

Assignment 1 Submission

Q1

```
library(Matrix)
library(MASS) # For Moore-Penrose pseudo-inverse ginv()
rankMatrix(matrix(c(1,1,2,2),nrow=2))
```

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

```
rankMatrix(matrix(c(1,1,2,2,1,2),nrow=3))
```

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

```
rankMatrix(matrix(c(1,1,2,0),nrow=2))
```

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

```
rankMatrix(matrix(c(1,2,3,2,0,2),nrow=3))
```

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

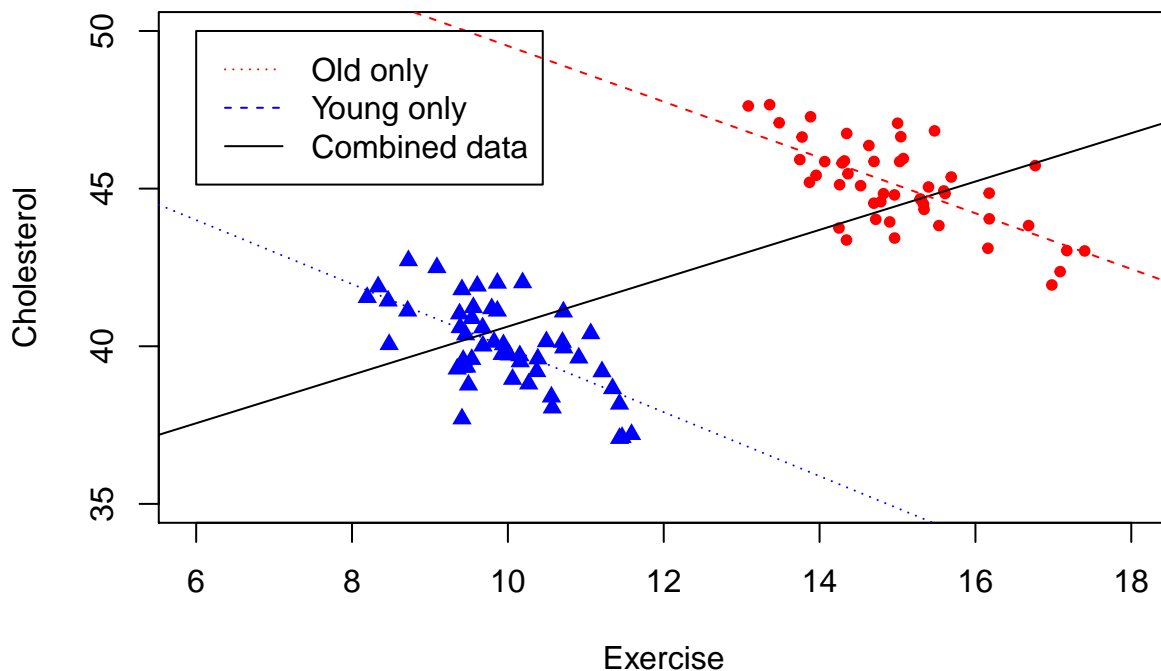
Q2

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))

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.

The 3 lines are giving conflicting insights. The young and old only lines suggest that Cholesterol and Exercise has negative association while the combined data suggesting positive association. The inconsistency occurs because Cholesterol is positively associated with Age so that the two groups are different. Furthermore, the old only group's exercise data is in the range between 13 - 18 while the young only group's data ranging from 7 - 12.

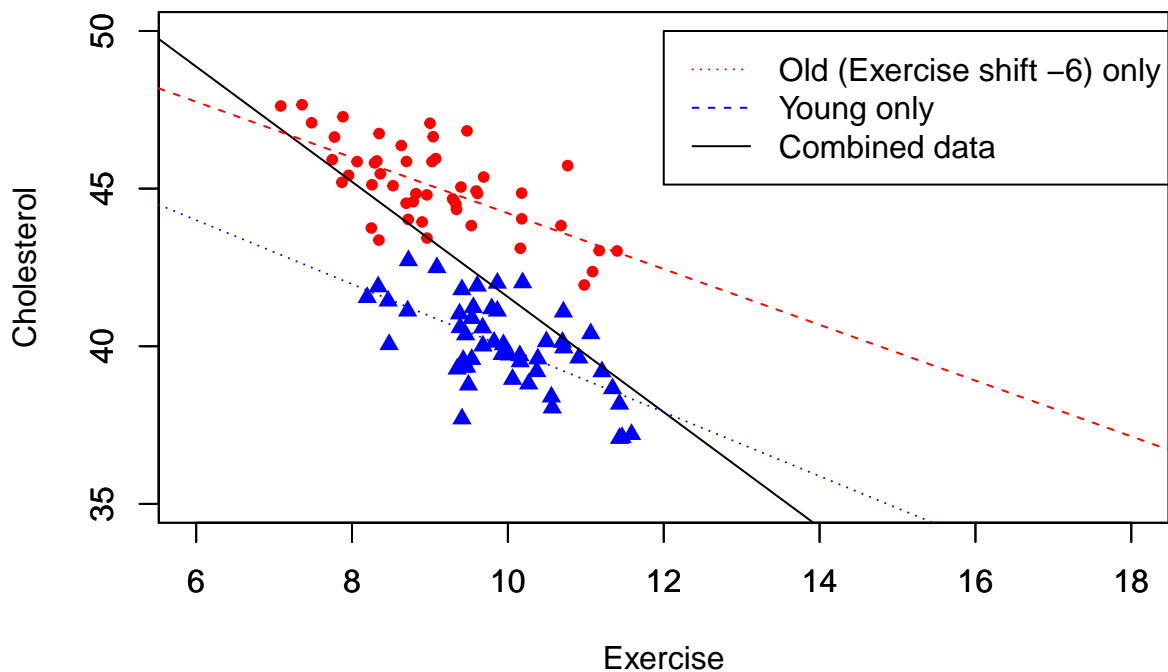
III. Can you propose an alternative estimation using all the data in order to obtain the same insight as the two separate regressions using only the old or the young people?

Maybe, collect old only data on the same range as young could solve the problem?

```
df_alter=df
df_alter[df_alter$Age=="Old",]$Exercise = df_alter[df_alter$Age=="Old",]$Exercise - 6
old_df_alter = old_df
old_df_alter$Exercise = old_df_alter$Exercise - 6

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

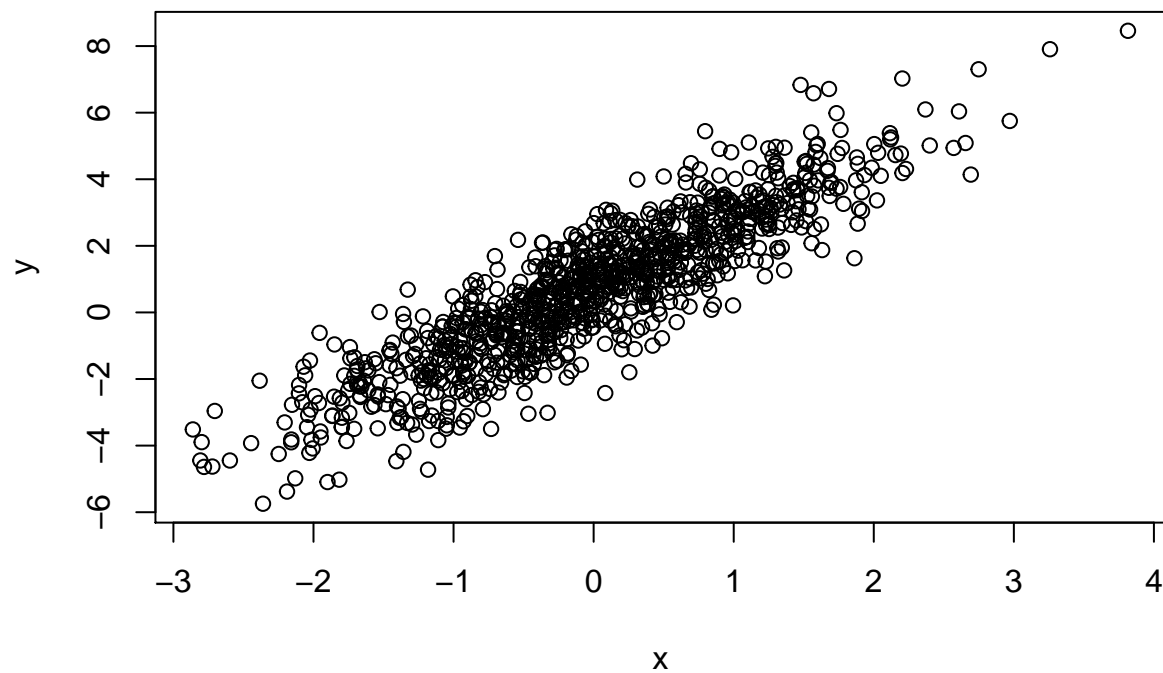
all_model = lm(Cholesterol ~ Exercise, data=df_alter)
young_model = lm(Cholesterol ~ Exercise, data=young_df)
old_model = lm(Cholesterol ~ Exercise, data=old_df_alter)
abline(all_model$coefficients[1],all_model$coefficients[2], lty="solid")
abline(young_model$coefficients[1],young_model$coefficients[2], lty="dotted",col="blue")
abline(old_model$coefficients[1],old_model$coefficients[2], lty="dashed",col="red")
legend(12, 50, legend=c("Old (Exercise shift -6) only", "Young only","Combined data"),
col=c("red", "blue","black"), lty=c("dotted","dashed","solid"), cex=1)
```



Q3

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

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



Use R's

lm() function:

```
# 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
```

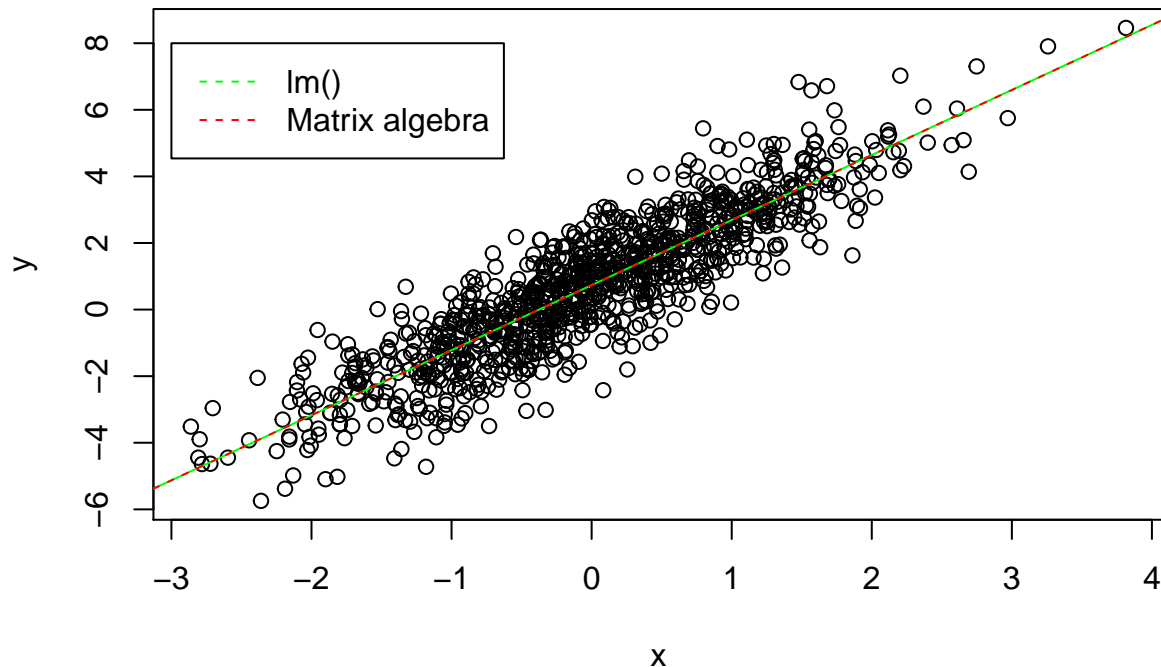
```
# y = 0.734 + 1.954 * x
```

Use matrix algebra: $\hat{\beta} = (X^T X)^{-1} X^T Y$

```
# 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 # \beta_0 = 0.734, \beta_1 = 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"))
```



$\text{tim}(): \hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1,\dots,n} (y_i - \beta_0 - \beta_1 \times x_i)^2$

Use op-

```
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) # \beta_0 = 0.734, \beta_1 = 1.954
```

```
## $par
## [1] 0.734061 1.953818
##
## $value
## [1] 1031.934
```

```
##
## $counts
## function gradient
##      57      NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Use formula for 1 variable regression: $\bar{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}$, $\hat{\beta}_1 = \frac{Cov(x_i, y_i)}{Var(x_i)}$

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

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

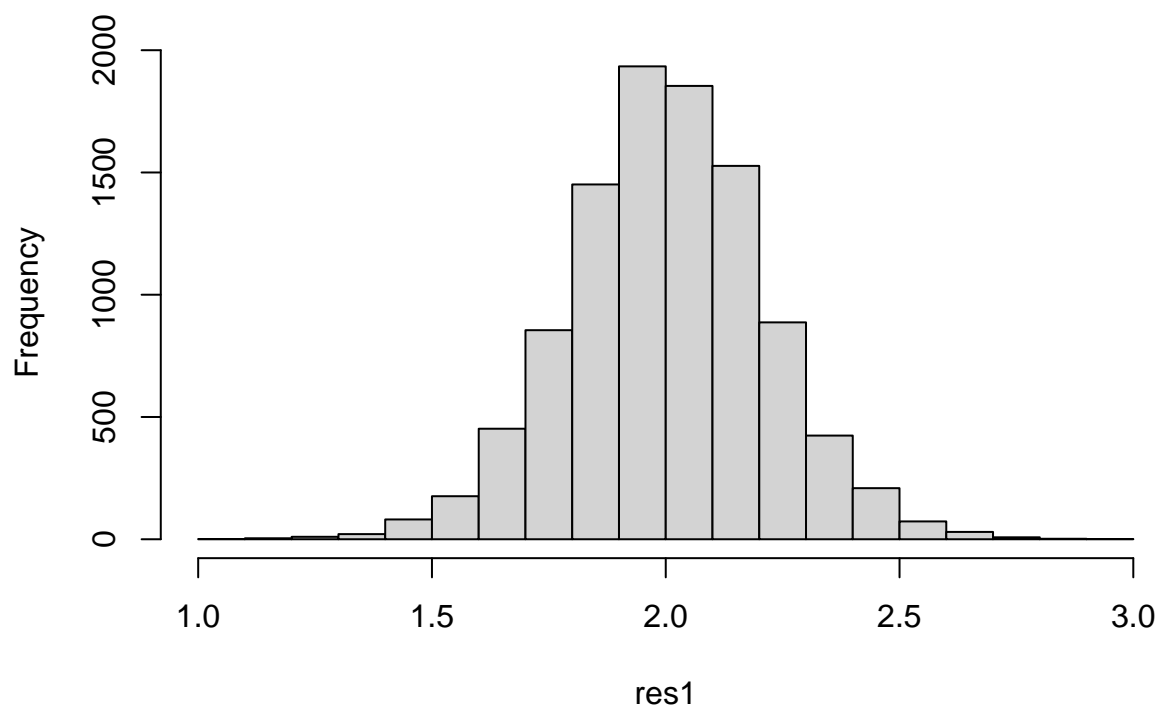
```
beta1 # 1.954
```

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

```
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)

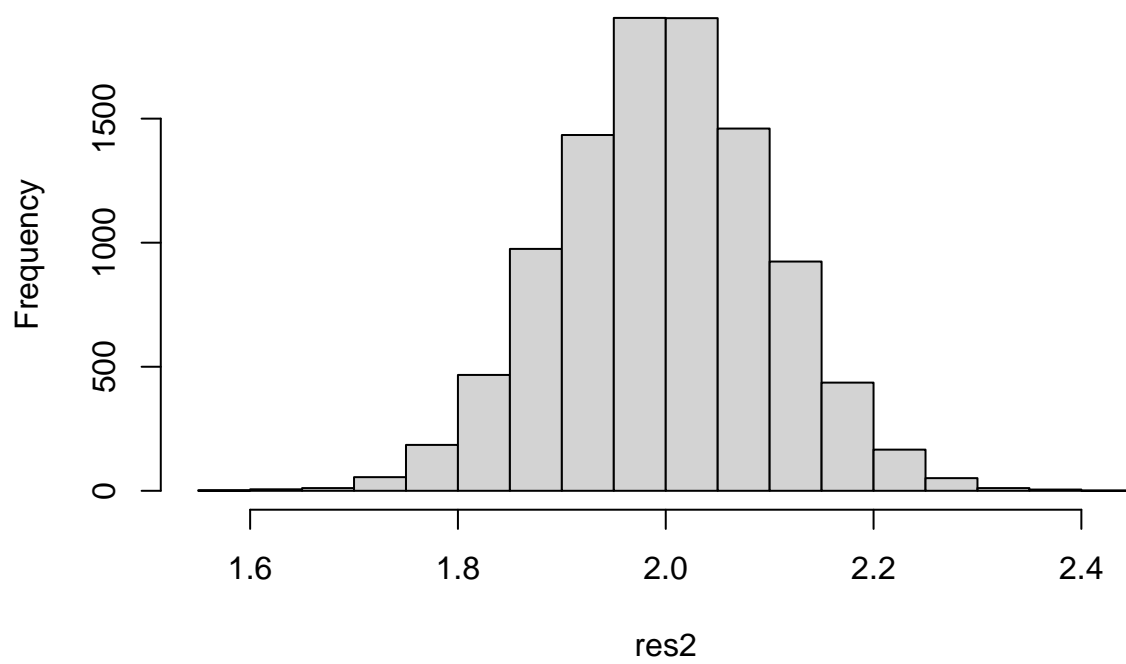
res1 = vec_loop(rep(25,S))
hist(res1)
```

Histogram of res1



```
res2 = vec_loop(rep(100,S))  
hist(res2)
```

Histogram of res2



```

mean(res1) # 2.002271

## [1] 2.002271

mean(res2) # 1.998513

## [1] 1.998513

var(res1) # 0.04465469

## [1] 0.04465469

var(res2) # 0.01046778

## [1] 0.01046778

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

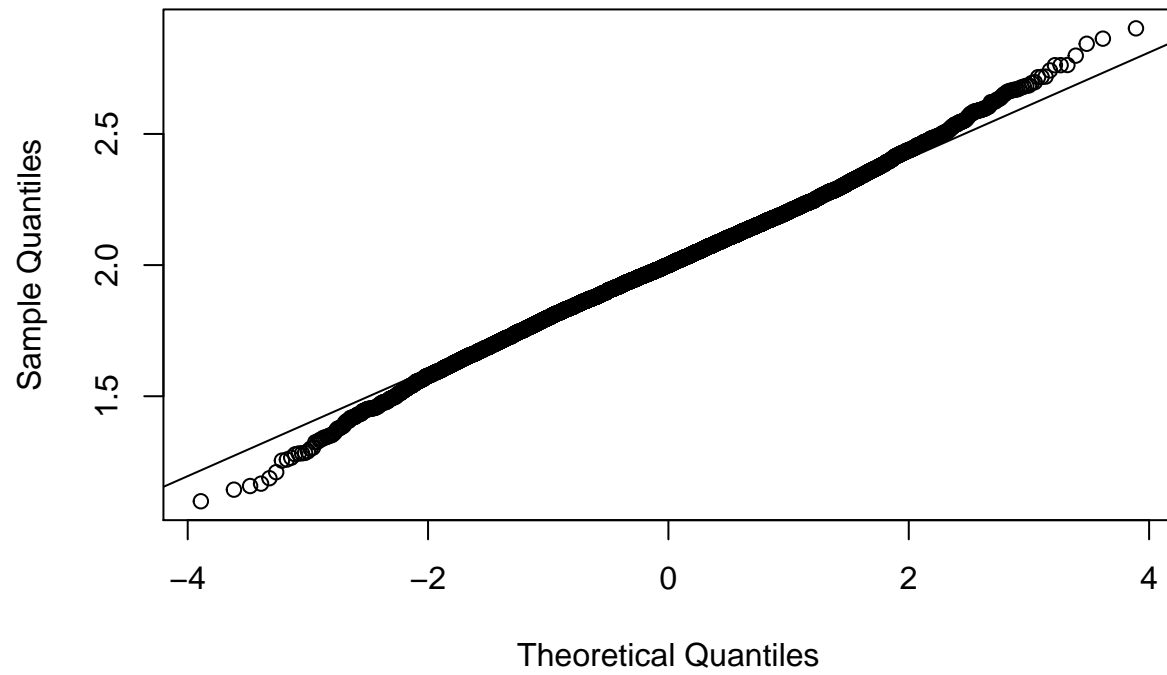
# Test normality
library(nortest)
#Anderson-Darling normality test
ad.test(res1) # p-value = 2.064e-08

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

qqnorm(res1)
qqline(res1)

```


Normal Q-Q Plot

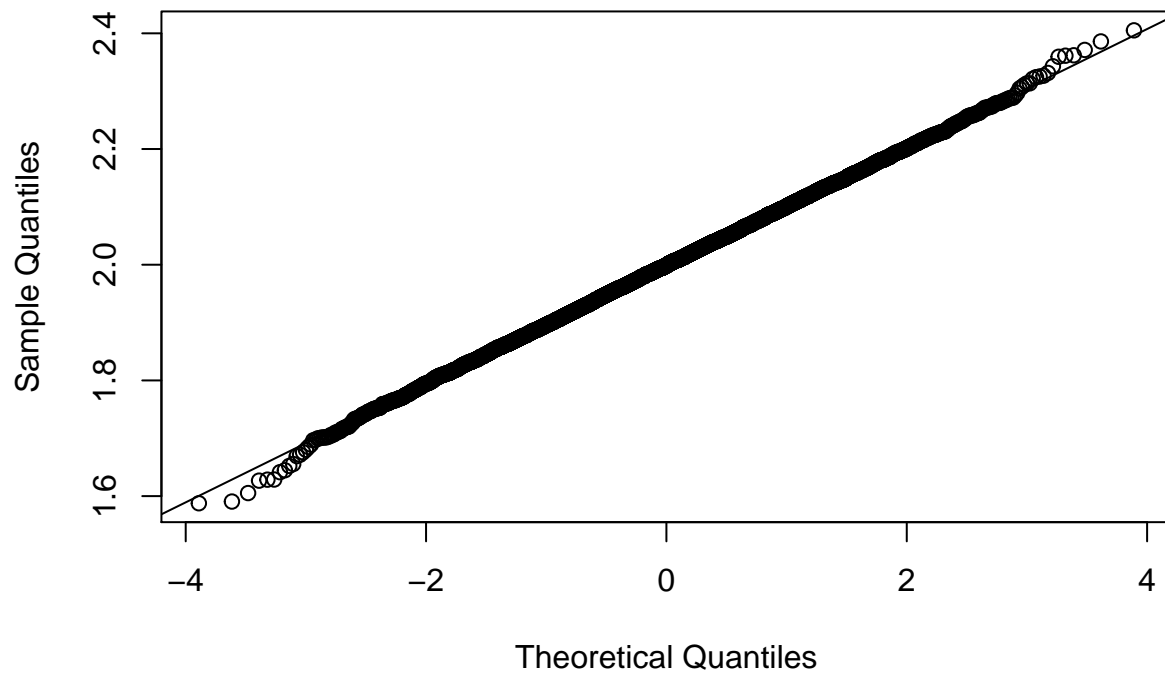


```
ad.test(res1) # p-value = 2.064e-08
```

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

```
qqnorm(res2)  
qqline(res2)
```

Normal Q-Q Plot

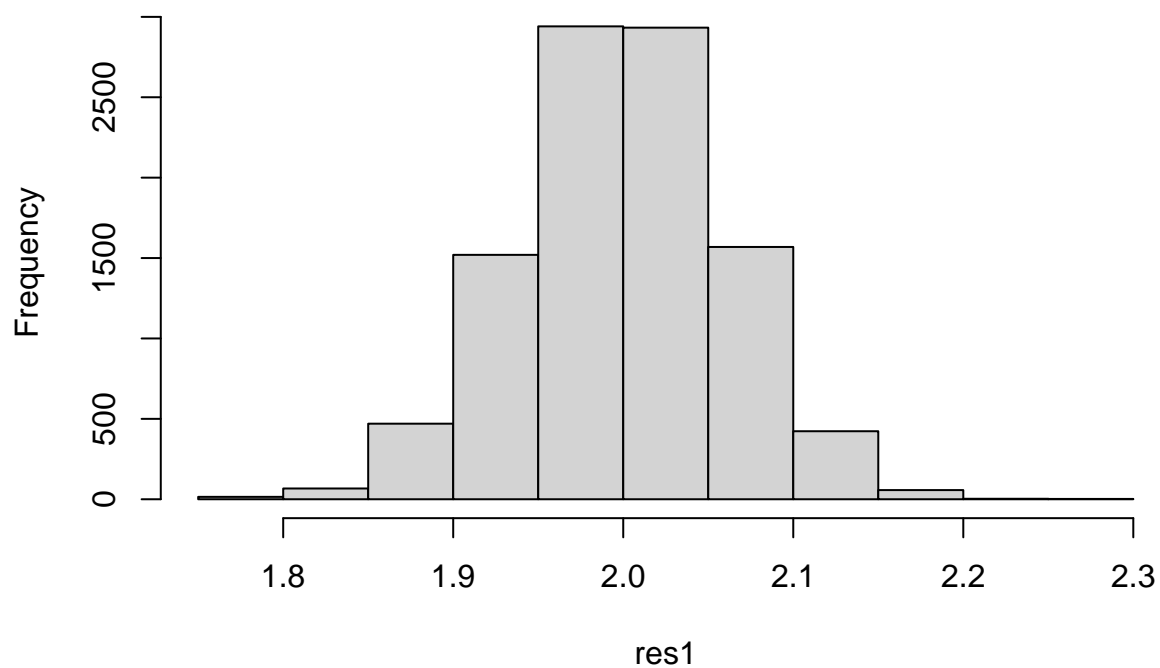


Use uniformly distributed error

```
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)

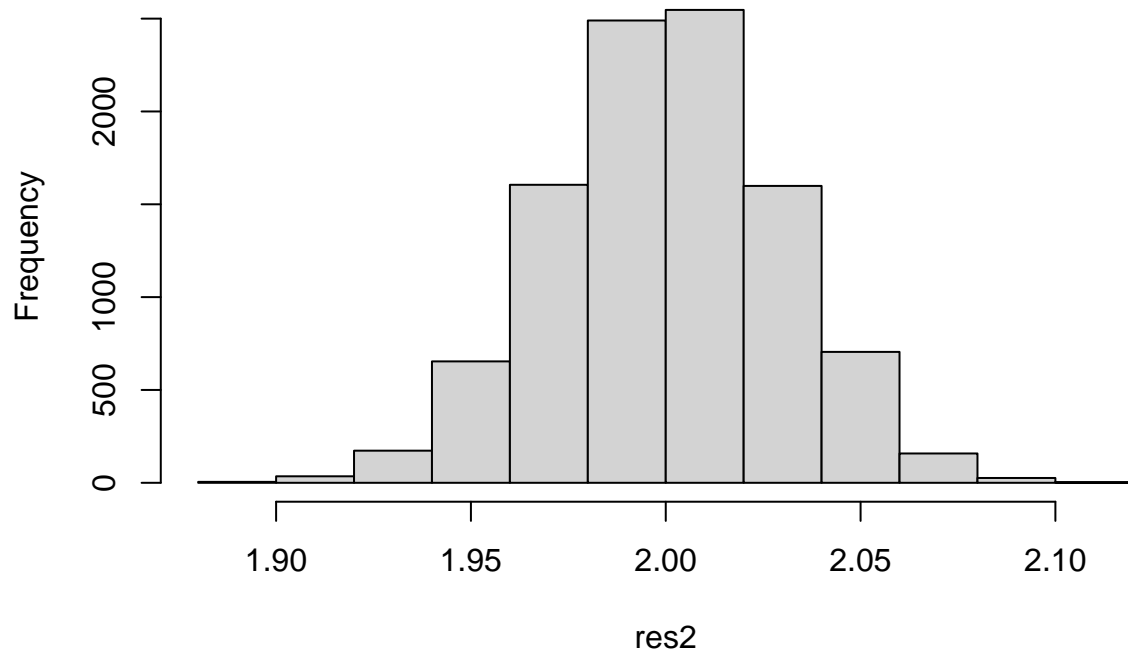
res1 = vec_loop(rep(25,S))
hist(res1)
```

Histogram of res1



```
res2 = vec_loop(rep(100,S))  
hist(res2)
```

Histogram of res2



```

mean(res1) # 1.999372

## [1] 1.999372

mean(res2) # 2.000016

## [1] 2.000016

var(res1) # 0.003772373

## [1] 0.003772373

var(res2) # 0.0008630136

## [1] 0.0008630136

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

##
## Two Sample t-test
##
## data: res1 and res2
## t = -0.94514, df = 19998, p-value = 0.3446
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.0019779794 0.0006910131
## sample estimates:
## mean of x mean of y
## 1.999372 2.000016

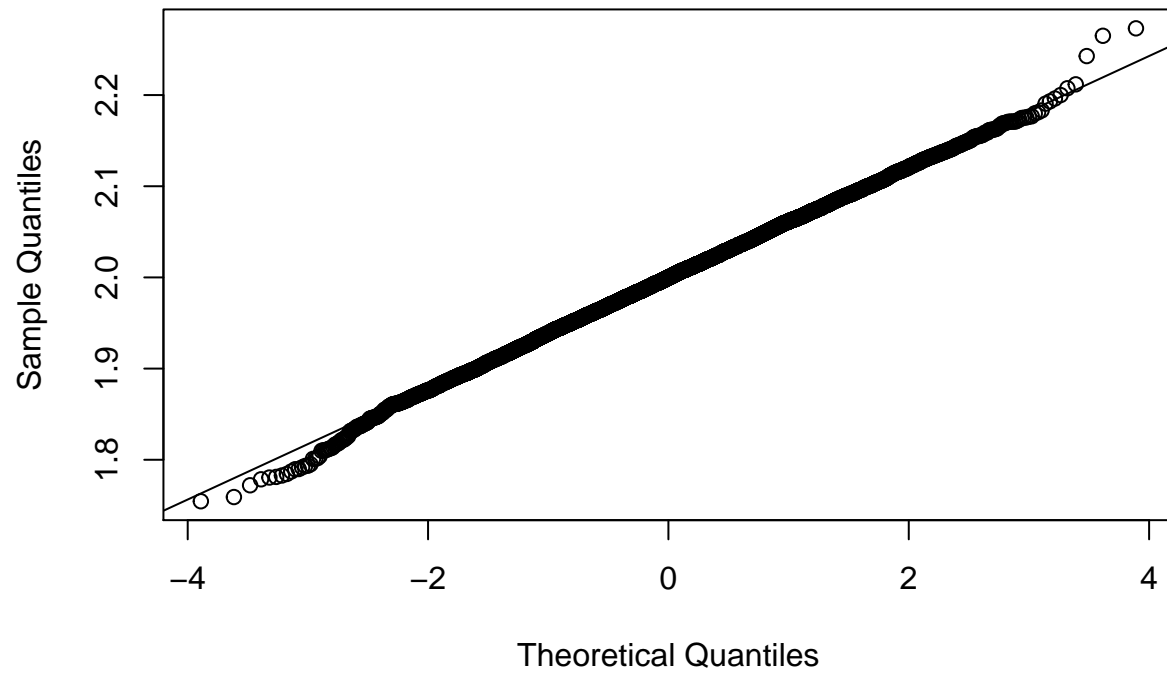
#Anderson-Darling normality test
ad.test(res1) # p-value = 0.4207

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

qqnorm(res1)
qqline(res1)

```

Normal Q-Q Plot

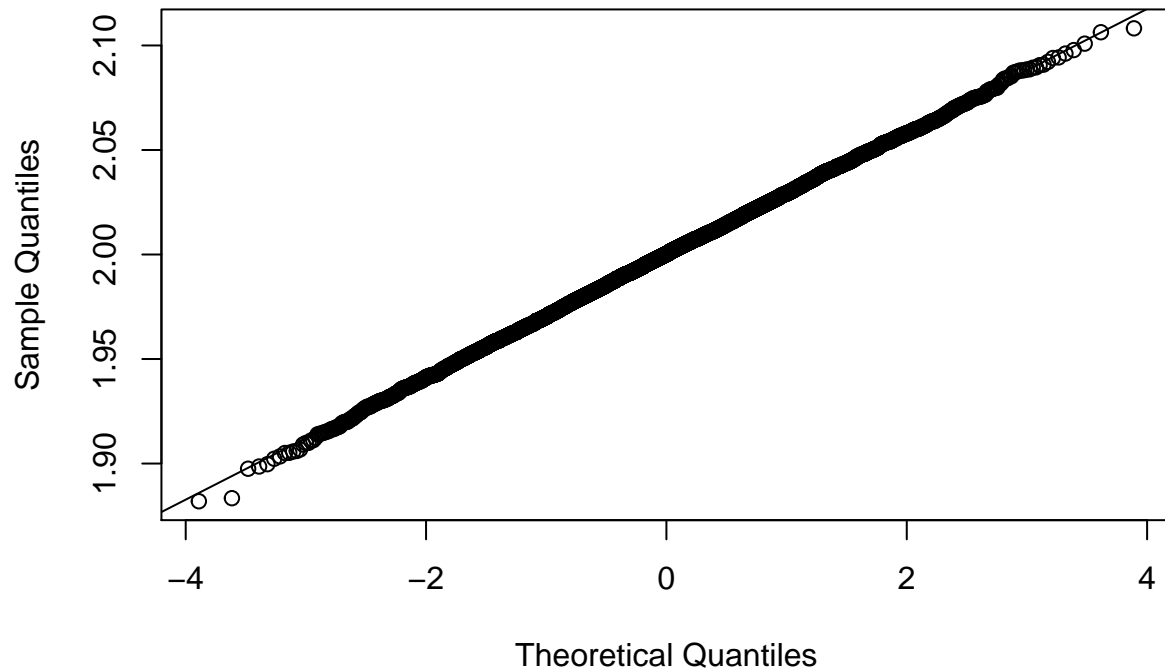


```
ad.test(res1) # p-value = 0.4207
```

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

```
qqnorm(res2)  
qqline(res2)
```

Normal Q-Q Plot

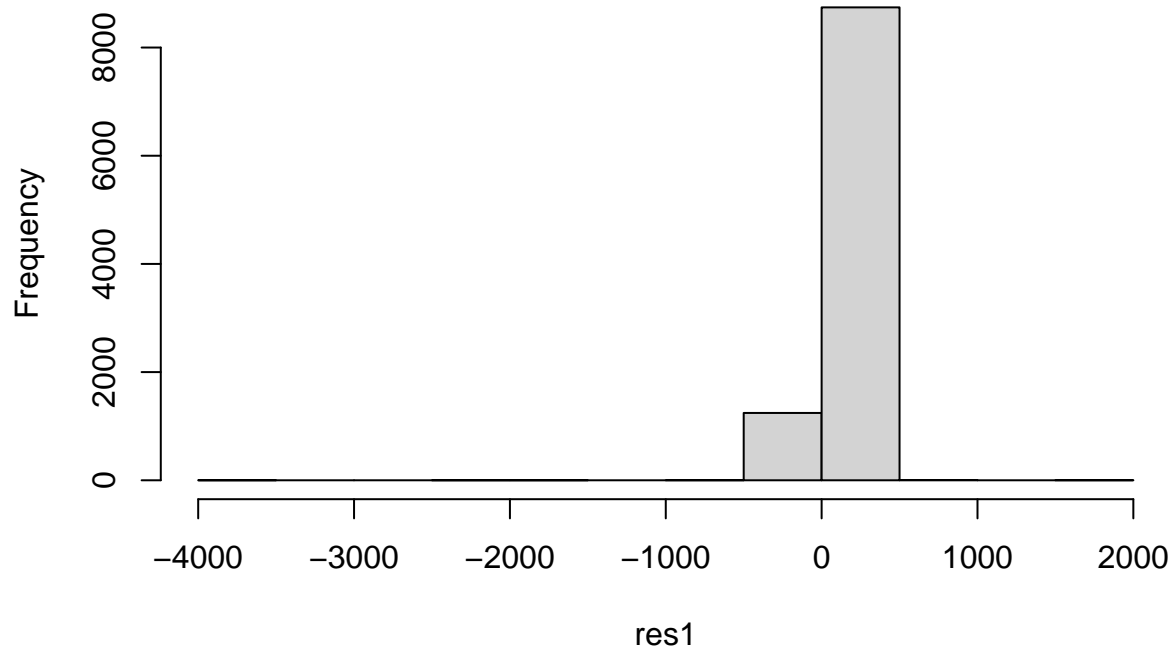


Use Cauchy distributed error

```
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)

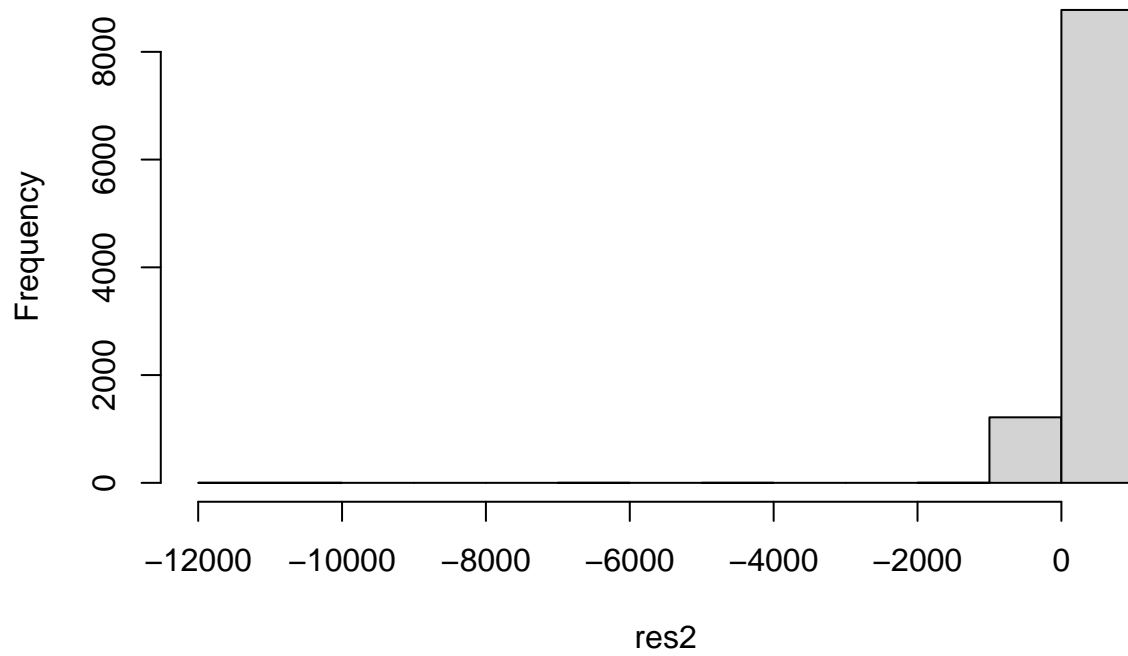
res1 = vec_loop(rep(25,S))
hist(res1)
```

Histogram of res1



```
res2 = vec_loop(rep(100,S))  
hist(res2)
```

Histogram of res2



```

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

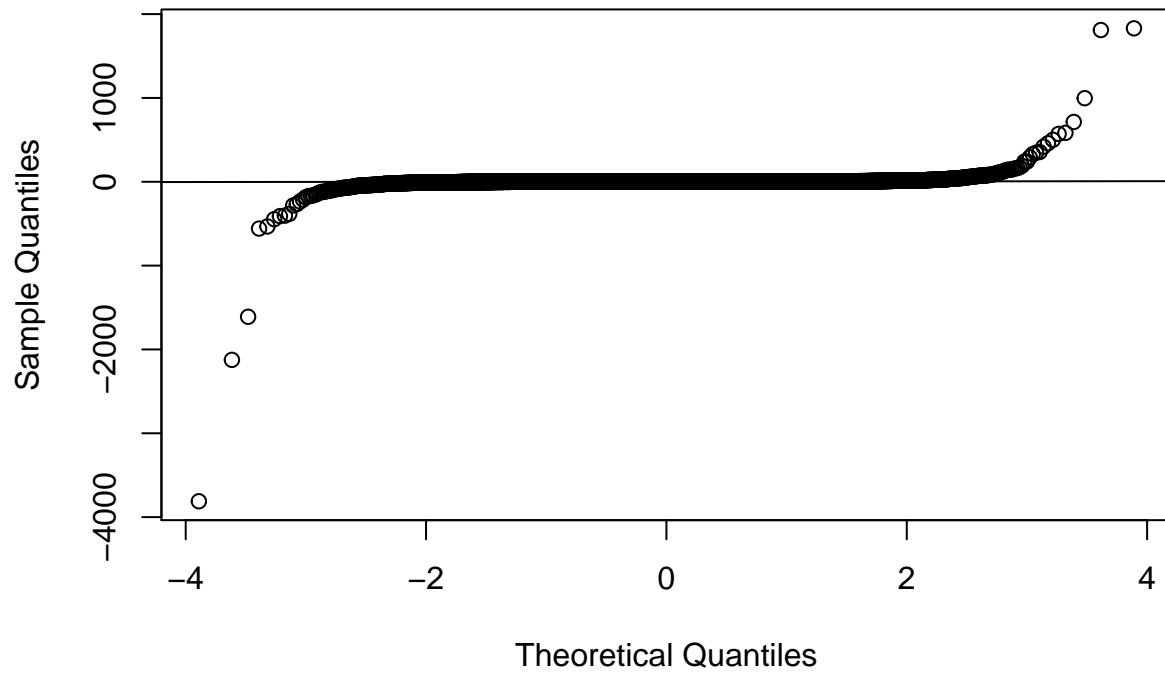
#Anderson-Darling normality test
ad.test(res1) # p-value = 2.2e-16

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

qqnorm(res1)
qqline(res1)

```


Normal Q-Q Plot



```
ad.test(res1) # p-value = 2.2e-16
```

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

```
qqnorm(res2)  
qqline(res2)
```

Normal Q-Q Plot

