Assignment 1 Submission

Gao Haochun A0194525Y Ge Siqi A0194550A Wang Pei A0194486M
Wei Yifei A0203451W

```
# Load packages
library(Matrix) # Matrix operations
library(MASS) # For Moore-Penrose pseudo-inverse ginv()
library(doParallel) # Parallel computing
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(nortest) # Anderson-Darling normality test
library(numDeriv) # genD()
```

Q1

I. Which of the 4 data violate(s) the rank condition in OLS regression?

The full assumption can be denoted by rank(X) = k, where X is an n x k matrix, n observations, k independent variables. Thus, full rank assumption can be interpreted as: all the columns are linearly independent. In other words, all independent variables (including the constant) are linearly independent.

```
A = matrix(c(1,1,2,2),nrow=2)
B = matrix(c(1,1,2,2,1,2),nrow=3)
C = matrix(c(1,1,2,0),nrow=2)
D = matrix(c(1,2,3,2,0,2),nrow=3)
rankMatrix(A)
```

```
## [1] 1
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 4.440892e-16
```

```
rankMatrix(B)
```

```
## [1] 2
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.661338e-16
```

```
rankMatrix(C)
```

```
## [1] 2
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 4.440892e-16
```

```
rankMatrix(D)
```

```
## [1] 2
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.661338e-16
```

A: The reduced form has one zero column. The two columns are linearly dependent and $Rank(A) = 1 \neq 2$. Therefore, A is not full column rank and A violates rank condition in OLS regression.

B: The reduced form does not have zero column. Therefore, two columns are linearly independent and $Rank(B) = 2$. Thus, B is full column rank and does not violate rank condition in OLS regression.

C: The reduced form does not have zero column. Therefore, two columns are linearly independent and $Rank(C) = 2$. Thus, C is full column rank and does not violate rank condition in OLS regression.

D: The reduced form has one zero column. The two columns are linearly dependent and $Rank(D) = 2$. Therefore, D is not full column rank and violates rank condition in OLS regression.

II. Suppose that you wanted to build a model of the approval ratings of major party nominees for US president. You included the following 4 independent variables: Years holding elected office; Party; Gender; Indicator variable for being married to a former president. Does this violate the rank condition? Explain.

Yes, this violates the rank condition.

The assumption of full rank condition is violated if one independent variable is perfectly linearly dependent on another independent variable, or an independent variable is perfectly jointly determined by other independent variables.

In this context, the indicator variable for being married to a former president is perfectly linearly dependent on the gender of the nominees. Based on the history of US presidents, all past presidents are male and there is no cases of a man marrying to male US past presidents. Therefore, if an nominee is male (Gender

= Male), the indicator variable for being married to a former president can only be 0 (not married to a former president). In other words, the indicator variable for being married to a former president is perfectly dependent on the gender. If both "gender" and "being married to a former president" are included as independent variables in the matrix X, the column of "gender" and the column of "being married to a former president" will be linearly dependent. Since there is perfect linear dependence between independent variables, this violates the rank condition in OLS regression.

III. Give a different real-world example where the rank condition fails.

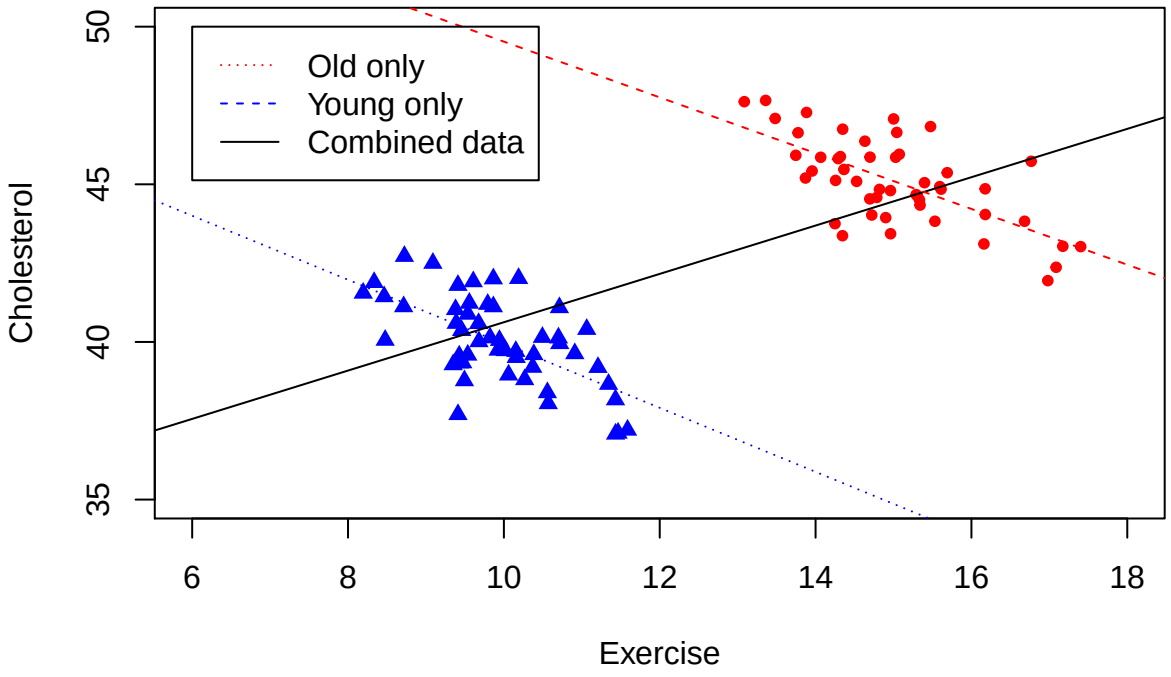
One example is to compare the weight of secondary school student between Cedar Girls' Secondary School (girls school) and Victoria School (boys school). In this case, the dependent variable is the weight of a student, and the independent variables include a dummy variable indicating the student's school, gender, age, height etc. The independent variable school is defined as school = 0 if the student is from Cedar Girls' Secondary School, school = 1 if the student is from Victoria School. The independent variable gender is defined as gender = 0 if the student is female, gender = 1 if the student is male.

Because all the students from Cedar Girls' Secondary School are girls and all those from Victoria School are boys, the independent variable school is perfectly linearly dependent on another independent variable gender. If we include both "gender" and "school" as independent variables in the matrix X, the column of "school" and the column of "gender" will be linearly dependent. The assumption of full rank condition is violated if one independent variable is perfectly linearly dependent on another independent variable. Therefore, the rank condition fails in this example.

Q2

I. Reproducing the plot

```
df = read.csv("exercise_and_cholesterol.csv")
old_df = df[df$Age=="Old",]
young_df = df[df$Age=="Young",]

attach(df)
plot(Cholesterol ~ Exercise,col=ifelse(Age=="Old","red","blue"),
     pch=ifelse(Age=="Old",20,17),ylim=c(35,50),xlim=c(6,18))

all_model = lm(Cholesterol ~ Exercise, data=df)
young_model = lm(Cholesterol ~ Exercise, data=young_df)
old_model = lm(Cholesterol ~ Exercise, data=old_df)
abline(all_model$coefficients[1],all_model$coefficients[2], lty="solid")
abline(young_model$coefficients[1],young_model$coefficients[2], lty="dotted",col="blue")
abline(old_model$coefficients[1],old_model$coefficients[2], lty="dashed",col="red")
legend(6, 50, legend=c("Old only", "Young only","Combined data"),
       col=c("red", "blue","black"), lty=c("dotted","dashed","solid"), cex=1)
```



II. Are the three lines giving consistent or conflicting insights? Explain why.

Yes, the three lines are giving conflicting insights. Below are the insights from the three lines:

- The regression line based on the combined data implies that there is a positive linear correlation between cholesterol and exercise.
- The regression line based on the data from only young people implies that there is a negative linear correlation between cholesterol and exercise.
- The regression line based on the data from only old people implies that there is a negative linear correlation between cholesterol and exercise.
- Old people tend to have higher cholesterol level than the young. This can be implied from the regression line of old only data being higher than the regression line of young only data.

The insights drawn from subgroups are different from that drawn from the combined data. This is because age is another factor affecting the level of cholesterol as well as the amount of exercise. To be specific, age is positively correlated with cholesterol level. The conflict in insights can be due to omitted variable bias in the model of ***Cholesterol*** ~ ***Exercise***.

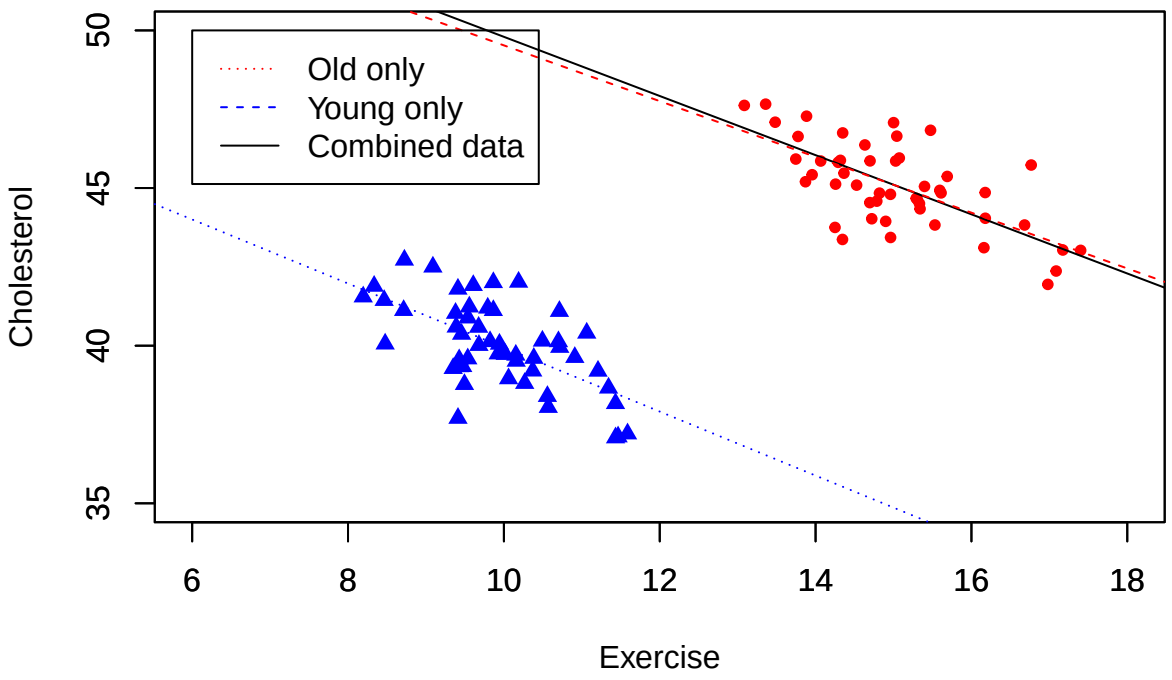
III. Can you propose an alternative estimation using all the data in order to obtain the same insight as the two separate regressions using only the old or the young people?

An alternative estimation is to include age as an independent variable in the model and make it a dummy variable: age = 1 if one is young, age = 0 if one is old. Run a linear regression of cholesterol against age and exercise using all the data. This gives us three regression lines with similar trend as shown on the graph below. Now, the same insights can be generated.

In addition, by running the linear regression using the model ***Cholesterol*** ~ ***Exercise + Age***, the coefficient of the independent variable exercise is -0.9384, which suggests a negative correlation between exercise and cholesterol of a person i.e. the increase in the amount of exercise by 1 unit results in a decrease in the

cholesterol level of that particular person by 0.9384 units in average. Also, the coefficient of the independent variable Age is -9.8511, which suggests that the young has a lower cholesterol level than old people by 9.8511 units in average. Then, this further proves that we obtain the same insight as the two separate regressions.

```
plot(young_df$Cholesterol ~ young_df$Exercise,col="blue",pch=17,
     ylim=c(35,50),xlim=c(6,18),xlab="",ylab="")
par(new=TRUE)
plot(old_df$Cholesterol ~ old_df$Exercise,col="red",pch=20,
     ylim=c(35,50),xlim=c(6,18),xlab="Exercise",ylab="Cholesterol")

all_model = lm(Cholesterol ~ Exercise + Age, data=df)
young_model = lm(Cholesterol ~ Exercise, data=young_df)
old_model = lm(Cholesterol ~ Exercise, data=old_df)
abline(all_model$coefficients[1],all_model$coefficients[2], lty="solid")
abline(young_model$coefficients[1],young_model$coefficients[2], lty="dotted",col="blue")
abline(old_model$coefficients[1],old_model$coefficients[2], lty="dashed",col="red")
legend(6, 50, legend=c("Old only", "Young only","Combined data"),
       col=c("red", "blue","black"), lty=c("dotted","dashed","solid"), cex=1)
```



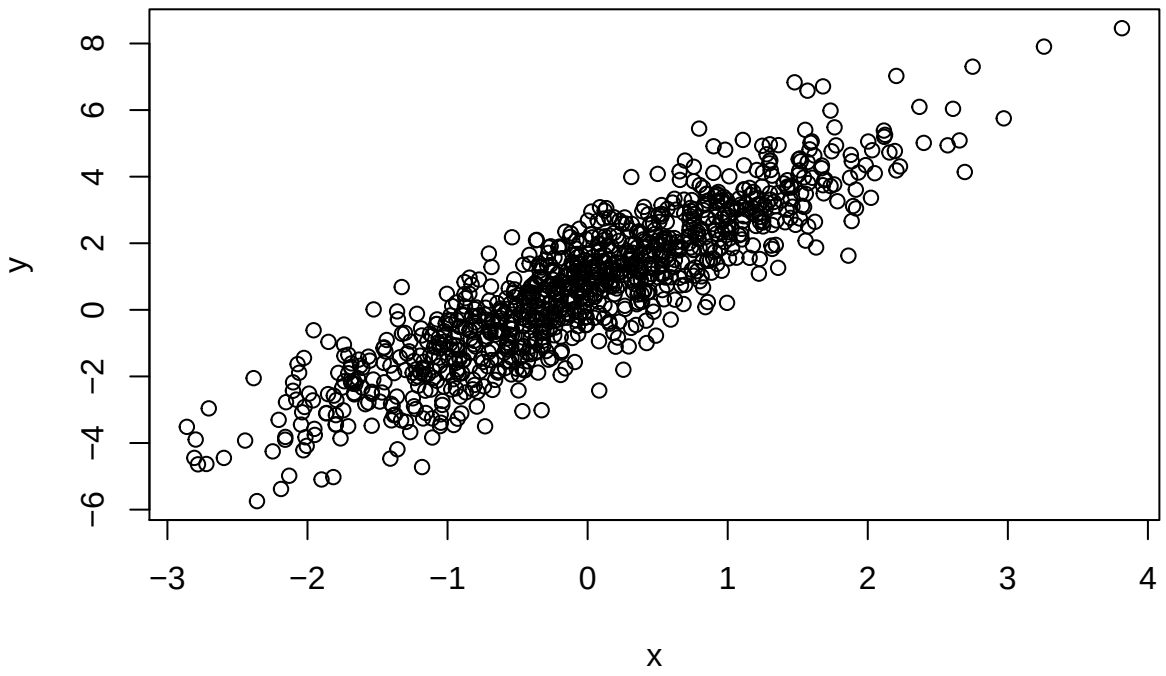
Q3

I. Generate a random sample with 1000 observations. Using the simulated data and the following four computational methods to estimate the value of the model parameters. Do the four methods give the same estimates?

Data generation

```
set.seed(37)
# Data generation
x = rnorm(1000) # Sample 1000 points from N(0, 1)
```

```r
e = rnorm(1000)
y = 0.7 + 2*x + e
plot(y ~ x)
```



a. Use R native function lm()

$$\hat{y} = 0.734 + 1.954x$$

```r
# Use lm()
model = lm(y ~ x)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3179 -0.6318  0.0383  0.6764  3.2130
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.73407    0.03216   22.82   <2e-16 ***
## x            1.95394    0.03165   61.74   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.017 on 998 degrees of freedom
## Multiple R-squared:  0.7925, Adjusted R-squared:  0.7923
## F-statistic:  3812 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
# y = 0.734 + 1.954 * x
```

Using lm(), $\hat{\beta_0}$ is 0.73407 and $\hat{\beta_1}$ is 1.95394.

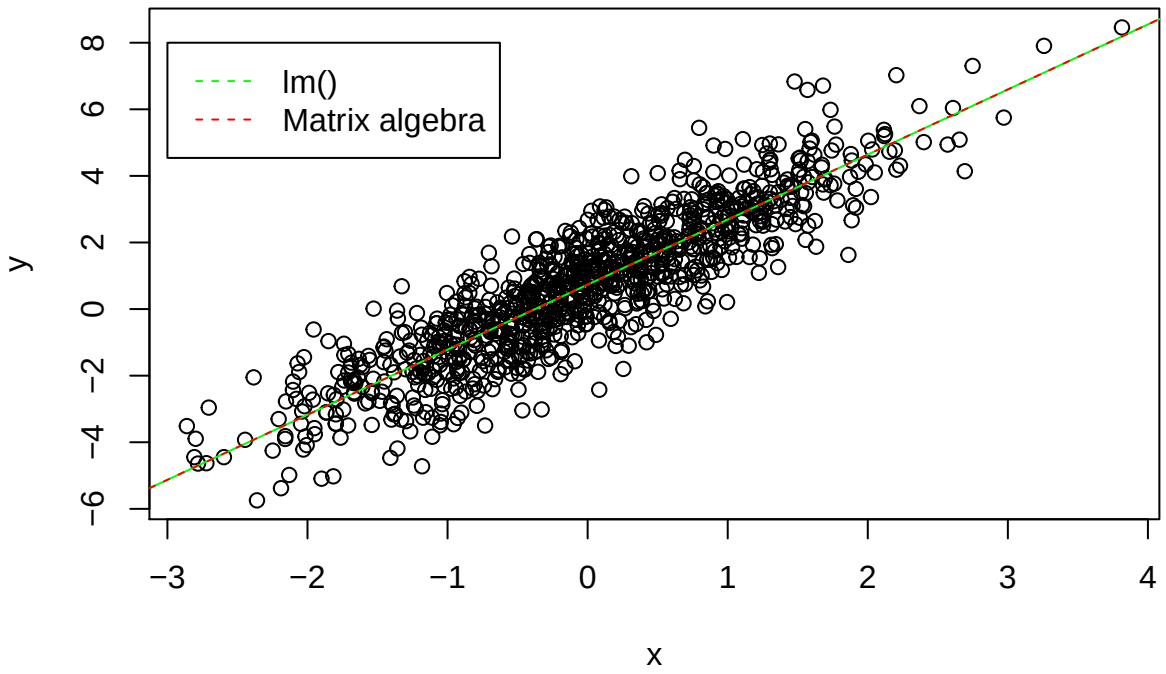
b. Use matrix algebra

$$\hat{\beta} = (X^T X)^{-1} X^T Y \implies \hat{y} = 0.734 + 1.954x$$

```
library(MASS)
# Use matrix algebra
X = cbind(rep(1,1000),x) # Add bias(constant for intercept) to data matrix
Y = y
beta_h = ginv(t(X)%*%X)%*%t(X)%*%Y
beta_h # \beta0 = 0.734, \beta1 = 1.954
```

```
##           [,1]
## [1,] 0.7340735
## [2,] 1.9539367
```

```
plot(y ~ x)
abline(model$coefficients[1],model$coefficients[2],col="green")
abline(beta_h[1],beta_h[2],col="red",lty="dashed")
legend(-3,8,legend=c("lm()", "Matrix algebra"),col=c("green","red"),lty=c("dashed","dashed"))
```



Using matrix algebra, $\hat{\beta_0}$ is 0.73407 and $\hat{\beta_1}$ is 1.95394.

c. Use R's optim()

$$\hat{\beta} = argmin_{\beta} \sum_{i=1,\ldots,n} (y_i - \beta_0 - \beta_1 \times x_i)^2 \implies \hat{y} = 0.734 + 1.954x$$

```
X = cbind(rep(1,1000), x)
Y = y
# Objective function
fun = function(beta) {
  sum((Y - X%*%beta)^2)
}
# Start optimization with random initialization
optim(runif(2),fun) # \beta0 = 0.734, \beta1 = 1.954
```

```
## $par
## [1] 0.734061 1.953818
##
## $value
## [1] 1031.934
##
## $counts
## function gradient
##       57       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Using R's optim(), $\hat{\beta}_0$ is 0.73406 and $\hat{\beta}_1$ is 1.95382.

d. Use formula for 1 variable regression:

$$\hat{\beta}_1 = \frac{Cov(x_i, y_i)}{Var(x_i)} \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \implies \hat{y} = 0.734 + 1.954x$$

```
beta1 = cov(x,y)/var(x)
beta0 = mean(y) - beta1*mean(x)
beta0 # 0.734
```

```
## [1] 0.7340735
```

```
beta1 # 1.954
```

```
## [1] 1.953937
```

Using formula for 1 variable regression, $\hat{\beta}_0$ is 0.73407 and $\hat{\beta}_1$ is 1.95394.

In conclusion, based on the values of estimates from all four methods, all four methods give the same estimates.

II. Now generate S = 10000 independent random samples each with N observations. For each of the random samples, run an OLS regression. Record all the S estimates on x.

Plot the distributions when N=25, 100. Compare the means and variances of the distributions. Are they different? Do they follow normal distributions?

a.

```
set.seed(37)
S = 10000
single_loop = function(N) {
  x = rnorm(N)
  e = rnorm(N)
  Y = 0.7 + 2*x + e
  X = cbind(rep(1,N),x)
  beta_h = ginv(t(X)%*%X)%*%t(X)%*%Y
  beta1 = beta_h[2]
}
vec_loop = Vectorize(single_loop)


res1 = vec_loop(rep(25,S))
res2 = vec_loop(rep(100,S))

c(mean(res1),mean(res2)) # 2.002271,  1.998513
```

```
## [1] 2.002271 1.998513
```

```
c(var(res1),var(res2)) # 0.04465469,  0.01046778
```

```
## [1] 0.04465469 0.01046778
```

```
# Assume normality, do var test
var.test(res1, res2, alternative = "two.sided") # p-value < 2.2e-16
```

```
##
##  F test to compare two variances
##
## data:  res1 and res2
## F = 4.2659, num df = 9999, denom df = 9999, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  4.101914 4.436477
## sample estimates:
## ratio of variances
##           4.265917
```

```
t.test(res1, res2, alternative = "two.sided", var.equal = FALSE) # p-value = 0.1095
```

```
##
```

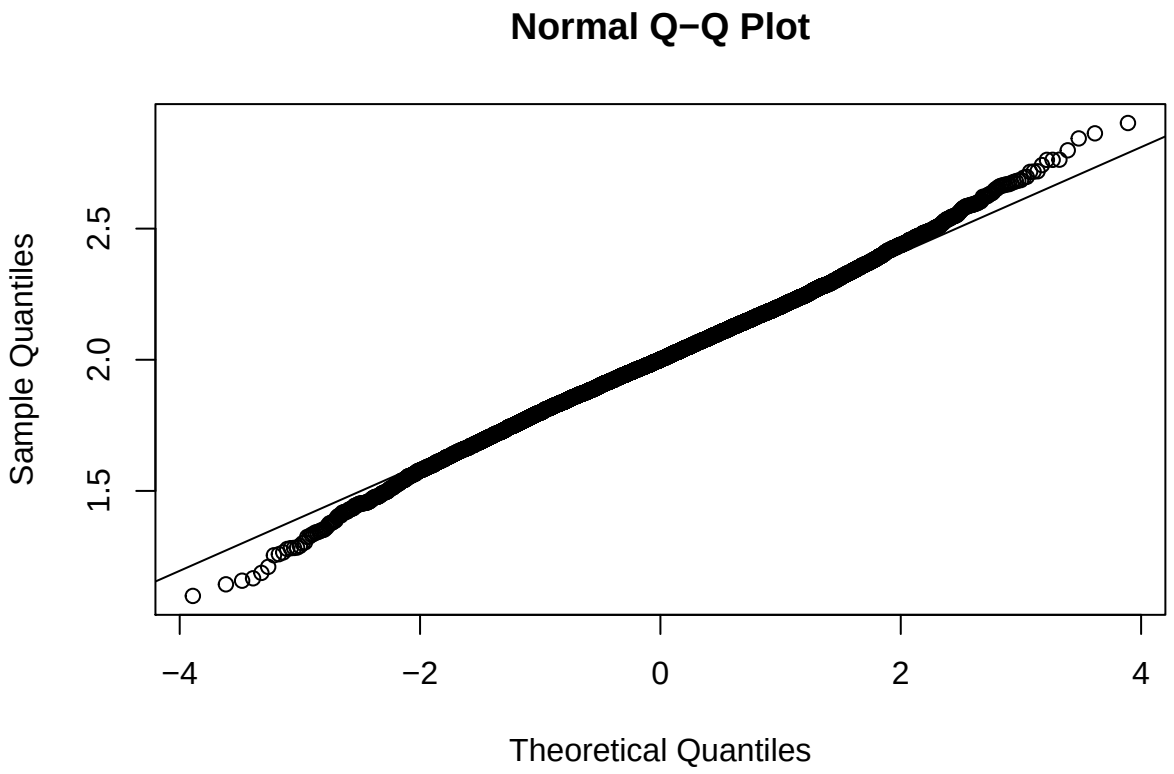
```
##  Welch Two Sample t-test
##
## data:  res1 and res2
## t = 1.6006, df = 14443, p-value = 0.1095
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.0008440232  0.0083600243
## sample estimates:
## mean of x mean of y
##  2.002271  1.998513
```

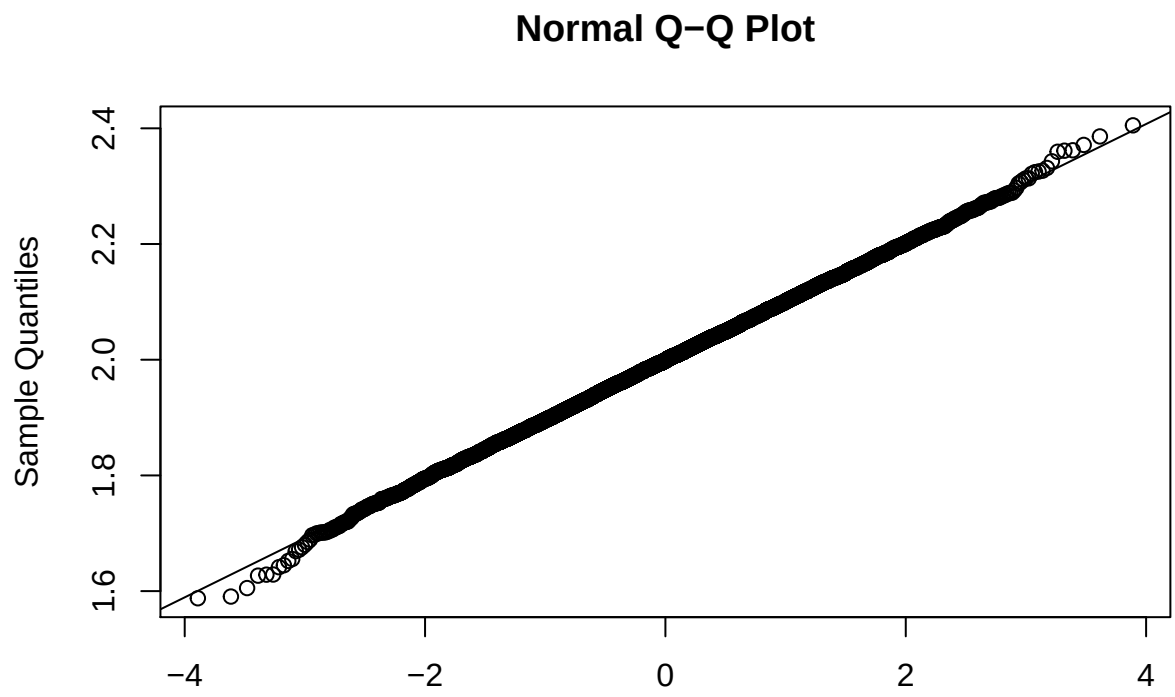
Check distribution and normality of $\hat{\beta}_1$

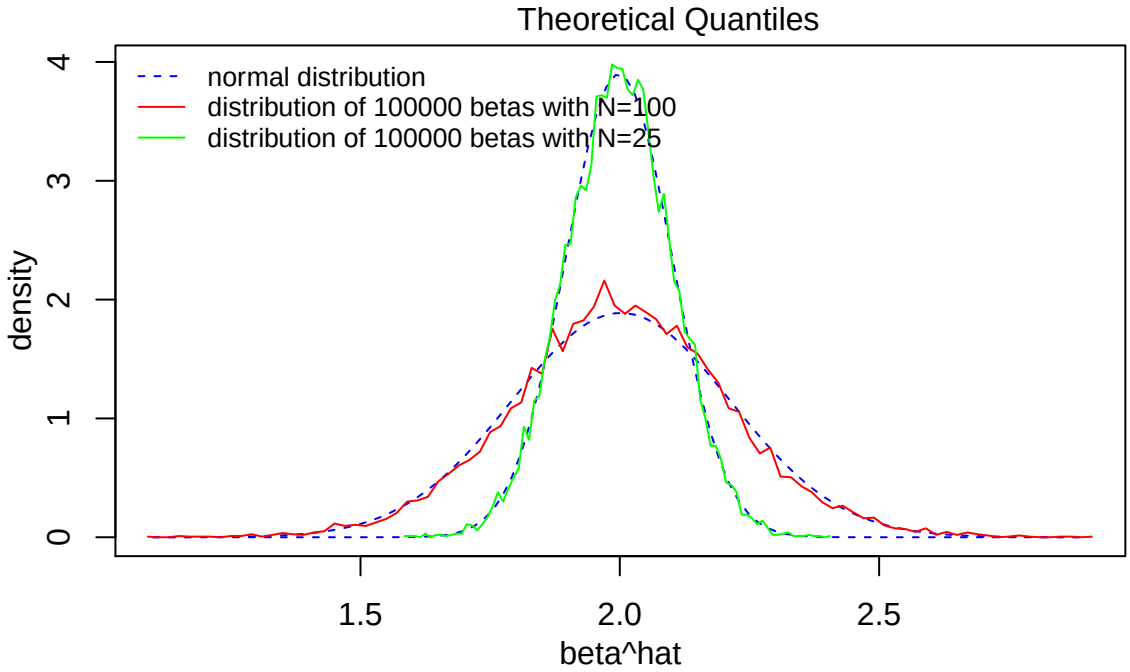
```
normality_check = function(res1, res2) {
  # Test normality
  #Anderson-Darling normality test
  print(ad.test(res1))
  print(ad.test(res2))
  qqnorm(res1)
  qqline(res1)
  qqnorm(res2)
  qqline(res2)

  # Plot density
  # Call lm_reg and do the computation
  cores=detectCores()
  cl <- parallel::makeCluster(2, setup_strategy = "sequential")
  registerDoParallel(cl)
  stopCluster(cl)

  # Plot the distribution of beta1^hat and normal distribution
  x = seq(min(res1), max(res1), length=100)
  hx1 = dnorm(x,mean=mean(res1),sd=sd(res1))
  hx2 = dnorm(x,mean=mean(res2),sd=sd(res2))
  res1_hist=hist(res1, breaks = 10^2, plot=FALSE)
  res2_hist=hist(res2, breaks = 10^2, plot=FALSE)

  plot(x,hx1,xlab="",ylab="",main="",
       col="blue",lty=2,type="l",lwd=1,
       ylim=c(0,max(c(res1_hist$density, res2_hist$density))))
  mtext("beta^hat",side=1,line=2)
  mtext("density",side=2, line=2)

  lines(x,hx2,col="blue",lty=2,type="l",lwd=1)
  lines(res1_hist$mids,res1_hist$density,
        col="red",lty=1,lwd=1)
  lines(res2_hist$mids,res2_hist$density,
        col="green",lty=1,lwd=1)

  legend("topleft",
         legend=c("normal distribution",
                  "distribution of 100000 betas with N=100",
                  "distribution of 100000 betas with N=25"),
         col=c("blue","red","green"),
```

```
         lty=c(2,1,1),cex=0.8,bty = "n")

}
normality_check(res1,res2)
```

```
##
##  Anderson-Darling normality test
##
## data:  res1
## A = 3.3628, p-value = 2.064e-08
##
##
##  Anderson-Darling normality test
##
## data:  res2
## A = 0.20902, p-value = 0.8635
```







According to the calculations and plots above, res1 (N=25) has a mean of 2.002271 and res2 (N=100) has a mean of 1.998513. We can conclude that mean differs little at about 1e-03 level. Moreover, by running two-sided t-test of res1 and res2, we get p-value = 0.1095 > 0.05, thus we cannot reject null hypothesis that the mean of two groups are statistically equal at 5% significance level. Therefore, we may conclude that the mean of two results i.e. res1 and res2 are approximately equal.

Regarding the variance, res1 has a variance of 0.04465469 while res2 has a variance of 0.01046778. By comparing the variance of two results which differs at about 1e-02 level and differs by 0.03418691, we may conclude that the variance is different. Further, through F.test (assuming the data is normally distributed), we get p-value = 2.2e-16 < 0.05, thus we can reject the null hypothesis that the variance of two groups are

statistically equal i.e. the variance of res1 and res 2 are statistically different at 5% significance level. Also, we can conclude that the estimated beta 1 with greater sample size N=100 has a lower variance that the estimated beta 1 with smaller sample size N=25. A larger sample size gives a less spread-out result i.e. data with smaller dispersion.

By observing the QQ-plot and density plot, when N = 25, many points fall away from the benchmark line in the respective plots. When N=100, the points are nearly on the benchmark line in the respective plots. Also, by running Anderson-Darling normality test, the p-value for N=25 case is 2.064e-08 < 0.05, thus we should reject Anderson-Darling test null hypothesis at 5% significance level. In contrast, the p-value for N=100 is 0.8635 > 0.05, which is normal. So we conclude that the estimated beta 1 in res1 (N=25) does not follow normal distribution but the estimated beta 1 in res2 (N=100) follows a normal distribution.

b.1 Use uniformly distributed error

```
set.seed(37)
S = 10000
single_loop = function(N) {
  x = rnorm(N)
  e = runif(N)
  Y = 0.7 + 2*x + e
  X = cbind(rep(1,N),x)
  beta_h = ginv(t(X)%*%X)%*%t(X)%*%Y
  beta1 = beta_h[2]
}
vec_loop = Vectorize(single_loop)

res1 = vec_loop(rep(25,S))
res2 = vec_loop(rep(100,S))

c(mean(res1),mean(res2)) # 1.999372,  2.000016
```

```
## [1] 1.999372 2.000016
```

```
c(var(res1),var(res2)) # 0.0037723373,  0.0008630136
```

```
## [1] 0.0037723731 0.0008630136
```

```
var.test(res1, res2, alternative = "two.sided") # p-value < 2.2e-16
```

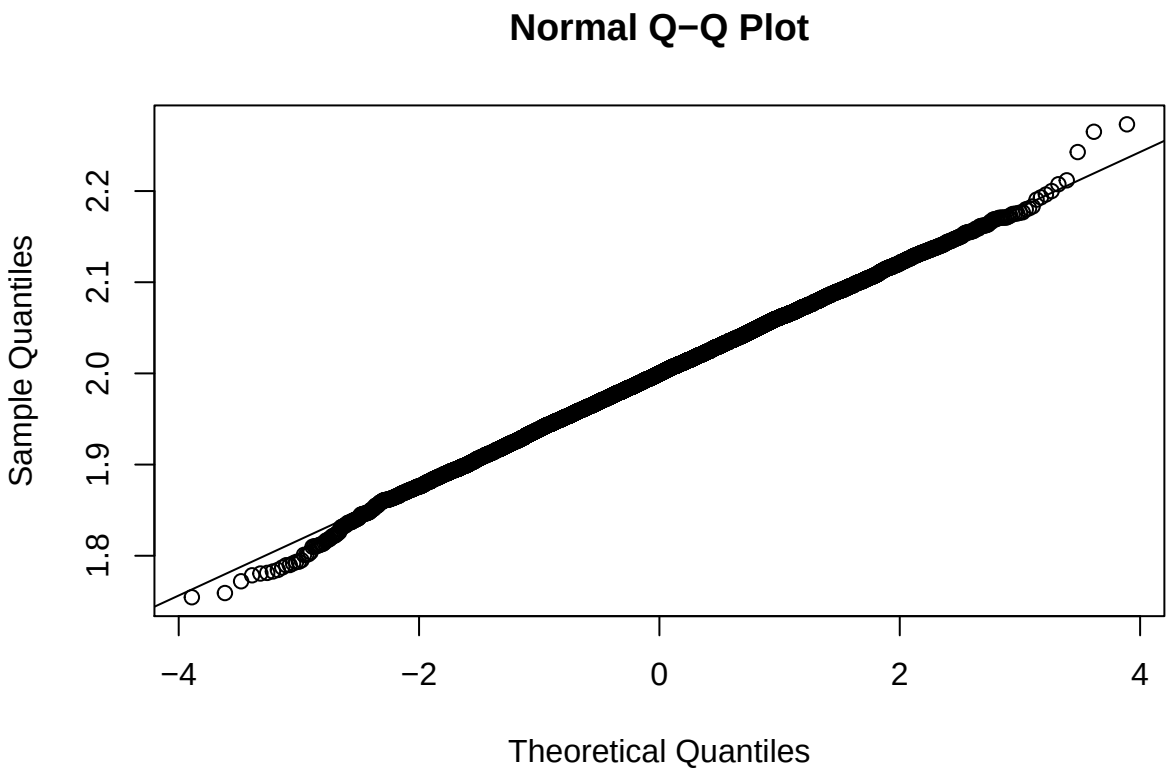
```
##
##  F test to compare two variances
##
## data:  res1 and res2
## F = 4.3712, num df = 9999, denom df = 9999, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  4.203114 4.545931
## sample estimates:
## ratio of variances
##           4.371163
```

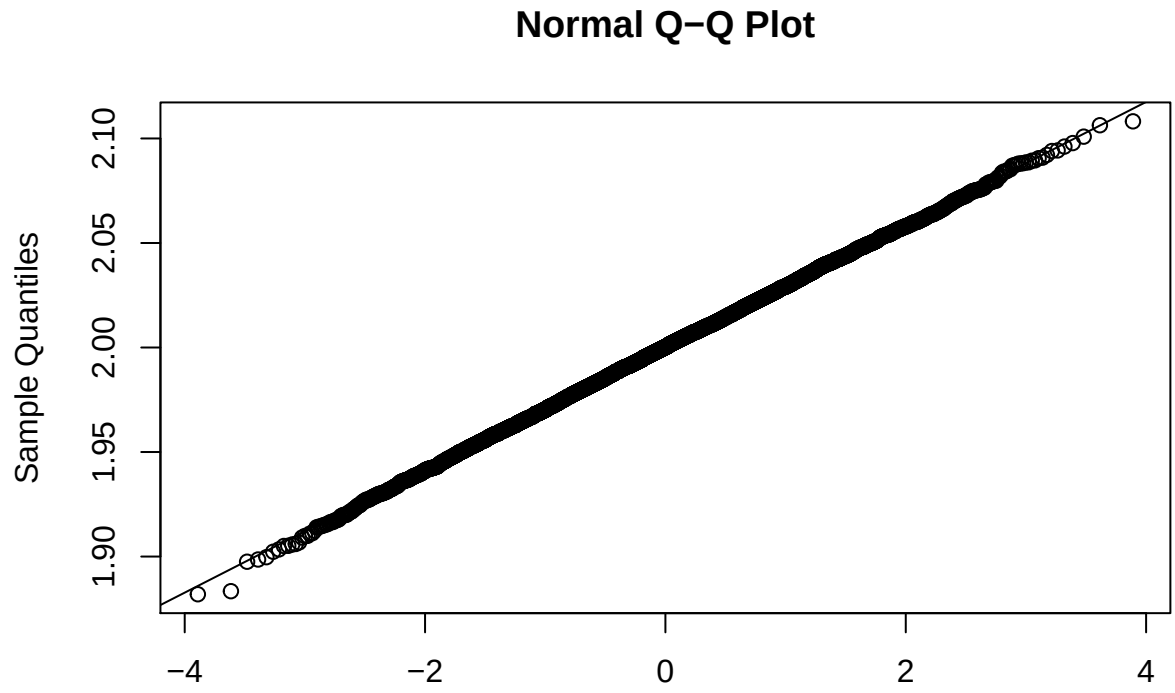
```r
t.test(res1, res2, alternative = "two.sided", var.equal = FALSE) # p-value = 0.3446
```

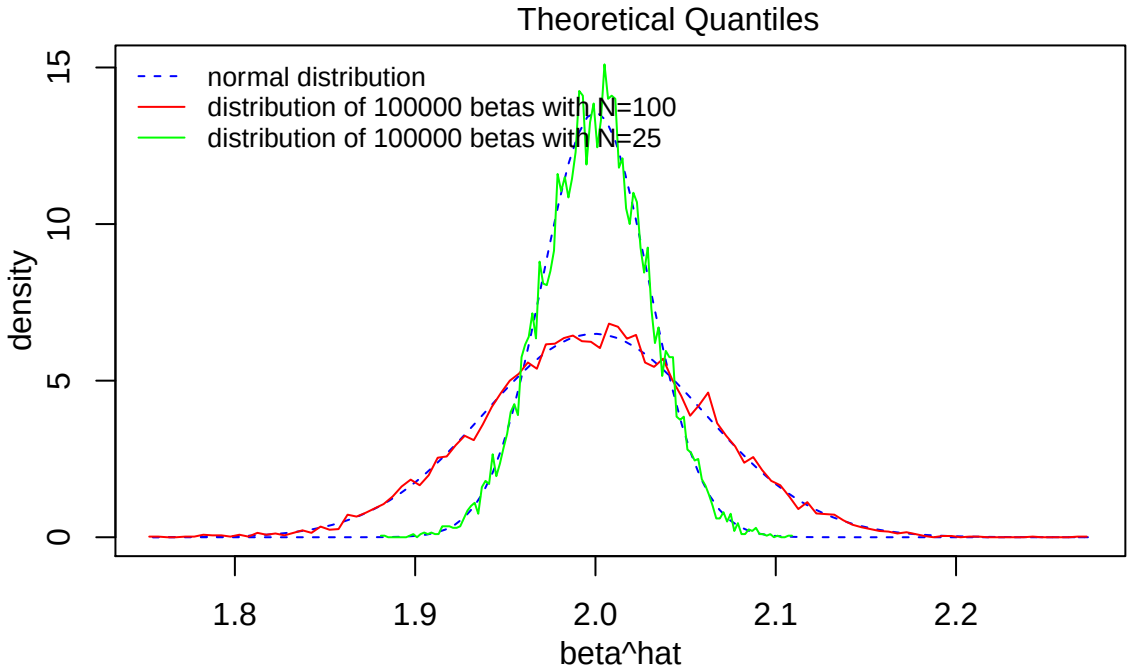
```
##
##  Welch Two Sample t-test
##
## data:  res1 and res2
## t = -0.94514, df = 14346, p-value = 0.3446
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.001978011  0.000691045
## sample estimates:
## mean of x mean of y
##  1.999372  2.000016
```

```r
normality_check(res1,res2)
```

```
##
##  Anderson-Darling normality test
##
## data:  res1
## A = 0.37211, p-value = 0.4207
##
##
##  Anderson-Darling normality test
##
## data:  res2
## A = 0.35184, p-value = 0.4682
```







When the error term follows uniform distribution, res1 (with N=25) has a mean of 1.999372 and res2 (with N=100) has a mean of 2.000016. We can conclude that mean differs little at about 1e-04 level. Moreover, by running two-sided t-test of res1 and res2, we get p-value=0.3446 > 0.05, thus we cannot reject null hypothesis that the mean of two groups are statistically equal at 5% significance level. Therefore, we may conclude that the mean of two results i.e. res1 and res2 are approximately equal. The answer in II does not change for mean comparison.

Regarding the variance, res1 has a variance of 0.003772373 while res2 has a variance of 0.0008630136. By comparing the variance of two results which differs at about 1e-03 level and differs by 0.002909359, we may conclude that the variance is different. Further, through F.test (assuming the data is normally distributed),

we get p-value < 2.2e-16, thus we can reject the null hypothesis that the variance of two groups are statistically equal i.e. the variance of res1 and res 2 are statistically different. Also, we can conclude that the estimated beta 1 with greater sample size N=100 has a lower variance that the estimated beta 1 with smaller sample size N=25. A larger sample size gives a less spread-out result i.e. data with smaller dispersion. The answer in II does not change for variance comparison.

By observing the QQ-plot and density plot, when N = 25 and N = 100, the points are nearly on the benchmark line in the respective plots. Also, by running Anderson-Darling normality test, the p-value for N=25 case is 0.4207 > 0.05, thus we should not reject Anderson-Darling test null hypothesis at 5% significance level i.e. follow normal distribution. The p-value for N=100 is 0.4682 > 0.05, which is also normal. So we conclude that both the estimated beta 1 follow normal distributions when N = 25 and N = 100.

b.2 Use Cauchy distributed error

```
set.seed(37)
S = 10000
single_loop = function(N) {
  x = rnorm(N)
  e = rcauchy(N)
  Y = 0.7 + 2*x + e
  X = cbind(rep(1,N),x)
  beta_h = ginv(t(X)%*%X)%*%t(X)%*%Y
  beta1 = beta_h[2]
}
vec_loop = Vectorize(single_loop)

res1 = vec_loop(rep(25,S))
res2 = vec_loop(rep(100,S))

mean(res1) # 1.907382
```

```
## [1] 1.907382
```

```
mean(res2) # -1.599096
```

```
## [1] -1.599096
```

```
var(res1) # 3427.213
```

```
## [1] 3427.213
```

```
var(res2) # 29851.75
```

```
## [1] 29851.75
```

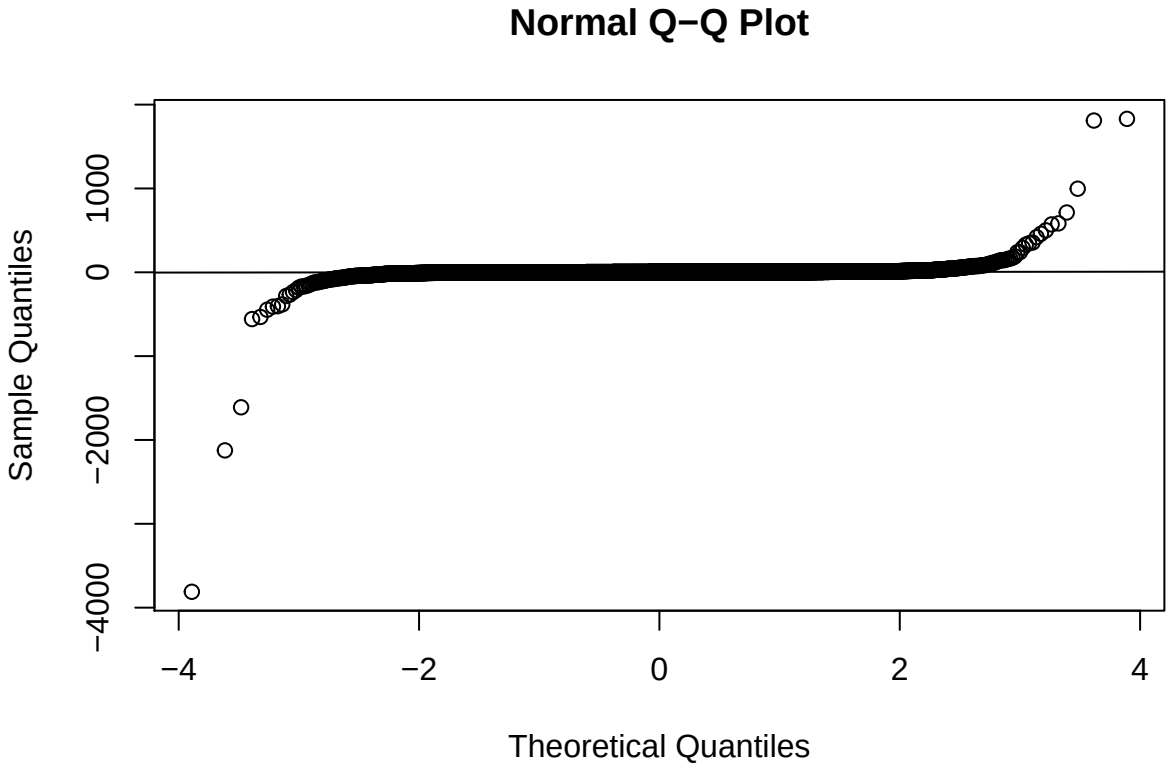
```
t.test(res1, res2, alternative = "two.sided", var.equal = FALSE) # p-value = 0.05461
```

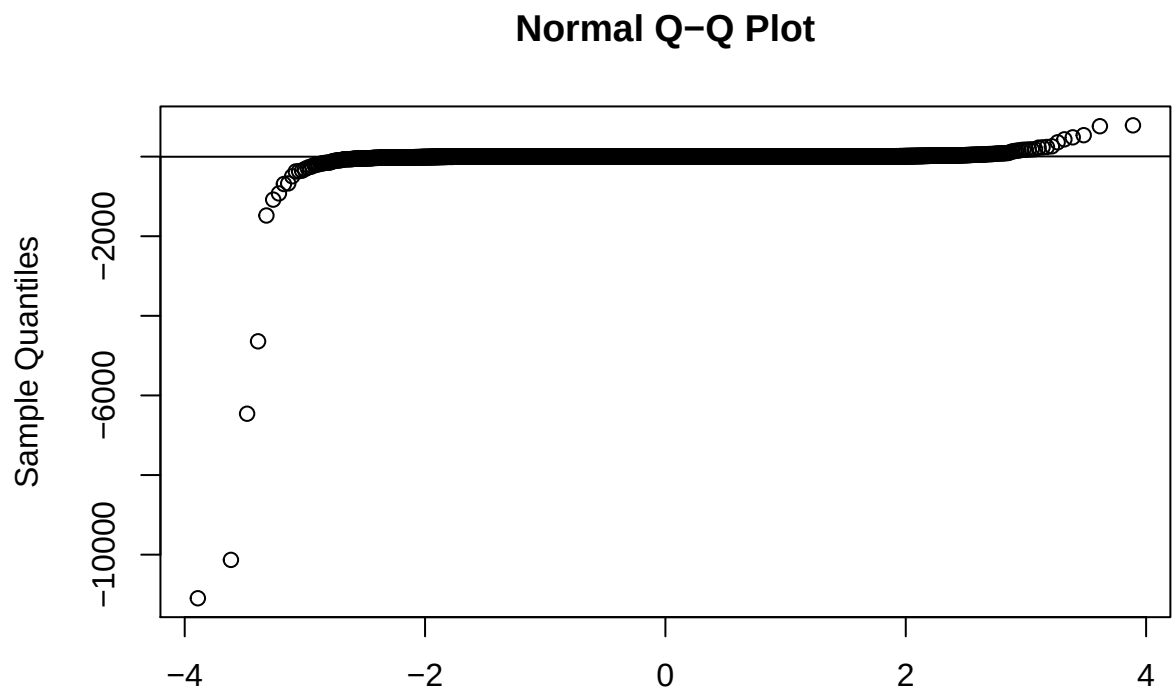
```
##
##  Welch Two Sample t-test
```

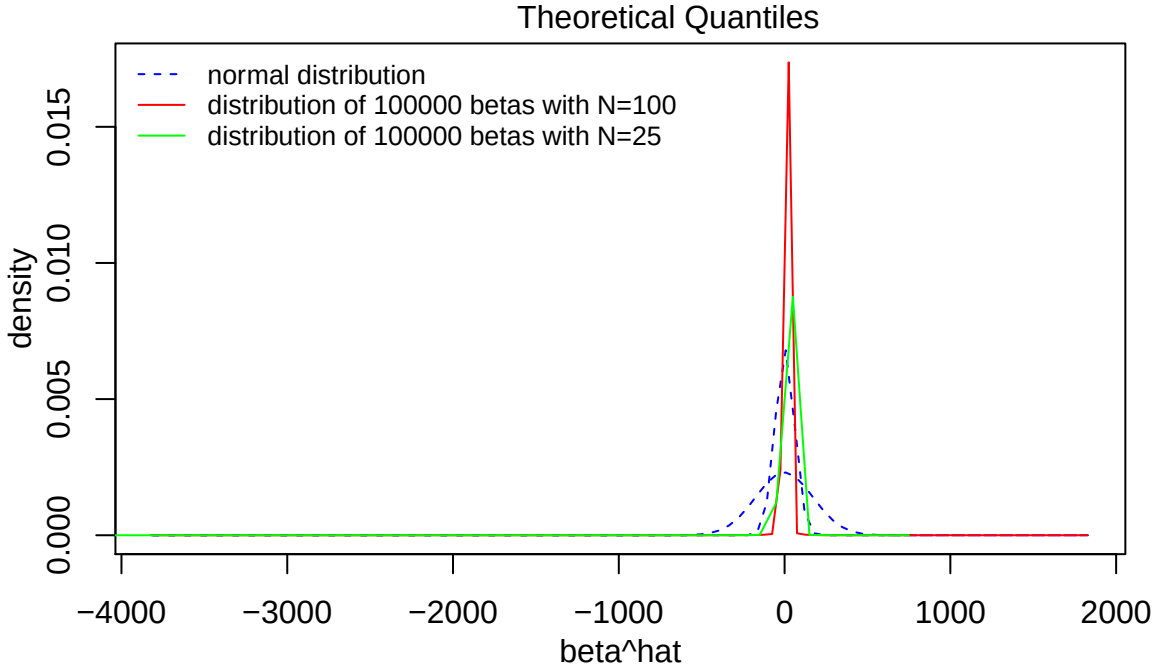
```
##
## data:  res1 and res2
## t = 1.9221, df = 12265, p-value = 0.05461
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.06934377  7.08229918
## sample estimates:
## mean of x mean of y
##  1.907382 -1.599096
```

```
normality_check(res1,res2)
```

```
##
##  Anderson-Darling normality test
##
## data:  res1
## A = 3277, p-value < 2.2e-16
##
##
##  Anderson-Darling normality test
##
## data:  res2
## A = 3620.1, p-value < 2.2e-16
```







When the error term follows Cauchy distribution, res1 (with N=25) has a mean of 1.907382 and res2 (with N=100) has a mean of -1.599096. We can conclude that mean differs. Moreover, by running two-sided t-test of res1 and res2, we get p-value = 0.05461 $<$ 0.1, thus we can reject null hypothesis that the mean of two groups are statistically equal at 10% significance level. Therefore, we may conclude that the mean of res1 and res2 are statistically different. The answer in II changes for mean comparison.

Regarding the variance, res1 has a variance of 3427.213 while res2 has a variance of 29851.75. By comparing the variance of two results which differs much, we may conclude that the variance is different. Further, through F.test (assuming the data is normally distributed), we get p-value $<$ 2.2e-16, thus we can reject the null hypothesis that the variance of two groups are statistically equal i.e. the variance of res1 and res 2 are

statistically different. Also, we can conclude that the estimated beta 1 with greater sample size N=100 has a lower variance that the estimated beta 1 with smaller sample size N=25. A larger sample size gives a less spread-out result i.e. data with smaller dispersion. The answer in II does not change for variance comparison. In addition, the difference between two mean values and 2 variance values are bigger when using Cauchy distribution than the previous 2 methods.

By observing the QQ-plot and density plot, when N = 25 and N = 100, many points fall away from the benchmark line in the respective plots. Also, by running Anderson-Darling normality test, the p-values for both N = 25 and N = 100 are 2.2e-16 < 0.05, thus we should reject Anderson-Darling test null hypothesis at 5% significance level. So we conclude that the estimated beta 1 in res1(N=25) and res2(N=100) do not follow normal distributions.

Q4

A car loan of $10000 was repaid in 60 monthly payments of $250, starting one month after the loan was made. Find the monthly interest rate r (error < 1e-8).

Recurrent formula of compound interest rate:

$$x_{n+1} = x_n(1+r) - P \implies x_n = x_0(1+r)^n + P\frac{1-(1+r)^n}{r}$$

Given:

$$x_0 = 10000, x_{60} = 0, P = 250 \implies r = 0.01439478$$

```
func1 = function(r, x0=10000,p=250, n=60) {
  # Iteration method to calculate xn
  x = x0
  for(i in 1:n) {
    x = x*(1+r) - p
  }
  res = x
}

func2 = function(r, x0=10000,p=250, n=60) {
  # Close form of xn
  xn = x0*(1+r)^n + p*(1-(1+r)^n)/r
}
```

func1 and func2 are equivalent to each other. For simplicity, func2 will be used for calculation.

```
bisection_method = function(f,left,right,tol=1e-10,n=1000) {
  if(sign(f(left))==sign(f(right))) {
    print("Bad Initial Points!")
    stop()
  }

  history = right
  for (i in 1:n) {
    mid = (left+right)/2
    if (f(mid)==0 || abs(left-right) < tol) {
      history = c(history,mid)
      res = list('optim r' = mid, 'iterations' = i,'history'=history)
```

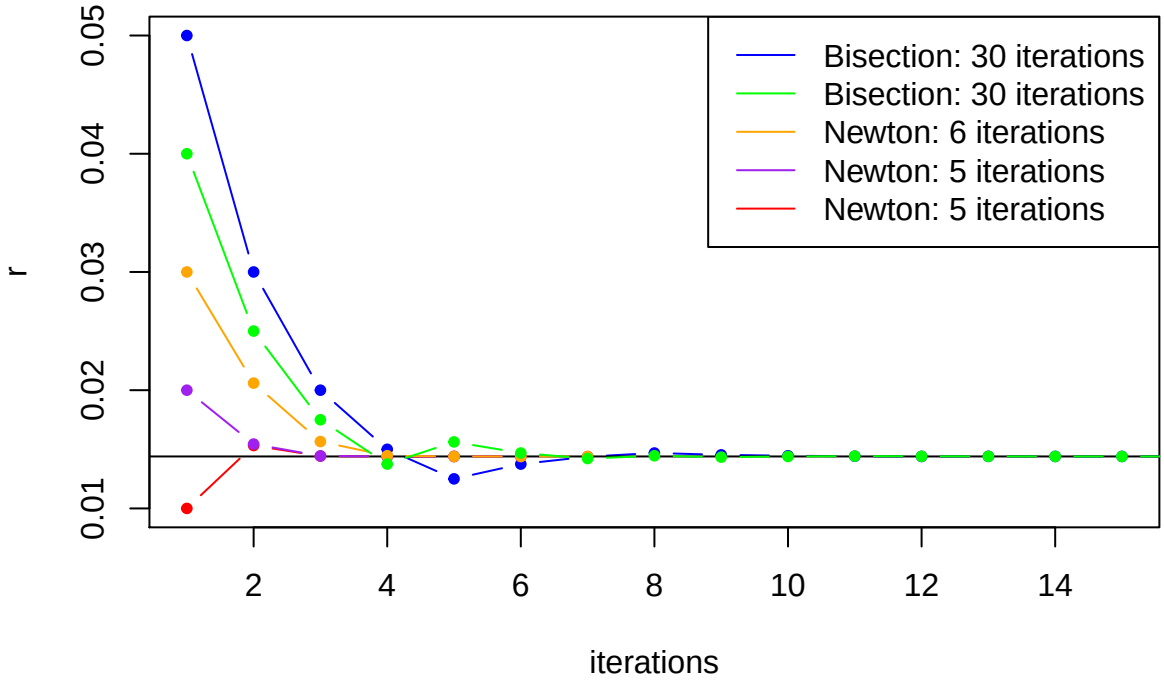
```
      return(res)
    } else if (sign(f(mid))==sign(f(right))) {
      right = mid
      history = c(history,right)
    } else {
      left = mid
      history = c(history,left)
    }

  }
  print("Maximum number of iteration reached")
  res = list('optim r' = mid, 'iterations' = i,'history'=history)
  return(res)
}

newton_method = function(f, r0, tol=1e-8,n=1000) {
  history = r0
  for(i in 1:n) {
    deriv = genD(func = f, x = r0)$D
    r1 = r0 - f(r0)/deriv[1]
    history = c(history,r1)
    if(abs(r1-r0) < tol) {
      res = list('optim r' = r1, 'iterations' = i,'history'=history)
      return(res)
    }
    r0 = r1
  }
  print("Maximum number of iteration reached")
  res = list('optim r' = r1, 'iterations' = i,'history'=history)
  return(res)
}

# Start searching from 1% interest rate
sol1 = newton_method(func2, 0.01)
sol2 = newton_method(func2, 0.02)
sol3 = newton_method(func2, 0.03)
sol4 = bisection_method(func2, 0.01, 0.04)
sol5 = bisection_method(func2, 0.01, 0.05)
c(sol1$`optim r`, sol2$`optim r`, sol3$`optim r`,
  sol4$`optim r`, sol5$`optim r`)
```

```
## [1] 0.01439478 0.01439478 0.01439478 0.01439478 0.01439478
```

Bisection method and Newton method come to the same result: monthly interest rate r = 0.01439478.

```
plot(1:length(sol5$history),sol5$history,type="b",pch=20,col="blue",
     xlim=c(1,15),ylim=c(0.01,0.05),xlab="iterations",ylab="r")
abline(sol1$`optim r`,0,col="black")
lines(1:length(sol1$history),sol1$history,type="b",pch=20,col="red")
lines(1:length(sol2$history),sol2$history,type="b",pch=20,col="purple")
lines(1:length(sol3$history),sol3$history,type="b",pch=20,col="orange")
lines(1:length(sol4$history),sol4$history,type="b",pch=20,col="green")
legend("topright",
```

```
       legend=c(paste("Bisection:",sol5$iterations,"iterations"),
                paste("Bisection:",sol4$iterations,"iterations"),
                paste("Newton:",sol3$iterations,"iterations"),
                paste("Newton:",sol2$iterations,"iterations"),
                paste("Newton:",sol1$iterations,"iterations")),
       col=c("blue","green","orange","purple","red"),lty="solid")
```



As shown in the plot, Newton method converges faster than bisection method for the function in this question.

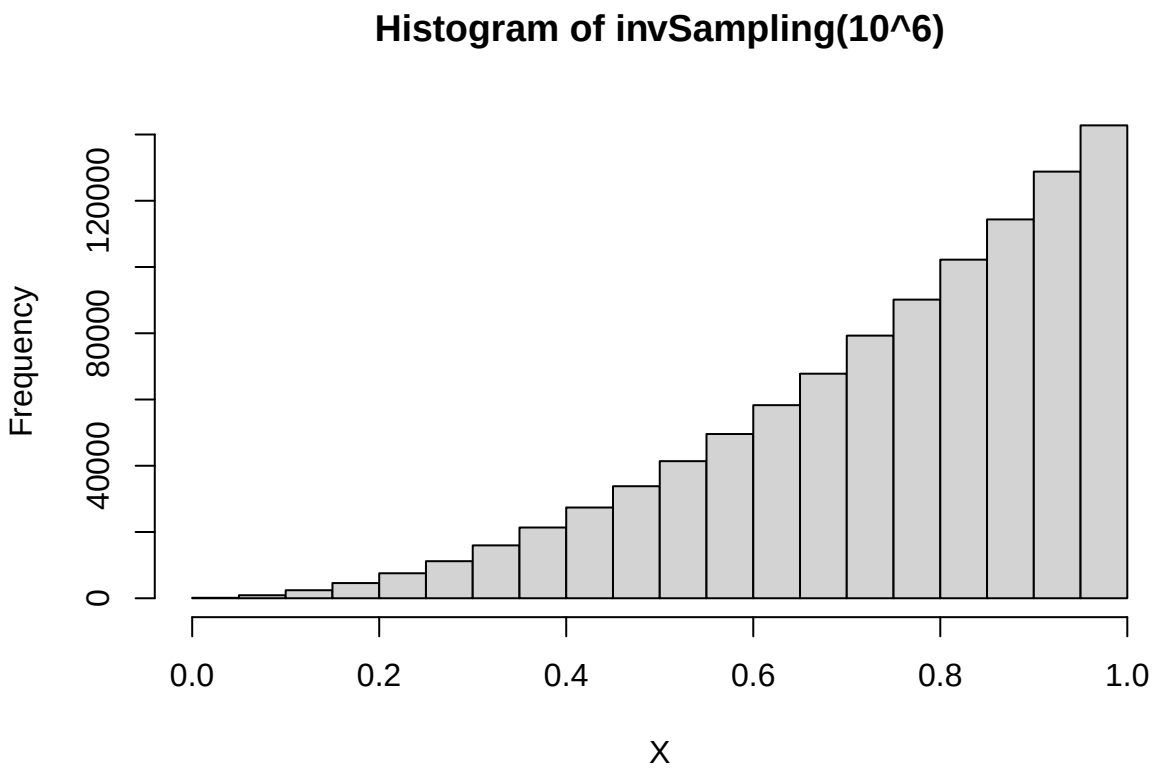
Q5

I. Use inversion sampling to simulate 10. random draws for market condition x.

Find the inverse function of X

$$f(x) = 3x^2 \cdot I(0,1) \implies F(x) = \int_0^x f(x)dx = x^3, x \in (0,1) \implies x = F^{-1}(u) = u^{1/3}$$

```
set.seed(37)

invSampling = function(N) {
  # Returns a vector of N elements sampled by inversion.
  return(runif(N)^(1/3))
}

hist(invSampling(10^6),xlab="X")
```



II. Calculate the expected returns using your draws. Plot the expected returns for y in (0,1).

Expected return given y:

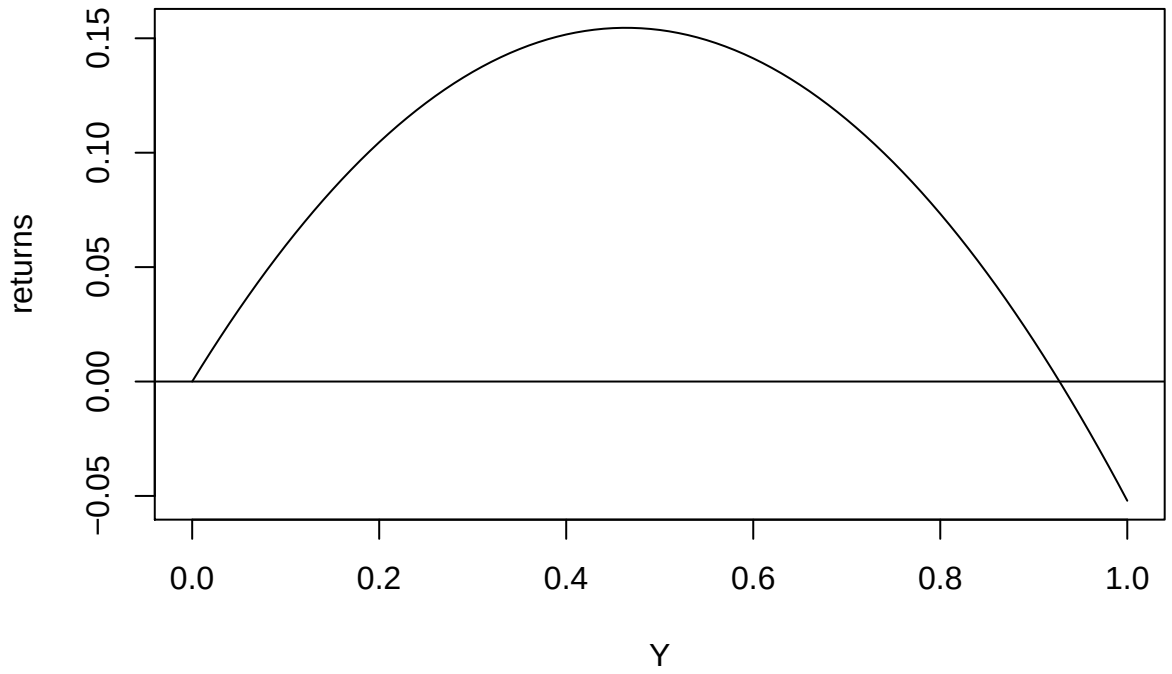
$$E[p(x,y)|y] = \int_0^1 p(x,y)f(x)dx$$

```
set.seed(37)

revenue = function(x,y) {
  return(y*(log(x)+1)-y^2*sqrt(1-x^4))
}

X = invSampling(1000000)
expectedReturn = function(y) {
  # Compute expected return given y
  N = 1000000
  #X = invSampling(N)
  Y = rep(y,N)

  # Take mean of 10^6 sampled return to approximate expectation
  res = sum(mapply(revenue, X, Y))/N
}
```

Plot **E[p(x,y)|y] ~ y**

```r
# take y from 0 to 1 with step 0.01
Y = seq(0,1,0.01)
returns = sapply(Y,expectedReturn)
```

```r
plot(returns ~ Y, type="l")
abline(0,0)
```



III. Compute the optimal investment y.

$$y^* = argmax\ E[p(x,y)|y],\ y \in (0,1) \implies y* = 0.4637619$$

```r
newton_optim = function(f, x0, tol=1e-10, n=1000) {
  history = x0

  for(i in 1:n) {
    deriv = genD(func=f, x=x0)$D
    x1 = x0 - deriv[1]/deriv[2]
    history = c(history, x1)
    if(abs(x1 - x0) < tol) {
      return(list(`optim x`=x1, `iterations`=i, `history`=history))
    }
    # Continue iteration
    x0 = x1
  }
  print("Maximum number of iteration reached")
  return(list(`optim x`=x1, `iterations`=i, `history`=history))
}

newton_optim(expectedReturn, 0)
```

```
## $`optim x`
## [1] 0.4637619
##
## $iterations
## [1] 2
##
## $history
## [1] 0.0000000 0.4637619 0.4637619
```

```
# 0.4637619
```

The optimal investment y is 0.4637619.