0.9937
```

```r
# Create a vector with all the summed values
total_sum = c(sum10, sum100, sum1000, sum10000)

ggplot() + geom_line(aes(x=log_func, y=total_sum))
```



Here we are neither calculating a probability, nor is it converging to 0. The value is converging to 1, which is the SD of the normal distribution, and within the range of our pseudo-uniform distribution. Hence, we do not converge in probability to Y but we do in distribution as $n \to \infty$ as we can see in the plot, it is headed towards 1.0.

Q5. Basic R Programming for Big Data (20 points)

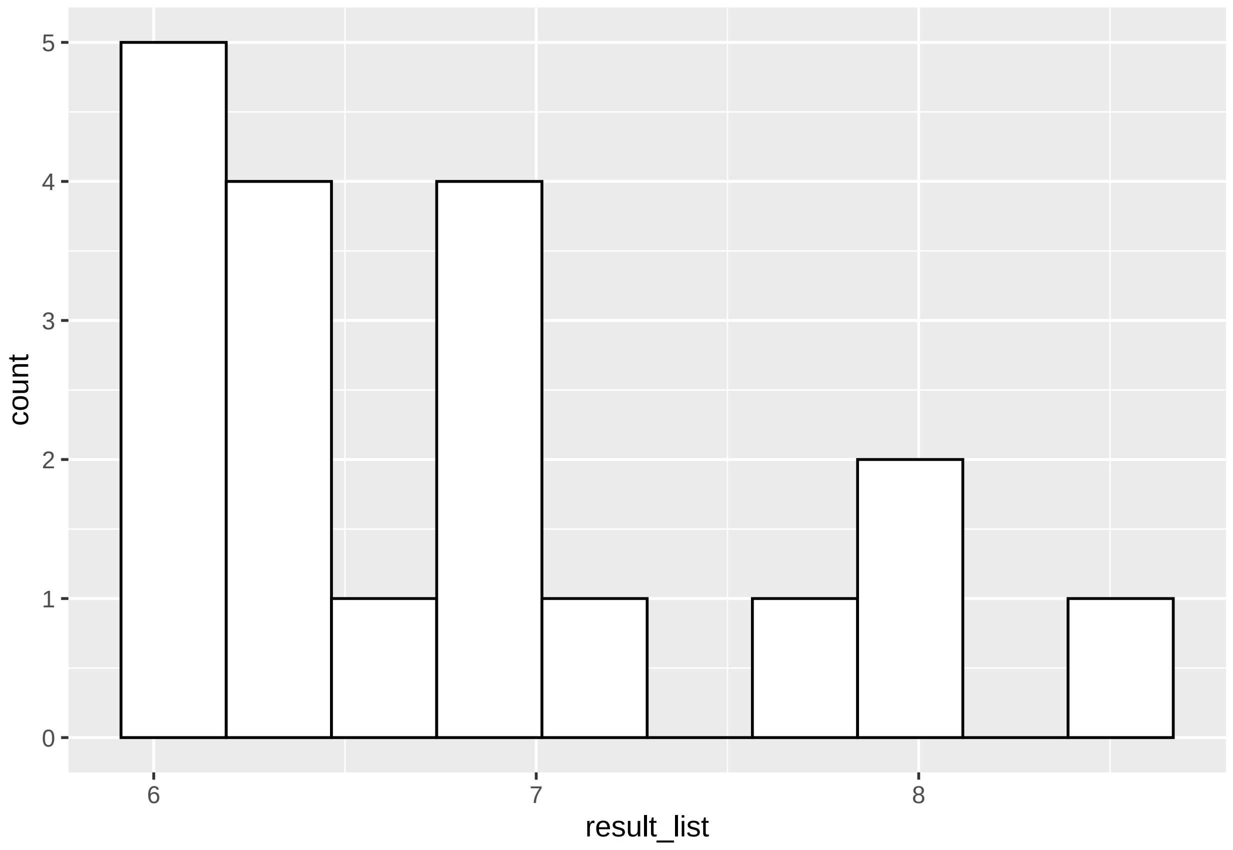
Answers:

A. Part 1 and Part 2

```r
library(bigmemory)

# set working directory COMMENT OUT/CHANGE AS NEED BE
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")

# read in the data
ratings_dat = read.big.matrix("ratings.dat", sep=",")
```

```
## Warning in read.big.matrix("ratings.dat", sep = ","): Because type was not
## specified, we chose integer based on the first line of data.
```

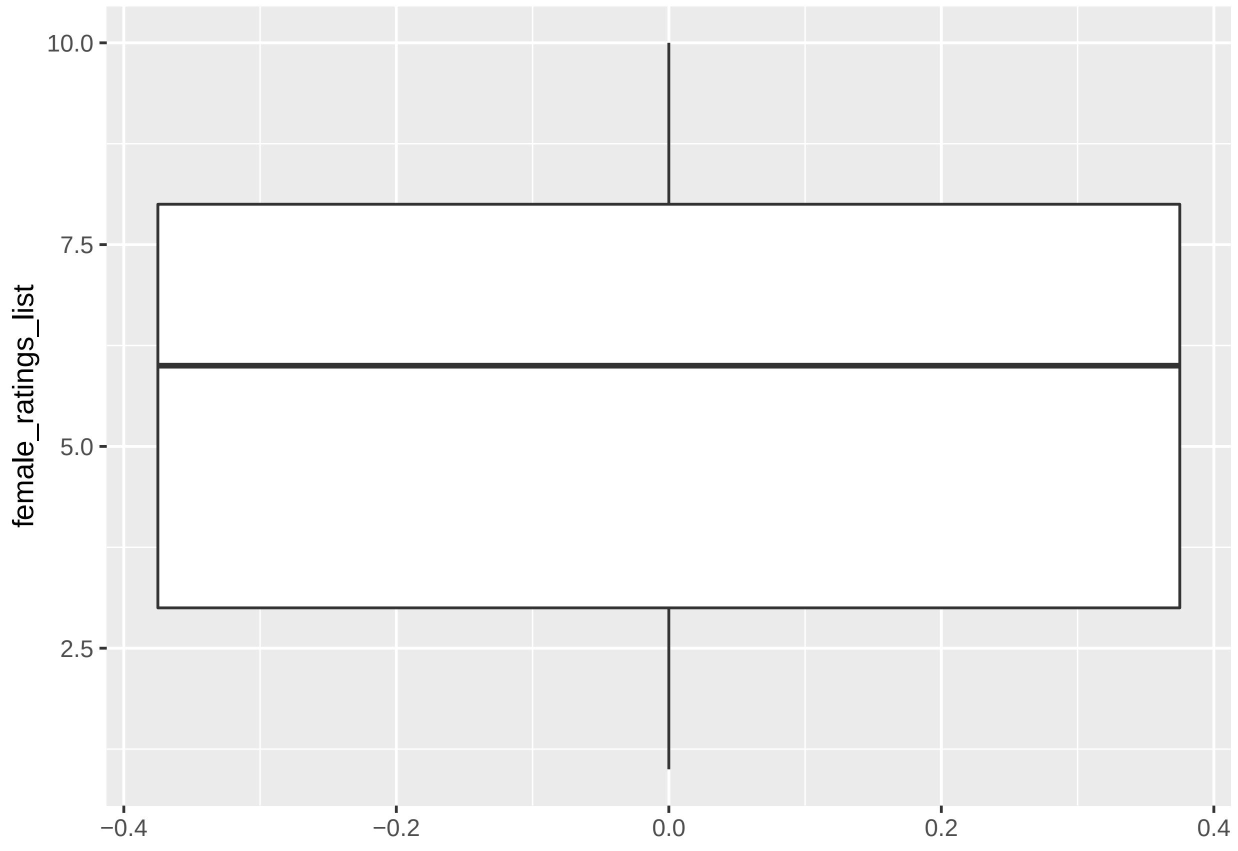
```
# assign column names
options(bigmemory.allow.dimnames = TRUE)
colnames(ratings_dat) = c("UserID", "ProfileID", "Rating")

overall_ratings_avg = (1/3000000)*(sum(ratings_dat[,3]))

weighted.rank <- function(x){
  # Get all indices with given ProfileID
  profile_index = which(ratings_dat[,2] == x)
  # Get v = number of votes for the profile
  num_of_index = length(profile_index)
  temp_rating_sum = 0
  # Sum all of the ratings for given Profile ID
  for(i in 1:num_of_index){
    temp_rating_sum = temp_rating_sum + ratings_dat[i,3]
  }
  # Get average for the profile
  profile_avg = (1/num_of_index)*temp_rating_sum
  # Get overall avg for the whole data (DOING IT ABOVE FUNCTION CAUSE IT IS A STATIC VALUE)

  weighted_rank = (((num_of_index/(num_of_index + 4182))*profile_avg) + (4182/(num_of_index + 4182)))*o
  return(weighted_rank)
}
# test for weighted.rank, result = 5.958786
#y=weighted.rank(8035)

# Part 2 of (a), compute weighted ranks for all the profiles who were rated by UserID 100, then plot a

# First get all the indexes that UserID 100 appears in
user_100_indexes = which(ratings_dat[,1] == 100)


# Create a list to store the weighted rank results for each of the indices
# First get number of indices for size of vector
numof_100indices = length(user_100_indexes)


profile_list = list(a = 1:numof_100indices)

for(j in 1:numof_100indices){
  profile_list[j] = ratings_dat[user_100_indexes[j], 2]
}

result_list = lapply(profile_list, weighted.rank)

result_list = unlist(result_list, use.names = FALSE)

result_list = as.numeric(result_list)

ggplot(data = NULL, df, mapping = aes(x=result_list)) + geom_histogram(fill="white",color="black", bins
```



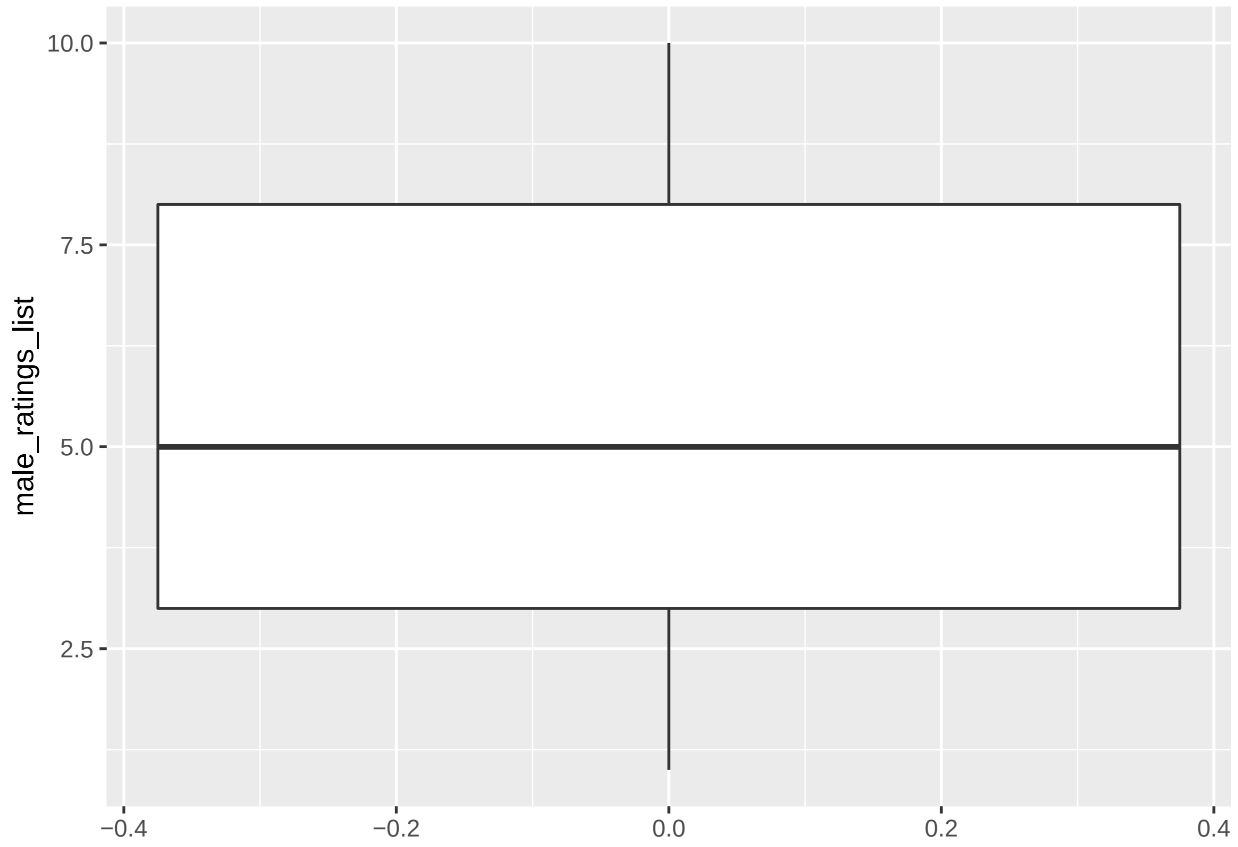
B.

```
# First load in users.Rdata
# set working directory COMMENT OUT/CHANGE AS NEED BE
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")
load(file = "users.RData")

# Do grep + regex to get the indexes of what we want
male_and_newyork = intersect(grep("M", User$Gender, ignore.case=TRUE), union(grep("New York", User$Stat

cali = union(union(grep("California", User$State, ignore.case=TRUE), grep("CA", User$State, ignore.case

female_and_cali = intersect(grep("F", User$Gender, ignore.case=TRUE), cali)

# Get user id for our people
male_and_ny_index = length(male_and_newyork)
male_list = list(a = 1:male_and_ny_index)
for(i in 1:male_and_ny_index){
  male_list[i] = ratings_dat[User$UserID[male_and_newyork[i]], 1]
}

male_list = unlist(male_list, use.names = FALSE)

#male_list = as.numeric(male_list)

female_and_ca_index = length(female_and_cali)
```

```r
female_list = list(a = 1:female_and_ca_index)
for(j in 1:female_and_ca_index){
  female_list[j] = ratings_dat[User$UserID[female_and_cali[j]], 1]
}

female_list = unlist(female_list, use.names = FALSE)

#female_list = as.numeric(female_list)

# Now we get all the ratings these users have given out and store them in a vector to make a boxplot ou
male_ratings_list = c()

for(i in 1:length(male_list)){
  male_ratings_list = c(male_ratings_list, ratings_dat[which(ratings_dat[,1] == male_list[i]), 3])
}

female_ratings_list = c()

for(i in 1:length(female_list)){
  female_ratings_list = c(female_ratings_list, ratings_dat[which(ratings_dat[,1] == female_list[i]), 3]
}

ggplot(data = NULL, aes(y=female_ratings_list)) + geom_boxplot(outlier.colour="black")
```



```
ggplot(data = NULL, aes(y=male_ratings_list)) + geom_boxplot(outlier.colour="black")
```



C.

Will come back to this question later if time permits.

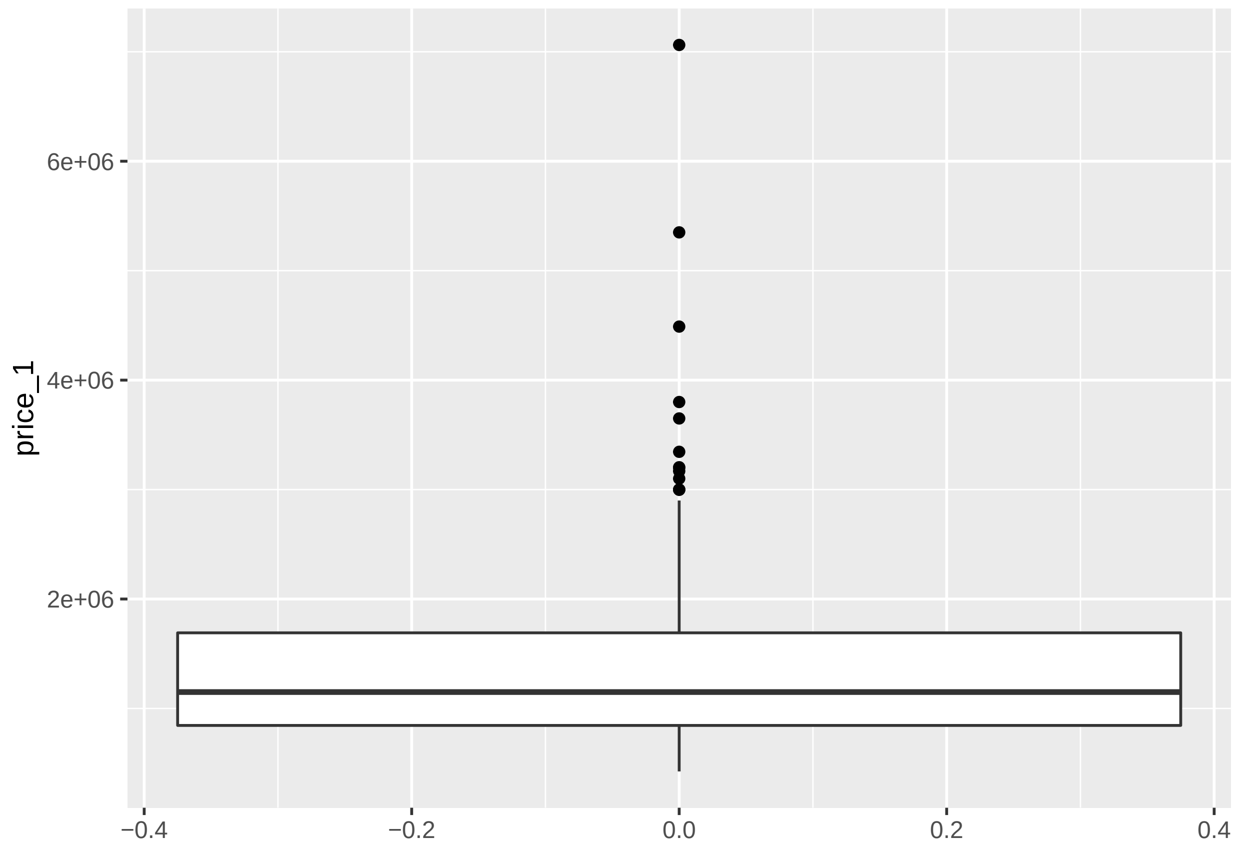
Q6. A Simple Linear Regression (25 points)

Answers:

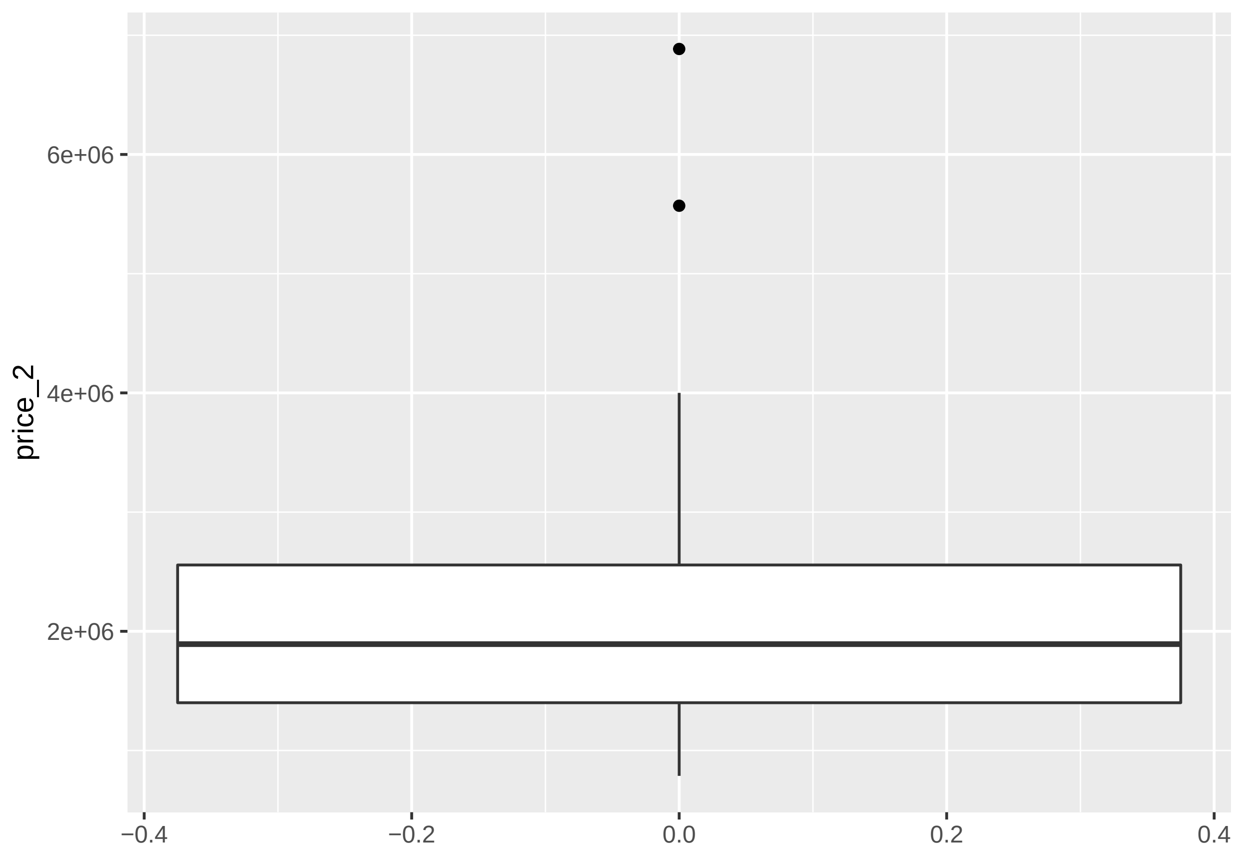
A.

```
# First we will read in housingprice.csv
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")
hp = read.csv("housingprice.csv")

library(tidyverse)

top_zipcodes = hp %>% group_by(zipcode) %>% summarize(meanprice=mean(price)) %>% arrange(desc(meanprice

#98039
prices1 = hp %>% filter(zipcode==98039)
price_2 = prices1$price
#98004
prices2 = hp %>% filter(zipcode==98004)
price_1 = prices2$price
```

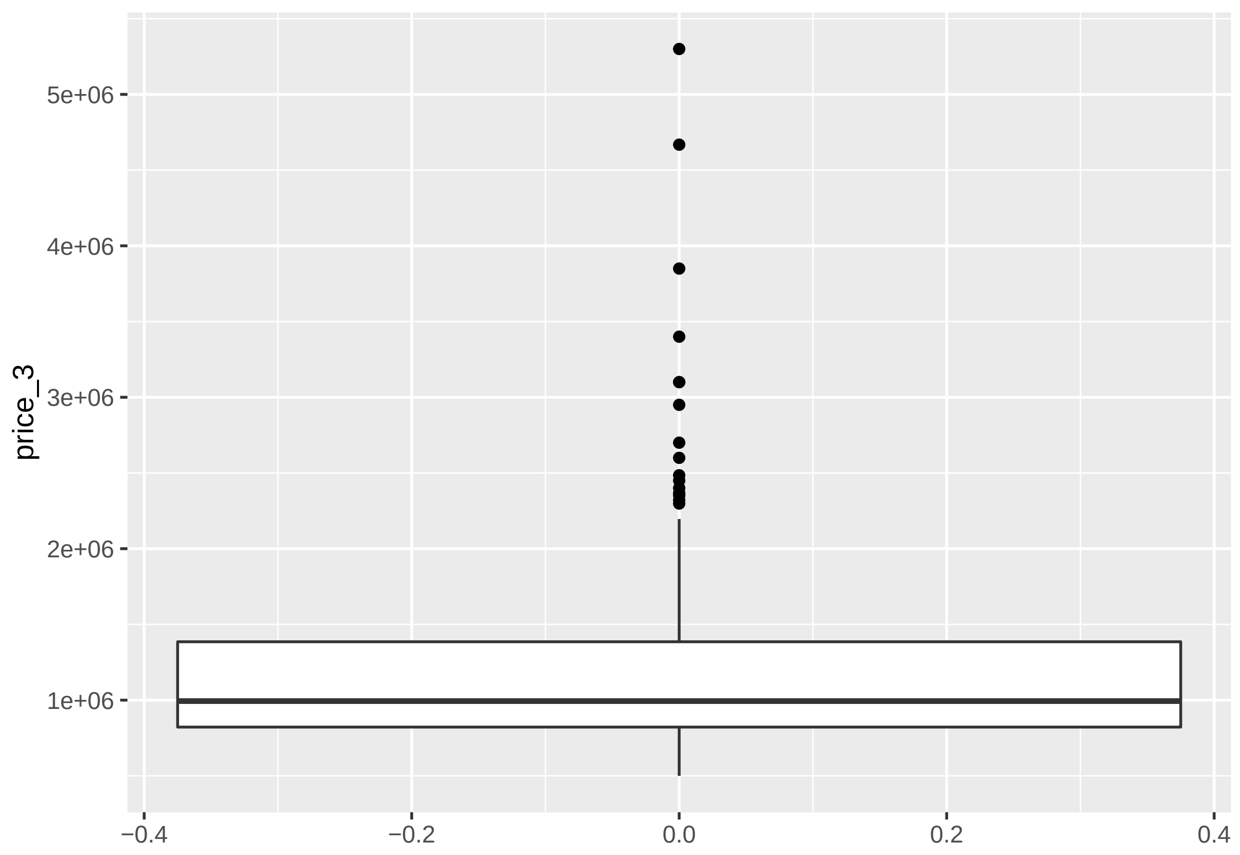
```
#98040
prices3 = hp %>% filter(zipcode==98040)
price_3 = prices3$price

ggplot(data = NULL, aes(y=price_1)) + geom_boxplot(outlier.colour="black")
```



```
ggplot(data = NULL, aes(y=price_2)) + geom_boxplot(outlier.colour="black")
```

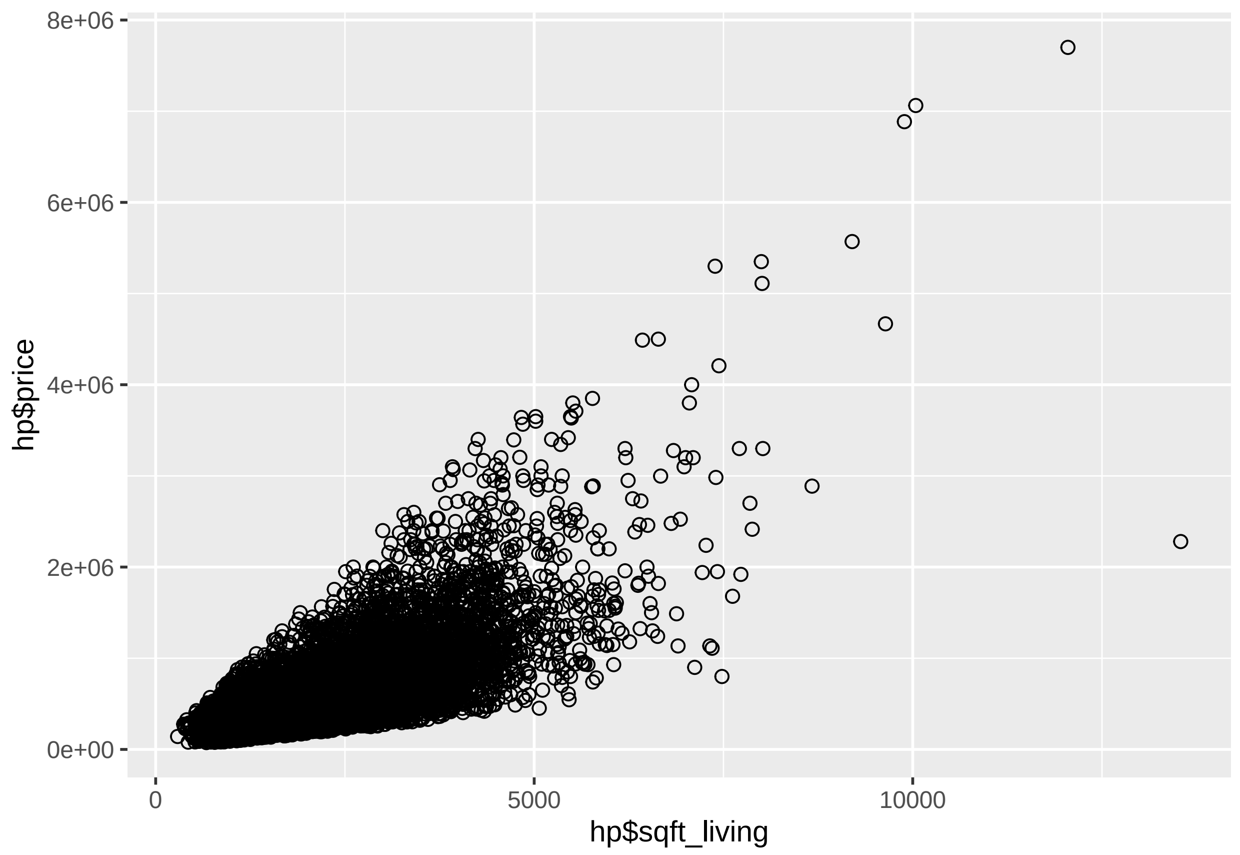


```
ggplot(data = NULL, aes(y=price_3)) + geom_boxplot(outlier.colour="black")
```



B.

```r
# Scatterplot between sqft_living and housing price
sp = ggplot(hp, aes(x=hp$sqft_living, y=hp$price)) + geom_point(size=2, shape=1)
sp
```



Not sure how to better format the plot. I have tried changing the limits on the x and y axis and but it still did not become more clear.

C.

```
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")
train_data = read.csv("train.data.csv")
test_data = read.csv("test.data.csv")
bill_gates = read.csv("fancyhouse.csv")

train_data_model = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot, data=train_data)

summary(train_data_model)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot,
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1571803  -143678   -22595   103133  4141210
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.083e+04  8.208e+03   9.848  < 2e-16 ***
## bedrooms    -5.930e+04  2.753e+03 -21.537  < 2e-16 ***
## bathrooms    3.682e+03  4.178e+03   0.881    0.378
## sqft_living  3.167e+02  3.750e+00  84.442  < 2e-16 ***
## sqft_lot    -4.267e-01  5.504e-02  -7.753 9.52e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 257200 on 15124 degrees of freedom
## Multiple R-squared:  0.5101, Adjusted R-squared:   0.51
## F-statistic:  3937 on 4 and 15124 DF,  p-value: < 2.2e-16
```

```
# As we can see by the summary above, the R-Squared on the Training Data is 0.5101.

# Calculate the test data RSQ using the sum of squares formula

# First get y_hat values
test.pred = predict(train_data_model, test_data)
# get y
test.y = test_data$price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.5081596
```

```
# The Test Data R-Squared is 0.5081.
```

As we can see by the summary above, the R-Squared on the Training Data is 0.5101. Similarly, for the Test Data the R-Squared is 0.5081.

D.

```
train_data_model_zip = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + zipcode, data=train_d

summary(train_data_model_zip)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     zipcode, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1638518  -141274   -22673   101293  4074728
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.460e+07  3.933e+06 -13.883  < 2e-16 ***
## bedrooms    -5.760e+04  2.739e+03 -21.034  < 2e-16 ***
## bathrooms    8.631e+03  4.167e+03   2.071   0.0383 *
## sqft_living  3.185e+02  3.729e+00  85.420  < 2e-16 ***
## sqft_lot    -3.443e-01  5.501e-02  -6.259 3.98e-10 ***
## zipcode      5.573e+02  4.008e+01  13.904  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 255600 on 15123 degrees of freedom
## Multiple R-squared:  0.5163, Adjusted R-squared:  0.5161
## F-statistic:  3228 on 5 and 15123 DF,  p-value: < 2.2e-16
```

```
# Calculate the test data RSQ using the sum of squares formula

# First get y_hat values
test.pred = predict(train_data_model_zip, test_data)
# get y
test.y = test_data$price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.5154904
```

```
# The Test Data R-Squared is 0.5154.
# Once we add in zipcode we get an R-Squared of 0.5163 for Training Data nad 0.5154 for the Test Data
```

Once we add in zip code we get an R-Squared of 0.5163 for Training Data and 0.5154 for the Test Data

E.

```
predict(train_data_model_zip, bill_gates)
```

```
##        1
## 15642273
```

```
# Our model predicts the house is worth $15 million. The average price of houses in his zipcode is arou

# (Although in real life I could see it being true hahaha)
```

Our model predicts the house is worth 15 million dollars. The average price of houses in his zip code is around 2 million dollars. Looking at the box-plot we made earlier, the furthest outlier was around $6 million. The fact that his house would be double that, I do not believe the predicted price is reasonable.

F.

Will come back to this question if time permits.

Q7. Feature Engineering (20 Points)

Answers:

A.

```
extra_rooms_train = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + zipcode + (bedrooms*bathi
summary(extra_rooms_train)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     zipcode + (bedrooms * bathrooms), data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2202454  -139444   -23520   100249  3685052
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -4.920e+07  3.928e+06 -12.526  < 2e-16 ***
## bedrooms           -1.216e+05  5.359e+03 -22.697  < 2e-16 ***
## bathrooms          -9.739e+04  8.694e+03 -11.203  < 2e-16 ***
## sqft_living         3.110e+02  3.745e+00  83.054  < 2e-16 ***
## sqft_lot           -3.502e-01  5.467e-02  -6.405 1.55e-10 ***
## zipcode             5.045e+02  4.001e+01  12.608  < 2e-16 ***
## bedrooms:bathrooms  3.107e+04  2.240e+03  13.871  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 254000 on 15122 degrees of freedom
## Multiple R-squared:  0.5224, Adjusted R-squared:  0.5222
## F-statistic:  2756 on 6 and 15122 DF,  p-value: < 2.2e-16
```

```
# Calculate the test data RSQ using the sum of squares formula

# First get y_hat values
test.pred = predict(extra_rooms_train, test_data)
# get y
test.y = test_data$price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.5185557
```

```
# The Test Data R-Squared is 0.5185.
```

Now we have an R-Squared value of 0.5224, which is a .01 increase from without this new variable. Furthermore, it has caused the p-value of all the variables to improve drastically. For the test data, the value continues to increase to 0.5185.

B.

```
extra_rooms_train_2 = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + zipcode + (bedrooms*ba
summary(extra_rooms_train_2)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     zipcode + (bedrooms * bathrooms) + (sqft_living * sqft_lot),
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1812396  -139614   -23222   101065  3650294
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -5.068e+07  3.923e+06  -12.92  < 2e-16 ***
## bedrooms             -1.274e+05  5.392e+03  -23.62  < 2e-16 ***
## bathrooms            -1.038e+05  8.709e+03  -11.92  < 2e-16 ***
## sqft_living           3.193e+02  3.869e+00   82.53  < 2e-16 ***
## sqft_lot              3.803e-01  1.039e-01    3.66 0.000253 ***
## zipcode               5.196e+02  3.997e+01   13.00  < 2e-16 ***
## bedrooms:bathrooms    3.295e+04  2.246e+03   14.67  < 2e-16 ***
## sqft_living:sqft_lot -2.745e-04  3.323e-05   -8.26  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 253400 on 15121 degrees of freedom
## Multiple R-squared:  0.5245, Adjusted R-squared:  0.5243
## F-statistic:  2383 on 7 and 15121 DF,  p-value: < 2.2e-16
```

```r
# Calculate the test data RSQ using the sum of squares formula

# First get y_hat values
test.pred = predict(extra_rooms_train_2, test_data)
# get y
test.y = test_data$price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.5220649
```

```r
# The Test Data R-Squared is 0.5185.
```

Looking at the p-values we got in Question 7. (a), let us try multiplying sqft_living by sqft_lot. Usually bigger living spaces will lead to bigger lots. Even if the living space is small, if the lot is big house extensions can be done which are a beneficial option.

This change causes the R-Squared of the model to jump to 0.5245, the adjusted R-Squared to also continue to rise, as well as the R-Squared of the test data to get extremely close to that of the model at 0.5220.

C.

```r
# Do the polynomial regression
poly_rooms_train = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + zipcode + poly(bedrooms, 
summary(poly_rooms_train)
```

```
##
```

```
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     zipcode + poly(bedrooms, 2) + poly(bathrooms, 3), data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3312253  -136245   -26067    98812  2733696
##
## Coefficients: (2 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -3.952e+07  3.865e+06 -10.224  < 2e-16 ***
## bedrooms            -5.316e+04  2.702e+03 -19.672  < 2e-16 ***
## bathrooms            2.250e+04  4.082e+03   5.512 3.61e-08 ***
## sqft_living          3.011e+02  3.736e+00  80.610  < 2e-16 ***
## sqft_lot            -4.209e-01  5.359e-02  -7.855 4.27e-15 ***
## zipcode              4.035e+02  3.940e+01  10.241  < 2e-16 ***
## poly(bedrooms, 2)1          NA         NA      NA       NA
## poly(bedrooms, 2)2   1.803e+06  2.556e+05   7.054 1.82e-12 ***
## poly(bathrooms, 3)1         NA         NA      NA       NA
## poly(bathrooms, 3)2  7.116e+06  2.576e+05  27.621  < 2e-16 ***
## poly(bathrooms, 3)3  2.093e+05  2.492e+05   0.840    0.401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 248600 on 15120 degrees of freedom
## Multiple R-squared:  0.5423, Adjusted R-squared:  0.5421
## F-statistic:  2240 on 8 and 15120 DF,  p-value: < 2.2e-16
```

```r
# Calculate the test data RSQ using the sum of squares formula

# First get y_hat values
test.pred = predict(poly_rooms_train, test_data)
```

```
## Warning in predict.lm(poly_rooms_train, test_data): prediction from a rank-
## deficient fit may be misleading
```

```r
# get y
test.y = test_data$price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.5280011
```

```r
# The Test Data R-Squared is 0.5280.
```

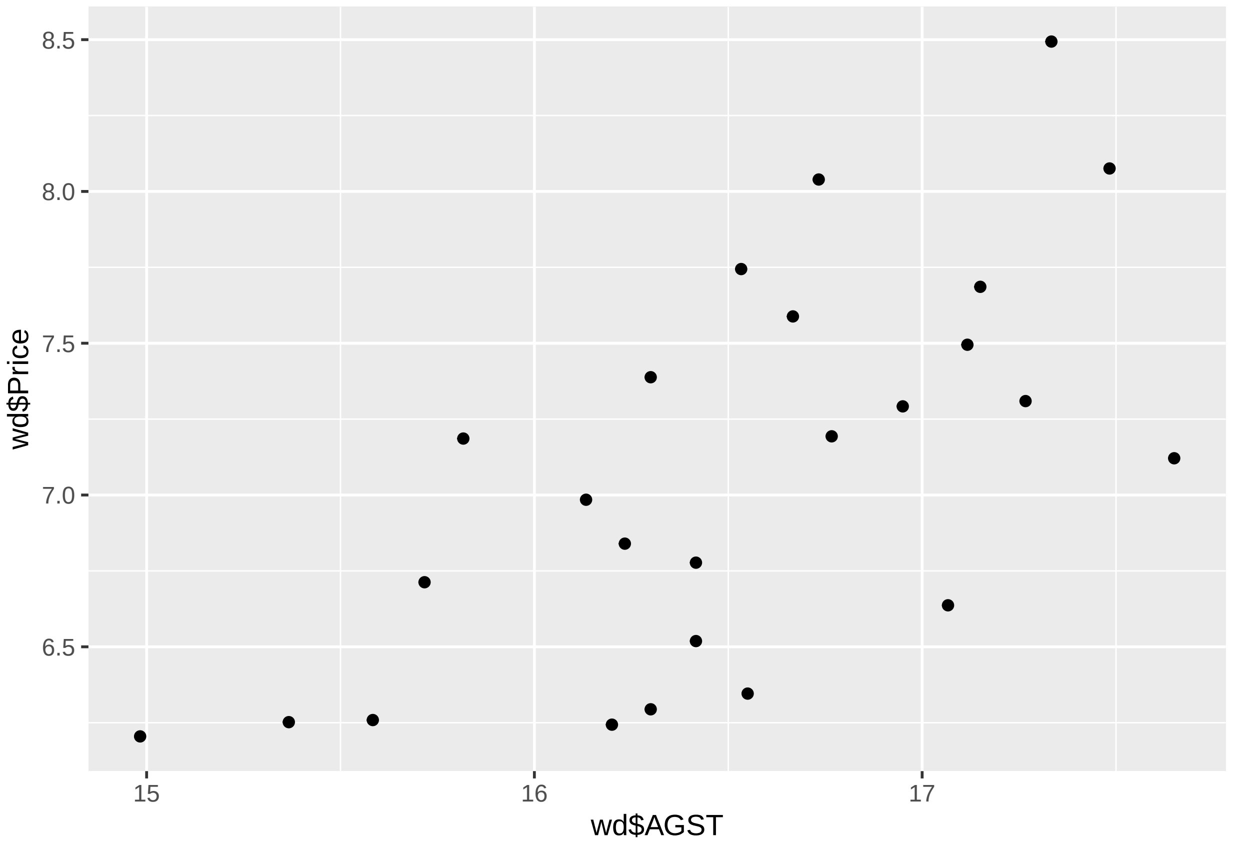
This results in the best R-Squared and Adjusted R-Squared value thus far at 0.5423 and 0.5421 respectively. Furthermore, the test data R-Squared is also the largest it's been at 0.5280.

Q8. Wine Pricing (20 Points)

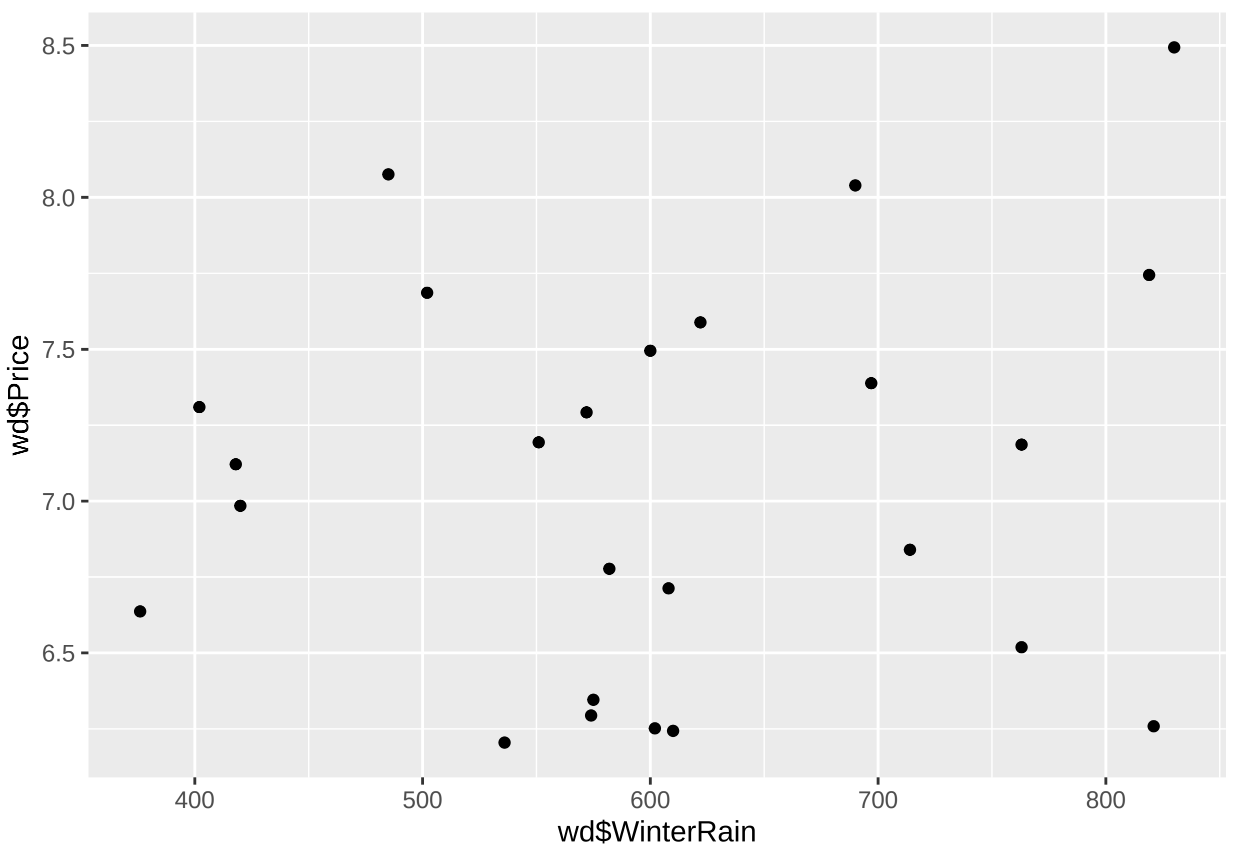
Answers:

Part 1. Preliminary Analysis

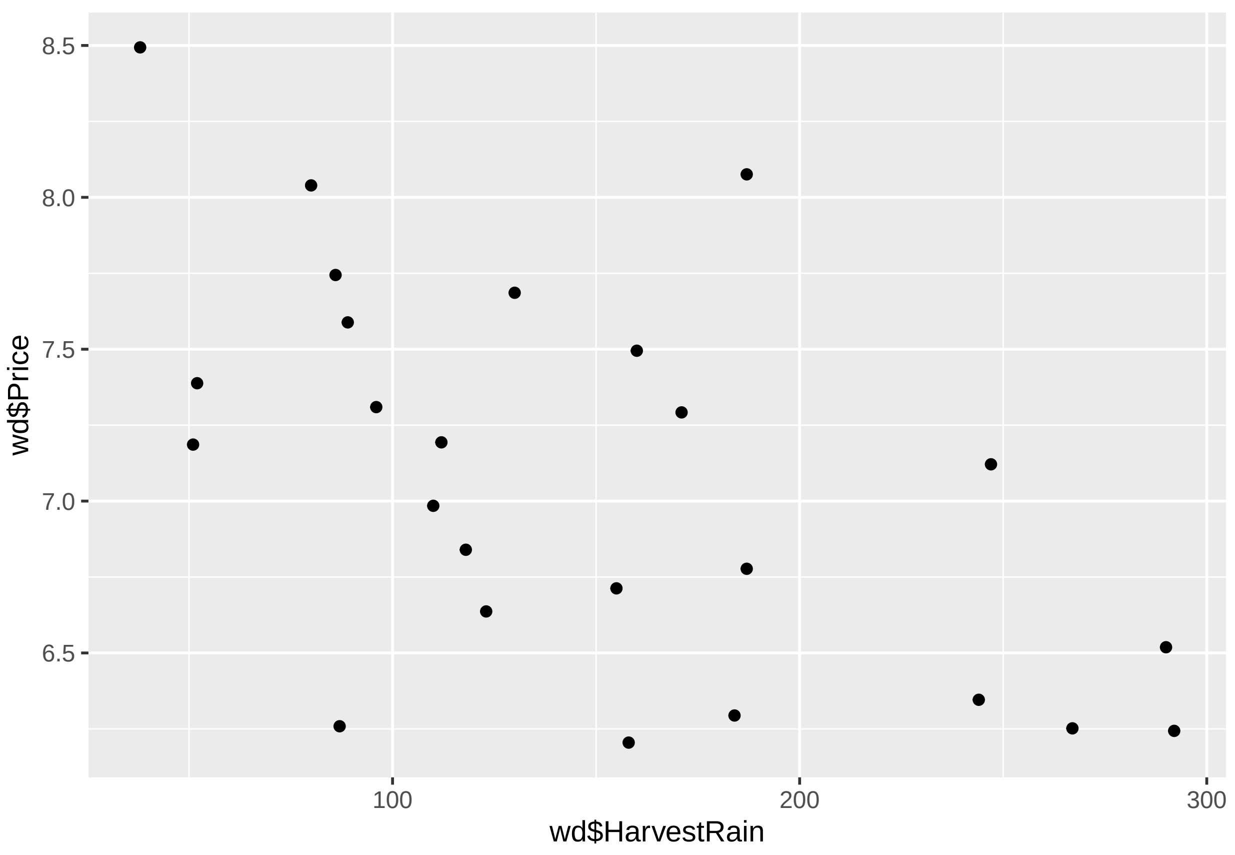
```r
# Load wine.csv
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")
wd = read.csv("wine.csv")

# Scatter plots for Price v.s. AGST, Price v.s. WinterRain, Price v.s. HarvestRain and Price v.s. Age.
# Price is on y-axis for all
ggplot(wd, aes(x=wd$AGST, y=wd$Price)) + geom_point()
```



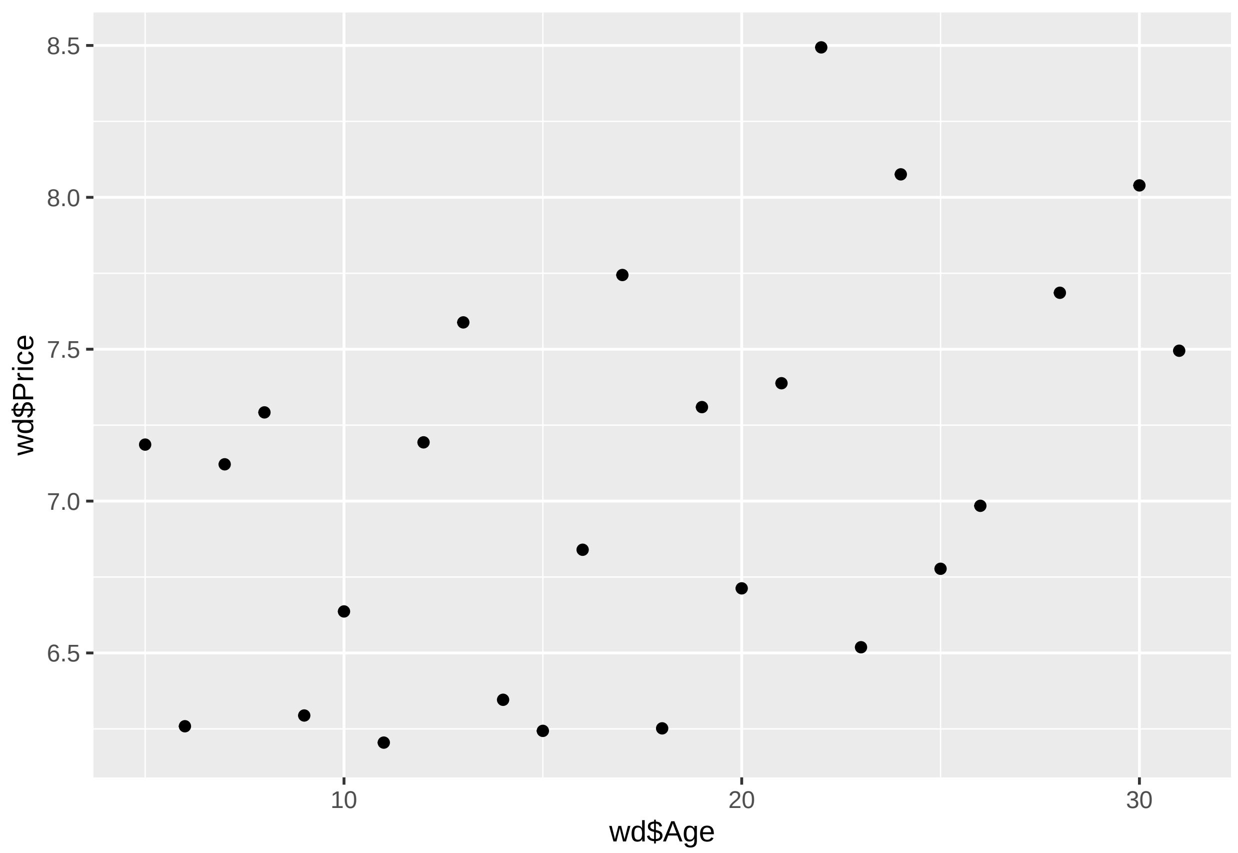
```r
ggplot(wd, aes(x=wd$WinterRain, y=wd$Price)) + geom_point()
```



```r
ggplot(wd, aes(x=wd$HarvestRain, y=wd$Price)) + geom_point()
```



```r
ggplot(wd, aes(x=wd$Age, y=wd$Price)) + geom_point()
```



AGST is the variable most correlated with Price as you can see a relatively clear positive trend in AGST as price increases. The second most would be HarvestRain with a negative trend.

```
# Getting the Pearson Coefficient
covPriceAGST = cov(wd$Price, wd$AGST)
pricevar = var(wd$Price)
agstvar = var(wd$AGST)

pearson_coeff = covPriceAGST/(sqrt(pricevar)*sqrt(agstvar))
pearson_coeff
```

```
## [1] 0.6595629
```

As we can see above the Pearson Coefficient is 0.65 which leans towards a positive correlation. Since Pearson's coefficient ranges from -1 to 1 (with 0 implying no correlation) we can see that there is, in-fact, a positive correlation.

Part 2. Marginal Regression Analysis.

```
library("margins")
wd_reg_1 = lm(Price ~ AGST, data=wd)

summary(wd_reg_1)
```

```
##
## Call:
## lm(formula = Price ~ AGST, data = wd)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max 
## -0.78450 -0.23882 -0.03727  0.38992  0.90318 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  -3.4178     2.4935  -1.371 0.183710    
## AGST          0.6351     0.1509   4.208 0.000335 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.4993 on 23 degrees of freedom
## Multiple R-squared:  0.435,  Adjusted R-squared:  0.4105 
## F-statistic: 17.71 on 1 and 23 DF,  p-value: 0.000335
```

```
margins(wd_reg_1)
```

```
## Average marginal effects

## lm(formula = Price ~ AGST, data = wd)

##    AGST
##  0.6351
```

The R-Squared for the regression is 0.435, and the marginal fitted coefficient values are 0.6351 for AGST.

Part 3. Multiple Regression Analysis.

```
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")
wtd = read.csv("winetest.csv")

wd_reg_2 = lm(Price ~ AGST + HarvestRain, data=wd)
summary(wd_reg_2)
```

```
## 
## Call:
## lm(formula = Price ~ AGST + HarvestRain, data = wd)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.88321 -0.19600  0.06178  0.15379  0.59722 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.20265    1.85443  -1.188 0.247585    
## AGST         0.60262    0.11128   5.415 1.94e-05 ***
## HarvestRain -0.00457    0.00101  -4.525 0.000167 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.3674 on 22 degrees of freedom
## Multiple R-squared:  0.7074, Adjusted R-squared:  0.6808 
## F-statistic: 26.59 on 2 and 22 DF,  p-value: 1.347e-06
```

```
# First get y_hat values
test.pred = predict(wd_reg_2, wtd)
# get y
test.y = wtd$Price
```

```r
# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.3613324
```

```r
#######################################################################################################

wd_reg_3 = lm(Price ~ AGST + Age, data=wd)
summary(wd_reg_3)
```

```
##
## Call:
## lm(formula = Price ~ AGST + Age, data = wd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66213 -0.32661 -0.01914  0.28574  0.83949
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.66805    2.37636  -1.123 0.273649
## AGST         0.56296    0.14655   3.841 0.000887 ***
## Age          0.02565    0.01287   1.994 0.058746 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4699 on 22 degrees of freedom
## Multiple R-squared:  0.5215, Adjusted R-squared:  0.478
## F-statistic: 11.99 on 2 and 22 DF,  p-value: 0.0003013
```

```r
# First get y_hat values
test.pred = predict(wd_reg_3, wtd)
# get y
test.y = wtd$Price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.4210777
```

```r
#######################################################################################################

wd_reg_4 = lm(Price ~ AGST + WinterRain, data=wd)
summary(wd_reg_4)
```

```
##
```

```
## Call:
## lm(formula = Price ~ AGST + WinterRain, data = wd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.77970 -0.41353  0.05376  0.34062  0.64113
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.5567664  2.5227794  -2.599   0.0164 *
## AGST         0.7552047  0.1420822   5.315 2.46e-05 ***
## WinterRain   0.0019100  0.0007255   2.633   0.0152 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4452 on 22 degrees of freedom
## Multiple R-squared:  0.5704, Adjusted R-squared:  0.5313
## F-statistic:  14.6 on 2 and 22 DF,  p-value: 9.203e-05
```

```
# First get y_hat values
test.pred = predict(wd_reg_4, wtd)
# get y
test.y = wtd$Price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.8134612
```

```
############################################################################################

wd_reg_5 = lm(Price ~ AGST + FrancePop, data=wd)
summary(wd_reg_5)
```

```
##
## Call:
## lm(formula = Price ~ AGST + FrancePop, data = wd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67201 -0.32895 -0.03006  0.26631  0.82395
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.864e-01  3.043e+00   0.226 0.823641
## AGST         5.559e-01  1.459e-01   3.811 0.000956 ***
## FrancePop   -5.629e-05  2.688e-05  -2.094 0.048018 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4662 on 22 degrees of freedom
## Multiple R-squared:  0.5289, Adjusted R-squared:  0.4861
## F-statistic: 12.35 on 2 and 22 DF,  p-value: 0.0002536
```

```r
# First get y_hat values
test.pred = predict(wd_reg_5, wtd)
# get y
test.y = wtd$Price

# Calculate all of the variables
ss.total = sum((test.y - mean(test.y))^2)
ss.residual = sum((test.y - test.pred)^2)
ss.regression = sum((test.pred - mean(test.y))^2)
ss.total = (ss.regression+ss.residual)
test.rsq = 1 - ss.residual/ss.total
test.rsq
```

```
## [1] 0.3726983
```

+HarvestRain: R-Squared for the model rises hugely to 0.7074, but the test data R-Squared is a lot lower at 0.3613.

+Age: R-Squared for the model rises again to 0.5215, and the test data R-Squared is at 0.4210.

+WinterRain: R-Squared for the model is now at a relatively high 0.5704, but the test data rises hugely to 0.8134

+FrancePop: R-Squared for the model is now 0.4662 and the test data R-squared is 0.3727

As we can see by the above results, the most important factor to wine prices is clearly WinterRain, which has the highest R-Squared, and when the model is validated with the test data, it is even more clearly the most important factor.

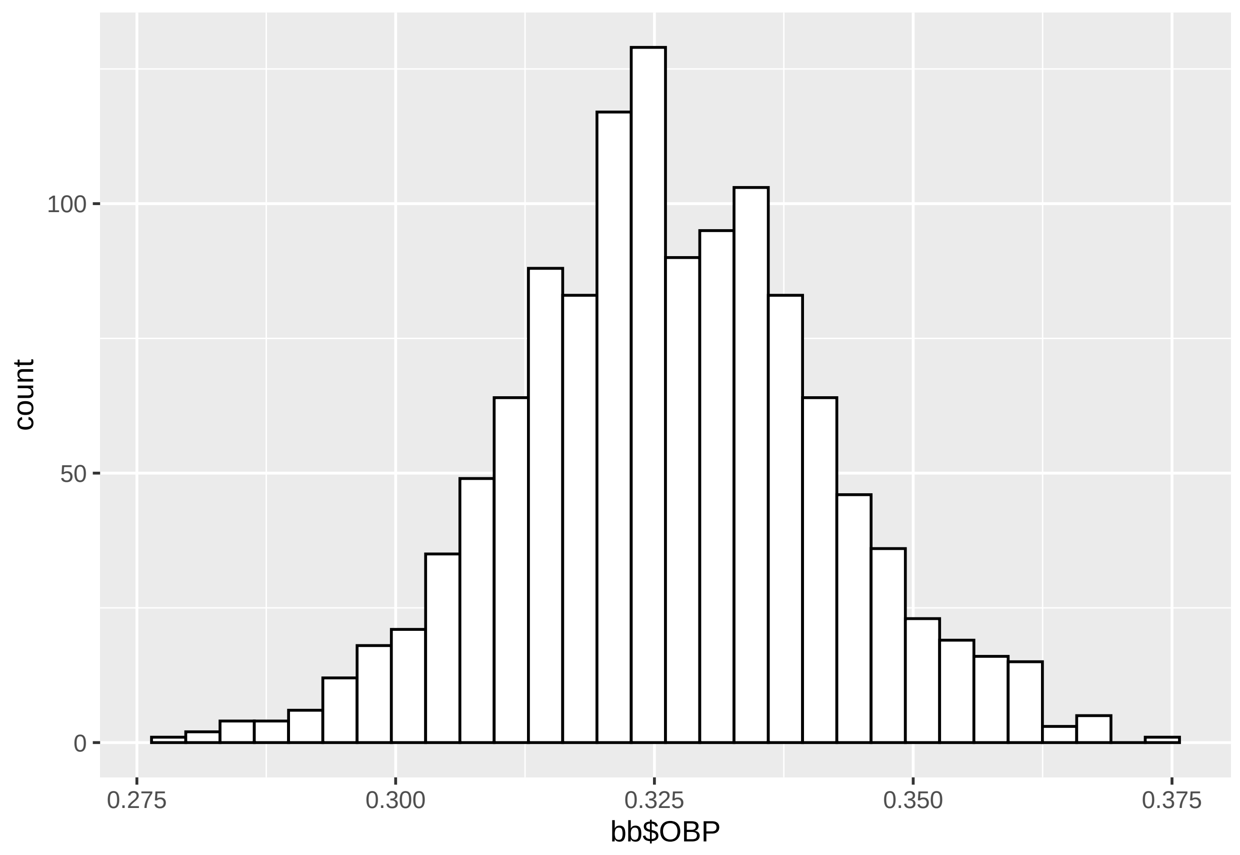
Q9. Moneyball: The Analytics Edge in Sports (30 points)

Answer:

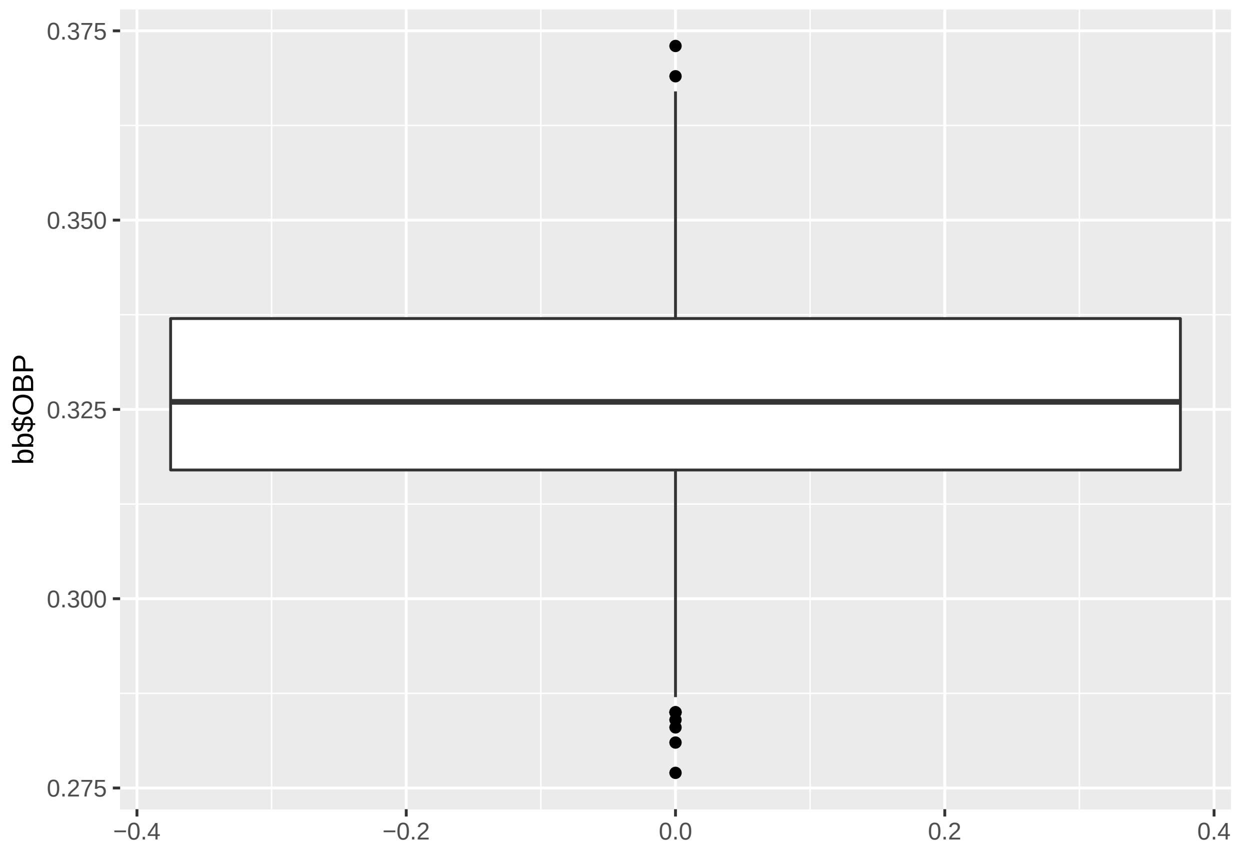
Part 1. Preliminary Analysis.

```r
# setwd("C:/Users/Cougar/Dropbox/University/Term 11/STAD80/hw1_files/")
bb = read.csv("baseball.csv")

# On-Base Percentage (OBP) plots, mean and median
ggplot(data = NULL, df, mapping = aes(x=bb$OBP)) + geom_histogram(fill="white",color="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
ggplot(data = NULL, aes(y=bb$OBP)) + geom_boxplot(outlier.colour="black")
```



```
obp_mean = mean(bb$OBP)
obp_median = median(bb$OBP)
obp_mean
```

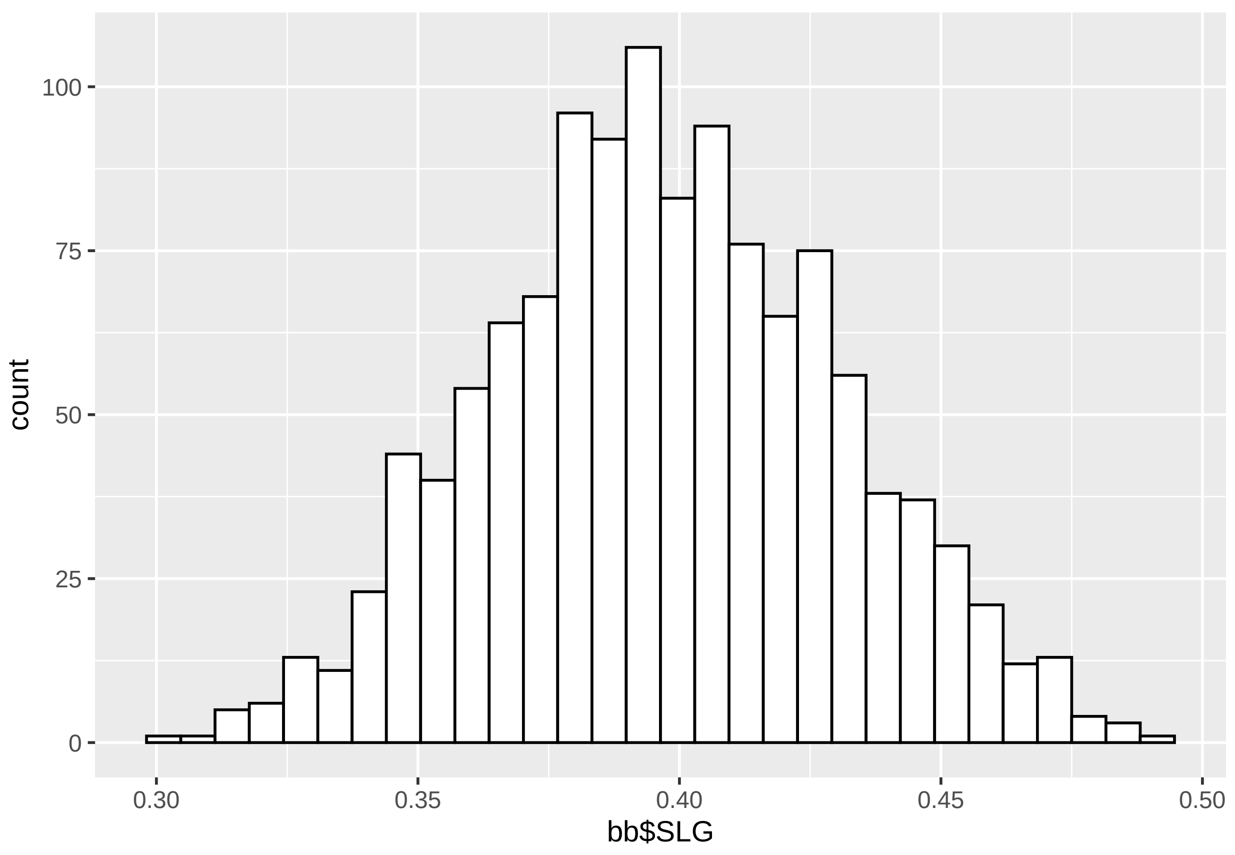
```
## [1] 0.3263312
```

```
obp_median
```

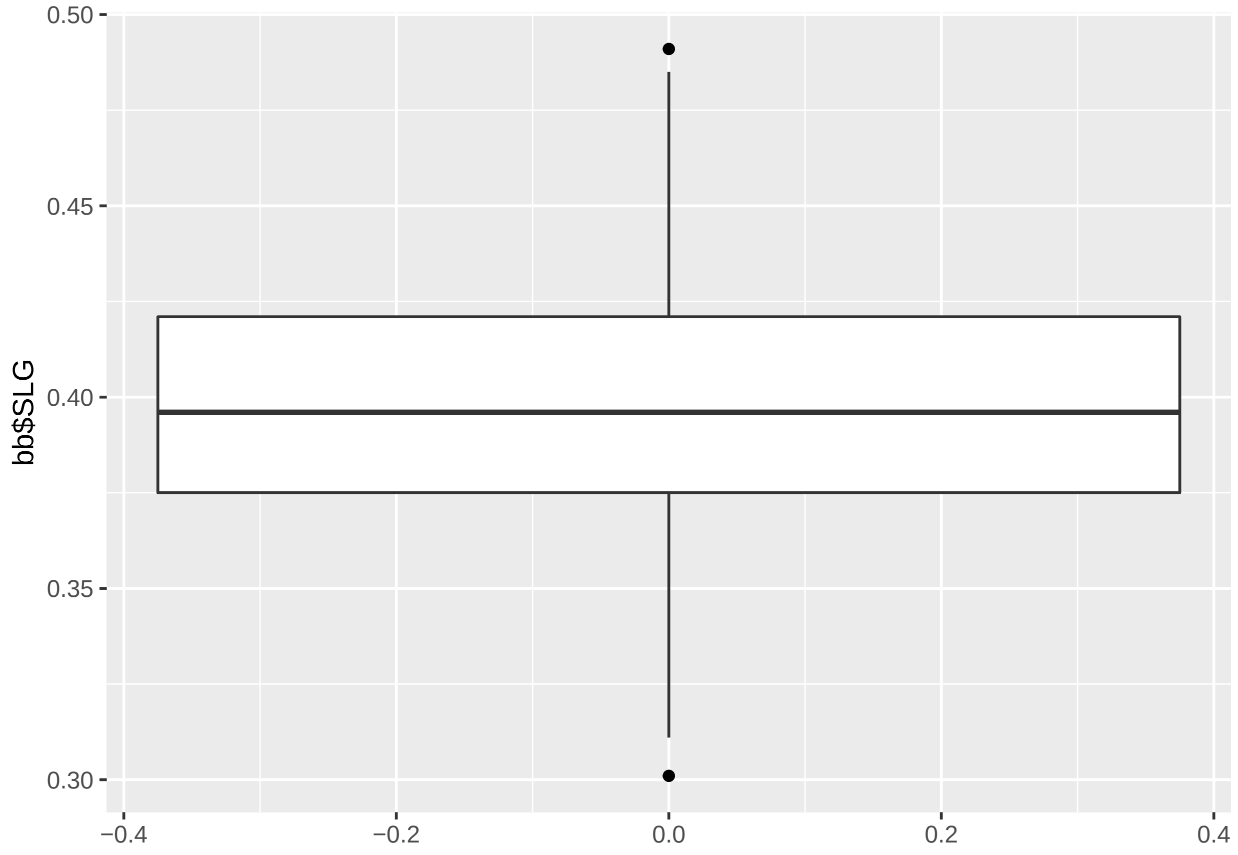
```
## [1] 0.326
```

```
# Slugging Percentage (SLG) plots, mean and median
ggplot(data = NULL, df, mapping = aes(x=bb$SLG)) + geom_histogram(fill="white",color="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
ggplot(data = NULL, aes(y=bb$SLG)) + geom_boxplot(outlier.colour="black")
```



```
slg_mean = mean(bb$SLG)
slg_median = median(bb$SLG)
slg_mean
```

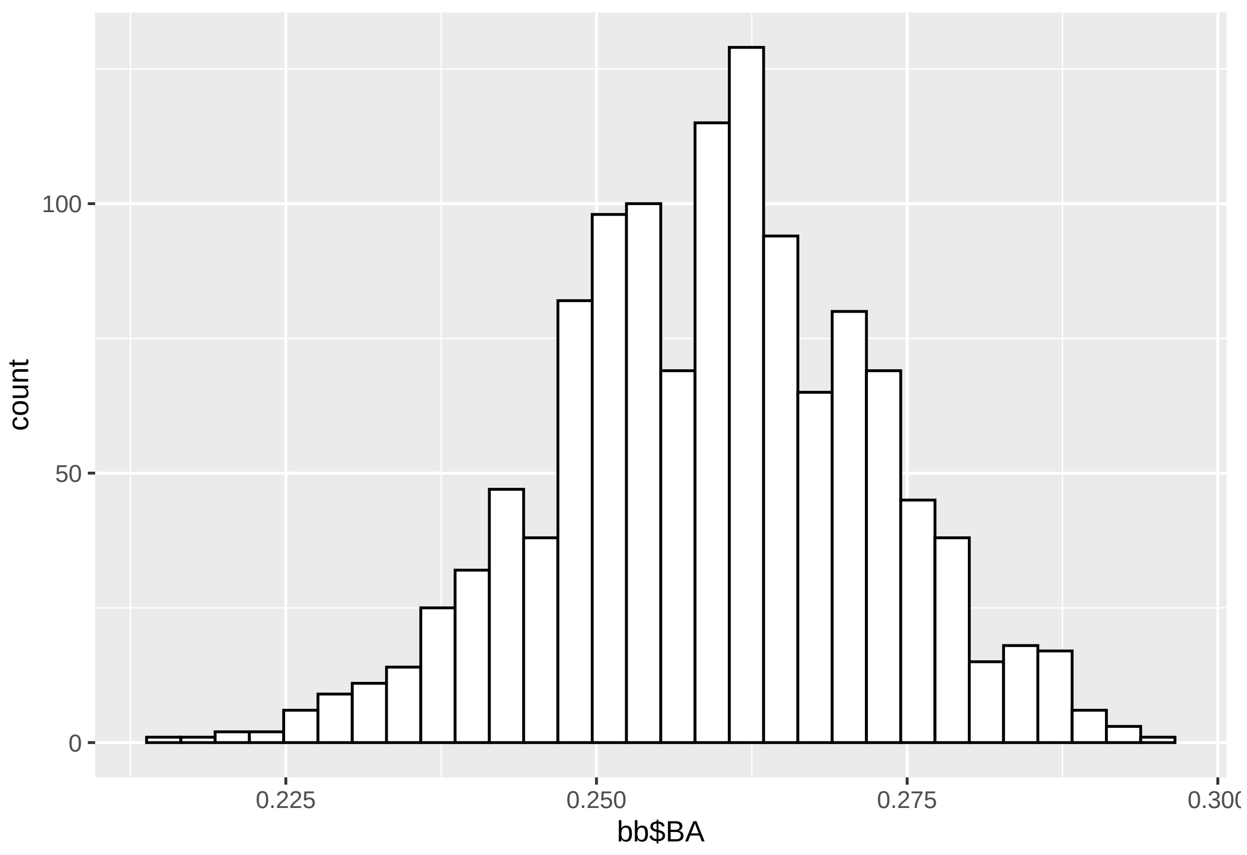
```
## [1] 0.3973417
```

```
slg_median
```

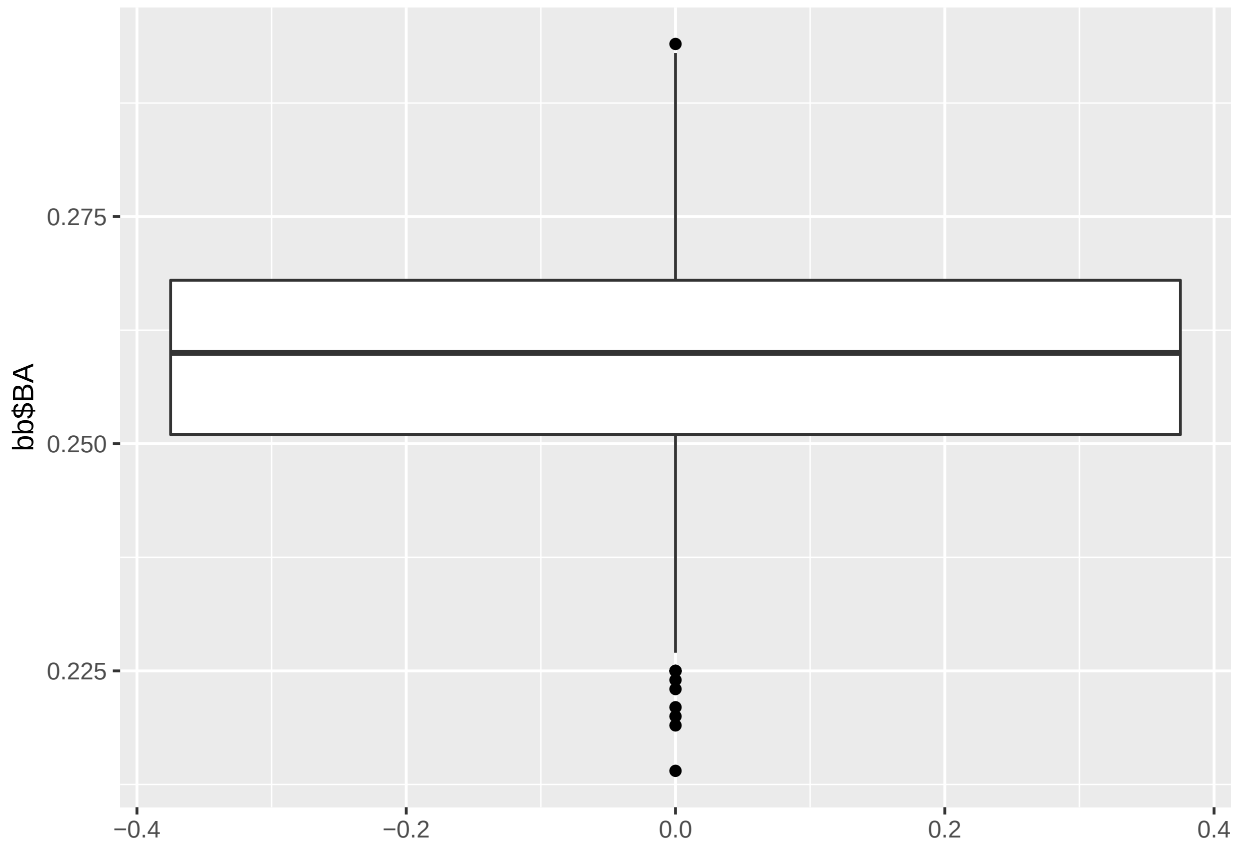
```
## [1] 0.396
```

```
# Batting Average (BA) plots, mean and median
ggplot(data = NULL, df, mapping = aes(x=bb$BA)) + geom_histogram(fill="white",color="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
ggplot(data = NULL, aes(y=bb$BA)) + geom_boxplot(outlier.colour="black")
```



```
ba_mean = mean(bb$BA)
ba_median = median(bb$BA)
ba_mean
```

```
## [1] 0.2592727
```

```
ba_median
```

```
## [1] 0.26
```

OBP(mean, median) = (0.326, 0.326) SLG(mean, median) = (0.397, 0.396) BA(mean, median) = (0.259, 0.260)

Part 2. Marginal Regression Analysis.

```
# Regression of RS on BA, OBP, and SLG
bb_reg_1 = lm(RS ~ BA, data=bb)
bb_reg_2 = lm(RS ~ OBP, data=bb)
bb_reg_3 = lm(RS ~ SLG, data=bb)

summary(bb_reg_1)
```

```
##
## Call:
## lm(formula = RS ~ BA, data = bb)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -158.429  -36.057   -1.064   35.018  179.518
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -805.51      29.51  -27.30   <2e-16 ***
## BA           5864.84     113.68   51.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.48 on 1230 degrees of freedom
## Multiple R-squared:  0.6839, Adjusted R-squared:  0.6837
## F-statistic:  2662 on 1 and 1230 DF,  p-value: < 2.2e-16
```

```
margins(bb_reg_1)
```

```
## Average marginal effects

## lm(formula = RS ~ BA, data = bb)

##    BA
##  5865
```

```
summary(bb_reg_2)
```

```
##
## Call:
## lm(formula = RS ~ OBP, data = bb)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -122.129  -27.110    1.284   26.441  135.265
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1076.6       24.7  -43.59   <2e-16 ***
## OBP           5490.4       75.6   72.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.82 on 1230 degrees of freedom
## Multiple R-squared:  0.8109, Adjusted R-squared:  0.8107
## F-statistic:  5274 on 1 and 1230 DF,  p-value: < 2.2e-16
```

```
margins(bb_reg_2)
```

```
## Average marginal effects

## lm(formula = RS ~ OBP, data = bb)

##   OBP
##  5490
```

```
summary(bb_reg_3)
```

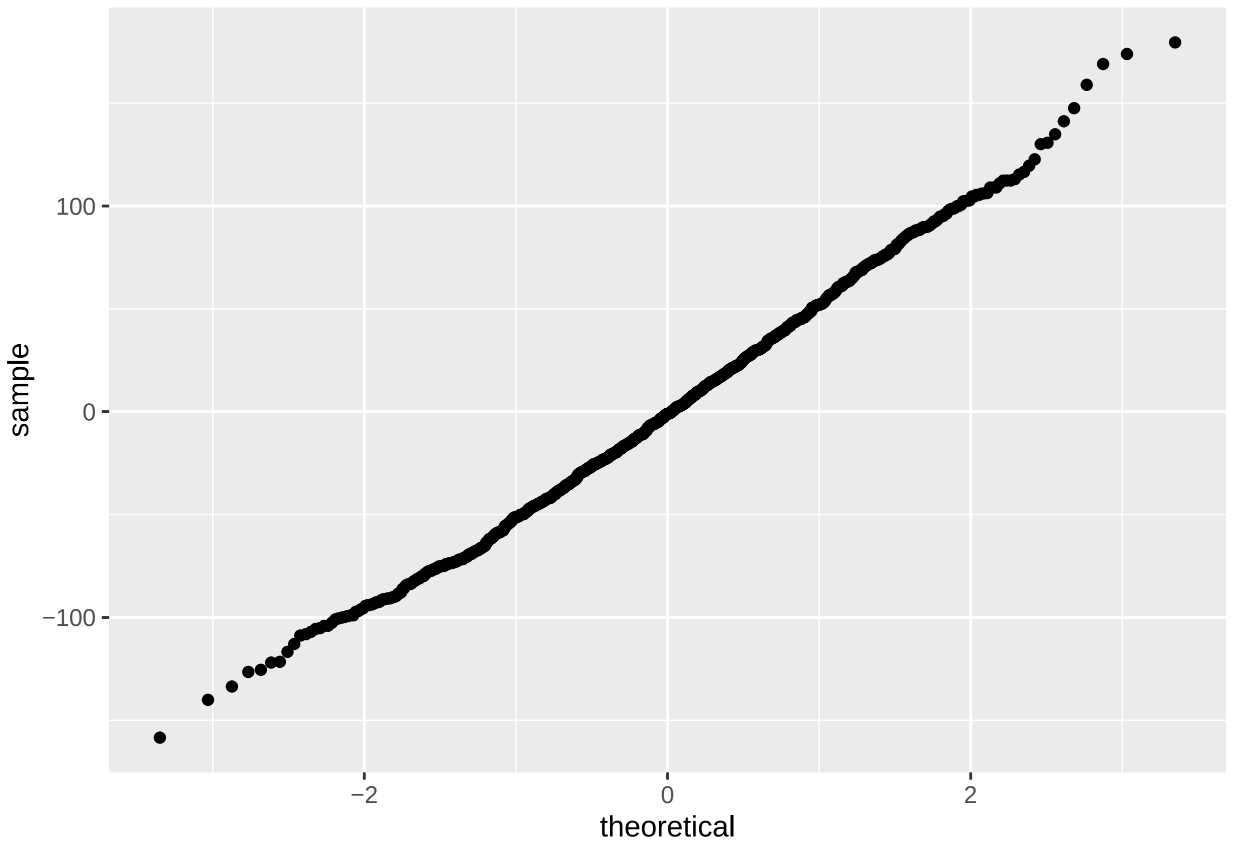
```
##
## Call:
## lm(formula = RS ~ SLG, data = bb)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -119.919  -23.666   -1.541   22.353  131.812
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -289.37      12.35  -23.43   <2e-16 ***
## SLG          2527.92      30.98   81.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.16 on 1230 degrees of freedom
## Multiple R-squared:  0.8441, Adjusted R-squared:  0.844
## F-statistic:  6659 on 1 and 1230 DF,  p-value: < 2.2e-16
```

```
margins(bb_reg_3)
```

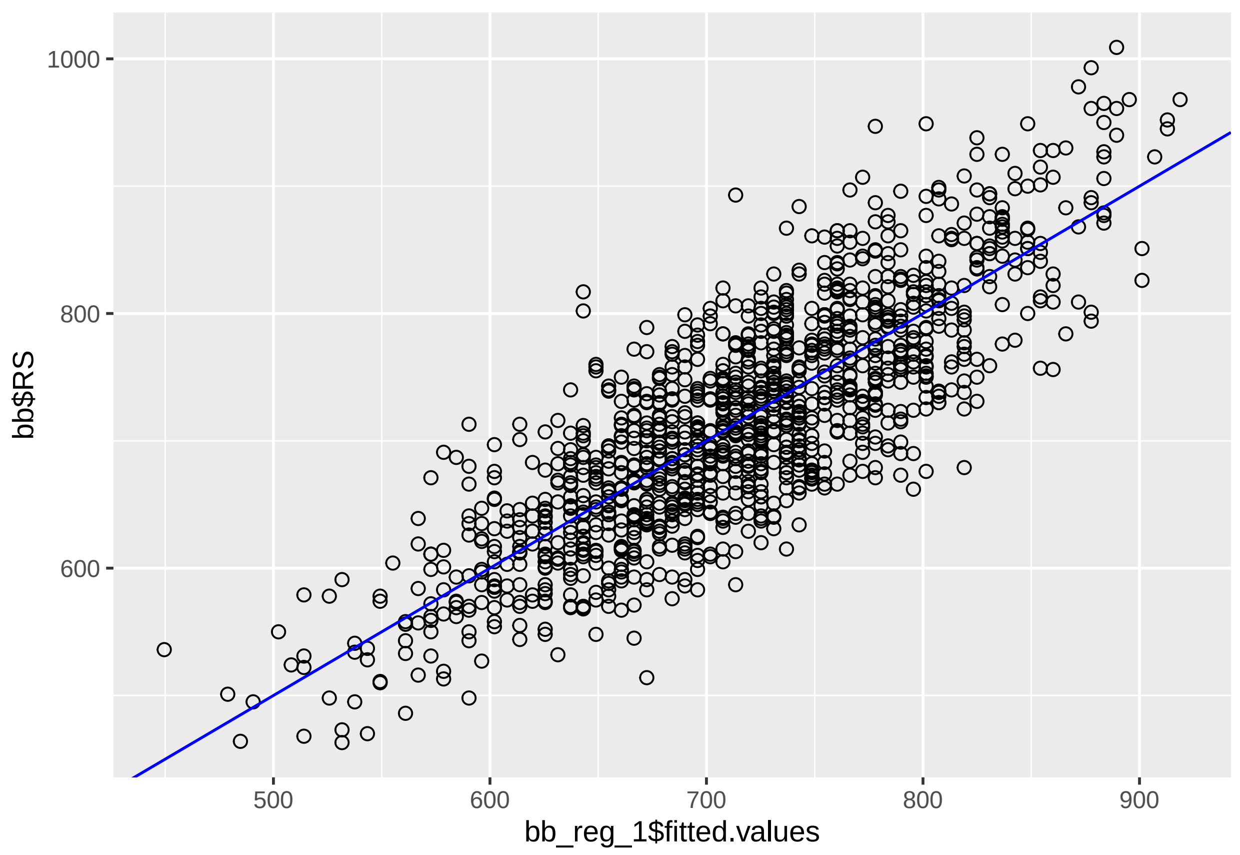
```
## Average marginal effects

## lm(formula = RS ~ SLG, data = bb)

##   SLG
##  2528
```

```
ggplot(bb, aes(sample = bb_reg_1$residuals)) + stat_qq()
```

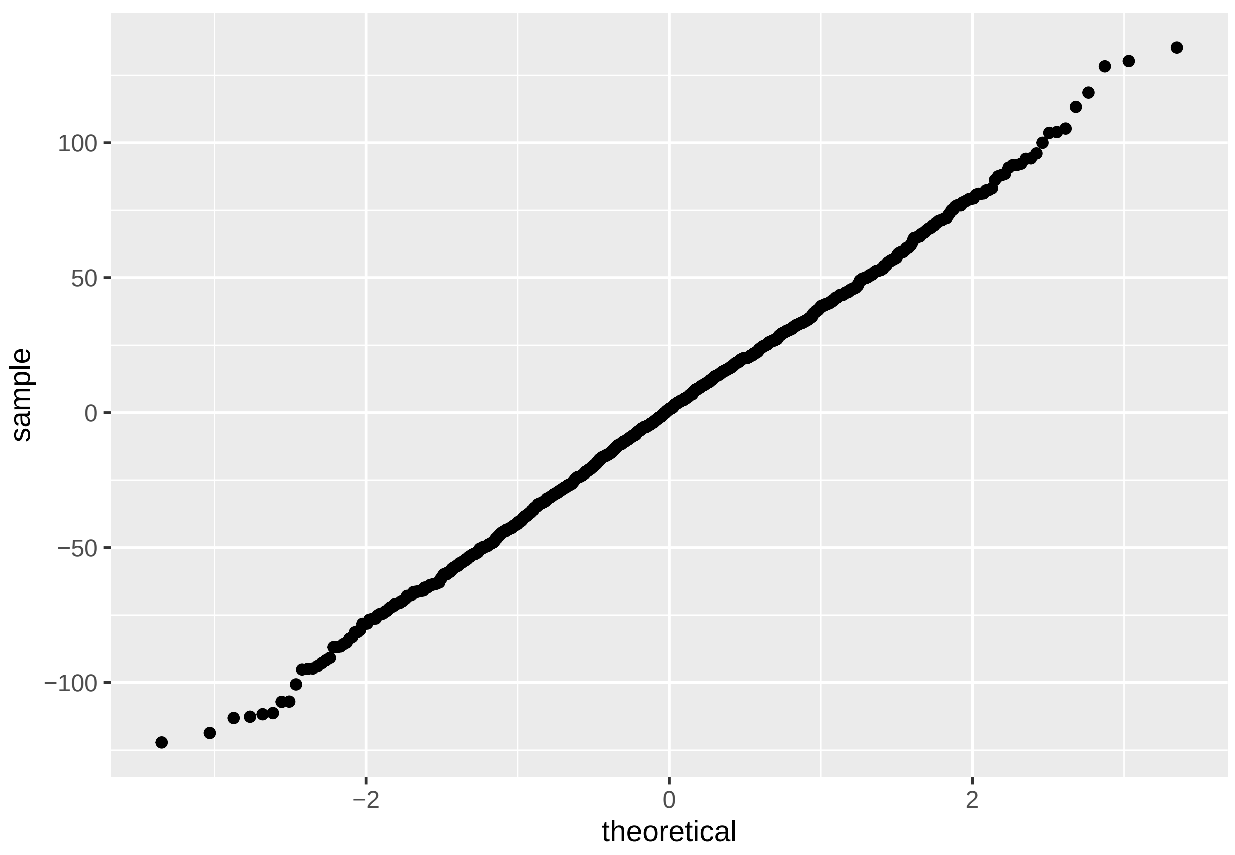


```
ggplot(bb, aes(x=bb_reg_1$fitted.values, y=bb$RS)) + geom_point(size=2, shape=1) + geom_abline(method="
```

```
## Warning: Ignoring unknown parameters: method, formula
```

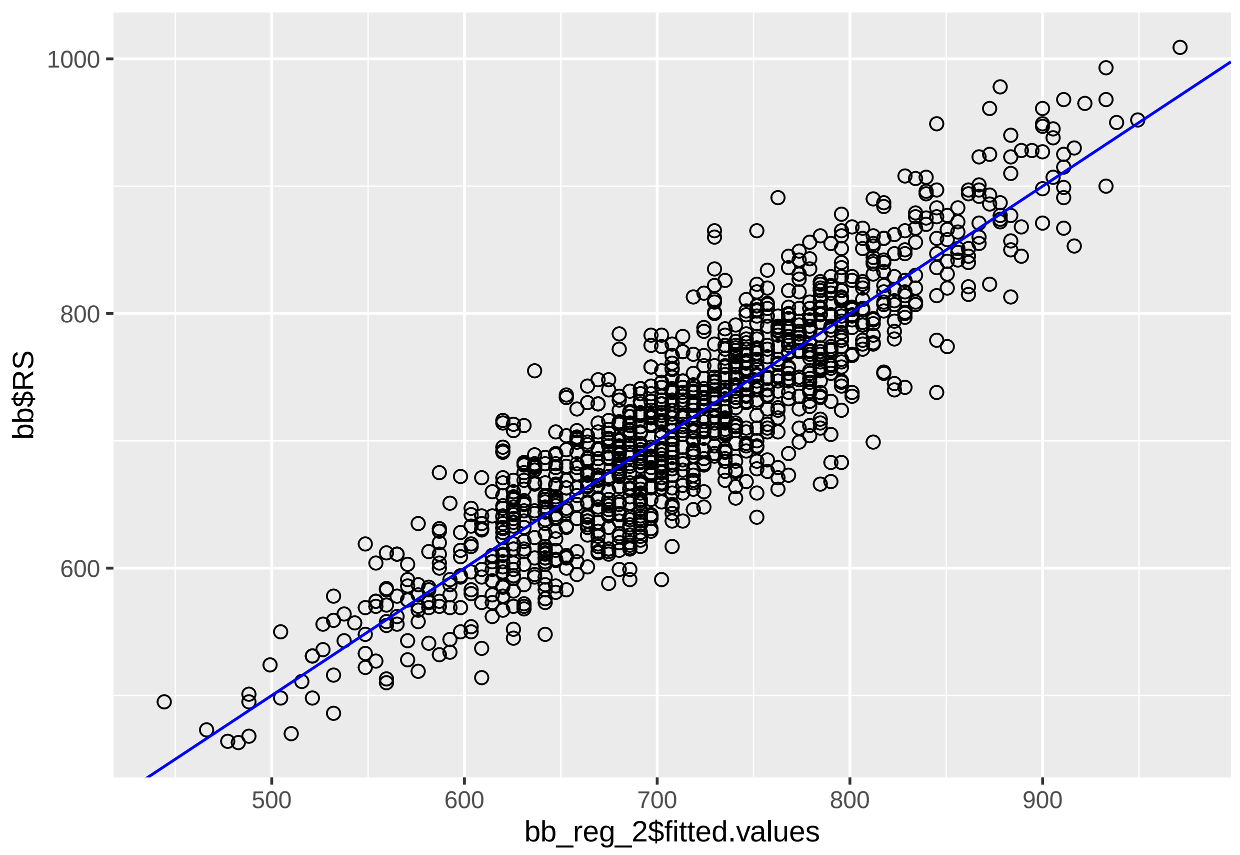


```r
ggplot(bb, aes(sample = bb_reg_2$residuals)) + stat_qq()
```

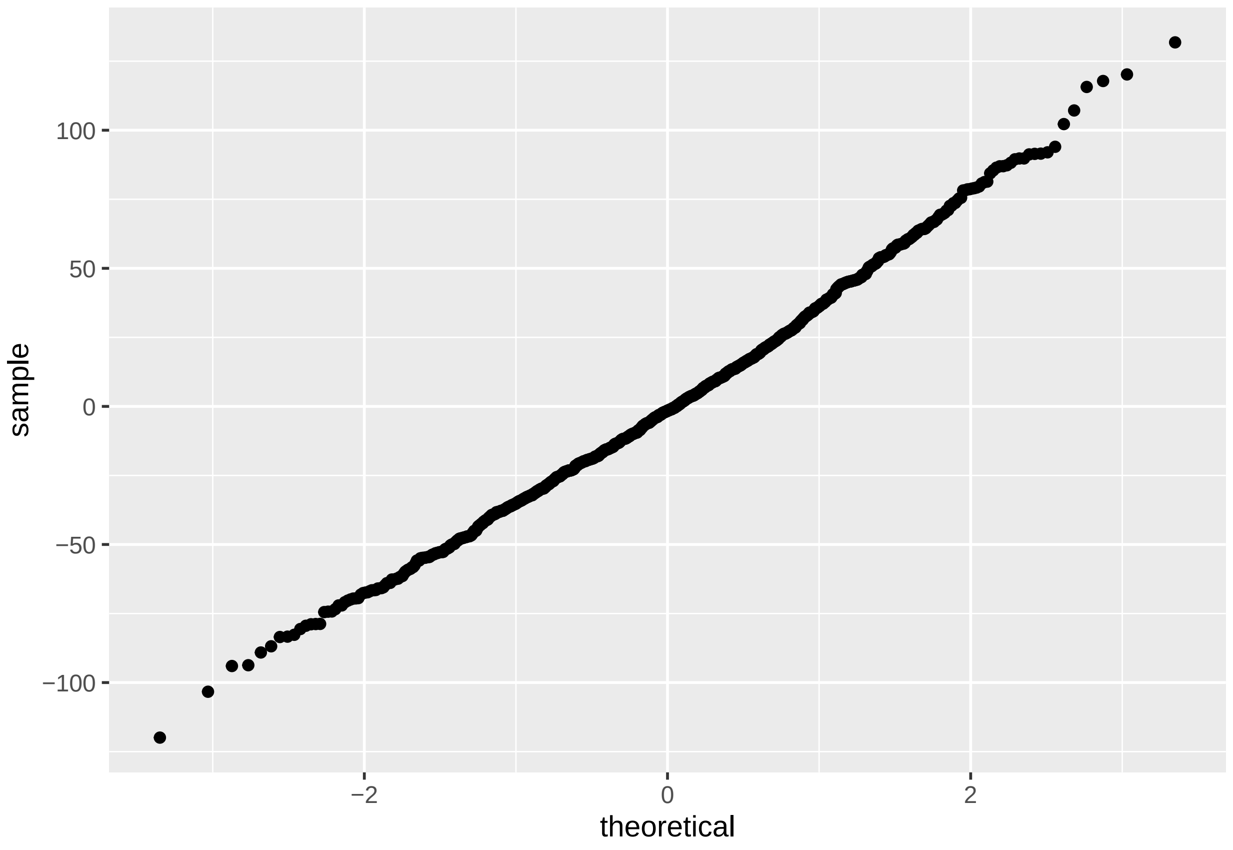


```
ggplot(bb, aes(x=bb_reg_2$fitted.values, y=bb$RS)) + geom_point(size=2, shape=1) + geom_abline(method="
```

```
## Warning: Ignoring unknown parameters: method, formula
```

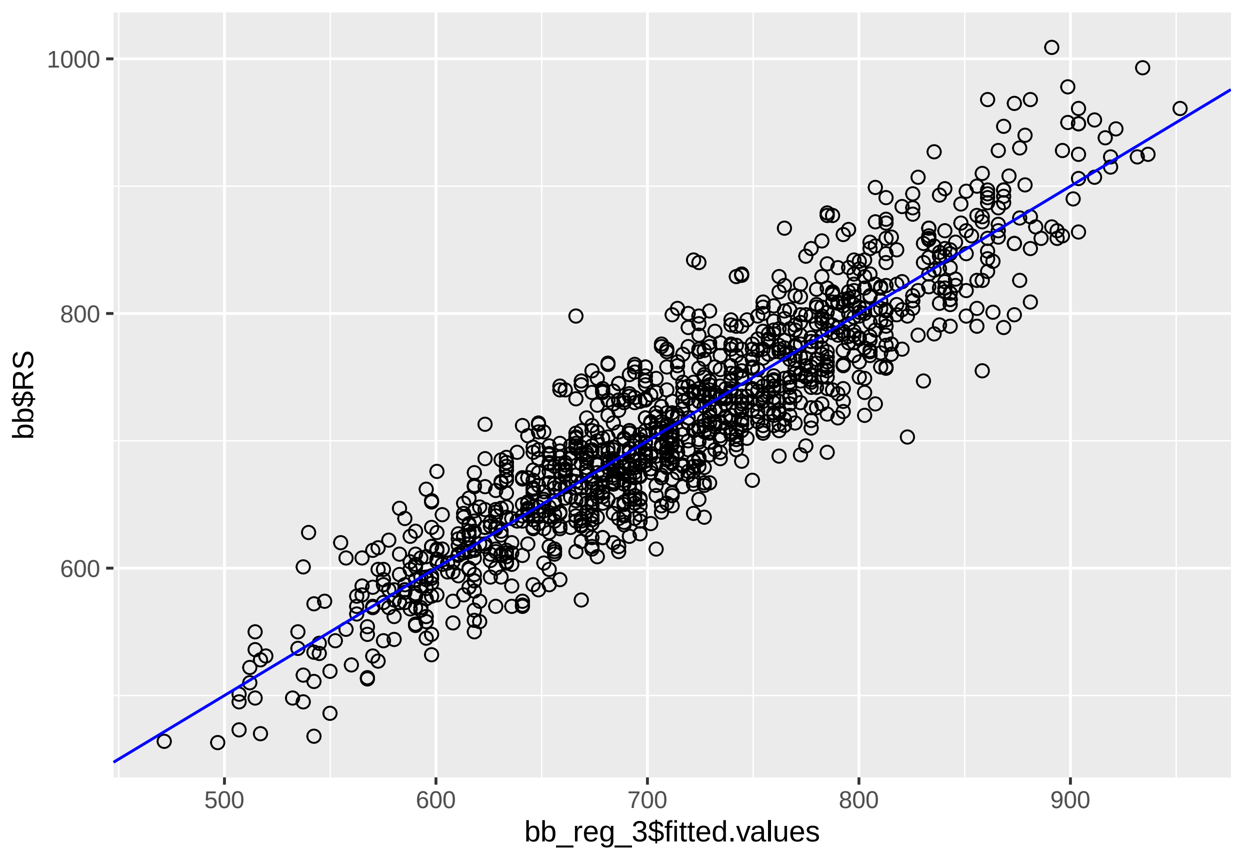


```
ggplot(bb, aes(sample = bb_reg_3$residuals)) + stat_qq()
```



```
ggplot(bb, aes(x=bb_reg_3$fitted.values, y=bb$RS)) + geom_point(size=2, shape=1) + geom_abline(method="
```

```
## Warning: Ignoring unknown parameters: method, formula
```



The marginal coefficients for BA, OBP, and SLG are 5864.84, 5490.386, and 2527.925 respectively.

This intuition does not since the R^2 we obtain for RS ~ BA is the lowest. By our regression analysis it is actually SLG that reflects how well the batter hits the ball.

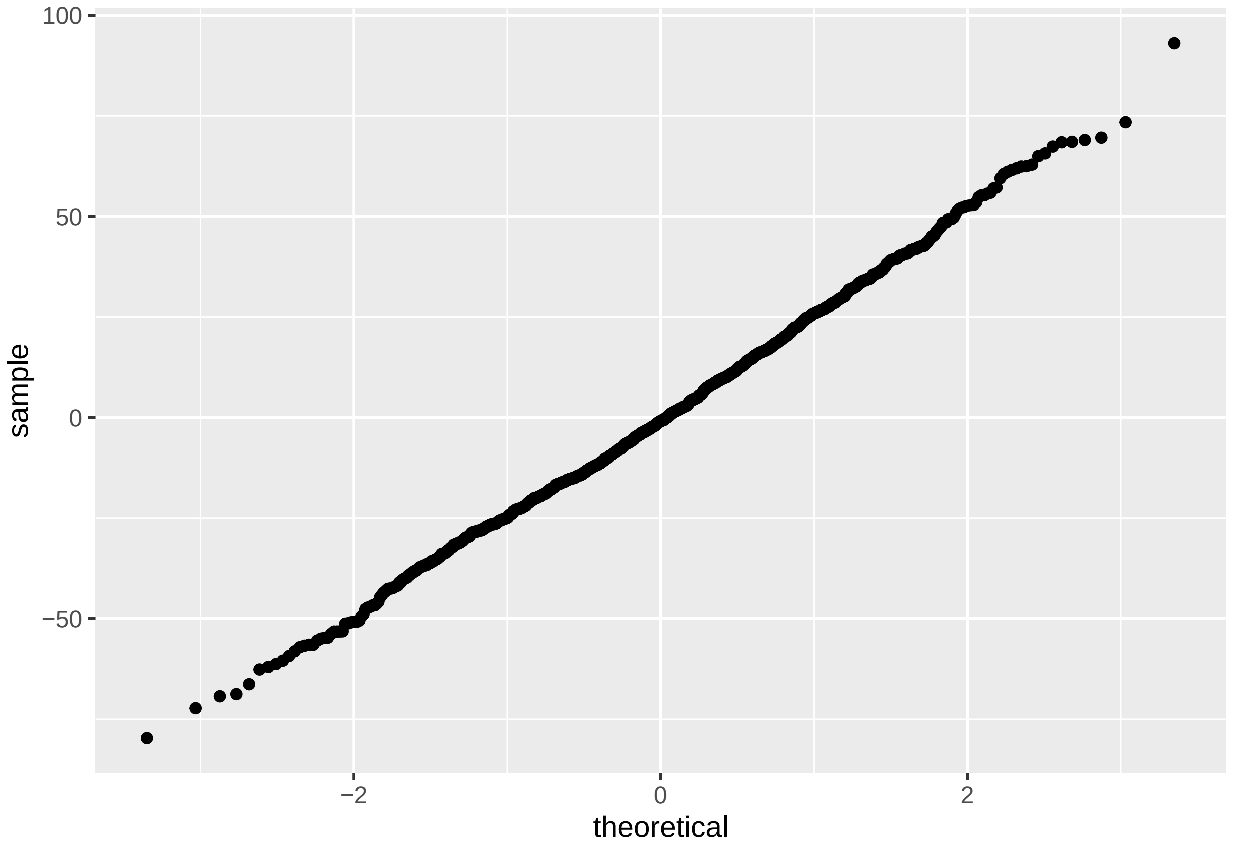
Part 3. Multiple Regression Analysis.

```r
bb_reg_4 = lm(RS ~ BA + SLG + OBP, data=bb)

summary(bb_reg_4)
```

```
##
## Call:
## lm(formula = RS ~ BA + SLG + OBP, data = bb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -79.693 -16.667  -0.892  16.556  93.068
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -806.08      17.39 -46.348   <2e-16 ***
## BA           -134.90     113.73  -1.186    0.236
## SLG          1533.88      37.76  40.623   <2e-16 ***
## OBP          2900.94      97.87  29.640   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 25.12 on 1228 degrees of freedom
## Multiple R-squared:  0.9249, Adjusted R-squared:  0.9247
## F-statistic:  5040 on 3 and 1228 DF,  p-value: < 2.2e-16
```

```
ggplot(bb, aes(sample = bb_reg_4$residuals)) + stat_qq()
```



```
bb_reg_5 = lm(RS ~ BA + SLG, data=bb)

summary(bb_reg_5)
```

```
##
## Call:
## lm(formula = RS ~ BA + SLG, data = bb)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -115.432  -23.284   -2.048   21.068  113.415
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -551.08      19.79  -27.85   <2e-16 ***
## BA           1904.66     118.56   16.07   <2e-16 ***
## SLG          1943.77      46.00   42.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.88 on 1229 degrees of freedom
## Multiple R-squared:  0.8711, Adjusted R-squared:  0.8709
## F-statistic:  4154 on 2 and 1229 DF,  p-value: < 2.2e-16
```

Estimated Coefficients: BA - -134.90 at 0.236 significance (Not very good) SLG - 1533.88 at very good significance <2e-16 OBP - 2900.94 at very good significance <2e-16

The residual plot is comparable to the residual plot of our RS ~ BA regression. The head-end is just slightly different.

The main difference is the fitted coefficient for BA, when done regression solely on it we get a coefficient of 5864.84 at <2e-16, which is very good however now it's a negative value with really bad significance, not even one significant code.

The R^2 for RS ~ BA + SLG + OBP is a good bit higher at 0.9429 meanwhile the R^2 for RS ~ BA + SLG is at 0.8711. Both are good but I would prefer the former model.

Part 4. Back to 2001 and Reshape the Baseball World.

```
# Note: Oakland athletics had 103 wins in 2002

# First create RD = RS - RA and add it to the bb dataframe

RD = bb$RS - bb$RA
bb$RD = RD

bb_2001 = bb %>% filter(Year < 2002)

bb_reg_6 = lm(W ~ RD, data=bb_2001)
bb_reg_7 = lm(RS ~ OBP + SLG, data=bb_2001)
bb_reg_8 = lm(RA ~ OOBP + OSLG, data=bb_2001)

# oakland_oobp_oslg = data.frame(OOBP=0.307, OSLG=0.373)

# oakland_obp_slg = data.frame(OBP=0.349, SLG=0.430)

# oakland_ra = predict(bb_reg_8, oakland_oobp_oslg)

# oakland_rs = predict(bb_reg_7, oakland_obp_slg)

# oakland_rd = oakland_rs - oakland_ra

# oakland_w = predict(bb_reg_6, RD=oakland_rd)
# oakland_w

# mean(oakland_w)
# mean(oakland_rs)
# mean(oakland_rd)
# mean(oakland_ra)

oakland_ra = bb_reg_8$coefficients['(Intercept)'] + bb_reg_8$coefficients['OOBP']*0.307 + bb_reg_8$coeff

oakland_rs = bb_reg_7$coefficients['(Intercept)'] + bb_reg_7$coefficients['OBP']*0.349 + bb_reg_7$coeff

oakland_w = bb_reg_6$coefficients['(Intercept)'] +bb_reg_6$coefficients['RD']*(oakland_rs - oakland_ra)

oakland_w
```

```
## (Intercept)
##    103.1386
```

Our prediction of 103.1386 (Rounded down to 103) is the exact same as the number of wins Oakland received in 2002.