stats191 homework 4

Zolboo Chuluunbaatar

Question 1

1. Fit a linear regression model to the observed counts as a linear function of T and P.

```
asthma.table = read.table("http://stats191.stanford.edu/data/asthma.table", header=TRUE, sep='')
head(asthma.table)
```

```
##    T  P Y
## 1 56 38 3
## 2 57 39 2
## 3 58 72 2
## 4 60 43 1
## 5 61 39 1
## 6 62 44 4
```

```
asthma.lm <- lm(Y ~ T + P, data = asthma.table)
summary(asthma.lm)
```

```
##
## Call:
## lm(formula = Y ~ T + P, data = asthma.table)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8204 -1.0570  0.2916  1.1167  2.7060
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.48541    2.60957  -1.336   0.1928
## T            0.05544    0.03558   1.558   0.1308
## P            0.04736    0.01812   2.614   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.566 on 27 degrees of freedom
## Multiple R-squared:  0.2576, Adjusted R-squared:  0.2026
## F-statistic: 4.684 on 2 and 27 DF,  p-value: 0.01793
```

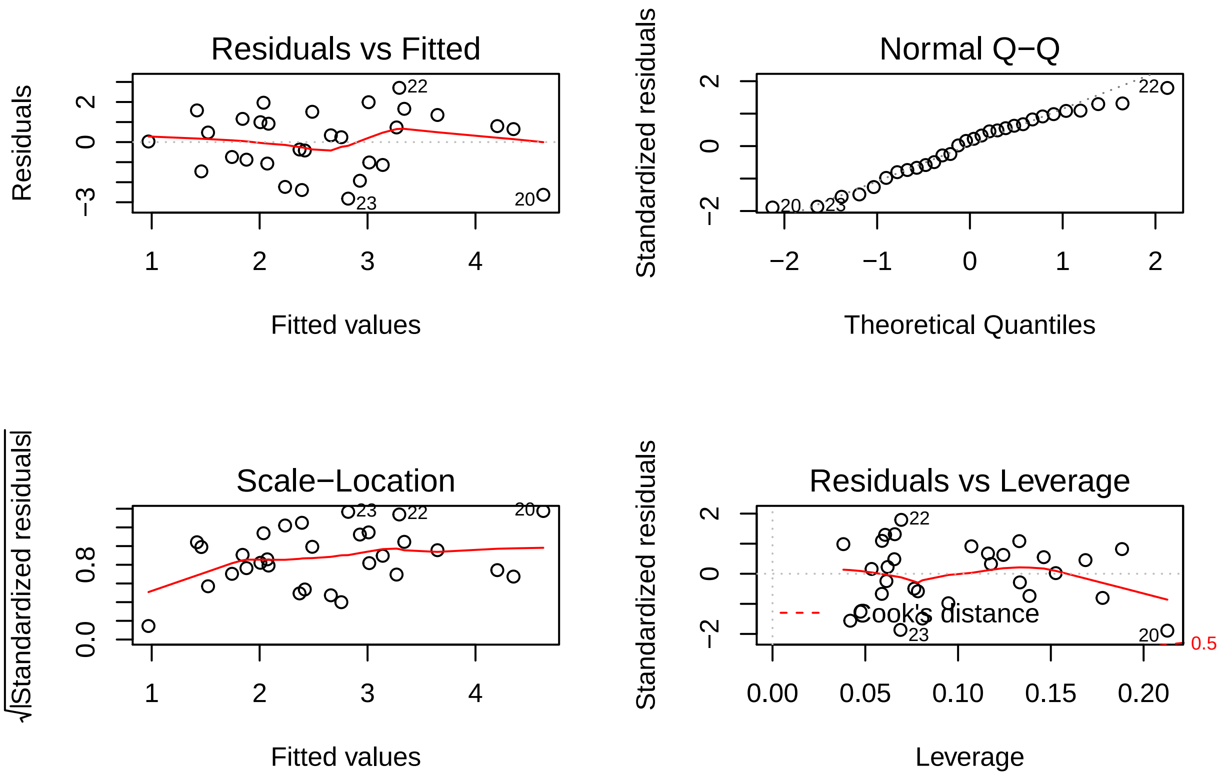
2. Looking at the usual diagnostics plots, does the constant variance assumption seem justified?

We look at the "Residuals vs Fitted value" 1st plot, where the red broken line indicates that the constant variance assumption doesn't seem justified.

Moreover, we look at the Breusch–Pagan test result, (with p-value less than 0.05), we reject the null hypothesis that the constant variance assumtion holds.

```r
par(mfrow = c(2,2))
plot(asthma.lm)
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```



```r
bptest(asthma.lm)  # Constant variance test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  asthma.lm
## BP = 2.7452, df = 2, p-value = 0.2534
```

3. The outcomes are counts, for which a common model is the so-called Poisson model which says that Var(Y)=E(Y) . In words, this says that the variance of the outcome is equal to the expected value of the outcome. Using a two-stage procedure, fit a weighted least squares regression to Y as a function of T and P with weights being inversely proportional to the fitted values of the initial model in 1.

```
asthma.weights = 1.0 / fitted.values(asthma.lm)

asthma1.lm <- lm(Y ~ T + P , weights = asthma.weights, data = asthma.table)
summary(asthma1.lm)
```

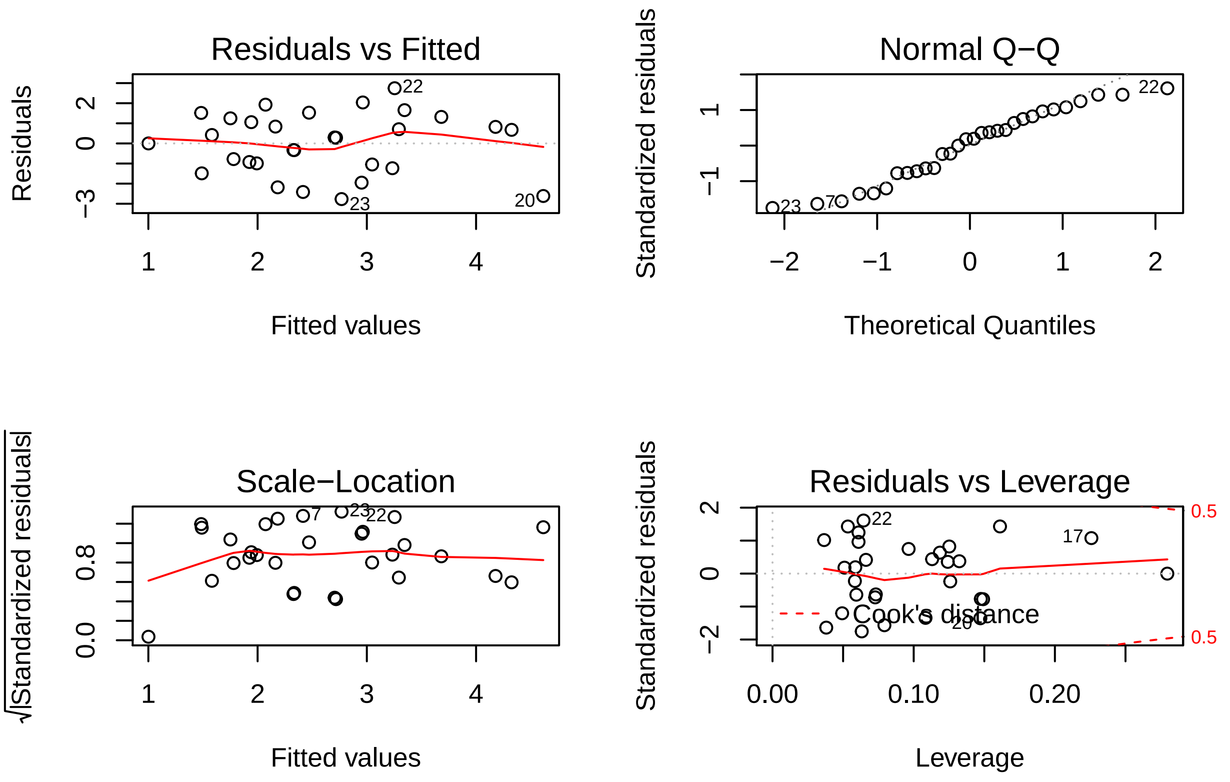
```
##
## Call:
## lm(formula = Y ~ T + P, data = asthma.table, weights = asthma.weights)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6487 -0.6863  0.1753  0.7325  1.5127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.14233    2.32661  -1.351   0.1880
## T            0.04965    0.03223   1.540   0.1351
## P            0.04856    0.01831   2.652   0.0132 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9717 on 27 degrees of freedom
## Multiple R-squared:  0.2585, Adjusted R-squared:  0.2036
## F-statistic: 4.706 on 2 and 27 DF,  p-value: 0.01764
```

4. Looking at the usual diagnostics plots of this model (which takes the weights into account), does the constant variance assumption seem more reasonable? (The change may not be astonishing – the point of the problem is to try using weighted least squares.)

We look at the "Residuals vs Fitted value" 1st plot, where the red broken line indicates that the constant variance assumption doesn't seem justified.

Moreover, if we look at the Breusch–Pagan test result, the p-value = 0.25 is less than 0.05, we reject the null hypothesis that the constant variance assumtion holds.

```
par(mfrow = c(2,2))
plot(asthma1.lm)
```



```r
library(lmtest)

bptest(asthma1.lm)  # Constant variance test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  asthma1.lm
## BP = 2.7452, df = 2, p-value = 0.2534
```

###. 5 Using the weighted least squares fit, test the hypotheses at level α=0.05 that the number of asthma cases is uncorrelated to the temperature allowing for pollutants; the number of asthma cases is uncorrelated to the atmospheric pollutants allowing for temperature.

Null hypothesis: H_0 : the number of asthma cases is uncorrelated to the temperature allowing for pollutants.

By the correlation test, we find that the sample correlation between Y and T is 0.2555195, and the p-value is statistically not significant $0.1729396 > 0.05$. Therefore, we reject the null hypothesis that Y and T are uncorrelated.

```r
library(weights)
wtd.cor(asthma.table$Y, asthma.table$T, weight = asthma.weights)
```

```
##   correlation   std.err  t.value   p.value
## Y   0.2555195 0.1827088 1.398507 0.1729396
```

Null hypothesis: H_0 : the number of asthma cases is uncorrelated to the atmospheric pollutants allowing for temperature.

By the correlation test, we find that the sample correlation between Y and P is 0.439697, and the p-value is statistically significant $0.0150455 < 0.05$. Therefore, we fail to reject the null hypothesis that Y and P are uncorrelated.

```
wtd.cor(asthma.table$Y, asthma.table$P, weight = asthma.weights)
```

```
##   correlation   std.err  t.value   p.value
## Y    0.439697 0.1697337 2.590512 0.0150455
```

Question 2.

1. Fit a linear regression model connecting DJIA with Time using all 262 trading days in 1996. Is the linear trend model adequate? Examine the residuals for time dependencies, including a plot of the autocorrelation function.

We see a linear trend in the following plot.

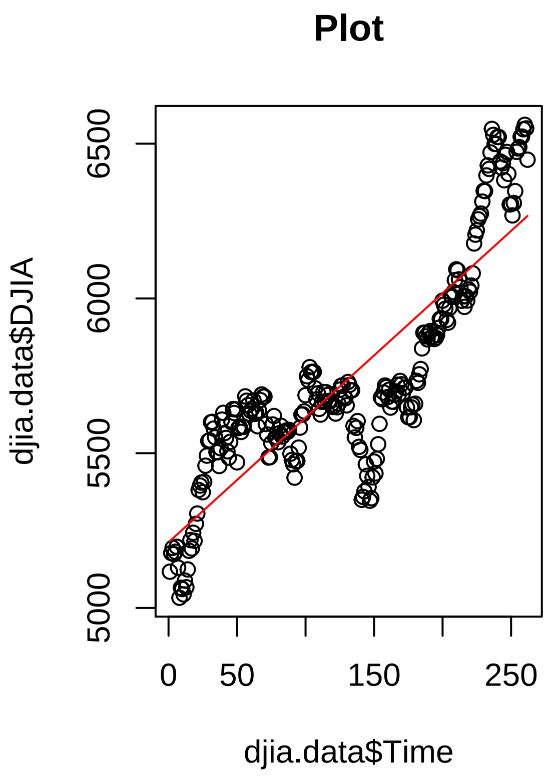
```r
file <- "http://www1.aucegypt.edu/faculty/hadi/RABE5/Data5/P229-30.txt"
djia.data <- read.delim(file, header = TRUE, sep = "\t")
head(djia.data)
```

```
##     Date    DJIA Time
## 1 1/1/96 5117.12    1
## 2 1/2/96 5177.45    2
## 3 1/3/96 5194.07    3
## 4 1/4/96 5173.84    4
## 5 1/5/96 5181.43    5
## 6 1/8/96 5197.68    6
```

```r
djia.lm <- lm(DJIA ~ Time, data = djia.data)
summary(djia.lm)
```

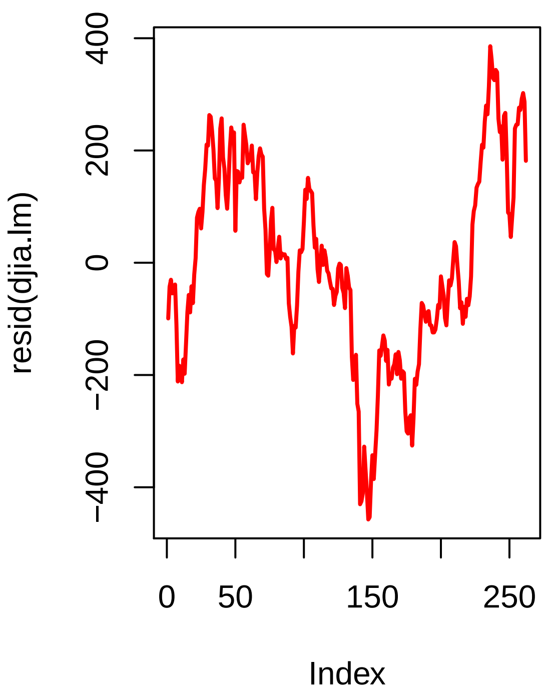
```
##
## Call:
## lm(formula = DJIA ~ Time, data = djia.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -457.33 -111.62   -9.81  145.71  385.75
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5212.2995    22.1965   234.8   <2e-16 ***
## Time           4.0243     0.1463    27.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 179.1 on 260 degrees of freedom
## Multiple R-squared:  0.7442, Adjusted R-squared:  0.7432
## F-statistic: 756.5 on 1 and 260 DF,  p-value: < 2.2e-16
```

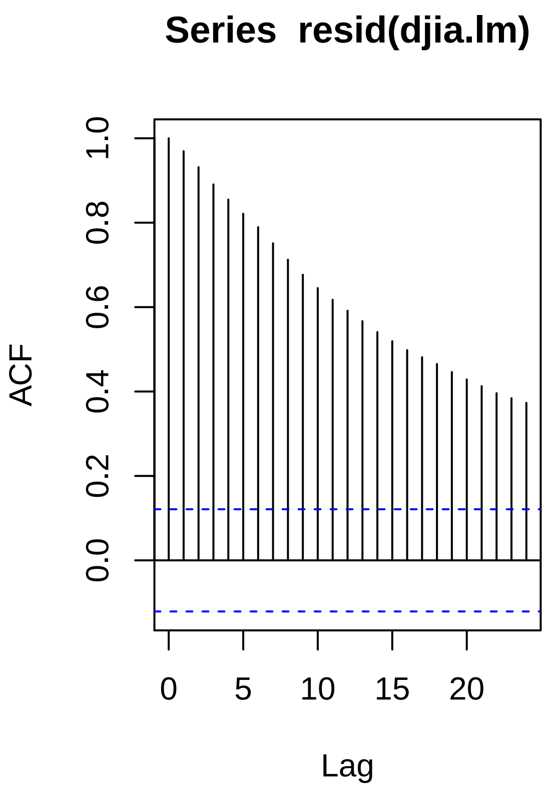
```r
par(mfrow=c(1,2))
plot(djia.data$DJIA ~ djia.data$Time, main= "Plot")
lines(djia.data$Time, predict(djia.lm),col='red')
```



ACF of residuals From the autoplot, we see increasing and decreasing trend in the time series. ACF plot reveals spikes everywhere. The residuals look highly correlated from the autocorrelation plot.

```r
par(mfrow=c(1,2))
plot(resid(djia.lm), type='l', lwd=2, col='red')
acf(resid(djia.lm))
```





Also, DW = 0.056 indicates the high autocorrelation.

```
dwtest(djia.lm)
```

```
##
##  Durbin-Watson test
##
## data:  djia.lm
## DW = 0.055891, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

2. Regress DJIA[t] against its lagged by one version DJIA[t-1]. Is this an adequate model? Are there any evidences of autocorrelation in the residuals?

We convert the data.frame to time series as follows.

```
library(xts)
djia.ts <- xts(djia.data$DJIA, as.Date(djia.data$Date, format = '%m/%d/%Y'))
colnames(djia.ts) <- "DJIA"
head(djia.ts)
```

```
##              DJIA
## 96-01-01 5117.12
## 96-01-02 5177.45
## 96-01-03 5194.07
## 96-01-04 5173.84
## 96-01-05 5181.43
## 96-01-08 5197.68
```

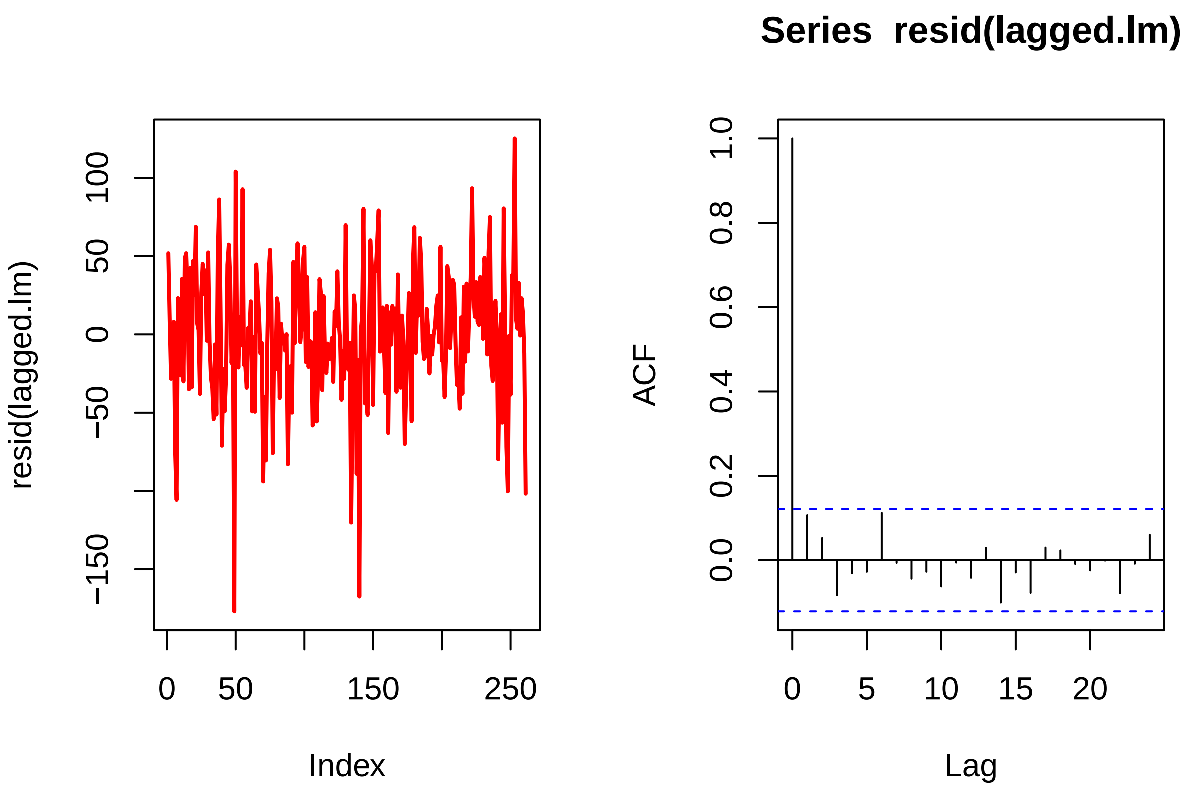
We regress DJIA[t] against its lagged by one version DJIA[t-1].

```r
djia <- as.numeric(djia.ts$DJIA)
lagged <- as.numeric(lag(djia.ts$DJIA, 1))

lagged.lm <- lm(djia ~ lagged, data = djia.ts)
summary(lagged.lm)
```

```
##
## Call:
## lm(formula = djia ~ lagged, data = djia.ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -176.878  -22.397   -0.641   26.478  125.139
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.898384  42.989100   0.858    0.392
## lagged       0.994459   0.007477 133.002   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.37 on 259 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9856, Adjusted R-squared:  0.9855
## F-statistic: 1.769e+04 on 1 and 259 DF,  p-value: < 2.2e-16
```

ACF of residuals (for regression DJIA[t] ~ DJIA[t-1]). ACF plot reveals spike only at lag 0.So there is no evidence of autocorrelation.

```r
par(mfrow=c(1,2))
plot(resid(lagged.lm), type='l', lwd=2, col='red')
acf(resid(lagged.lm))
```



By Durbin-Watson test, D-W Statistic is 1.7586 (near 2). So, we fail to reject the null hypothesis. In other words, there is no evidence of autocorrelation in the residuals(for regression DJIA[t] ~ DJIA[t-1]).

```
library(car)
durbinWatsonTest(lagged.lm)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.1066715      1.758642   0.026
##  Alternative hypothesis: rho != 0
```

3.The variability (volatility) of the daily DJIA is large, and to accomodate this phenomenon the analysis is crried out on the logarithm of the DJIA. Repeat 2. above using log(DJIA) instead of DJIA.

We regress log(DJIA[t]) against its lagged by one version log(DJIA[t-1]).

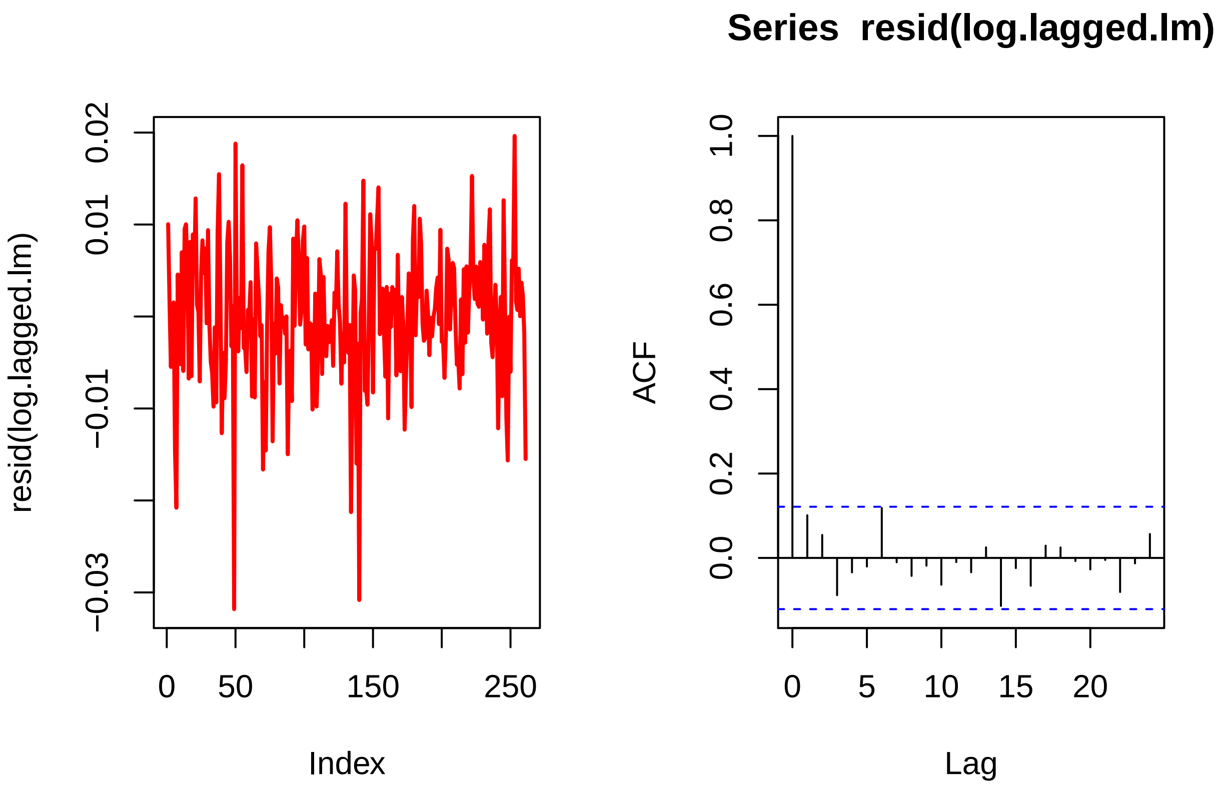
```
log.djia <- log(djia)
log.lagged <- log(lagged)

log.lagged.lm <- lm(log.djia ~ log.lagged, data = djia.ts)
summary(log.lagged.lm)
```

```
##
## Call:
## lm(formula = log.djia ~ log.lagged, data = djia.ts)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.031817 -0.003950  0.000008  0.004895  0.019632
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.062131   0.066402   0.936     0.35
## log.lagged  0.992922   0.007674 129.396   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007455 on 259 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9848, Adjusted R-squared:  0.9847
## F-statistic: 1.674e+04 on 1 and 259 DF,  p-value: < 2.2e-16
```

ACF of residuals (for regression DJIA[t] ~ DJIA[t-1]). ACF plot reveals spike only at lag 0. So there is no evidence of autocorrelation.

```
par(mfrow=c(1,2))
plot(resid(log.lagged.lm), type='l', lwd=2, col='red')
acf(resid(log.lagged.lm))
```



By Durbin-Watson test, D-W Statistic is 1.774118 (close to 2). So, we fail to reject the null hypothesis. In other words, there is no evidence of autocorrelation in the residuals (for regression log(DJIA[t]) ~ log(DJIA[t-1])).

```
durbinWatsonTest(log.lagged.lm)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1        0.101101      1.774118   0.064
##  Alternative hypothesis: rho != 0
```

4. A simplified version of the random walk model of stock prices states that the best prediction of the stock index at Time=t is the value of the index at Time=t-1. Show that this corresponds to a simple linear regression model for 2. with an intercept of 0 and a slope of 1.

From question 2, the fitted linear model function is

$$DJ\hat{I}A[t] = slope * DJIA[t-1] + intercept$$

By the simplified version of the random walk model of stack price,

$$DJ\hat{I}A[t] = DJIA[t-1]$$

From these two equations, we get $slope = 1$ and $intercept = 0$.

5. Carry out the the appropriate tests of significance at level alpha = 0.05 for 4. Test each coefficient separately (t -tests) , then test both simultaneously (i.e. an F test).

```
summary(lagged.lm)
```

```
##
## Call:
## lm(formula = djia ~ lagged, data = djia.ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -176.878  -22.397   -0.641   26.478  125.139
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.898384  42.989100   0.858    0.392
## lagged       0.994459   0.007477 133.002   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.37 on 259 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9856, Adjusted R-squared:  0.9855
## F-statistic: 1.769e+04 on 1 and 259 DF,  p-value: < 2.2e-16
```

From the linear regression fit summary above, we immediately see that p-value for Hypothesis test H_0 : $intercept = 0$ is 0.392.

Now we need to do hypothesis testing for whether the slope is equal to one. We can construct an offset by 1 on the "lagged" so that this will reset the slope to zero.

```
lagged.offset.lm <- lm(djia ~ 1 + lagged + offset(lagged), data = djia.ts)
summary(lagged.offset.lm)
```

```
##
## Call:
## lm(formula = djia ~ 1 + lagged + offset(lagged), data = djia.ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -176.878  -22.397   -0.641   26.478  125.139
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.898384  42.989100   0.858    0.392
## lagged      -0.005541   0.007477  -0.741    0.459
##
## Residual standard error: 42.37 on 259 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9856, Adjusted R-squared:  0.9855
## F-statistic: 1.769e+04 on 1 and 259 DF,  p-value: < 2.2e-16
```

From the summary of the new fit with an offset, we see that p-value for the slope is 0.459, and is statistically not significant.

Since P-values for the slope and the intercept are all statistically not significant, we fail to reject the null hypotheses.

Lastly, we perform Linear Hypothesis test which performs F test to compare the following two models:

$$\text{Model} = djia \sim lagged$$

$$\text{Restricted model} = djia \sim 1 + lagged + offset(lagged)$$

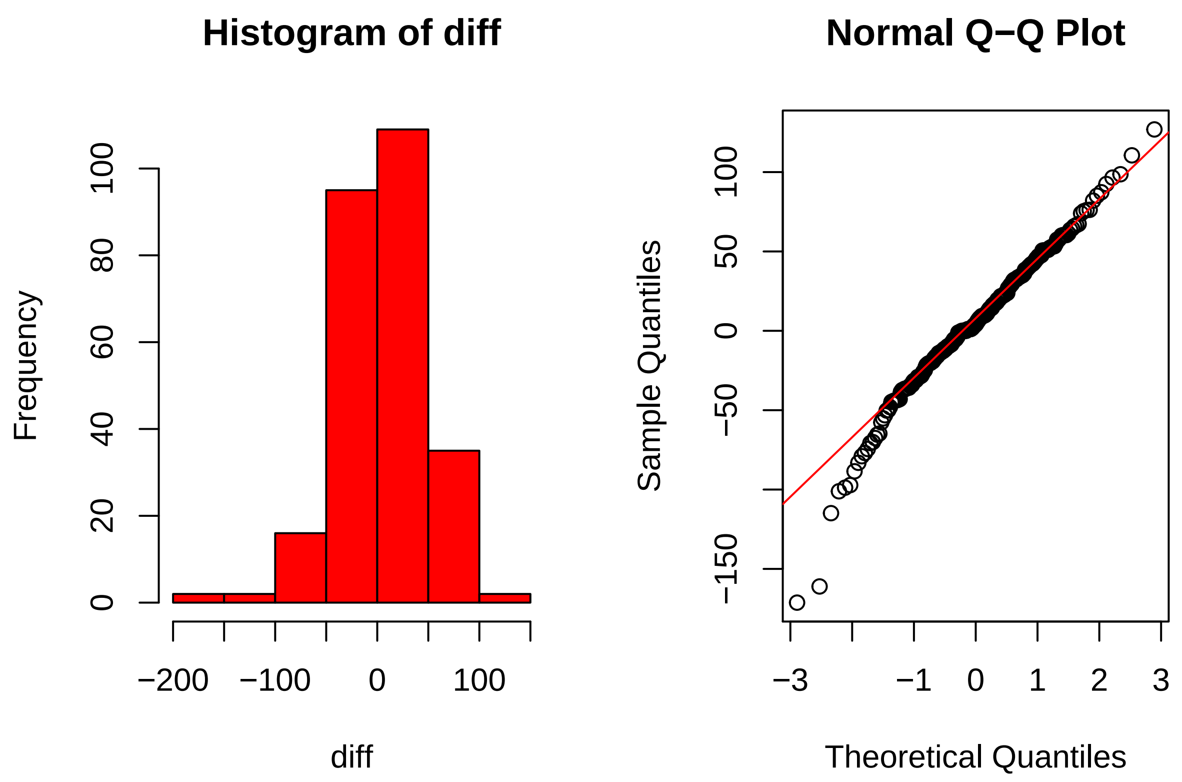
```
fit <- lm(djia ~ lagged, data = djia.ts)
linearHypothesis(fit, c("(Intercept) = 0", "lagged=1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## (Intercept) = 0
## lagged = 1
##
## Model 1: restricted model
## Model 2: djia ~ lagged
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    261 472757
## 2    259 464981  2      7775 2.1654 0.1168
```

P-value for this F-test is 0.1168 statistically insignificant, so we fail to reject the null hypothesis that slope = 0 and intercept = 1.

6. The random walk theory implies that the first differences of the index (the difference between successive values) should be independently normally distributed with mean zero and constant variance. What kind of plot can be used to visually assess this hypothesis? Provide the plot.

```
par(mfrow = c(1,2))
diff <- djia - lagged
hist(diff, col = "red")
qqnorm(diff)
qqline(diff, col = 2)
```



Shapiro test with pvalue = 0.0003357 implies that the null hypothesis that the first difference is NOT normally distributed.

```
shapiro.test(diff)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diff
## W = 0.97717, p-value = 0.0003357
```

Question 3

1.Write a function with the same regression function but errors that are not normally distributed using, say, rexp(n)-1. Use weighted least squares and construct the Z statistic (i.e. ignore degrees of freedom, pretending they are Inf) to test H0:β_1=2 . Does the Z -statistic have close to a N(0,1) distribution? How often is your Z statistic larger than the usual 5% threshold?

The function below returns an indicator variable for the pvalue with 0.05 threshold, and the Z score. Here count =1 if pvalue is greater than 0.05, count = 0 otherwise.

```
fun1 <- function() {
  X <- runif(100)
  err <- rexp(100) - 1
  Y <- 1 + 2 * X + err
  wts <- 1/fitted(lm(Y ~ X))^2
  weighted.lm <- lm(Y~ X, weights = wts)
  summary(weighted.lm)
  beta1 <- summary(weighted.lm)$coefficients[2,1]
  mu <- 2
  std <- summary(weighted.lm)$coefficients[2,2]
  Z <- (beta1 - mu)/std
  pvalue <- 2*pnorm(-abs(Z))
  count <- 0
  if (pvalue > 0.05) count <- 1
  ci <- 2 * qnorm(0.975) * std
  return(cbind(count, Z, ci))
}
```

We repeatedly perform the above function to get 100 data. Here, count/nism represents how often the p-value is greater than the usual 5%.

```
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, fun1()[2])
    count = count + fun1()[1]
}
```

In this experiment, the value is

```
print(count/nsim)
```

```
## [1] 0.93
```

We test if Z-statistic is normally distributed:

mean and standard deviation of the Z-statistic:

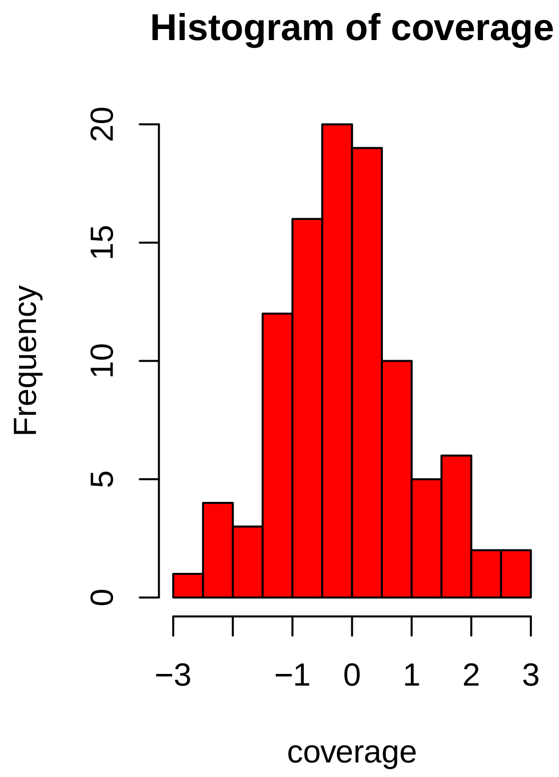
```
print(c(mean(coverage), sqrt(var(coverage))))
```

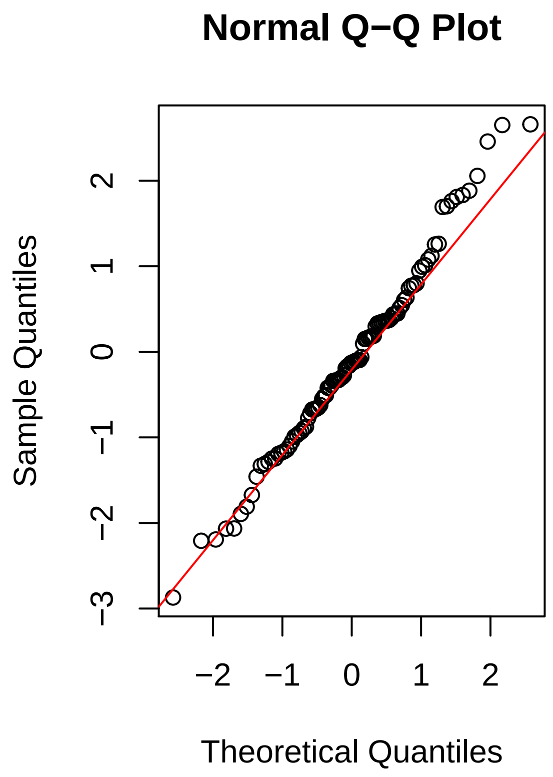
```
## [1] -0.08986187  1.10887869
```

Histogram, QQ-Plot

```
par(mfrow = c(1,2))
hist(coverage, col = "red")
```

```r
qqnorm(coverage)
qqline(coverage, col = 2)
```





Shapiro test for normality

```r
shapiro.test(coverage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coverage
## W = 0.98858, p-value = 0.5519
```

2. Write a new function with the same regression function but multiply the errors by sqrt(X+1) so the i -th variance (given X[i]) is 1+X[i] . Repeat the experiment in part 1.

The function below returns an indicator variable for the pvalue with 0.05 threshold, and the Z score. Here count =1 if pvalue is greater than 0.05, count = 0 otherwise.

```r
fun2 <- function() {
  X <- runif(100)
  err <- (rexp(100) - 1)*sqrt(X+1)
  Y <- 1 + 2 * X + err
  wts <- 1/(1 + X)
  weighted.lm <- lm(Y~ X, weights = wts)
  summary(weighted.lm)
  beta1 <- summary(weighted.lm)$coefficients[2,1]
  mu <- 2
  std <- summary(weighted.lm)$coefficients[2,2]
  Z <- (beta1 - mu)/std
  pvalue <- 2*pnorm(-abs(Z))
  count <- 0
  if (pvalue > 0.05) count <- 1
  ci <- 2 * qnorm(0.975) * std
   return(cbind(count, Z, ci))
}
```

We repeatedly perform the above function to get 100 data. Here, count/nism represents how often the p-value is greater than the usual 5%.

```
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, fun2()[2])
    count = count + fun2()[1]
}
```

In this experiment, the value is

```
print(count/nsim)
```

```
## [1] 0.94
```

We test if Z-statistic is normally distributed:

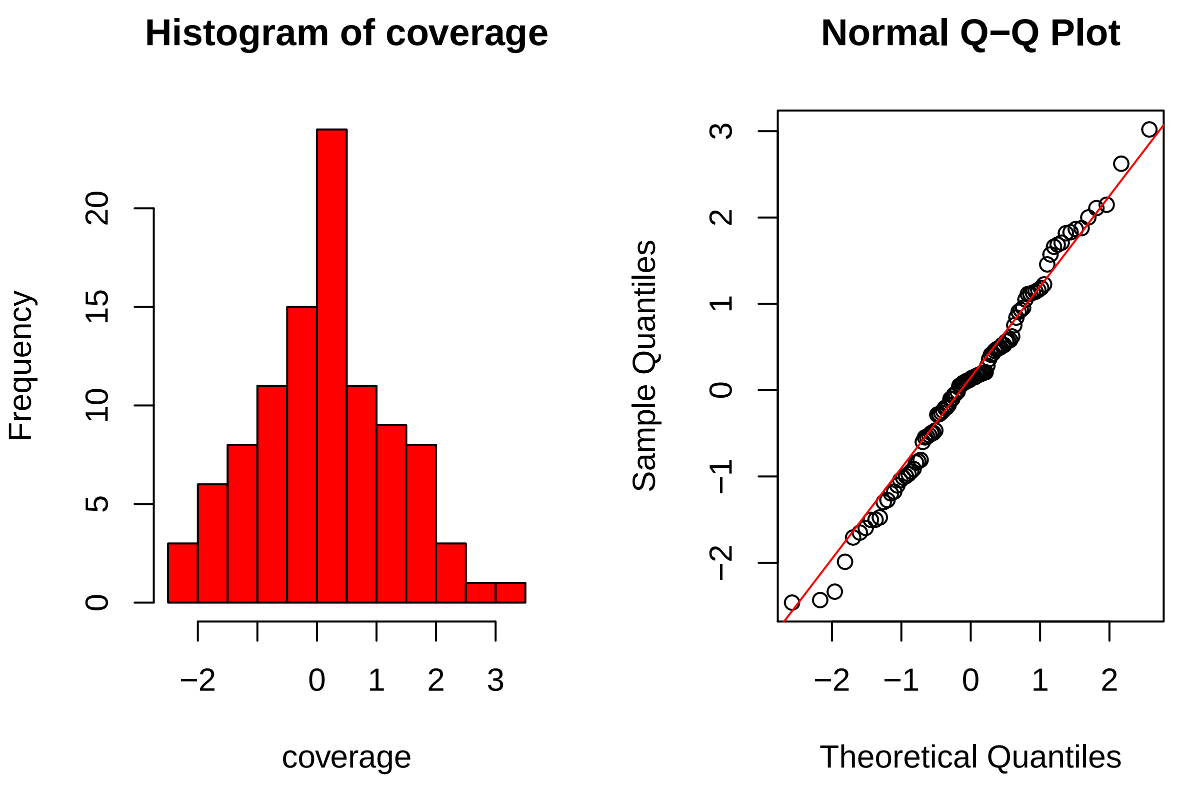
mean and standard deviation of the Z-statistic:

```
print(c(mean(coverage), sqrt(var(coverage))))
```

```
## [1] 0.1119019 1.1297199
```

Histogram, QQ-Plot

```
par(mfrow = c(1,2))
hist(coverage, col = "red")
qqnorm(coverage)
qqline(coverage, col = 2)
```



Shapiro test for normality

```
shapiro.test(coverage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coverage
## W = 0.99163, p-value = 0.7941
```

3. Redo part 2. replacing sqrt(X+1) by exp(0.5 * (1 + 5 * X)).

The function below returns an indicator variable for the pvalue with 0.05 threshold, and the Z score. Here count =1 if pvalue is greater than 0.05, count = 0 otherwise.

```
fun3 <- function() {
  X <- runif(100)
  err <- (rexp(100) - 1)*exp(0.5 * (1 + 5 * X))
  Y <- 1 + 2 * X + err
  wts <- 1/(exp(1 + 5 * X))
  weighted.lm <- lm(Y~ X, weights = wts)
  summary(weighted.lm)
  beta1 <- summary(weighted.lm)$coefficients[2,1]
  mu <- 2
  std <- summary(weighted.lm)$coefficients[2,2]
  Z <- (beta1 - mu)/std
  pvalue <- 2*pnorm(-abs(Z))
  count <- 0
  if (pvalue > 0.05) count <- 1
```

```
   ci <- 2 * qnorm(0.975) * std
   return(cbind(count, Z, ci))
}
```

We repeatedly perform the above function to get 100 data. Here, count/nism represents how often the p-value is greater than the usual 5%.

```
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, fun3()[2])
    count = count + fun3()[1]
}
```

In this experiment, the value is

```
print(count/nsim)
```

```
## [1] 0.97
```

We test if Z-statistic is normally distributed:

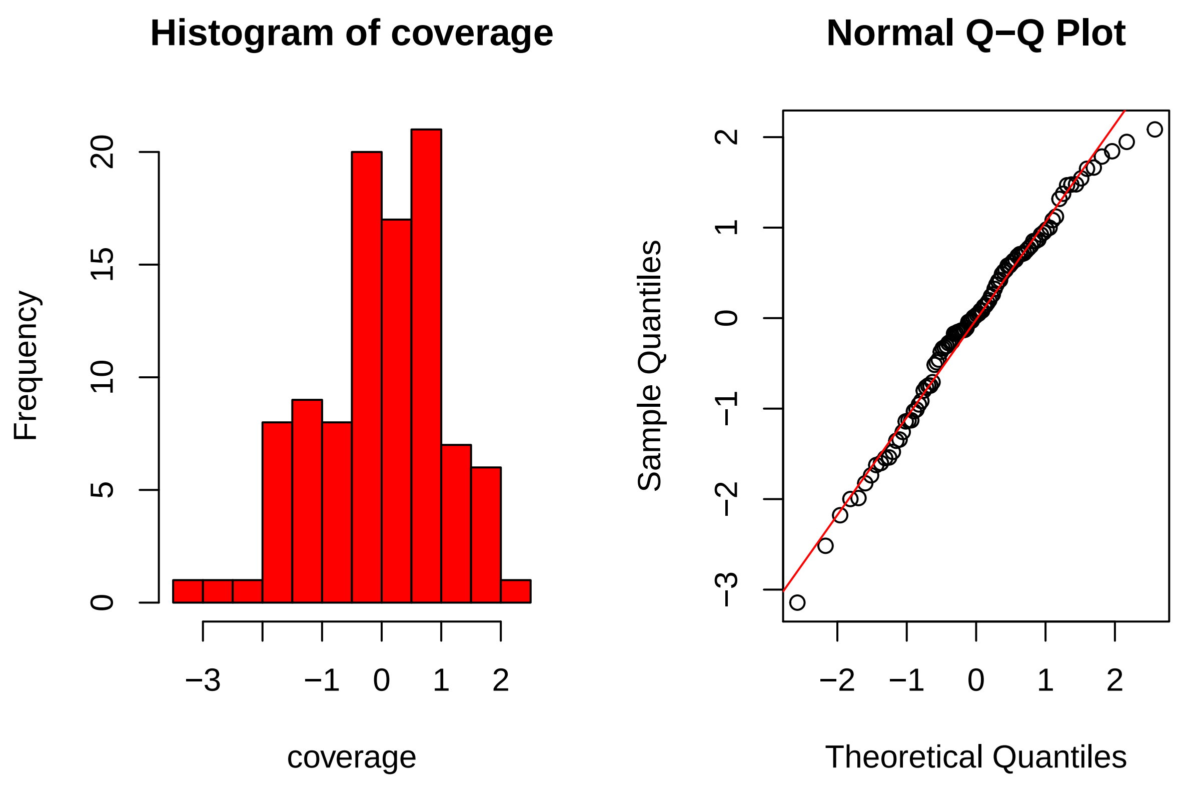
mean and standard deviation of the Z-statistic:

```
print(c(mean(coverage), sqrt(var(coverage))))
```

```
## [1] -0.02139735  1.06834434
```

Histogram, QQ-Plot

```
par(mfrow = c(1,2))
hist(coverage, col = "red")
qqnorm(coverage)
qqline(coverage, col = 2)
```



Shapiro test for normality

```r
shapiro.test(coverage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coverage
## W = 0.9845, p-value = 0.2922
```

Question 4

1. Repeat 1. of Question 3 above using the bootstrap estimate of standard error to form the Z statistic testing H0:β_1=2 .

```r
library(boot)
library(car)


noise = function(n) { return(rexp(n) - 1) }

simulate_correct1 = function(n=100, b=2) {

    X = runif(n)
    Y <- 1 + b * X + noise(n)
    Y.lm = lm(Y ~ X)
    summary(Y.lm)
    int_param = confint(Y.lm)[2,]
    D = data.frame(X, Y)
    bootstrap_stat = function(D, bootstrap_idx) {
       return(lm(Y ~ X, data=D[bootstrap_idx,])$coef)
    }
    boot_results = boot(D, bootstrap_stat, R=1000)

  boot_results
  beta1 <- boot_results$t0[2]

   pairs_SE <- sqrt(cov(boot_results$t)[2,2])

   Z <- (beta1 - b)/pairs_SE
   pvalue <- 2*pnorm(-abs(Z))
   count <- 0
   if (pvalue > 0.05) count <- 1
   int_pairs = c(coef(Y.lm)[2] - qnorm(0.975) * pairs_SE,
                 coef(Y.lm)[2] + qnorm(0.975) * pairs_SE)
   ci <- int_pairs[2] - int_pairs[1]
   return(cbind(count, Z, ci))
}
```

```r
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, simulate_correct1()[2])
    count = count + simulate_correct1()[1]
}
```

In this experiment, the value is

```r
print(count/nsim)
```

```
## [1] 0.98
```

We test if Z-statistic is normally distributed:

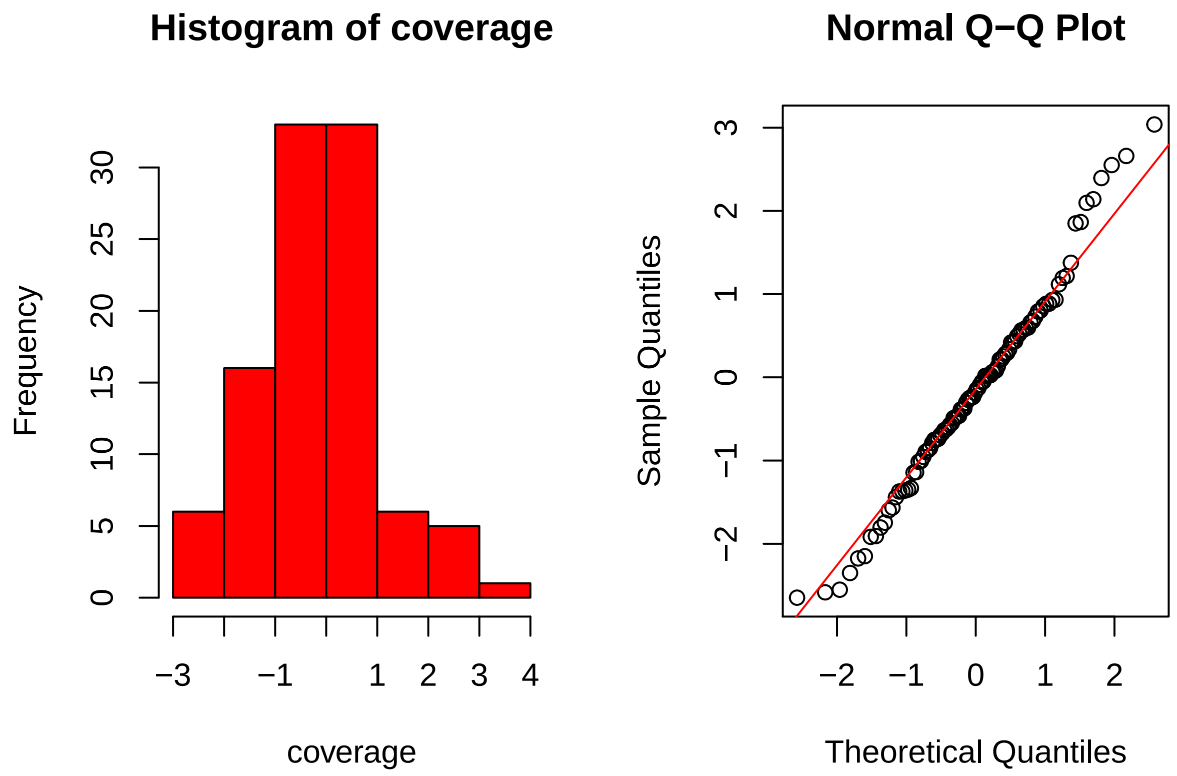
mean and standard deviation of the Z-statistic:

```r
print(c(mean(coverage), sqrt(var(coverage))))
```

```
## [1] -0.1515412  1.1779864
```

Histogram, QQ-Plot

```r
par(mfrow = c(1,2))
hist(coverage, col = "red")
qqnorm(coverage)
qqline(coverage, col = 2)
```



Shapiro test for normality

```r
shapiro.test(coverage)
```

```
## 
##  Shapiro-Wilk normality test
## 
## data:  coverage
## W = 0.98566, p-value = 0.3537
```

2. Repeat 2. of Question 3 above using the bootstrap for the two different data generating mechanisms for the variance 1+X.

```r
noise = function(n) { return(rexp(n) - 1) }

simulate_correct2 = function(n=100, b=2) {

    X = runif(n)
    Y <- 1 + b * X + noise(n) * sqrt(X + 1)
    Y.lm = lm(Y ~ X)
    summary(Y.lm)
    int_param = confint(Y.lm)[2,]
    D = data.frame(X, Y)
    bootstrap_stat = function(D, bootstrap_idx) {
       return(lm(Y ~ X, data=D[bootstrap_idx,])$coef)
    }
    boot_results = boot(D, bootstrap_stat, R=1000)

  boot_results
  beta1 <- boot_results$t0[2]

   pairs_SE <- sqrt(cov(boot_results$t)[2,2])
   Z <- (beta1 - b)/pairs_SE
   pvalue <- 2*pnorm(-abs(Z))
   count <- 0
   if (pvalue > 0.05) count <- 1
   int_pairs = c(coef(Y.lm)[2] - qnorm(0.975) * pairs_SE,
                 coef(Y.lm)[2] + qnorm(0.975) * pairs_SE)
   ci <- int_pairs[2] - int_pairs[1]
   return(cbind(count, Z, ci))
}
```

```
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, simulate_correct2()[2])
    count = count + simulate_correct2()[1]
}
```

In this experiment, the value is

```
print(count/nsim)
```

```
## [1] 0.93
```

We test if Z-statistic is normally distributed:

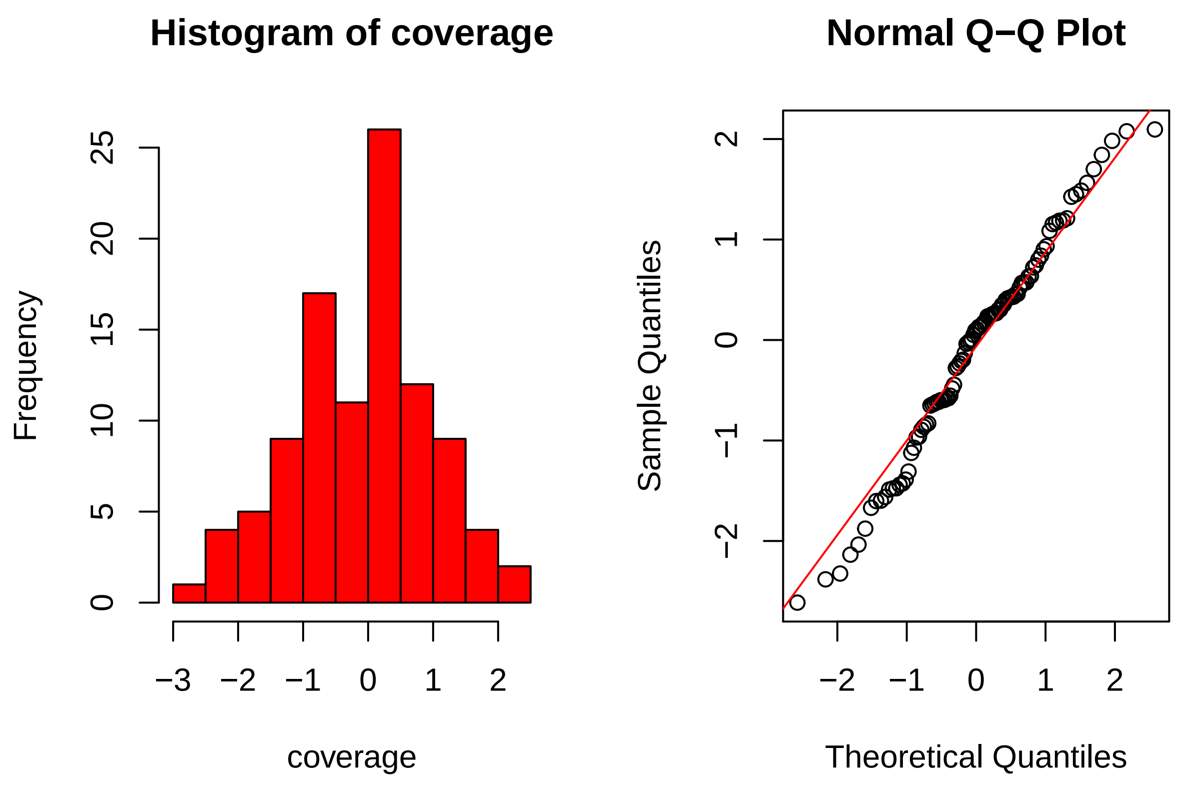
mean and standard deviation of the Z-statistic:

```
print(c(mean(coverage), sqrt(var(coverage))))
```

```
## [1] -0.08400472  1.05600208
```

Histogram, QQ-Plot

```
par(mfrow = c(1,2))
hist(coverage, col = "red")
qqnorm(coverage)
qqline(coverage, col = 2)
```



Shapiro test for normality

```
shapiro.test(coverage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coverage
## W = 0.98547, p-value = 0.3429
```

3. Repeat 2. of Question 3 above using the bootstrap for the two different data generating mechanisms for the variance exp(1+5X).

```
noise = function(n) { return(rexp(n) - 1) }

simulate_correct3 = function(n=100, b=2) {

    X = runif(n)
    Y <- 1 + b * X + noise(n) * exp(0.5 * (1 + 5 * X))
    Y.lm = lm(Y ~ X)
    summary(Y.lm)
    int_param = confint(Y.lm)[2,]
    D = data.frame(X, Y)
    bootstrap_stat = function(D, bootstrap_idx) {
       return(lm(Y ~ X, data=D[bootstrap_idx,])$coef)
    }
    boot_results = boot(D, bootstrap_stat, R=1000)

  boot_results
  beta1 <- boot_results$t0[2]

   pairs_SE <- sqrt(cov(boot_results$t)[2,2])
   Z <- (beta1 - b)/pairs_SE
   pvalue <- 2*pnorm(-abs(Z))
   count <- 0
   if (pvalue > 0.05) count <- 1
   int_pairs = c(coef(Y.lm)[2] - qnorm(0.975) * pairs_SE,
                 coef(Y.lm)[2] + qnorm(0.975) * pairs_SE)
   ci <- int_pairs[2] - int_pairs[1]
   return(cbind(count, Z, ci))
}
```

```
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, simulate_correct3()[2])
    count = count + simulate_correct3()[1]
}
```

In this experiment, the value is

```
print(count/nsim)
```

```
## [1] 0.9
```

We test if Z-statistic is normally distributed:

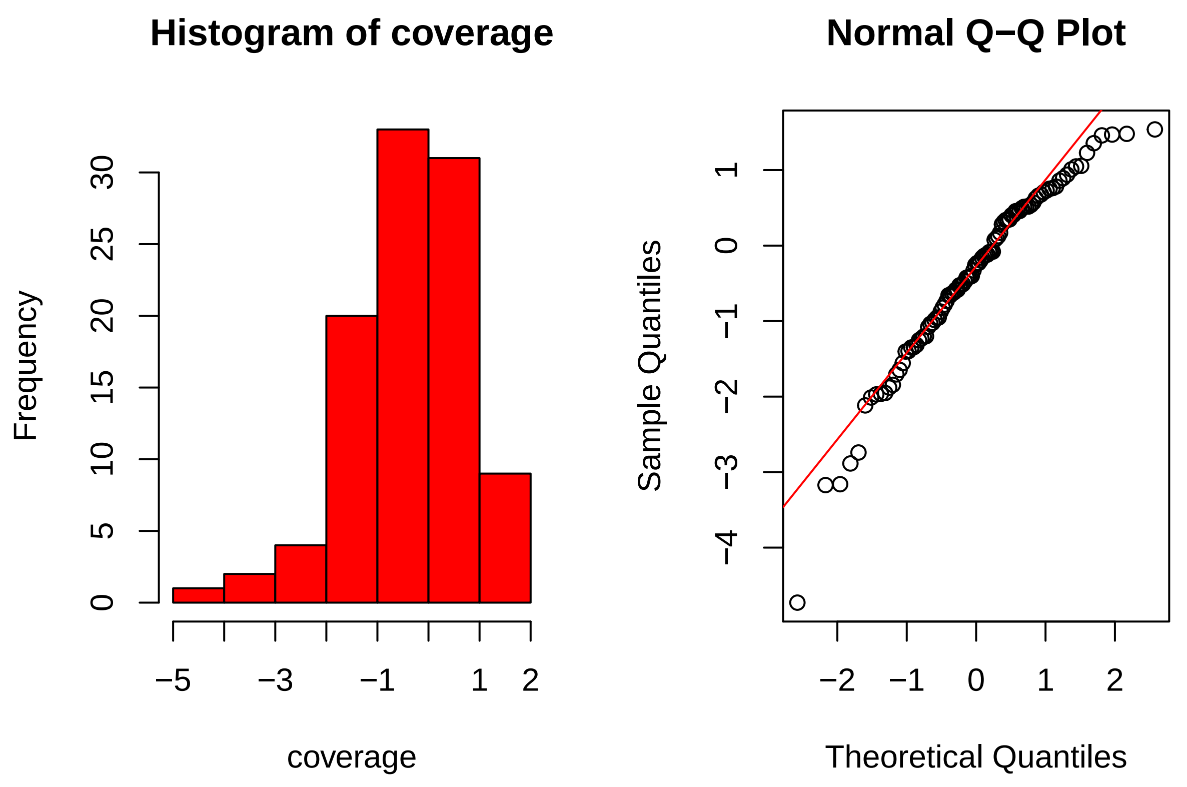
mean and standard deviation of the Z-statistic:

```
print(c(mean(coverage), sqrt(var(coverage))))
```

```
## [1] -0.3827841  1.1554853
```

Histogram, QQ-Plot

```
par(mfrow = c(1,2))
hist(coverage, col = "red")
qqnorm(coverage)
qqline(coverage, col = 2)
```



Shapiro test for normality

```
shapiro.test(coverage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coverage
## W = 0.95738, p-value = 0.002625
```

4. Compare the intervals formed using weighted least squares and those formed using the bootstrap estimate of standard error. Are one method's intervals shorter than the other?

Model in 3.1 versus 4.1 Model 3.1 seems to have longer interval than Modal 4.1.

```
nsim = 100
count1.ci = 0
for (i in 1:nsim) {
  if (fun1()[3] > simulate_correct1()[3]) count1.ci = count1.ci + 1
}

count1.ci
```

```
## [1] 63
```

Model in 3.2 versus 4.2 Model 3.2 seems to have a longer (about the same) than Modal 4.2.

```
nsim = 100
count2.ci = 0
for (i in 1:nsim) {
    if (fun2()[3] > simulate_correct2()[3]) count2.ci = count2.ci + 1
}
count2.ci
```

```
## [1] 51
```

Model in 3.3 versus 4.3 Model 3.3 seems to have a shorter Interval than Modal 4.3.

5. Write a new function with same regression function but errors that are not independent.Do this by first generating a vector of errors error and then returning a new vector whose first entry is error[1] but for i>1 the i -th entry is error[i-1] + error[i]. This is the same data generating mechanism we saw in Question 4 from Assignment 2 so you can reuse your code. Repeat the experiment in part 1. Does the Z -statistic approximately have the distribution it should have? Comment on whether the bootstrap can fix this problem of correlated errors.

```
nsim = 100
count3.ci = 0
for (i in 1:nsim) {
    if (fun3()[3] > simulate_correct3()[3]) count3.ci = count3.ci + 1
}
count3.ci
```

```
## [1] 3
```

```
library(boot)

noise = function(n) {
  errors <- c(0, n)
  err <- rt(n, 5)
  errors[1] <- err[1]
  for(i in 2:n) {
    errors[i] <- err[i-1] + err[i]
  }
  return(errors)
}

simulate_correct2 = function(n=100, b=2) {

    X = runif(n)
    Y <- 1 + b * X + noise(n)
    Y.lm = lm(Y ~ X)
    summary(Y.lm)
    int_param = confint(Y.lm)[2,]
    D = data.frame(X, Y)
    bootstrap_stat = function(D, bootstrap_idx) {
       return(lm(Y ~ X, data=D[bootstrap_idx,])$coef)
    }
    boot_results = boot(D, bootstrap_stat, R=1000)

  boot_results
  beta1 <- boot_results$t0[2]

   pairs_SE <- sqrt(cov(boot_results$t)[2,2])
   Z <- (beta1 - b)/pairs_SE
   pvalue <- 2*pnorm(-abs(Z))
   count <- 0
   if (pvalue > 0.05) count <- 1
   int_pairs = c(coef(Y.lm)[2] - qnorm(0.975) * pairs_SE,
                 coef(Y.lm)[2] + qnorm(0.975) * pairs_SE)
   ci <- int_pairs[2] - int_pairs[1]
   return(cbind(count, Z, ci))
}
```

```
nsim = 100
coverage = c()
count = 0
for (i in 1:nsim) {

    coverage = rbind(coverage, simulate_correct2()[2])
    count = count + simulate_correct2()[1]
}
```

In this experiment, the value is

```
print(count/nsim)
```

```
## [1] 0.94
```

We test if Z-statistic is normally distributed:

mean and standard deviation of the Z-statistic:

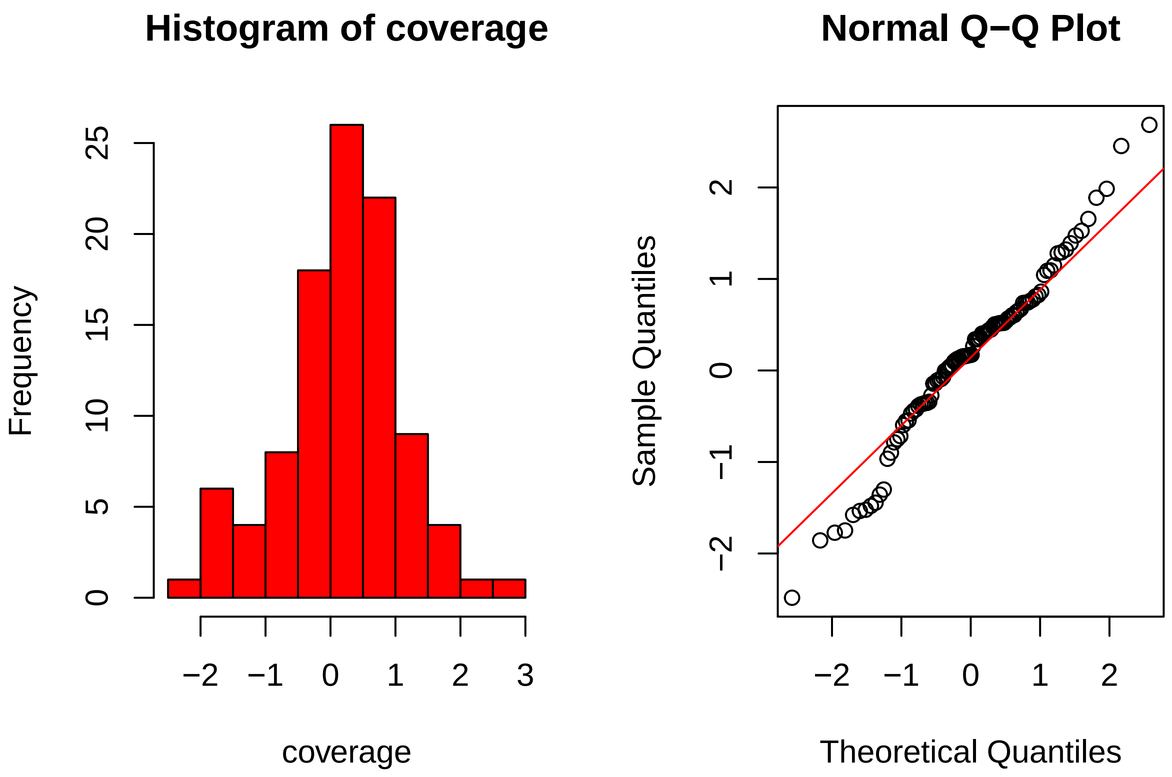
```
print(c(mean(coverage), sqrt(var(coverage))))
```

```
## [1] 0.1556767 0.9299060
```

Z-statistic seem to have close to a standard normal distribution.

Histogram, QQ-Plot

```
par(mfrow = c(1,2))
hist(coverage, col = "red")
qqnorm(coverage)
qqline(coverage, col = 2)
```



Shapiro test for normality

```r
shapiro.test(coverage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  coverage
## W = 0.97714, p-value = 0.07934
```