HW-4

STATS 500 - HOMEWORK 4

Vasudha Rohatgi

Code Published at https://github.com/Vasu2499/STATS-500-HW-4

PROBLEM 1

PART (a)

```
# install.packages("car")
# install.packages("lmtest")
library(car)
```

```
Loading required package: carData
```

```
library(lmtest)
```

```
Loading required package: zoo
```

```

Attaching package: 'zoo'
```

```
The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

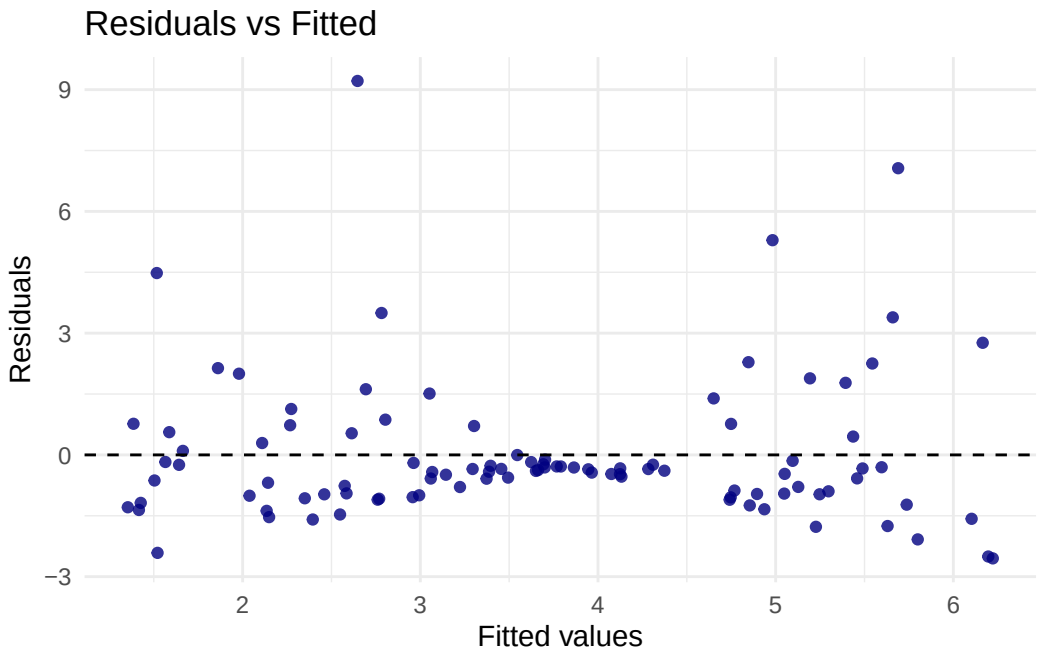
```
library(broom)
library(ggplot2)



## I created both vectors manually to avoid errors
## related to data loading/type mismatch

DATA_X_Y <- data.frame(
  x = c(0.172038167, 0.418466777, 0.054523322, 0.695505929, 0.980692884, 0.607070161,
        0.011947568, 0.435136626, 0.894557753, 0.83386622, 0.281672534, 0.33573479,
        0.045642636, 0.8848521, 0.96812355, 0.571297675, 0.716719676, 0.07434831,
        0.265655909, 0.170830833, 0.724962878, 0.421258879, 0.04217659, 0.619995055,
        0.765302866, 0.715242532, 0.784955135, 0.272284389, 0.026605156, 0.986976457,
        0.058915448, 0.559899994, 0.149852436, 0.372403622, 0.295098473, 0.197134535,
        0.695133297, 0.742518261, 0.825313237, 0.484748986, 0.675808315, 0.7916749,
        0.355265427, 0.044775931, 0.169300308, 0.795860181, 0.733278047, 0.569696626,
```

```
        0.342122064, 0.482784098, 0.257727252, 0.992103707, 0.353837485, 0.402662357,
        0.303802435, 0.25241684, 0.404298493, 0.221529326, 0.259792642, 0.234567108,
        0.878760028, 0.024527291, 0.497956215, 0.693973705, 0.502646435, 0.51743757,
        0.334757277, 0.018371782, 0.844523574, 0.356896266, 0.212493253, 0.296758583,
        0.474448235, 0.771659764, 0.872904745, 0.069980517, 0.16414597, 0.137929959,
        0.442823663, 0.53376598, 0.755756296, 0.855570187, 0.699383936, 0.907087586,
        0.805959126, 0.388349324, 0.195891048, 0.601747278, 0.469033639, 0.453257746,
        0.537691873, 0.299450343, 0.476787878, 0.569891095, 0.114090955, 0.756277627,
        0.484390114, 0.866153744, 0.838347831, 0.423012322),
  y = c(0.615463909, 2.786868902, 1.391114421, 5.513140781, 8.928013838, 4.068108402,
        0.063691378, 3.110613949, 4.511442248, 5.888070504, 4.311799621, 2.764826737,
        -0.892588409, 12.75371213, 4.528309356, 3.595944921, 3.609635666, 1.76275869,
        3.147503883, 1.456565682, 3.93219418, 2.972703801, 0.873412578, 3.984062378,
        4.949698937, 7.131532014, 7.079722233, 11.86116384, 0.244303005, 3.692677683,
        2.147189199, 3.605006326, 1.032537933, 2.655585392, 1.656532377, 3.404987867,
        3.701615913, 10.27326537, 7.169696402, 3.585939694, 6.043443262, 3.455010225,
        2.476542828, 5.997966752, 0.75690345, 4.276901278, 3.597403681, 3.648705215,
        1.998482562, 3.460836761, 1.81163011, 3.672050336, 4.563573561, 2.947486015,
        3.671598776, 1.079755993, 4.014472991, 0.803146673, 1.636552647, 1.488315055,
        9.048456133, 0.059128936, 3.483556904, 3.63638176, 3.505882982, 3.554139102,
        1.916388836, 2.152735695, 5.154818385, 2.645536336, 1.280925658, 1.688439264,
        3.256380613, 4.33833951, 3.877387878, 1.397218401, 2.402945759, 3.980154753,
        2.93424781, 3.59075253, 4.094248288, 7.796327544, 3.890117281, 3.718084517,
        4.400256847, 2.431933865, 2.998349391, 3.934755336, 3.446353977, 3.542439376,
        3.52940814, 6.278667985, 3.289264874, 3.793658899, 3.998930738, 4.583571992,
        3.392036397, 5.292731256, 4.880063631, 3.127386419)
)


# Fit linear model
model <- lm(y ~ x, data = DATA_X_Y)

# Summary of the model
summary(model)
```

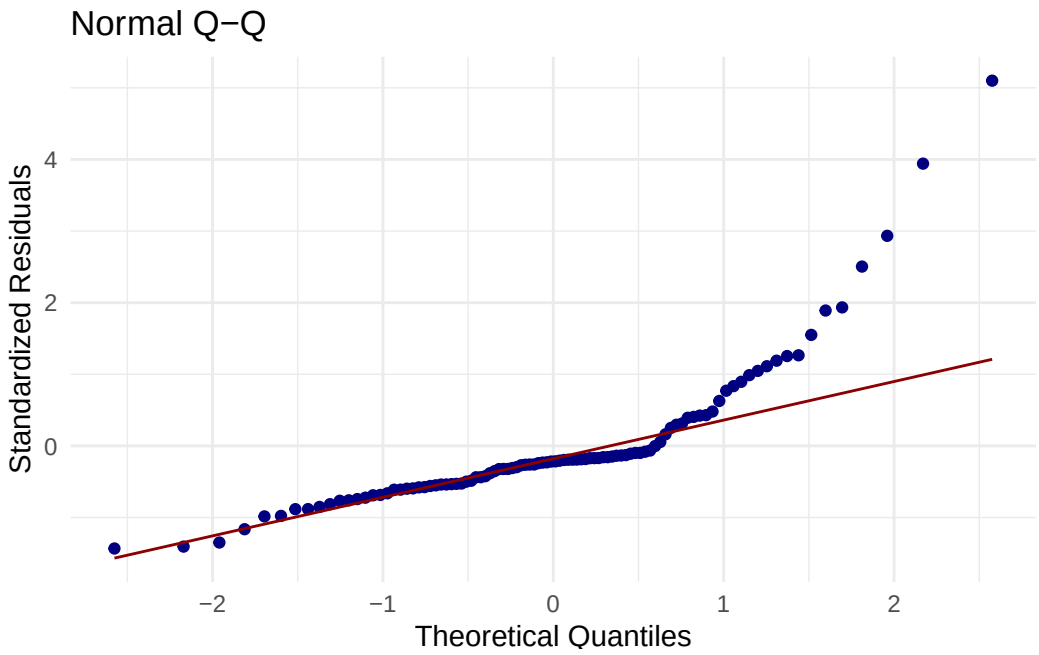
```
Call:
lm(formula = y ~ x, data = DATA_X_Y)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5499 -0.9772 -0.3922  0.3329  9.2143

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.2945     0.3627   3.569 0.000558 ***
```

```
x             4.9667     0.6543   7.591 1.88e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.821 on 98 degrees of freedom
Multiple R-squared:  0.3703,    Adjusted R-squared:  0.3638
F-statistic: 57.62 on 1 and 98 DF,  p-value: 1.875e-11
```

```r
# Diagnostic plots

diagnostics <- augment(model)

# Residuals vs. Fitted
ggplot(diagnostics, aes(.fitted, .resid)) +
  geom_point(color = "navy", alpha = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Residuals vs Fitted", x = "Fitted values",
       y = "Residuals") +
  theme_minimal()
```

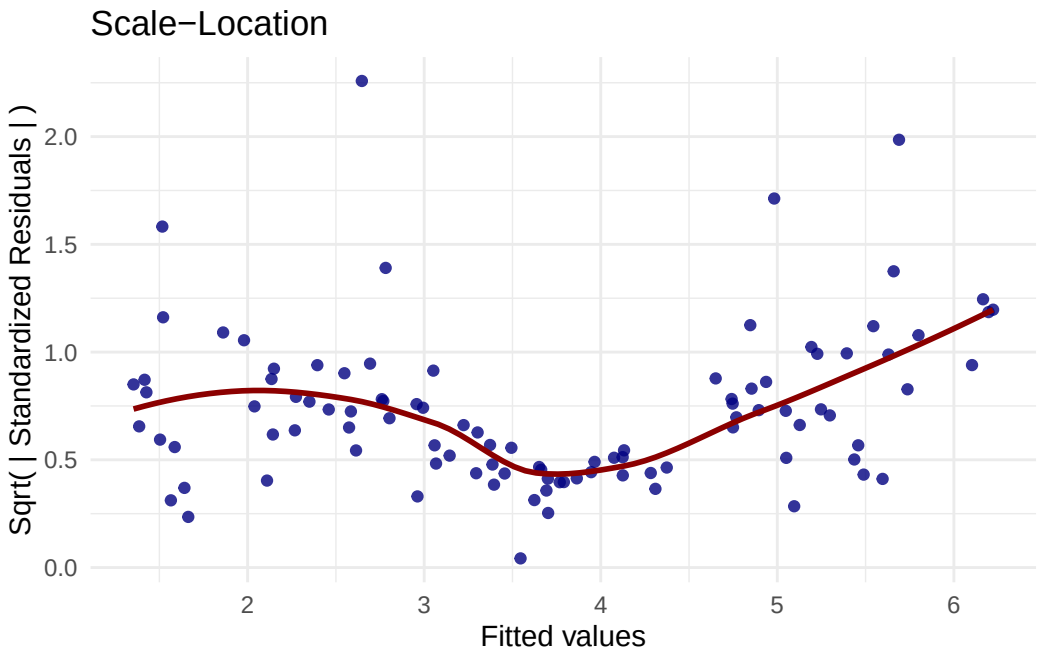



```r
# Normal Q-Q
ggplot(diagnostics, aes(sample = .std.resid)) +
  stat_qq(color = "navy") +
  stat_qq_line(color = "darkred") +
  labs(title = "Normal Q-Q", x = "Theoretical Quantiles",
       y = "Standardized Residuals") +
  theme_minimal()
```




```
# Scale-Location
ggplot(diagnostics, aes(.fitted, sqrt(abs(.std.resid)))) +
  geom_point(color = "navy", alpha = 0.8) +
  geom_smooth(se = FALSE, color = "darkred") +
  labs(title = "Scale-Location", x = "Fitted values",
       y = "Sqrt( | Standardized Residuals | )") +
  theme_minimal()
```

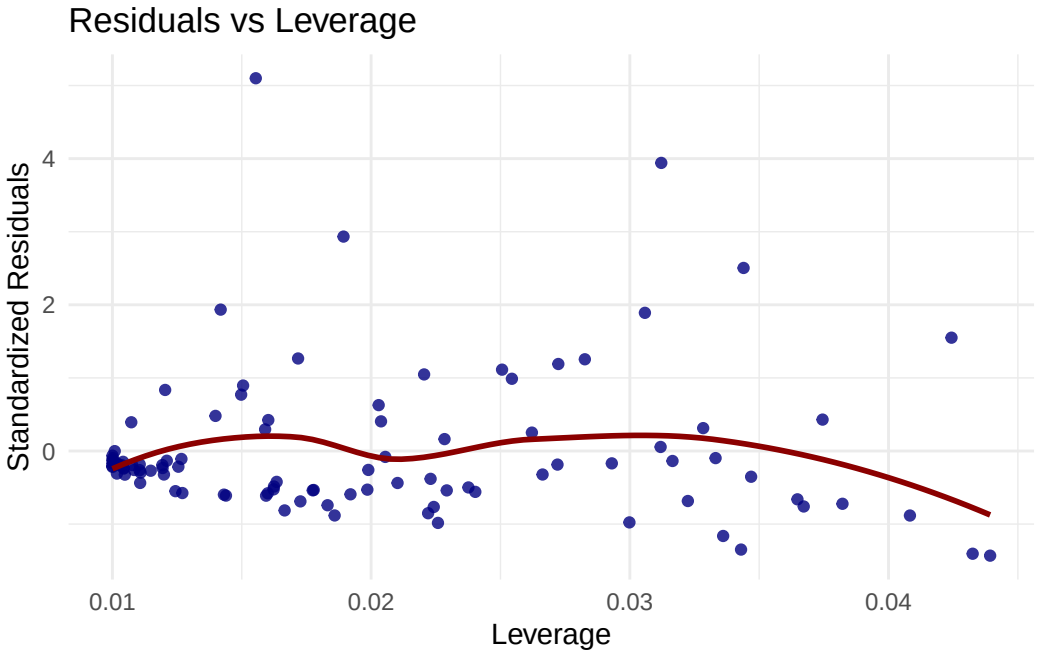
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
# Residuals vs. Leverage
ggplot(diagnostics, aes(.hat, .std.resid)) +
  geom_point(color = "navy", alpha = 0.8) +
  geom_smooth(se = FALSE, color = "darkred") +
```

```
  labs(title = "Residuals vs Leverage", x = "Leverage",
       y = "Standardized Residuals") +
  theme_minimal()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Some observations

- **Residuals vs. Fitted Plot**:
  - Mostly a straight line with slight deviations, indicating a mostly valid linear model with some mild non-linearity concerns.
- **Q-Q Plot**:
  - Resembles an exponential curve, suggesting non-normality of residuals, particularly right-skewed, which may affect inference.
- **Scale-Location Plot**:
  - Generally a straight horizontal line with a mild V-shape, indicating potential mild heteroskedasticity, which could lead to unreliable standard errors.
- **Residuals vs. Leverage Plot**:
  - A dense cluster of points along a line with many outliers, indicating that while most points fit well, the outliers could significantly influence the regression results.

The model shows potential issues with non-normality and heteroskedasticity, alongside influential outliers, which could undermine the reliability of coefficient estimates and inference.

PART (b) and PART(c)

80% Confidence Interval for the Slope Using Default Standard Error
(attached handwritten calculation)

NEXT SEGMENT OF PROBLEM - 1 : DATA GENERATION

```
set.seed(123)
n <- 100
B <- 10000
x <- DATA_X_Y$x
results <- numeric(B)

for (b in 1:B) {
  epsilon <- rnorm(n, mean = 0, sd = sqrt(0.75))
  y <- 1 + 5 * x + epsilon
  model <- lm(y ~ x)
  results[b] <- coef(model)[2]
}

# Calculating the coverage of the 80% CI
lower_bound <- quantile(results, 0.1)
upper_bound <- quantile(results, 0.9)
coverage <- mean((lower_bound <= 5) & (upper_bound >= 5))
print(coverage)
```

```
[1] 1
```

I WANT TO VISUALIZE THE POTENTIAL EFFECTS OF VARYING A FEW OF THE SIMULATION PARAMETERS BEFORE PROCEEDING TO PART (D)

- **Checking the effect of varying the distribution of errors**

```
set.seed(123)
n <- 100
B <- 10000
x <- DATA_X_Y$x
results_log_normal <- numeric(B)

for (b in 1:B) {
  epsilon <- rlnorm(n, meanlog = 0, sdlog = 0.5) - 1
  y <- 1 + 5 * x + epsilon
  model <- lm(y ~ x)
  results_log_normal[b] <- coef(model)[2]
}

lower_bound_log_normal <- quantile(results_log_normal, 0.1)
upper_bound_log_normal <- quantile(results_log_normal, 0.9)
coverage_log_normal <- mean((lower_bound_log_normal <= 5) &
                            (upper_bound_log_normal >= 5))

print(c(lower_bound_log_normal, upper_bound_log_normal, coverage_log_normal))
```

```
      10%       90%
```

```
4.728939 5.284067 1.000000
```

- **Checking the effect of change in sample size**

```r
sample_sizes <- c(50, 100, 200) # Different sample sizes
coverage_results <- numeric(length(sample_sizes))

for (j in 1:length(sample_sizes)) {
  n <- sample_sizes[j]
  results <- numeric(B)

  for (b in 1:B) {
    epsilon <- rnorm(n, mean = 0, sd = sqrt(0.75))
    y <- 1 + 5 * x[1:n] + epsilon # Ensure x matches the sample size
    model <- lm(y ~ x[1:n])
    results[b] <- coef(model)[2]
  }

  lower_bound <- quantile(results, 0.1)
  upper_bound <- quantile(results, 0.9)
  coverage_results[j] <- mean((lower_bound <= 5) & (upper_bound >= 5))
}

# Output
data.frame(Sample_Size = sample_sizes, Coverage = coverage_results)
```

```
  Sample_Size Coverage
1          50        1
2         100        1
3         200        1
```

- **Checking the impact of outliers**

```r
set.seed(123) # For reproducibility
n <- 100
B <- 10000
x <- DATA_X_Y$x
results_with_outliers <- numeric(B)

for (b in 1:B) {
  epsilon <- rnorm(n, mean = 0, sd = sqrt(0.75))
  y <- 1 + 5 * x + epsilon

  # Introduce outliers
  if (b %% 100 == 0) {
    y[c(1, n)] <- y[c(1, n)] + c(15, -15)
  }

  model <- lm(y ~ x)
```

```
  results_with_outliers[b] <- coef(model)[2]
}

lower_bound_outliers <- quantile(results_with_outliers, 0.1)
upper_bound_outliers <- quantile(results_with_outliers, 0.9)
coverage_outliers <- mean((lower_bound_outliers <= 5) &
                            (upper_bound_outliers >= 5))

# Output
print(c(lower_bound_outliers, upper_bound_outliers, coverage_outliers))
```

```
     10%      90%
4.603758 5.406768 1.000000
```

Attached handwritten Analysis of above visualizations

Simulation-1 and interpretation of results (PART(d) - PART(h))

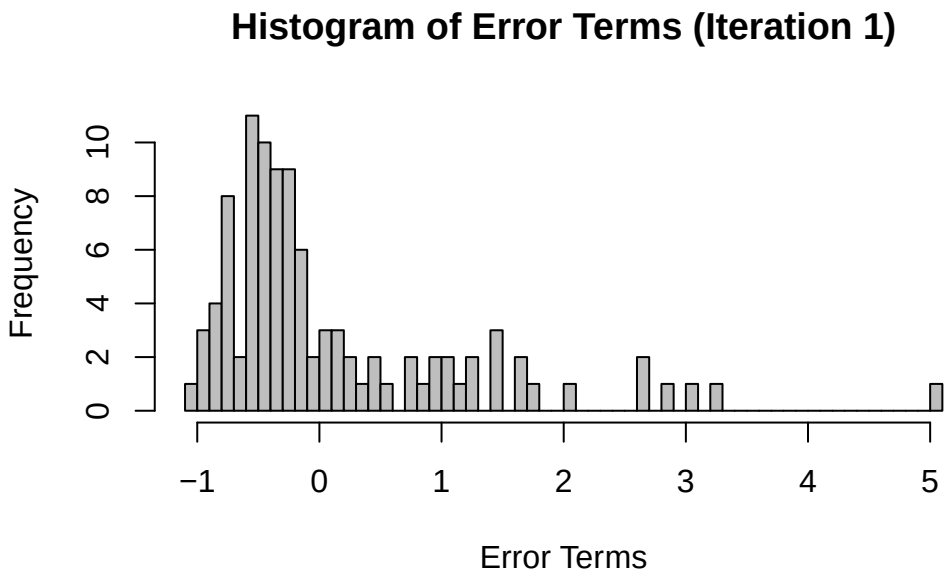
```
library(sandwich)
#read in the data
dat = read.csv("xy.csv")
x = dat$x
y = dat$y
n = length(x)

###############
#Population slope and intercept for simulations
#############

beta0 = 1
beta1 = 5

##########
#Simulation for d-h
#########

sigma.error = sqrt(.75)
SD.b1.skew = sigma.error/(sqrt(n-1)*sd(x))

nsim = 10000
b1.skew = rep(0, nsim)
se.skew= rep(0, nsim)
se.skew.hc = rep(0, nsim)
Epsilon.skew = matrix(0, n, nsim)

for(i in 1:nsim)
{
```

```
  errors = exp(rnorm(n, 0, sqrt(log(1.5)))) - exp(log(1.5)/2)
  Y = beta0 + beta1*x + errors
  lm.temp = lm(Y~x)
  b1.skew[i] = lm.temp$coef[2]
  se.skew[i] = summary(lm.temp)$coef[2,2]
  se.skew.hc[i] = sqrt(diag(vcovHC(lm.temp, type = "HC2")))[2]
  Epsilon.skew[,i] = errors
}
```

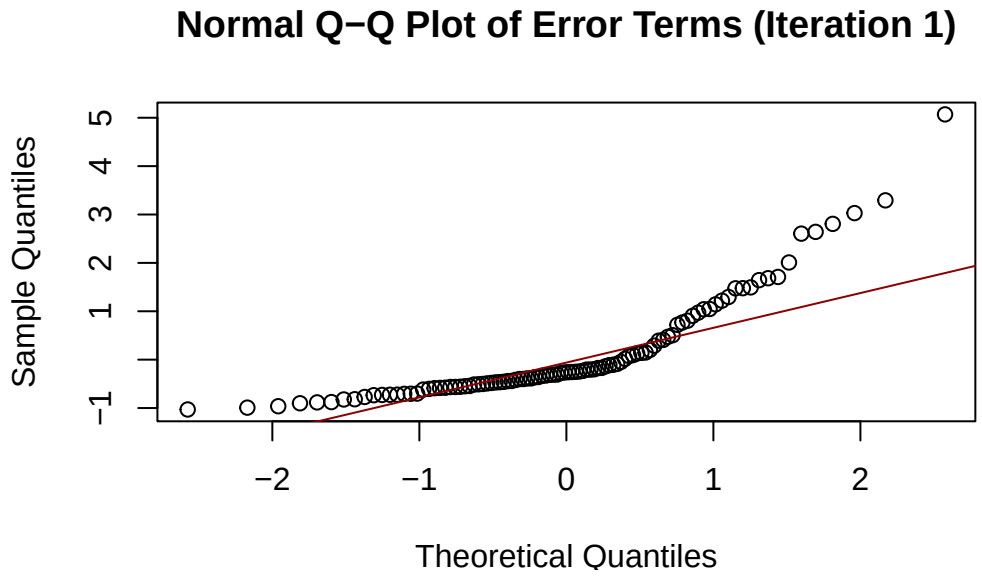
```
sigma.error = sqrt(.75)
SD.b1.skew = sigma.error/(sqrt(n-1)*sd(x))

nsim = 10000
b1.skew = rep(0, nsim)
se.skew= rep(0, nsim)
se.skew.hc = rep(0, nsim)
Epsilon.skew = matrix(0, n, nsim)

for(i in 1:nsim)
{
  errors = exp(rnorm(n, 0, sqrt(log(1.5)))) - exp(log(1.5)/2)
  Y = beta0 + beta1*x + errors
  lm.temp = lm(Y~x)
  b1.skew[i] = lm.temp$coef[2]
  se.skew[i] = summary(lm.temp)$coef[2,2]
  se.skew.hc[i] = sqrt(diag(vcovHC(lm.temp, type = "HC2")))[2]
  Epsilon.skew[,i] = errors
}
```

PART(d)

```
# Histogram of error terms from the first iteration

hist(Epsilon.skew[, 1],
     main = "Histogram of Error Terms (Iteration 1)",
     xlab = "Error Terms", breaks = 50, col = "grey")
```

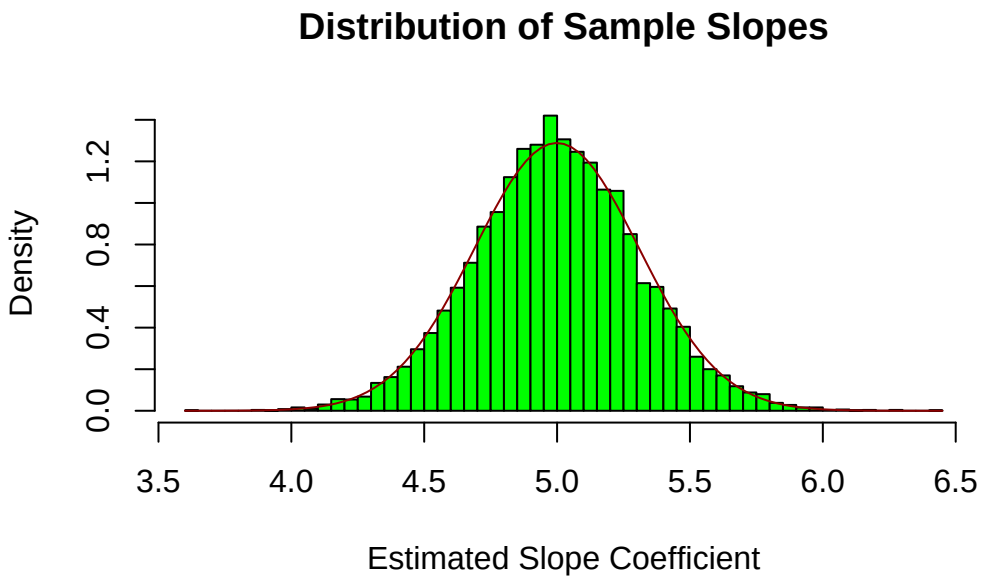



```
# Normal Quantile Plot

qqnorm(Epsilon.skew[, 1],
       main = "Normal Q-Q Plot of Error Terms (Iteration 1)")
qqline(Epsilon.skew[, 1], col = "darkred")
```



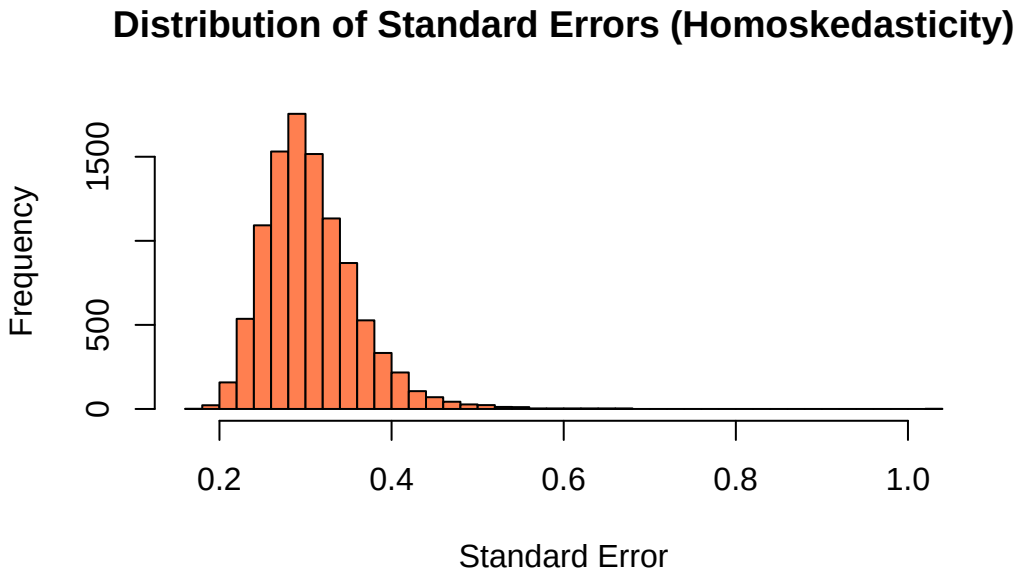
PART(e)

```
# Histogram of estimated slopes
hist(b1.skew, probability = TRUE,
     main = "Distribution of Sample Slopes",
     xlab = "Estimated Slope Coefficient", col = "green", breaks = 55)

# Overlay normal density
curve(dnorm(x, mean = mean(b1.skew), sd = sd(b1.skew)), add = TRUE, col = "darkred")
```

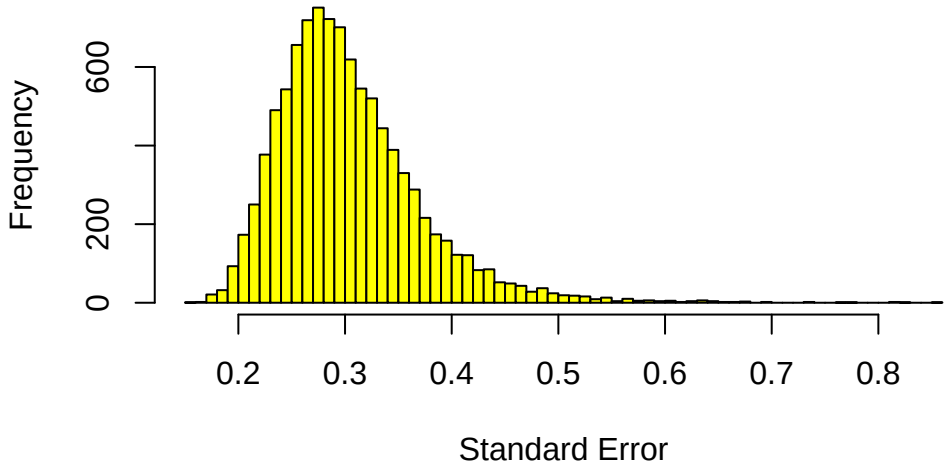


PART(f)

```
# Histogram for se.skew
hist(se.skew,
     main = "Distribution of Standard Errors (Homoskedasticity)",
     xlab = "Standard Error", col = "coral", breaks = 60)
```




```
# Histogram for se.skew.hc
hist(se.skew.hc,
     main = "Distribution of Standard Errors (Heteroskedasticity Consistent)",
     xlab = "Standard Error", col = "yellow", breaks = 50)
```




```
# Check means
mean_se_skew = mean(se.skew)
mean_se_skew_hc = mean(se.skew.hc)

cat("Mean of se.skew:", mean_se_skew, "\n")
```

```
Mean of se.skew: 0.3064303
```

```
cat("Mean of se.skew.hc:", mean_se_skew_hc, "\n")
```

```
Mean of se.skew.hc: 0.303425
```

PART(g) - Confidence Intervals

```
true_slope = 5

# Confidence intervals using se.skew
lower_bound_hom = b1.skew - qt(0.975, df = n - 2) * se.skew
upper_bound_hom = b1.skew + qt(0.975, df = n - 2) * se.skew

# Confidence intervals using se.skew.hc
lower_bound_het = b1.skew - qt(0.975, df = n - 2) * se.skew.hc
upper_bound_het = b1.skew + qt(0.975, df = n - 2) * se.skew.hc
```

PART(h) - Coverage calculation

```
coverage_hom = mean(lower_bound_hom <= true_slope &
                      upper_bound_hom >= true_slope)
coverage_het = mean(lower_bound_het <= true_slope &
                      upper_bound_het >= true_slope)

cat("Coverage using se.skew:", coverage_hom, "\n")
```

```
Coverage using se.skew: 0.9508
```

```
cat("Coverage using se.skew.hc:", coverage_het, "\n")
```

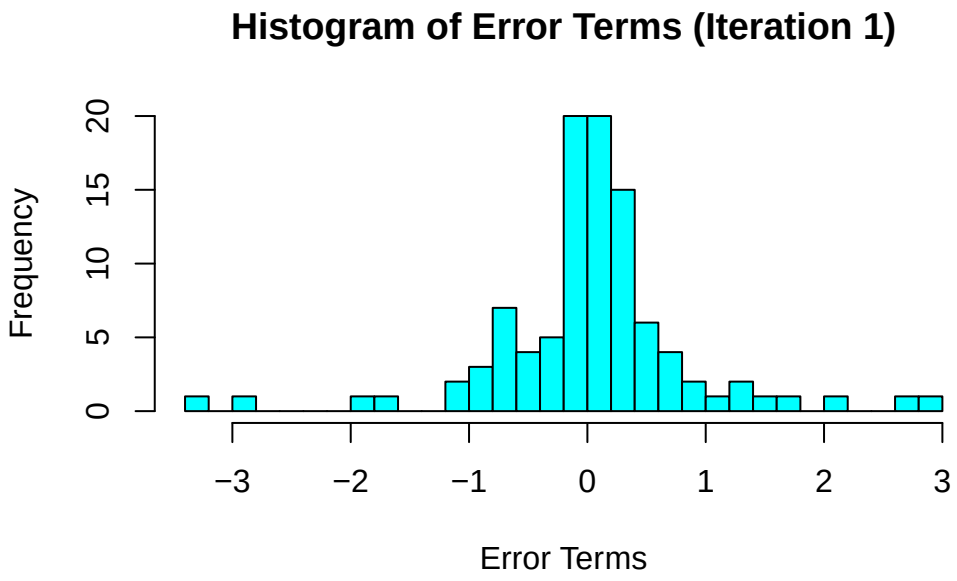
```
Coverage using se.skew.hc: 0.952
```

Simulation-2 and interpretation of results

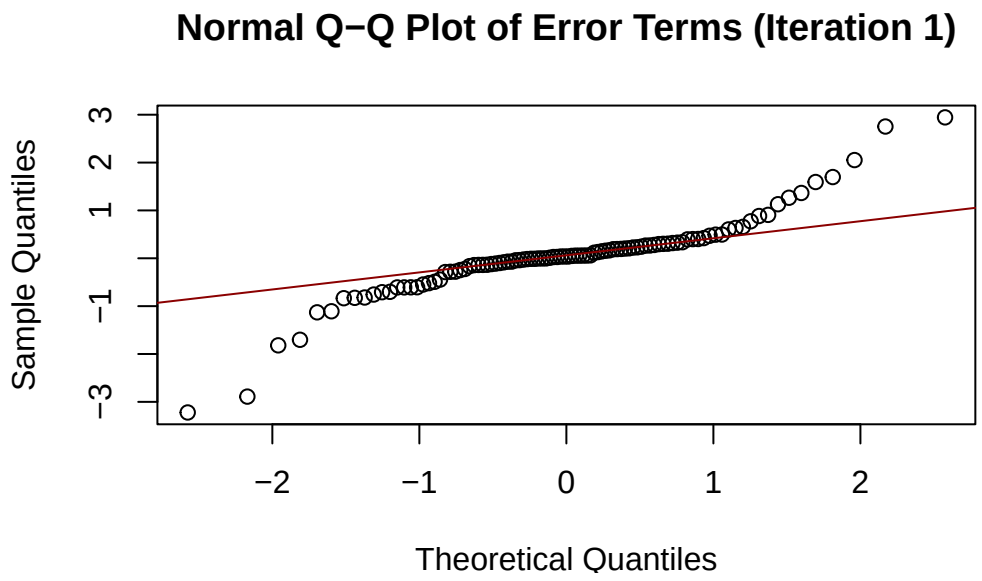
```
Sigma = diag((3.6*sqrt(.75)*abs(x - 1/2)))^2
Xmat = cbind(rep(1,n), x)
SD.b1.het = sqrt((solve(t(Xmat)%*%Xmat)%*%t(Xmat)%*%
                    Sigma%*%Xmat%*%solve(t(Xmat)%*%Xmat))[2,2])

nsim = 10000
b1.het = rep(0, nsim)
se.het = rep(0, nsim)
se.het.hc = rep(0, nsim)
Epsilon.het = matrix(0, n, nsim)
for(i in 1:nsim)
{
  errors = rnorm(n, 0, 3.6*sqrt(.75)*abs(x - 1/2))
  Y = beta0 + beta1*x + errors
  lm.temp = lm(Y~x)
  b1.het[i] = lm.temp$coef[2]
  se.het[i] = summary(lm.temp)$coef[2,2]
  se.het.hc[i] = sqrt(diag(vcovHC(lm.temp, type = "HC2")))[2]
  Epsilon.het[,i] = errors
}
```

Some Initial Visualizations & Histograms

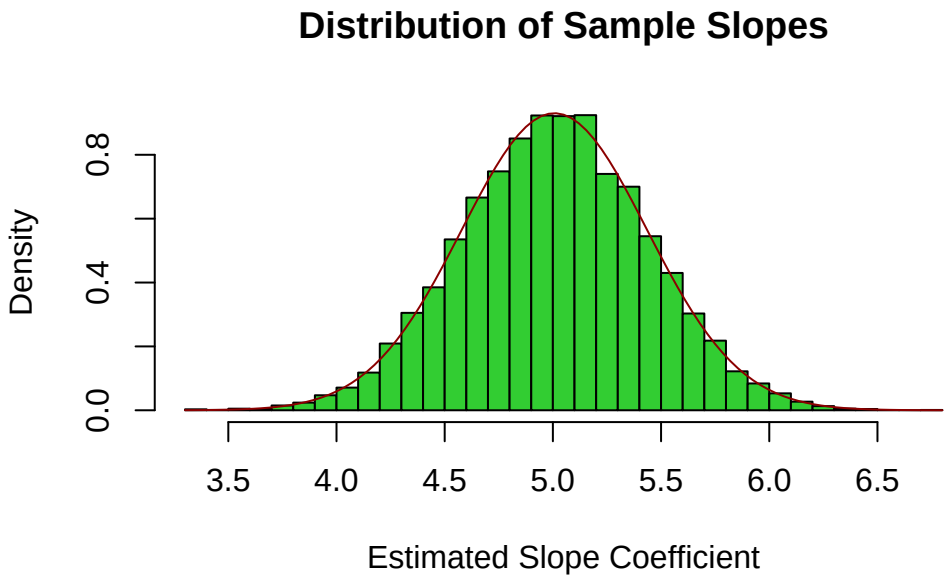
```
hist(Epsilon.het[, 1], main = "Histogram of Error Terms (Iteration 1)",
     xlab = "Error Terms", breaks = 40, col = "cyan")
```



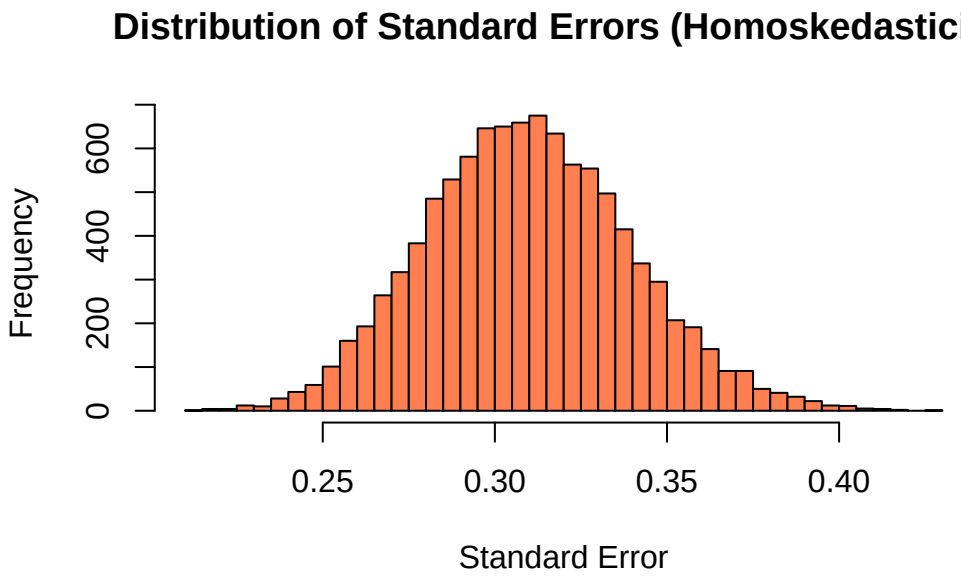
```r
# Normal Quantile Plot
qqnorm(Epsilon.het[, 1], main = "Normal Q-Q Plot of Error Terms (Iteration 1)")
qqline(Epsilon.het[, 1], col = "darkred")
```



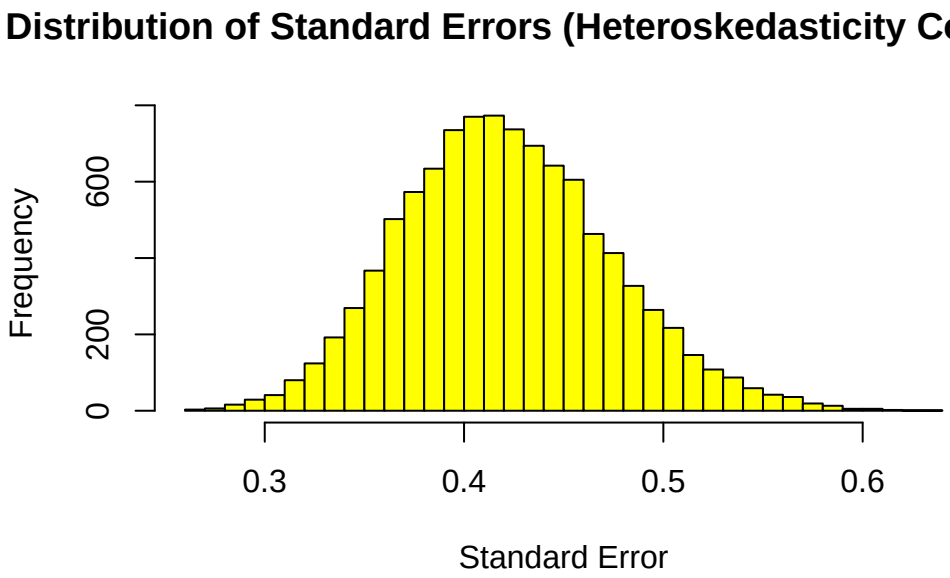

```r
# Histogram of estimated slopes
hist(b1.het, probability = TRUE,
     main = "Distribution of Sample Slopes",
     xlab = "Estimated Slope Coefficient", col = "limegreen", breaks = 40)

# Overlay normal density
curve(dnorm(x, mean = mean(b1.het), sd = sd(b1.het)), add = TRUE, col = "darkred")
```




```r
# Histogram for se.het
hist(se.het,
     main = "Distribution of Standard Errors (Homoskedasticity)",
     xlab = "Standard Error", col = "coral", breaks = 40)
```




```
# Histogram for se.het.hc
hist(se.het.hc,
     main = "Distribution of Standard Errors (Heteroskedasticity Consistent)",
     xlab = "Standard Error", col = "yellow", breaks = 40)
```




```
mean_se_het = mean(se.het)
mean_se_het_hc = mean(se.het.hc)

cat("Mean of se.het:", mean_se_het, "\n")
```

```
Mean of se.het: 0.3097563
```

```
cat("Mean of se.het.hc:", mean_se_het_hc, "\n")
```

```
Mean of se.het.hc: 0.4213828
```

```
# True population slope
true_slope = beta1
```

```r
# Confidence intervals using se.het
lower_bound_hom_het = b1.het - qt(0.975, df = n - 2) * se.het
upper_bound_hom_het = b1.het + qt(0.975, df = n - 2) * se.het

# Confidence intervals using se.het.hc
lower_bound_het_hc = b1.het - qt(0.975, df = n - 2) * se.het.hc
upper_bound_het_hc = b1.het + qt(0.975, df = n - 2) * se.het.hc

# Coverage calculation
coverage_hom_het = mean(lower_bound_hom_het <= true_slope &
                          upper_bound_hom_het >= true_slope)
coverage_het_hc = mean(lower_bound_het_hc <= true_slope &
                         upper_bound_het_hc >= true_slope)

cat("Coverage using se.het:", coverage_hom_het, "\n")
```

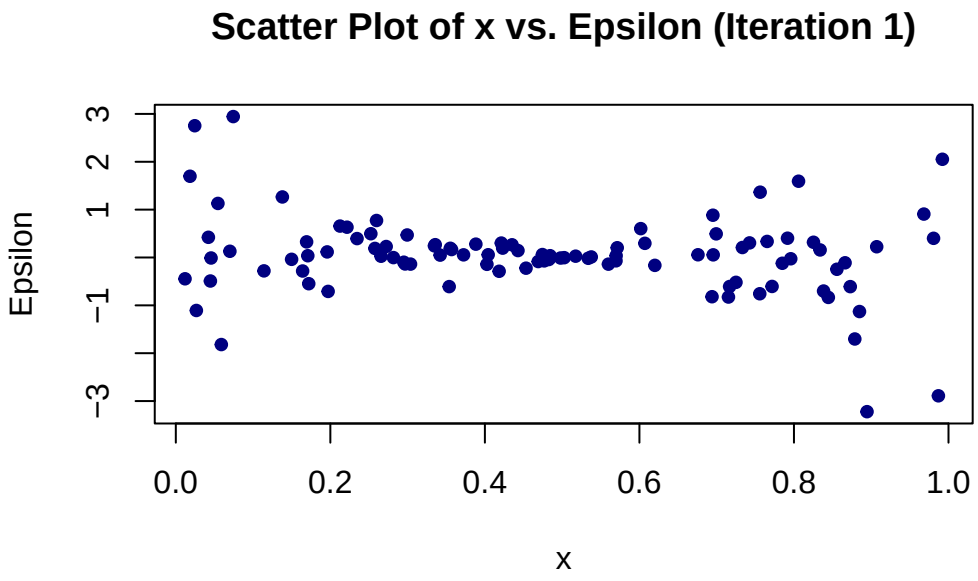
```
Coverage using se.het: 0.845
```

```r
cat("Coverage using se.het.hc:", coverage_het_hc, "\n")
```

```
Coverage using se.het.hc: 0.9438
```

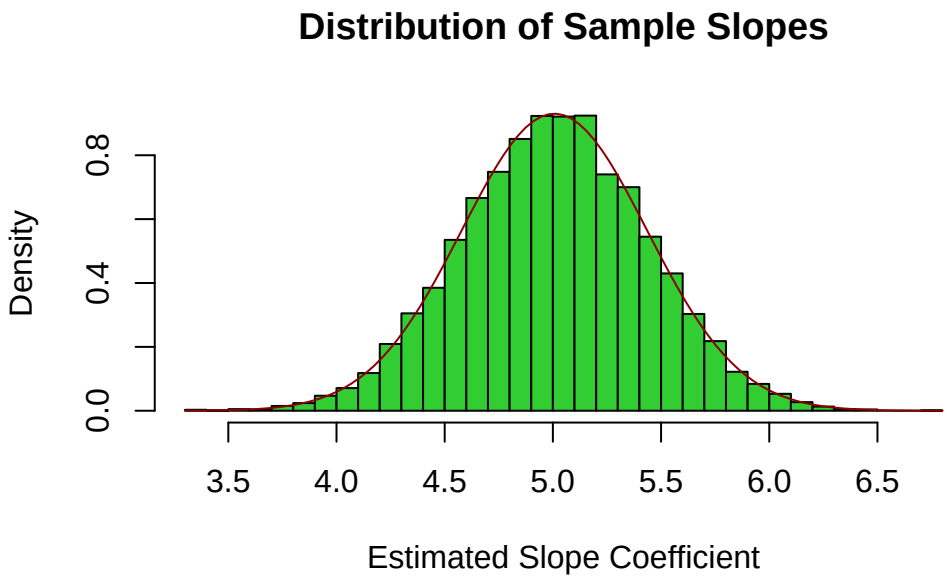
PART(i)

```r
plot(x, Epsilon.het[, 1], main = "Scatter Plot of x vs. Epsilon (Iteration 1)",
     xlab = "x", ylab = "Epsilon", col = "navy", pch = 19, cex=0.8)
```



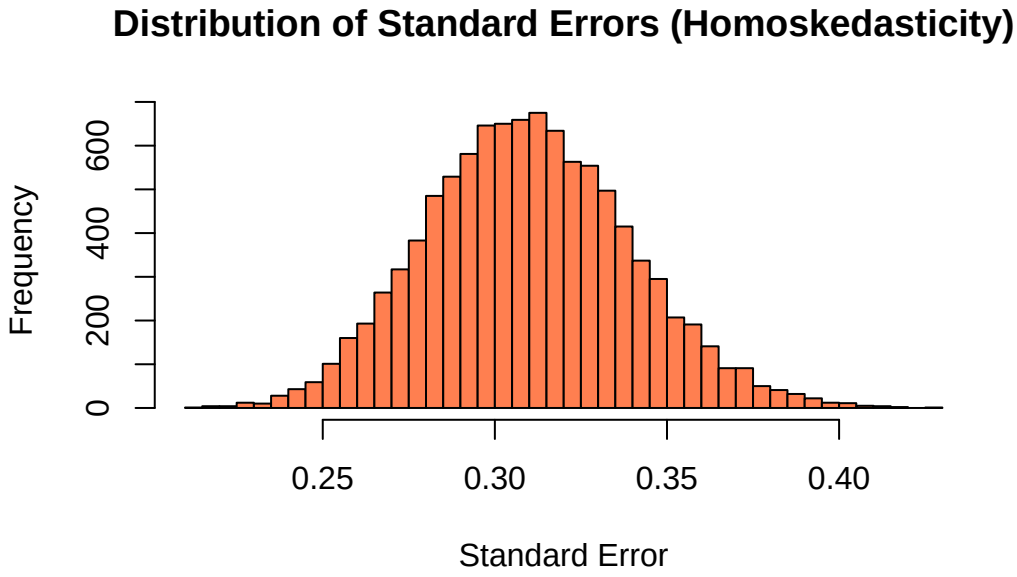
PART(j)

```r
hist(b1.het, probability = TRUE, main = "Distribution of Sample Slopes",
     xlab = "Estimated Slope Coefficient", col = "limegreen", breaks = 30)

curve(dnorm(x, mean = mean(b1.het), sd = sd(b1.het)), add = TRUE, col = "darkred")
```

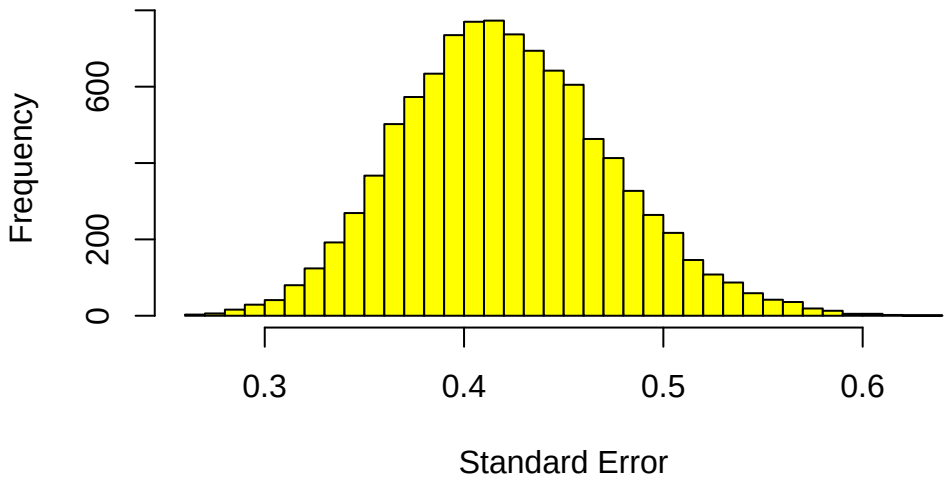


PART(k)

```
hist(se.het,
     main = "Distribution of Standard Errors (Homoskedasticity)",
     xlab = "Standard Error", col = "coral", breaks = 40)
```




```
hist(se.het.hc,
     main = "Distribution of Standard Errors (Heteroskedasticity Consistent)",
     xlab = "Standard Error", col = "yellow", breaks = 40)
```




```
mean_se_het = mean(se.het)
mean_se_het_hc = mean(se.het.hc)

cat("Mean of se.het:", mean_se_het, "\n")
```

```
Mean of se.het: 0.3097563
```

```
cat("Mean of se.het.hc:", mean_se_het_hc, "\n")
```

```
Mean of se.het.hc: 0.4213828
```

```
cat("True SD of beta1:", 0.427, "\n")
```

```
True SD of beta1: 0.427
```

PART(l)

```
true_slope = 5
lower_bound_hom_het = b1.het - qt(0.975, df = n - 2) * se.het
upper_bound_hom_het = b1.het + qt(0.975, df = n - 2) * se.het
lower_bound_het_hc = b1.het - qt(0.975, df = n - 2) * se.het.hc
upper_bound_het_hc = b1.het + qt(0.975, df = n - 2) * se.het.hc
```

PART(m)

```
coverage_hom_het = mean(lower_bound_hom_het <= true_slope &
                          upper_bound_hom_het >= true_slope)
coverage_het_hc = mean(lower_bound_het_hc <= true_slope &
                         upper_bound_het_hc >= true_slope)

cat("Coverage using se.het:", coverage_hom_het, "\n")
```

```
Coverage using se.het: 0.845
```

```r
cat("Coverage using se.het.hc:", coverage_het_hc, "\n")
```

```
Coverage using se.het.hc: 0.9438
```

Results from all of the above code is discussed in attached handwritten note.