Investigating How Sample Size and Simulation Runs Influence the Estimate of Model Parameters in Simple and Multiple Linear Regression

Faria Khandaker

5/30/2021

Linear Regression is a commonly used tool in both statistics and machine learning. In both disciplines it is used to find the relationship between datapoints after which, predictions can be made,based on the relationships(s) found. In this report we simulate our own data points with preset values of sigma(the population mean), different Betas (representing different characteristics of the population), error variances and correlation coefficients and we run both simple and multiple linear regression over the data points and calculate the estimated values of the betas in an effort to measure how close or far away the estimated values are from the true (predefined) values. Task 1 of this report covers the implementation of simple linear regression and task 2 covers the implementation of multiple linear regression and compares the results to the results of simple linear regression. The outputs and explanation of results are shown in the main part of the report. Some concluding points can be found at the end and code to produce the results can be found in the appendix.

Task 1

Part A: Simulating parameters using code

```
## Simulation
set.seed(1000557774)
beta0<-rnorm(1,mean=0, sd=1) #the population beta0
beta1<-runif(n=1, min=1, max=3) #the population beta1
sig2<- rchisq(n=1,df=25) ## the error variance sigma squared

## Multiple simulation may require loops
nsample <- 5 # sample size
n.sim<- 100 #number of simulations
sigX<- 0.2 # The variances of X

#Simulate the predictor variable
X<- rnorm(nsample,mean=0, sd=sqrt(sigX))
```

Part B:

- Generate epsilon and Y.
- Execute 100 simulations and estimate regression coefficients for each simulation.

- Calculate mean of the estimates from the different simulations.
- Comment on observations

```
## [1] "Estimated Mean of beta0 and beta1 respectively"
```

```
## [1] -0.4920835  2.0287829
```

```
## [1] "True Beta0"
```

```
## [1] -0.6485048
```

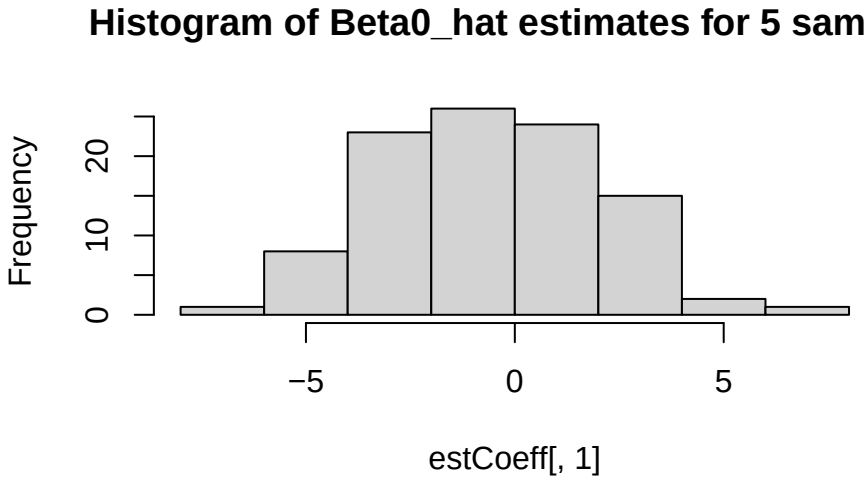
```
## [1] "True Beta1"
```

```
## [1] 2.285801
```

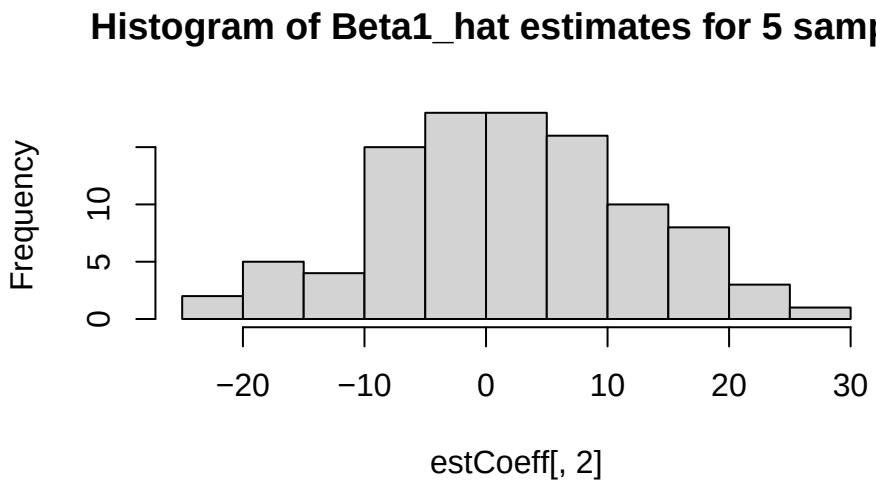
The estimated means from the 100 different simulation is off by approximately 0.2 for both beta0 and beta1, when compared to their true values. In terms of scale, it seems the estimate of beta1 is fairly close to it's true population coefficient. This does not seem to be the case for Beta0.

Part C:

- Plot the histogram of each of the regression parameter estimates.
- Explain the pattern of the distribution.



Estimates of the Beta0s from the 100 simulations appear to be centered but slightly right skewed. This is also inline with mean estimated obtained for beta0_hat as the estimate was slightly higher than the true value. There doesn't appear to be a wide spreading of the data points



The beta1 estimates appear to be right skewed or positively skewed, which fits with the mean estimate obtained as it was higher than the true value of beta1.

Part D:

- Obtain the variance of the regression parameter estimator (i.e Beta0 and Beta1) from the simulations (calculate the sample variances of the regression parameter estimates from the 100 simulations.)
- Is this variance approximately equal to the true variances of the regression parameter estimates? Explain

```
## [1] "True Variance of Beta0"

## [1] 6.38974

## [1] "Estimated Variance of Beta0"

## [1] 6.600514

## [1] "True Variance of Beta1"

## [1] 95.47248

## [1] "Estimated Variance of Beta1"

## [1] 108.1908
```

From the results presented above, it is apparent that the true variances and the estimated variances of both beta0 and beta1 are very close to each other. The estimated variance for beta0 is also higher than the true variance beta0, by approximately 0.2 points. The estimated variance of beta1 is 13 points higher than the true variance. The results here also match with those depicted in the histograms from part C as the histogram for beta1 estimates has more variation (as depicted by the higher number of bins) compared to the histogram for beta0 estimate. Increasing the sample size from 5 to a higher number can be useful in obtaining estimate which are closer to the true values.

Part E:

- Construct the 95% t and z confidence intervals for Beta0 and Beta1 during every simulation.
- What is the proportion of the intervals for each method containing the true value of the parameters?
- Is this consistent with the definition of confidence interval?
- What differences do you observe in the t and z confidence intervals?
- Does it help to increase the number of simulations from 100?

Confidence Intervals are a range of values you expect your estimated sample to fall within a certain percentage of the time, if you run your experiments again and again or sample the population in the same way. For a 95% confidence interval, for example, we expect that 95% out of 100 times, our estimated values will fall within a certain upper and lower limit. Here we try to see what propotions of our estimated values actually fall within the range we calculate. It was found that increasing simulations does help to increase the proportion of betas found within the confidence intervals. Explanations for specific simulation numbers can be found below.

Since simulated data is used for these experiments, we know the value of the population mean, sigma. Therefore, in this step, we calculate both t and z confidence intervals. The alpha value is set to 0.5 as we are calculating the 95% confidence intervals.

```
## [1] "True Value of Beta0"
```

```
## [1] -0.6485048
```

```
## [1] "Number of simulations out of 100 that beta0 fell into t-confidence intervals"
```

```
## B0ininterval
##           55
```

```
## [1] "10 rows from the B0 interval table"
```

```
##     upperlimit lowerlimit B0ininterval
## V1   0.0568175 -3.2480800            1
## V2   0.2880289 -4.5622493            1
## V3   2.2540566 -1.4368785            1
## V4   3.2200127 -0.9482803            1
```

```
## V5  -4.9162429 -6.0204449            0
## V6   4.5110889  0.6553748            0
## V7   2.6138999 -1.8077135            1
## V8   4.2371778  0.2825710            0
## V9   3.8757869 -0.9314280            1
## V10  1.4002478 -1.1966150            1
```

```
## [1] "True Value of Beta1"
```

```
## [1] 2.285801
```

```
## [1] "Number of simulations out of 100 that beta1 fell into t-confidence intervals"
```

```
## B1ininterval
##           47
```

```
## [1] "10 rows from the B1 interval table"
```

```
##     upperlimit  lowerlimit B1ininterval
## V1   26.026715 13.25187524            0
## V2   18.044223 -0.70417328            1
## V3   22.531263  8.26422244            0
## V4   13.913417 -2.19881607            1
## V5    6.220946  1.95273385            1
## V6   14.970204  0.06622259            1
## V7   13.761890 -3.32953435            1
## V8   23.275582  7.98933828            0
## V9   30.777163 12.19522398            0
## V10   1.353045 -8.68493839            0
```

There is information about both beta0 and beta1 in the data presented above. In order to calculate the percentage of times the true beta0 and beta1 values fell into the calcululated confidence intervals, I saved the upper and lower bounds to a dataframe and wrote a function to input dummy variables 1 if the true values fell within the confidence interval and 0 if they didnt. The first 10 rows for t-confidence interval for both beta0 and beta1 are shown, along with the number of times the true values fell within the upper and lower limits. The results above show that for 100 simulation runs, the true value of beta0 was contained within the calculated confidence intervals around 55% of the time. For beta1, that number is a little lower, with 47% of the time. This was the result of the t-confidence interval. below is the result for the Z-confidence interval.

```
## [1] "True Value of Beta0"
```

```
## [1] -0.6485048
```

```
## [1] "Number of simulations out of 100 that beta0 fell into z-confidence intervals"
```

```
## B0ininterval
##          100
```

```
## [1] "10 rows from the B0 interval table"
```

```
##     upperlimit lowerlimit B0ininterval
## V1   1.4807554 -1.9291863            1
## V2  -2.4036252 -5.8135669            1
## V3   0.2515392 -3.1584026            1
## V4  -0.9796389 -4.3895807            1
## V5   0.5058688 -2.9040730            1
## V6   3.8674403  0.4574985            1
## V7   3.1502448 -0.2596970            1
## V8   0.3582486 -3.0516932            1
## V9   2.6814771 -0.7284647            1
## V10  5.1329829  1.7230411            1
```

```
## [1] "True Value of Beta1"
```

```
## [1] 2.285801
```

```
## [1] "Number of simulations out of 100 that beta1 fell into z-confidence intervals"
```

```
## B1ininterval
##           47
```

```
## [1] "10 rows from the B1 interval table"
```

```
##     upperlimit  lowerlimit B1ininterval
## V1    7.008615  -6.1722658            1
## V2  -14.116271 -27.2971525            0
## V3   14.536528   1.3556473            1
## V4   12.040855  -1.1400261            1
## V5    5.762334  -7.4185474            1
## V6   13.770139   0.5892578            1
## V7   16.509876   3.3289952            0
## V8   -7.239714 -20.4205951            0
## V9    1.283618 -11.8972630            0
## V10   8.435854  -4.7450267            1
```

Similar to the output for the t-confidence interval, the output for the Z-confidence interval also contains the first 10 rows from tables which show the upper and lower limits of the intervals as well as dummy variables indicating 1 if the true value is found within those limit and 0 otherwise. This is shown for both beta0 and beta1.

According to the results above, for the intervals for beta0, the true beta0 value falls within the calculated intervals 100% of the time. For beta1, the percentage stays constant at 47% (as was seen for the t-intervals). The results for the beta1 intervals is very odd as the true value of sigma is used in the calculation of the upper and lower bounds so the percentage of times the true beta1 falls within the upper and lower bounds

should be higher than that of the t-confidence intervals since the estimated sample mean is used in that calculation.

With a sample number of 5 and 100 simulations, the Z interval is not very accurate. The results of beta0 make sense as the true population mean is being used to calculate the intervals and the true value of beta0, which is also from the population, is known.

Now to test the effect an increase in the number of simulations would have on the results, I ran the code two different times with a simulation of 1000 and 10,000 below. The sample number was kept constant.

```
## [1] "True Value of Beta0"
```

```
## [1] -0.6485048
```

```
## [1] "Number of simulations out of 1000 that beta0 fell into t-confidence interval"
```

```
## B0ininterval
##          477
```

```
## [1] "Percentage beta0 fell into t-confidence intervals"
```

```
## B0ininterval
##         47.7
```

```
## [1] "10 rows from the B0 interval table"
```

```
##     upperlimit lowerlimit B0ininterval
## V1   0.0568175 -3.2480800            1
## V2   0.2880289 -4.5622493            1
## V3   2.2540566 -1.4368785            1
## V4   3.2200127 -0.9482803            1
## V5  -4.9162429 -6.0204449            0
## V6   4.5110889  0.6553748            0
## V7   2.6138999 -1.8077135            1
## V8   4.2371778  0.2825710            0
## V9   3.8757869 -0.9314280            1
## V10  1.4002478 -1.1966150            1
```

```
## [1] "True Value of Beta1"
```

```
## [1] 2.285801
```

```
## [1] "Number of simulations out of 1000 that beta1 fell into t-confidence interval"
```

```
## B1ininterval
##          510
```

```
## [1] "Percentage beta1 fell into t-confidence intervals"
```

```
## B1ininterval
##           51
```

```
## [1] "10 rows from the B1 interval table"
```

```
##     upperlimit  lowerlimit B1ininterval
## V1   26.026715 13.25187524            0
## V2   18.044223 -0.70417328            1
## V3   22.531263  8.26422244            0
## V4   13.913417 -2.19881607            1
## V5    6.220946  1.95273385            1
## V6   14.970204  0.06622259            1
## V7   13.761890 -3.32953435            1
## V8   23.275582  7.98933828            0
## V9   30.777163 12.19522398            0
## V10   1.353045 -8.68493839            0
```

When the simulations were increased to 1000, the proportion of times the true beta0 is seen for t-confidence intervals actaully dropped from 55% for 100 simulations to 47.7% in 1000 simulations. The opposite occured for beta1 proportions. In 100 runs, 47% of the time the true value of beta1 was found within the t-confidence intervals.

However, as the simulation was increased to 1000, we can see that the number of times the true beta1 was observed increased to 51% for the t-confidence interval whereas for 100 simulations, the percentage for true beta1 observation was 47%.

```
## [1] "True Value of Beta0"
```

```
## [1] -0.6485048
```

```
## [1] "Number of simulations out of 1000 that beta0 fell into z-confidence interval"
```

```
## B0ininterval
##         1000
```

```
## [1] "Percentage beta0 fell into z-confidence interval"
```

```
## B0ininterval
##          100
```

```
## [1] "10 rows from the B0 interval table"
```

```
##     upperlimit lowerlimit B0ininterval
## V1   1.4807554 -1.9291863            1
## V2  -2.4036252 -5.8135669            1
## V3   0.2515392 -3.1584026            1
## V4  -0.9796389 -4.3895807            1
```

```
## V5   0.5058688 -2.9040730            1
## V6   3.8674403  0.4574985            1
## V7   3.1502448 -0.2596970            1
## V8   0.3582486 -3.0516932            1
## V9   2.6814771 -0.7284647            1
## V10  5.1329829  1.7230411            1
```

```
## [1] "True Value of Beta1"
```

```
## [1] 2.285801
```

```
## [1] "Number of simulations out of 1000 that beta1 fell into z-confidence interval"
```

```
## B1ininterval
##          485
```

```
## [1] "Percentage beta1 fell into z-confidence interval"
```

```
## B1ininterval
##         48.5
```

```
## [1] "10 rows from the B1 interval table"
```

```
##     upperlimit  lowerlimit B1ininterval
## V1    7.008615  -6.1722658            1
## V2  -14.116271 -27.2971525            0
## V3   14.536528   1.3556473            1
## V4   12.040855  -1.1400261            1
## V5    5.762334  -7.4185474            1
## V6   13.770139   0.5892578            1
## V7   16.509876   3.3289952            0
## V8   -7.239714 -20.4205951            0
## V9    1.283618 -11.8972630            0
## V10   8.435854  -4.7450267            1
```

The Z-confidence intervals are interesting. For beta0, when the number of simulations was increased to 1000, 100% of the times the true value of beta0 could be found within the z intervals,just like in the 100 simulations. The beta0 proportion is higher for z-intervals than the t-interval but the beta1 proportion dropped from 51% for t-intervals with 1000 simulations to 48.5% for z-intervals with 1000 simulations.

The beta1 scores appear to be improving with increasing simulations for both the t-confidence and z confidence intervals. The Beta0 scores appear to be constant for z-intervals and a decrease was observed for the t-intervals. Now we are going to do one final run with another 10-fold increase. These are the results for 10,000 simulation runs.

```
## [1] "True Value of Beta0"
```

```
## [1] -0.6485048

## [1] "Number of simulations out of 10,000 that beta0 was found within t-confidence interval"

## B0ininterval
##         4970

## [1] "Percentage beta0 was found within t-confidence intervals"

## B0ininterval
##         49.7

## [1] "10 rows from the B0 interval table"

##     upperlimit lowerlimit B0ininterval
## V1   0.0568175 -3.2480800            1
## V2   0.2880289 -4.5622493            1
## V3   2.2540566 -1.4368785            1
## V4   3.2200127 -0.9482803            1
## V5  -4.9162429 -6.0204449            0
## V6   4.5110889  0.6553748            0
## V7   2.6138999 -1.8077135            1
## V8   4.2371778  0.2825710            0
## V9   3.8757869 -0.9314280            1
## V10  1.4002478 -1.1966150            1

## [1] "True Value of Beta1"

## [1] 2.285801

## [1] "Number of simulations out of 10,000 that beta1 was found within t-confidence intervals"

## B1ininterval
##         4988

## [1] "Percentage for beta1 found within t-confidence intervals"

## B1ininterval
##        49.88

## [1] "10 rows from the B1 interval table"

##     upperlimit  lowerlimit B1ininterval
## V1   26.026715 13.25187524            0
## V2   18.044223 -0.70417328            1
## V3   22.531263  8.26422244            0
## V4   13.913417 -2.19881607            1
## V5    6.220946  1.95273385            1
## V6   14.970204  0.06622259            1
## V7   13.761890 -3.32953435            1
## V8   23.275582  7.98933828            0
## V9   30.777163 12.19522398            0
## V10   1.353045 -8.68493839            0
```

Between the 1000 and 10,000 simulations, beta0 found within t-intervals actually increased to 49.7% from 47%. However this is still lower than the 55% from the 100 simulations.

For beta1 there was decrease from the 1000 simulations; the proportions dropped from 51% (for 1000 runs) to 49.88% (for 10,000 runs). However, the proportion with 10,000 is higher than what was seen for 100 simulations(47%). There appears to be fluctuation of the results, instead of a straight lined improvement.

```
## [1] "True Value of Beta0"
```

```
## [1] -0.6485048
```

```
## [1] "Number of simulations out of 10,000 that beta0 fell into t-confidence interval"
```

```
## B0ininterval
##            0
```

```
## [1] "Percentage beta0 fell into z-confidence interval"
```

```
## B0ininterval
##            0
```

```
## [1] "10 rows from the B0 interval table"
```

```
##     upperlimit lowerlimit B0ininterval
## V1   1.4807554 -1.9291863            0
## V2  -2.4036252 -5.8135669            0
## V3   0.2515392 -3.1584026            0
## V4  -0.9796389 -4.3895807            0
## V5   0.5058688 -2.9040730            0
## V6   3.8674403  0.4574985            0
## V7   3.1502448 -0.2596970            0
## V8   0.3582486 -3.0516932            0
## V9   2.6814771 -0.7284647            0
## V10  5.1329829  1.7230411            0
```

```
## [1] "True Value of Beta1"
```

```
## [1] 2.285801
```

```
## [1] "Number of simulations out of 10,000 that beta1 fell into z-confidence interval"
```

```
## B1ininterval
##         5034
```

```
## [1] "Percentage beta1 fell into z-confidence interval"
```

```
## B1ininterval
##        50.34
```

```
## [1] "10 rows from the B1 interval table"
```

```
##     upperlimit  lowerlimit B1ininterval
## V1    7.008615  -6.1722658            1
## V2  -14.116271 -27.2971525            0
## V3   14.536528   1.3556473            1
## V4   12.040855  -1.1400261            1
## V5    5.762334  -7.4185474            1
## V6   13.770139   0.5892578            1
## V7   16.509876   3.3289952            0
## V8   -7.239714 -20.4205951            0
## V9    1.283618 -11.8972630            0
## V10   8.435854  -4.7450267            1
```

The results shown by the Z-intervals for 10,000 runs is a complete outlier to what has been observed so far. The proportion of runs where beta0 is found within the upper and lower limits for Z-intervals is being shown as 0. This is inconsistent with the other Z-interval results which had been constant at 100%. The proportion of times beta1 is found within the Z-intervals is 50.34% which is a slightly higher proportion than the proportion for t-intervals (49.88%). But when looking at beta1 proportions for z-intervals for other simulation runs, this result is the highest.

For both beta0 and beta1 the t-confidence proportions decreased from 100 to 1000 simulations but then increased from 1000 to 10000 simulations. For beta0 in Z-confidence intervals, the proportions remained consistent at 100% from 100 to 1000 and then went straight to 0 at 10,000 simulations. The beta1 proportions for Z-intervals actually increased from 100 to 10,000 simulation. There were no decreases.

Despite the fluctuations of proportions of confidence seen with higher numbers of simulations, the results still support the idea that increasing simulations leads to better confidence intervals. However, the results obtained do no match the definition of confidence interval because none of the results were close to the 95% proportion. The discrepancies can be explained by the small sample size of data being sampled from. A larger dataset may provide proportions closer to the 95% confidence level. In fact, Z-intervals are only appropriate to use when the sample size is greater than or equal to 30 so this can serve as an explanation as to why the beta0 values suddenly dropped to zero when the simulations were increased to a number as high as 10,000. This was all experimental. In real research, the conditions for conducting specific tests should be met before starting.

Part F: For steps (a)-(d) the sample size was fixed at 5.

- Start increasing the sample size (10,25,50,100) and run steps (a)-(d)
- Explain what happens to the mean, variance and distribution of the estimates as the sample size increases

All the simulations were kept at 100. All histograms and numbers can be found below. It was found that increasing sample sizes helps the estimated variance to converge very close to the true variance.

for sample size 10, the histogram shows the beta0_hat values to be very left skewed. The beta1_hat values are slightly left skewed (See sample 10 part C). Left skewness indicates negative skewness which can be seen when comparing the estimated means of the beta values to the true mean (see sample 10 part B). The estimated beta0 values are 0.23 points higher than the true beta0 value. For beta1, the estimated mean is 0.31 points lower than the true value. This is in line with the data represented in the histograms except that left skewness indicates values 'being more negative' than the mean and in the case of beta0_hat estimates, the mean(-0.4131) is actually less negative than the true value(-0.6485). This would mean that the beta1_hat histogram should be more right skewed. This is why it is important to double check findings from charts and graphs with calculations.

The estimated variance of beta0 is approximately 0.3 points higher than the true variance. The true variance is 3 points higher than and estimated variance for beta1 in 10 samples. There was a 13 point difference between the beta1 estimate and true value when the sample size was 5.

For sample size 25, the histograms are starting to look relatively bell shaped but the left skewness is still visible for beta0 estimates. Beta1 estimates, however are slightly more right skewed and has an outlier towards the right. The distance between the true betas and the estimated betas is similar to what was seen for sample size 10. The mean for beta0 is 0.2 points higher than the true beta0 values, which does not match the negative skewness of the histogram and the mean of the beta1_hat estimates is 0.01 points lower than the true beta1 value which also does not match the slight right skewness of beta1_hat estimates histogram in part c

The difference between the estimates and the true value of the variances is even smaller with a 25 sample dataset. The true value of beta0 is 0.05 point higher than its estimate and for beta1 the true value is 1.3 points lower than the estimated value. The Beta0 results are different from the other observations as estimated variance has always been higher than true variance so far.However, it is evident from this example and the other two examples above that the estimated variances are slowly converging to the true variance values with increasing sample size.

For sample size 50, the beta0 estimated mean is higher than the true value of beta0 by almost 0.23 points and the beta1_hat estimated mean is also higher than the true beta1 value by 0.15 points.

The histogram for beta0_hat estimates looks bimodal where as beta1_hat estimate histogram looks more bell-shaped, with a slight right skew with an outlier on the right side.

The difference in variance between the true beta0 and the estimated beta0s is approximately 0.06 points, with the true value being 0.06 points higher than the estimate. This is similar to what was seen in the previous sample with the true variance being higher than the estimated. The beta1 estimated variance is 0.35 points higher than the true variance.

For sample size 100, the estimated mean for beta0 is higher than the true beta0 value by approximately 0.24 points. The estimated mean for beta1 is also higher than the true value of beta1 by approximately 0.4 points. The histogram for beta0 estimates look more bell shaped than previously, however it looks slightly right skewed. The histogram for beta1 estimates are also right skewed, which explains the high difference between the estimated mean and the actual value. The variances for beta0 are a bit of an outlier for this sample size compared to what was seen previously. The estimated variance is 5 whole points higher than the true variance for beta0. The estimated variance of beta1 is approximately 0.3 points higher than the true variance and this is what was seen for sample size 50 as well.

For sample size 1000, it appears as if the estimated mean for beta0 has converged to the true value where the estimated mean is only 0.006 points higher than the true beta0. For beta1, the estimated mean is approximately .11 points higher than the true value. These are the lowest differences between the true value and the estimated values which confirms that increasing sample sizes allows estimated variance to converge to the true variance. The histogram for beta0 estimates is the most bellshaped with this sample size than any other. However, the beta1 estimates histogram looks nothing like bell-shape and has a outlier to the very right of the chart. Since the data was simulated, there can't be a data entry error to explain the outlier. This may be explained by how the data is being processed with the specific seed or perhaps the samples generated contains this type of variability. The estimated variance of beta0 is 0.006 points away from the

true variance of beta0 which is consistent with the other sample sizes. However the estimated value of beta1 is a full 1.0 point away from the true value of beta1 and this may be explained by the presence of the outlier in the beta1 estimates in part c. The results should be reanalyzed once the outlier is removed however I did not have time to perform this for this assignment.

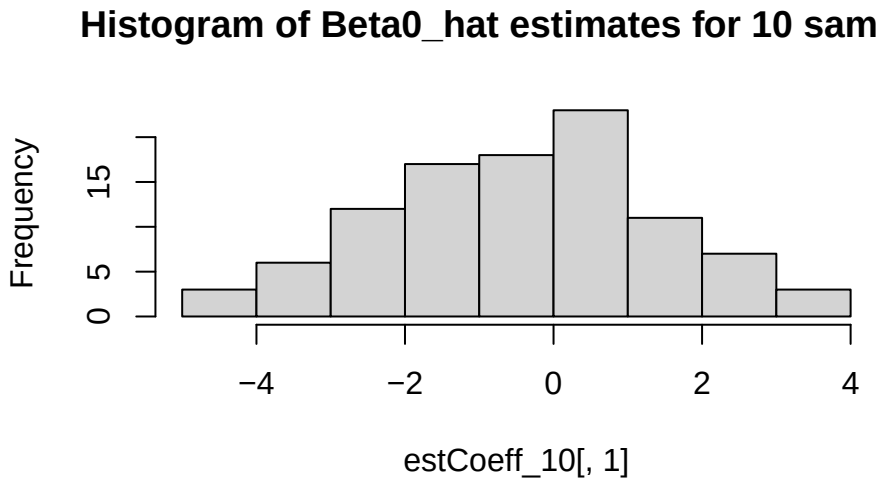
Sample 10 Part A

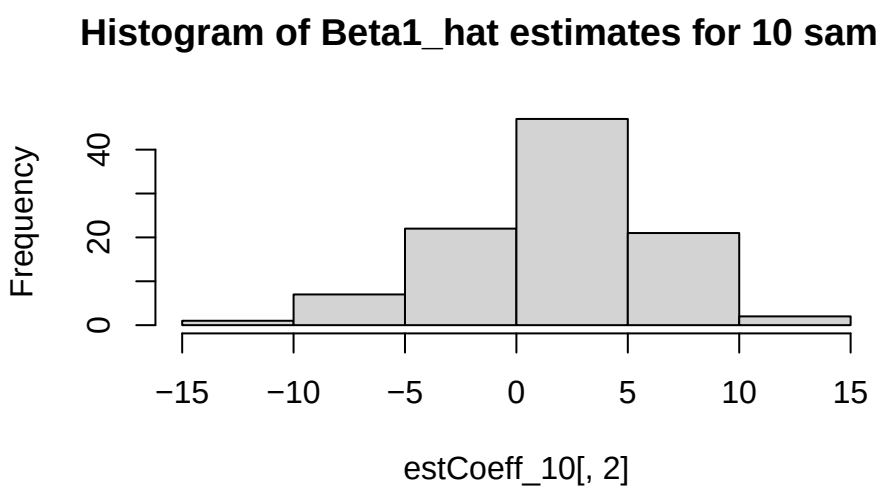
```
## Simulation
set.seed(1000557774)

## Mutiple simulation may require loops
nsample_10 <- 10 # sample size
n.sim<- 100 #number of simulations

#Simulate the predictor variable
X_10<- rnorm(nsample_10,mean=0, sd=sqrt(sigX))
```

Sample 10 Part B

```
## [1] "Estimated Mean of beta0 and beta1 respectively"

## [1] -0.41314  1.97203

## [1] "True Beta0"

## [1] -0.6485048

## [1] "True Beta1"

## [1] 2.285801
```

Sample 10 Part C





Sample 10 Part D

```
## [1] "True Variance of Beta0"

## [1] 3.152146

## [1] "Estimated Variance of Beta0"

## [1] 3.379332

## [1] "True Variance of Beta1"

## [1] 17.44936
```

```
## [1] "Estimated Variance of Beta1"
```

```
## [1] 20.32592
```

Sample 25 Part A

```
## Simulation
set.seed(1000557774)

## Multiple simulation may require loops
nsample_25 <- 25 # sample size

#Simulate the predictor variable
X_25<- rnorm(nsample_25,mean=0, sd=sqrt(sigX))
```

Sample 25 Part B

```
## [1] "Estimated Mean of beta0 and beta1 respectively"
```

```
## [1] -0.6273313  2.2788517
```

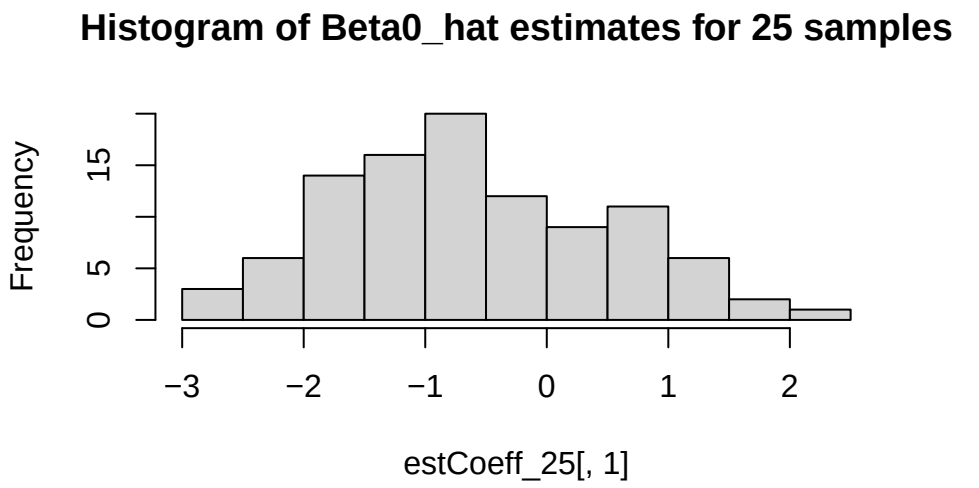
```
## [1] "True Beta0"
```

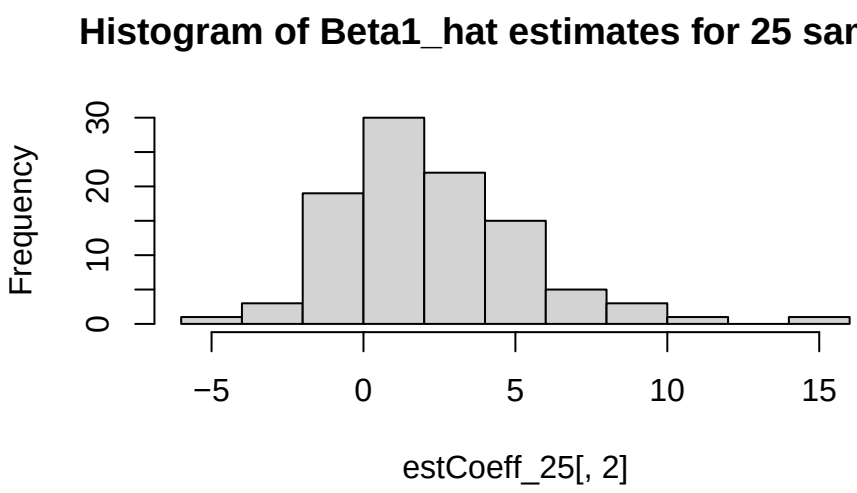
```
## [1] -0.6485048
```

```
## [1] "True Beta1"
```

```
## [1] 2.285801
```

Sample 25 Part C





Sample 25 Part D

```
## [1] "True Variance of Beta0"
```

```
## [1] 1.29446
```

```
## [1] "Estimated Variance of Beta0"
```

```
## [1] 1.242864
```

```
## [1] "True Variance of Beta1"
```

```
## [1] 8.855967
```

```
## [1] "Estimated Variance of Beta1"
```

```
## [1] 9.569599
```

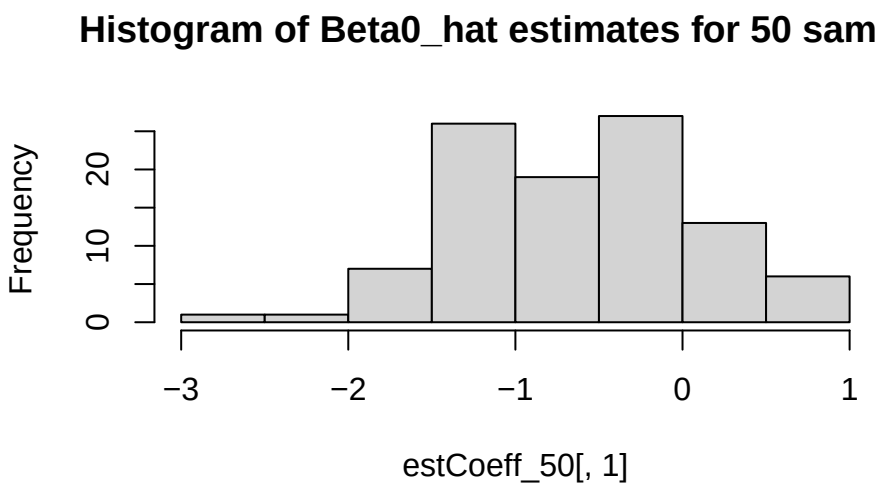
Sample 50 Part A

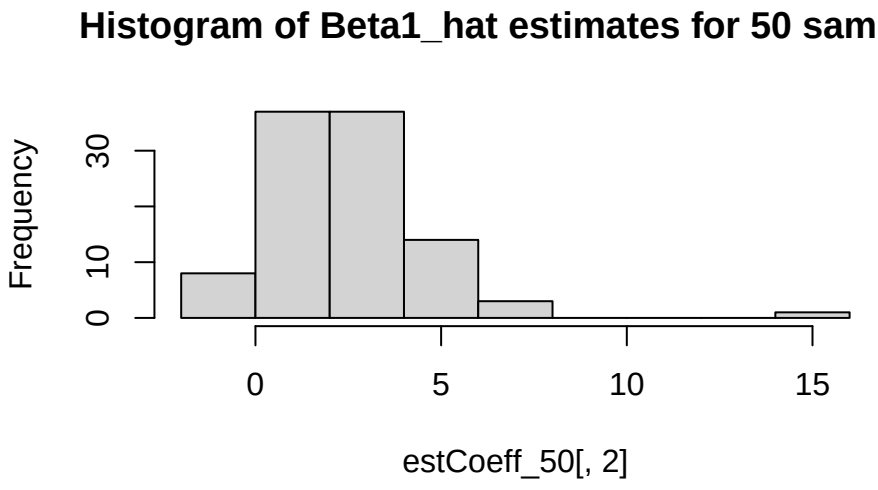
```
## Simulation
set.seed(1000557774)

## Mutiple simulation may require loops
nsample_50 <- 50 # sample size
#Simulate the predictor variable
X_50<- rnorm(nsample_50,mean=0, sd=sqrt(sigX))
```

Sample 50 Part B

```
## [1] "Estimated Mean of beta0 and beta1 respectively"

## [1] -0.6257015  2.4396389

## [1] "True Beta0"

## [1] -0.6485048

## [1] "True Beta1"

## [1] 2.285801
```

Sample 50 Part C





Sample 50 Part D

```
## [1] "True Variance of Beta0"
```

```
## [1] 0.6184373
```

```
## [1] "Estimated Variance of Beta0"
```

```
## [1] 0.5593924
```

```
## [1] "True Variance of Beta1"
```

```
## [1] 3.143115
```

```
## [1] "Estimated Variance of Beta1"
```

```
## [1] 4.497774
```

Sample 100 Part A

```
## Simulation
set.seed(1000557774)
## Mutiple simulation may require loops
nsample_100 <- 100 # sample size
#Simulate the predictor variable
X_100<- rnorm(nsample_100,mean=0, sd=sqrt(sigX))
```

Sample 100 Part B

```
## [1] "Estimated Mean of beta0 and beta1 respectively"
```

```
## [1] -0.4810379  2.3252369
```

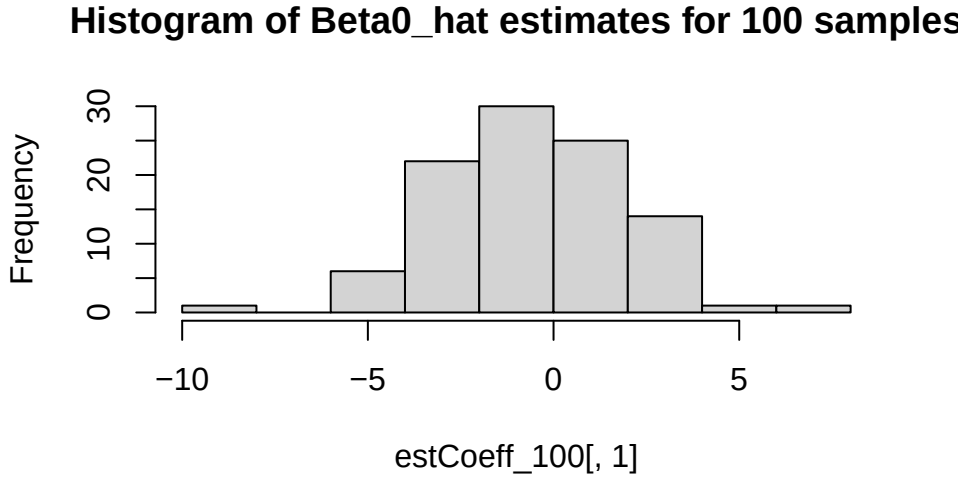
```
## [1] "True Beta0"
```

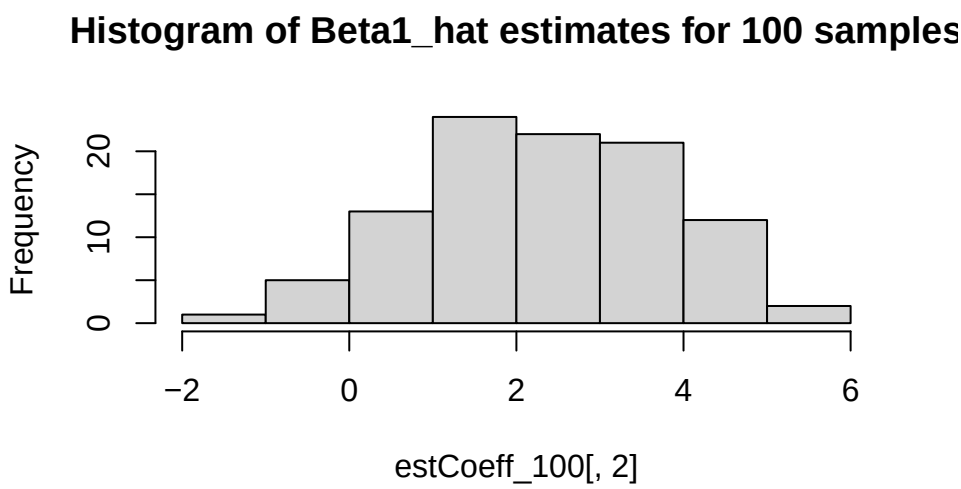
```
## [1] -0.6485048
```

```
## [1] "True Beta1"
```

```
## [1] 2.285801
```

Sample 100 Part C





Sample 100 Part D

```
## [1] "True Variance of Beta0"

## [1] 0.3123613

## [1] "Estimated Variance of Beta0"

## [1] 6.267411

## [1] "True Variance of Beta1"

## [1] 1.668272
```

```
## [1] "Estimated Variance of Beta1"
```

```
## [1] 1.95111
```

Sample 1000 Part A

```
## Simulation
set.seed(1000557774)

## Mutiple simulation may require loops
nsample_1000 <- 1000 # sample size

#Simulate the predictor variable
X_1000<- rnorm(nsample_1000,mean=0, sd=sqrt(sigX))
```

Sample 1000 Part B

```
## [1] "Estimated Mean of beta0 and beta1 respectively"
```

```
## [1] -0.6429605  2.3759050
```

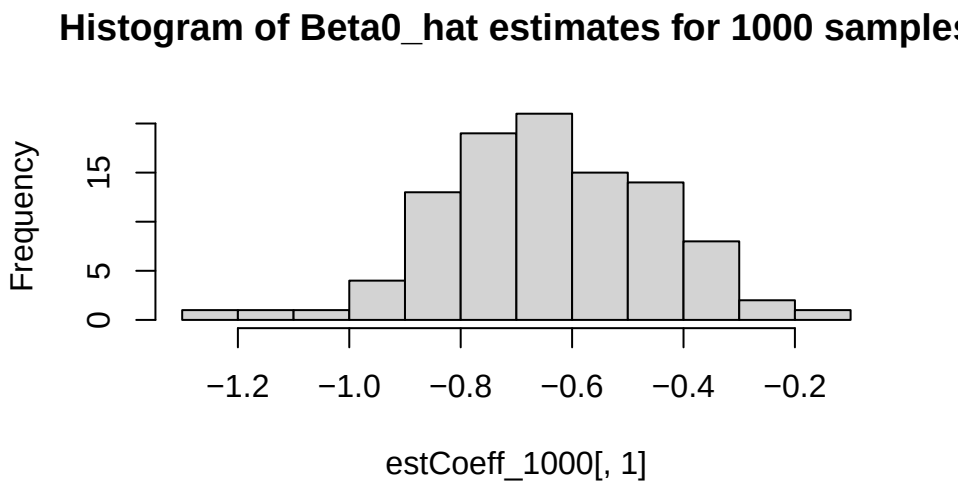
```
## [1] "True Beta0"
```

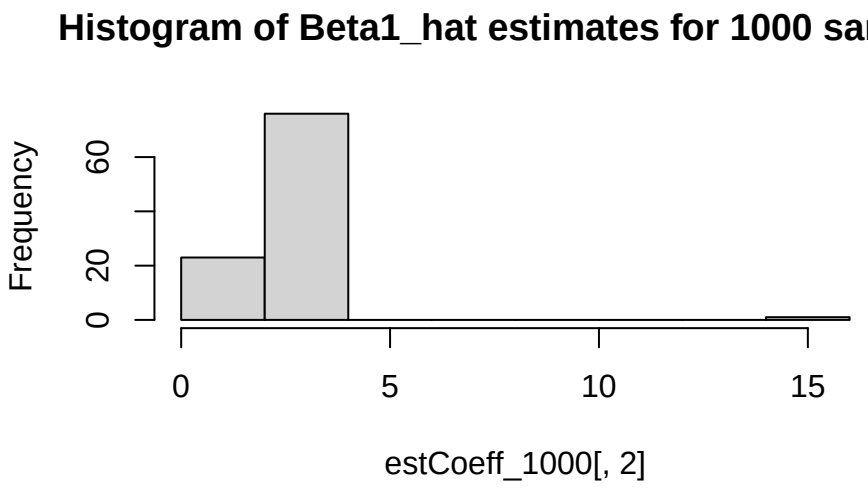
```
## [1] -0.6485048
```

```
## [1] "True Beta1"
```

```
## [1] 2.285801
```

Sample 1000 Part C





Sample 1000 Part D

```
## [1] "True Beta0"
```

```
## [1] 0.03072525
```

```
## [1] "Estimated Beta0"
```

```
## [1] 0.0366983
```

```
## [1] "True Beta1"
```

```
## [1] 0.1574622
```

```
## [1] "Estimated Beta1"
```

```
## [1] 1.652075
```

Part G: Choose the largest sample size used in step 6.

- Fix the sample size to that and start changing the error variance(sig2)
- You can increase and decrease the value of the error variance.
- For each value of error variance, execute steps 2-4.
- Explain what happens to the mean, variance and distribution of the estimates as the error variance changes

The error variance used so far had been 30, which was the approximate value of sig2. Here two different error variance is tested a value of 2 and a value of 100. Since the data points simulated for the sample size 1000 used sig2 as error variance, the results for the new variances will be compared to the results above.

The results below indicate that the higher the error variance, the further the estimated means and variances move from the true values of the regression parameters. The smaller the error variances, the smaller the distances between estimated values and true values. However, it has been observed that the beta1 estimates lose their spread when error variances are set to extreme values such as 2 and 100 which is slightly surprising for a sample size of 1000. I would have expected a bigger spread in the distribution of the estimates of the betas for a large sample size. From the results from the previous section where the error variance was approximately 30 and comparing the results from this section, it appears as if large samples sizes and small error variances help the estimates to converge close to the true values of the regression parameters but the distribution of the data no longer looks normally distributed.

There is an outlier in the beta1 estimates that has persisted in all the beta1 estimate calculations starting from sample size 25 and onwards. I could not find an explanation for this but this outlier appears to be the reason why the beta1 estimates are always skewed positively away from the true value.

With error variance of 2

```
#changed error variances and samples
set.seed(1000557774)
ev<-2
#Simulate the predictor variable
X_f<- rnorm(nsample_1000,mean=0, sd=sqrt(sigX))
```

Step2 (Part B)

```
## [1] "Estimated Mean of beta0 and beta1 respectively"
```

```
## [1] -0.6470887  2.3088148
```

```
## [1] "True Beta0"
```

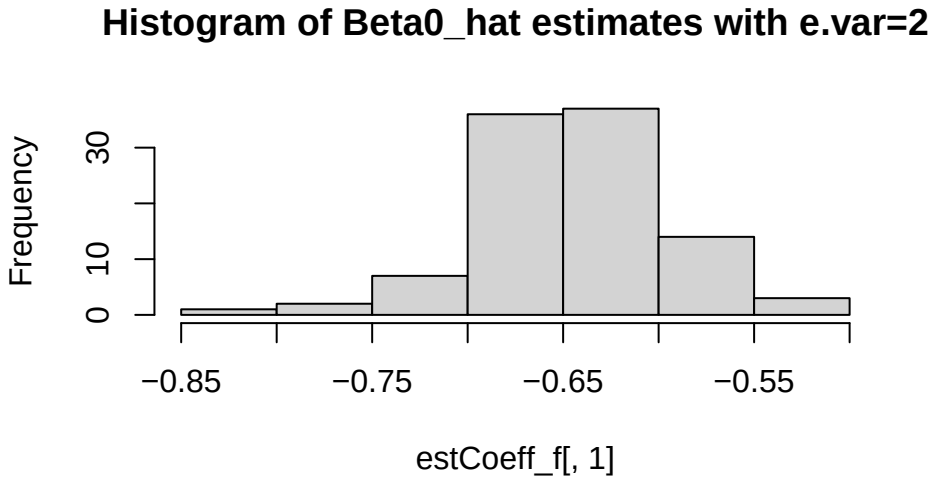
```
## [1] -0.6485048
```

```
## [1] "True Beta1"
```

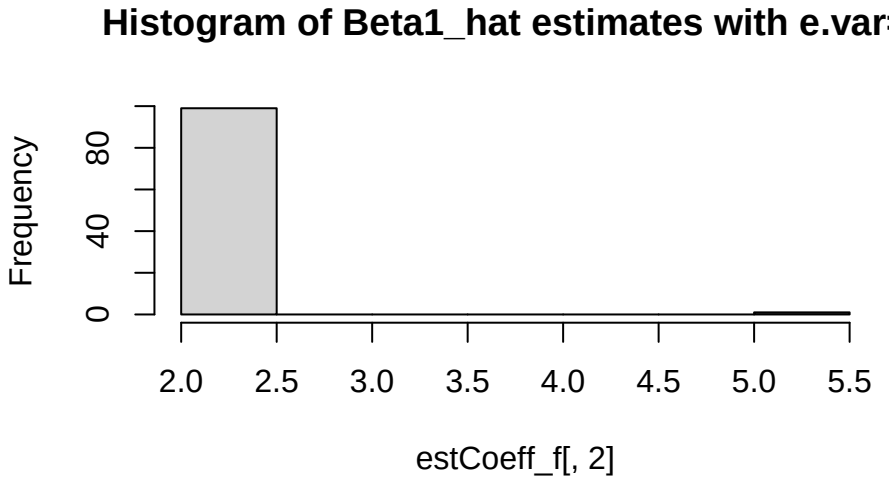
```
## [1] 2.285801
```

When the error variance is set to 2, the estimated means of both beta0 and beta1 are very close to the true value. In the case of beta0, the estimated mean is only 0.00015 points bigger than the true value. In the case of beta1, the estimated mean is only 0.02 points bigger than the true value.

Step 3 (Part C)



The histogram for beta0 estimates looks both right and left skewed at the tails but the center appears to have a very solid peak which may explain why the difference between the estimate and the true value was so small.



This chart no longer looks like a histogram and therefore does not have a describable shape. It appears as if majority of the estimated values fall between 2 and 2.5 with a single outlier value in the 5-5.5 region

Step 4 (Part D)

```
## [1] "True Variance of Beta0"

## [1] 0.03072525

## [1] "Estimated Variance of Beta0"

## [1] 0.002394109

## [1] "True Variance of Beta1"

## [1] 0.1574622

## [1] "Estimated Variance of Beta1"

## [1] 0.1077774
```

With an error variance of 2, the estimated variances are close to true values of beta0 and beta1, similar to the results of the estimated means.

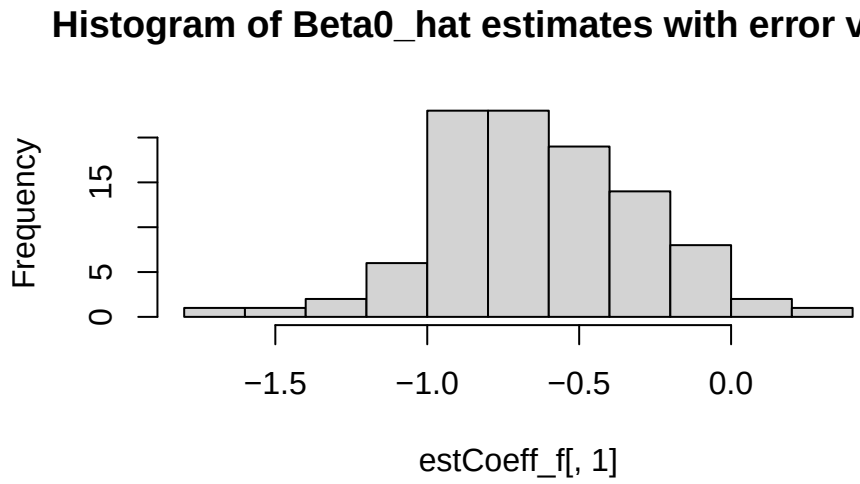
with error variance of 100

Step2 (Part B)

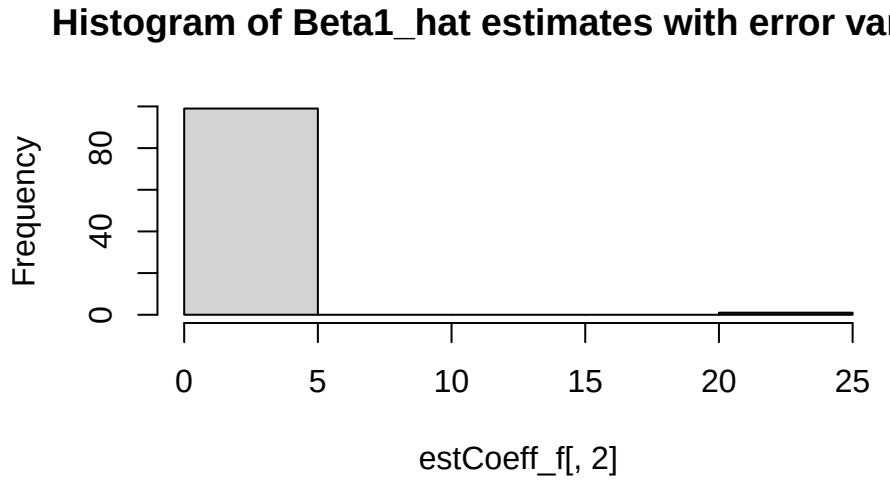
```
## [1] "Estimated Mean of beta0 and beta1 respectively"

## [1] -0.6384914  2.4485352

## [1] "True Beta0"

## [1] -0.6485048

## [1] "True Beta1"

## [1] 2.285801
```

With an error variance of 100, as expected, the estimated means of both beta0 and beta1 have diverged away from the true values by a few points.

Step 3 (Part C)



The histogram for beta0 estimates is positively skewed which fits the previous result as the mean of the beta0 estimates is higher than the true value



The beta1 estimates do not follow a distribution and seem to cluster around values between 0 and 5 with a single outlier in the 20-25 range.

Step 4 (Part D)

```
## [1] "True Variance of Beta0"
```

```
## [1] 0.03072525
```

```
## [1] "Estimated Variance of Beta0"
```

```
## [1] 0.1197055
```

```
## [1] "True Variance of Beta1"
```

```
## [1] 0.1574622
```

```
## [1] "Estimated Variance of Beta1"
```

```
## [1] 5.388871
```

When error variance is increased to 100, the estimated variances of the betas diverge far from the true variances. As can be seen above, estimated beta0 variance is .08 points higher which is a lot, considering the scale of the true variance. The difference can be seen clearly for beta1 estiamtes which a full 5.0 points away from the true beta1 variance. Increasing error variance has direct impacts on the estimated variances of the betas.

Task 2

Step A

- Assume correlation between the three standard predictors are zero(the values of the off diagonals are zero)
- With the simulation set to 100 and the sample size set to 10, generate Y for each simulation
- Run simple linear regression for each of the three betas separately
- Calculate the mean of all regression parameter estimates and check if the values are approximately equal to the true values and comment

```
## [1] "Means of Beta1, Beta2, and Beta3 respectively from SLR"
```

```
## [1] 0.6258332
```

```
## [1] -0.5256234
```

```
## [1] -0.2536893
```

```
## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Variances of Beta1, Beta2, and Beta3 respectively from SLR"
```

```
## [1] 0.005990698
```

```
## [1] 0.0123637
```

```
## [1] 0.02115411
```

```
## [1] "True Variances of Beta1, Beta2, Beta3"
```

```
## [1] 0.005841206
```

```
## [1] 0.01078769
```

```
## [1] 0.002064765
```

Estimated mean of beta1 could be considered an unbiased estimator as its value is very close to the true beta1 however the same cannot be said for betas2 and 3. The true value of beta3 is 0 but the estimated mean is shown as being close to -0.2. The estimated variances for beta1 and 2 and close to the true variances but the same cannot be said for beta3. None of the estimated variances are equal the the true variances and therefore none of the estimated betas are unbiased estimators of the their true variances.

Step B

- Fit a multiple linear regression model and obtain the regression parameter estimates along with the variances from each simulation.
- Check and comment on the unbiasedness and variances. Compare results to those found in Step A

```
## [1] "Estimates of beta1, beta2, beta3 respectively for MLR"
```

```
## [1] 0.6356037
```

```
## [1] 0.02213264
```

```
## [1] -0.006646931
```

```
## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Estimated variances of beta1, beta2, beta3 respectively for MLR"
```

```
## [1] 0.009124691
```

```
## [1] 0.01907796
```

```
## [1] 0.02296305
```

```
## [1] "True Variances of Beta1, Beta2, Beta3"
```

```
## [1] 0.005841206
```

```
## [1] 0.01078769
```

```
## [1] 0.002064765
```

After running the variables through multiple linear regression, it appears as if only the estimated mean of beta1 could be considered an unbiased estimator for the true value of beta 1. The estimated value and the true value are the same until the 100th decimal place.The same cannot be said for betas2 and 3. None of the estimated variances for the betas could be considered unbiased estimators as none of them equal the true value.

Step C

- Assume X1 and X2 are correlated. Select a value for correlation(r=0.2).
- Recreate sigmat matrix with covariance terms
- run simple linear reg again on each predictor and then run multiple linear reg
- Compare results with steps A and B
- increase the value of the coefficients(0.5,0.7,0.8) and perform steps 1 and 2 again
- How do estimates and standard errors of b1 and b2 change for simple and multiple linear reg as correlation changes

Simulation with correlation coefficient of 0.2 (r=0.2)

```
## [1] "Correlation between X1 and X2 variables"

## [1] 0.5976012

## [1] "**********Estimates and Standard Errors when r=0.2***************"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6250495

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.4201845

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4469841 0.16897675 2.645240 2.947266e-02
## X[, 1]      0.7362013 0.08653959 8.507104 2.798295e-05

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value  Pr(>|t|)
## (Intercept) -1.2239939  0.4181158 -2.927404 0.0190738
## X[, 2]       0.4681737  0.2548461  1.837084 0.1035071
```

Compared to the beta1 estimated mean obtained at step A, the beta1 estimated mean obtained after changing the correlations remained relatively the same. The beta2 estimate after correlation changed drastically. The beta2 estimated mean changed from a negative value in step A to a positive value here in step C. But similar to Step A, the beta2 estimated mean is very far from the true beta2 value.

```
## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6292725
```

```
## [1] "Estimated Means for Correlated B2 for MLR"
```

```
## [1] 0.002734836
```

```
## [1] "True value of betas"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Betas 1,2, and 3 from MLR"
```

```
##                Estimate Std. Error    t value    Pr(>|t|)
## (Intercept) -0.48996589  0.5605962 -0.8740086 0.415721209
## X[, 1]       0.58352681  0.1349095  4.3253211 0.004953882
## X[, 2]      -0.03744528  0.1391343 -0.2691305 0.796844773
## X[, 3]      -0.24254352  0.1347460 -1.8000051 0.121951563
```

After running multiple linear regression with correlated betas, the estimated mean of beta1 is the closest to its true value, compared to the other betas. This result is similar to what was observed in step B with when MLR was performed with uncorrelated betas.

When comparing the results of SLR and MLR from this step, it can be seen that the estimated mean for beta1 is consistent between the two models.The estimated mean of beta2 is closer to the true value in the results from the MLR model compared to the SLR model. If we look at the standard error for beta1 from slr, it is lower than the value shown above from the MLR model. The reverse is true for beta2; the error from the MLR model is lower than the error shown in the SLR model (0.255 vs 0.139). For both SLR and MLR, the p-value tells us that beta2 is not a significant predictor.

Simulation with correlation value of 0.5 (r=0.5)

As an be seen below, for SLR,when the correlation coefficient is increased to 0.5 the estimated mean for beta1 still remains close to the true value but the beta2 estimated mean diverges very far from the true value.

For MLR, the estimated mean of beta 1 remains close to the true value but now we can see that the estimated mean of beta2 is also converging towards its true value with only a difference of 0.0005 points.

```
## [1] "Correlation between X1 and X2 variables"
```

```
## [1] 0.7064763
```

```
## [1] "**********Estimates and Standard Errors when r=0.5**********"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6283082

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.6752489

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4042638 0.17426474 2.319826 4.893274e-02
## X[, 1]      0.6968397 0.07695123 9.055601 1.771043e-05

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value    Pr(>|t|)
## (Intercept) -2.0506509  0.4762852 -4.305511 0.002596351
## X[, 2]       0.8246224  0.2662054  3.097692 0.014717099




## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6273643

## [1] "Estimated Means for Correlated B2 for MLR"

## [1] 0.006145677

## [1] "True value of betas"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Betas 1,2, and 3 from MLR"

##                Estimate Std. Error     t value    Pr(>|t|)
## (Intercept) -0.54313547  0.4645771 -1.16909652 0.286704604
## X[, 1]       0.57366931  0.1175640  4.87963322 0.002767255
## X[, 2]      -0.01162233  0.1962388 -0.05922544 0.954695860
## X[, 3]      -0.24254352  0.1347460 -1.80000509 0.121951563
```

Simulation with correlation value of 0.7 (r=0.7)

As can be seen below for SLR, When correlation is increased to 0.7, the results for SLR match the results obtained for r=0.5; the beta1 estimated is close to the true value and the beta2 mean is far from its true value. For the MLR model, the increase in correlation moved the estimated mean of beta1 even closer but very slightly above the true mean. But the estimated mean of beta2 moved further away from its true value, when compared with the results obtained for r=0.5

Between the SLR and MLR, the standard errors for beta1 and beta2 is higher in the MLR model than the SLR model.

```
## [1] "Correlation between X1 and X2 variables"

## [1] 0.8626407

## [1] "**********Estimates and Standard Errors when r=0.7**********"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6272125

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.547569

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##               Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4463386 0.16835518 2.651172 2.920265e-02
## X[, 1]      0.7342462 0.08469345 8.669457 2.437852e-05
```

```
## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value    Pr(>|t|)
## (Intercept) -1.4404619  0.2887430 -4.988734 0.001067758
## X[, 2]       0.6235157  0.1639738  3.802532 0.005217809




## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6309534

## [1] "Estimated Means for Correlated B2 for MLR"

## [1] 0.001834091

## [1] "True value of betas"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Betas 1,2, and 3 from MLR"

##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept) -0.44085424  0.7339586 -0.6006528 0.57004887
## X[, 1]       0.60681936  0.2066146  2.9369620 0.02605026
## X[, 2]      -0.04828447  0.1967189 -0.2454491 0.81428866
## X[, 3]      -0.24254352  0.1347460 -1.8000051 0.12195156
```

Simulation with correlation value of 0.8 (r=0.8)

The results for this run with r=0.8 follows the same pattern as above. Estimated mean of beta1 and beta2 are closer to the true value in the results from the MLR model than the SLR model and the SLR model shows lower standard errors for both beta1 and beta2 than the MLR model.

```
## [1] "Correlation between X1 and X2 variables"

## [1] 0.9103557

## [1] "**********Estimates and Standard Errors when r=0.8****************"

## [1] "Estimated Means for Correlated B1 for SLR"
```

```
## [1] 0.627734

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.5878358

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4432115 0.16788129 2.640029 2.971197e-02
## X[, 1]      0.7308309 0.08304668 8.800243 2.184924e-05

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -1.5172248  0.2600368 -5.834654 0.0003894998
## X[, 2]       0.6681524  0.1465222  4.560076 0.0018497242




## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.631494

## [1] "Estimated Means for Correlated B2 for MLR"

## [1] 0.001344333

## [1] "True value of betas"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Betas 1,2, and 3 from MLR"

##                Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.42706635  0.8495099 -0.5027208 0.6330760
## X[, 1]       0.61240457  0.2459767  2.4896857 0.0471823
## X[, 2]      -0.05220263  0.2407601 -0.2168243 0.8355298
## X[, 3]      -0.24254352  0.1347460 -1.8000051 0.1219516
```

An interesting observation was that for r=0.2 and r= 0.5,the standard errors for beta1 is higher in the results from the MLR than they are from SLR. The opposite is true for the standard errors of beta2; the standard errors are higher from SLR than MLR. However, for r=0.7 and r=0.8, the standard errors from the SLR results are lower than the MLR results. However, when it comes to the unbiasedness of estimators, it is consistently shown in the results above that the estimates of beta1 and beta2 returned from the MLR model are closer to the real beta values. The estimated betas from the MLR model are more likely to be unbiased estimators of the true beta values.

Step D

- Assume X1 and X2 are uncorrelated and X1 and X3 are correlated.
- Select an arbitrary value for r and change the values of sigma[1,3] and sigmat[3,1]
- Perform steps A and B and compare with results from step C; comment on similarities and differences
- start increasing values of correlation coefficients (0.5, 0.7, 0.8, 0.9, 0.95)
- How do the estimates and standard error of B1 and B2 change for simple and multiple linear regression as the correlation changes?

Since correlation was being increased in the previous step and in this step, a negative abritatry correlation was selected to see what would happen to the results.

Simulation with correlation value of -0.5 (r=-0.5)

In the previous step, all the correlations were positive. This correlation is negative. With r=-0.5,the estimated mean for beta1 is close to the true mean. It appears to be off by 0.0071 points which is not too bad. The estimated value of beta2 is off by a full 2 points from the true value, meaning the estimate is very far from the true value. This seems to be consistent with the general pattern of the estimates as seen from SLR models in this experiment.

In step C, both SLR and MLR was run with the value of r= +0.5. The estimated beta1 values are not very different between r=0.5 and r=-0.5 but the estimated beta2 values are close to 6 points different, with the estimated beta2 value being higher for r=0.5. Between the two opposing correlation coefficients, the standard errors, however, are in pretty close range of each other. The pattern seen on this run is consistent with the patterns when r=0.5; estimated mean of beta1 is closer to the true values than the estimated mean of beta2 is to its true value. Standard error is lower for beta1 in the SLR model and standard error is lower for beta2 in the MLR model.

The similarities in results are surprising considering that different parameters are correlated for this section of the experiments.

```
## [1] "Correlation between X1 and X3 variables"
```

```
## [1] -0.7653413
```

```
## [1] "**********Estimates and Standard Errors when r=0.4**********"
```

```
## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6203468

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] -0.2518075

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4388016 0.16786793 2.613969 3.093895e-02
## X[, 1]      0.7307641 0.08569612 8.527389 2.750171e-05

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.1774348  0.4462772 -0.3975887 0.7013313
## X[, 2]      -0.4295216  0.2902715 -1.4797236 0.1772133

## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6301674

## [1] "Estimated Means for Correlated B2 for MLR"

## [1] 0.02699652

## [1] "True value of betas"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Betas 1,2, and 3 from MLR"

##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept)  0.37880691  0.4639735  0.8164408 0.44544508
## X[, 1]       0.59653213  0.1667509  3.5773852 0.01167981
## X[, 2]      -0.22768090  0.1296377 -1.7562859 0.12955920
## X[, 3]       0.04992605  0.1645064  0.3034901 0.77176488
```

Simulation with correlation value of 0.5 (r=0.5)

Here r=0.5 is being run once more although it was run in step C because the correlated variables are different in this case. The estimated beta values are similar and close to the true value in the runs r=0.5 and r=-0.5 (above) for SLR. The difference is in estimated mean for beta2 which is approximately 5 point higher in this run than when r=-0.5. The standard errors follow the same intervals for both r=0.5 and r=-0.5. The results for MLR are the same as the patterns for SLR. The estimated mean of beta1 is the similar to the true beta and they are similar to each other when r=0.5 and when r=-0.5. But the estimated mean of beta2 differ between the two correlation coefficients for the MLR models. For this run, the estimated mean of beta2 is negative value where was when r=-0.5, the estimated mean of beta 2 is a positive value. This is counter-intuitive but it may have something to do with the which variables are correlated to each other. The standard errors for the MLR models from both positive and negative coefficients follow the same interval and pattern: standard errors for beta1 is lower in the SLR model and standard errors for beta2 is lower in the MLR model

```
## [1] "Correlation between X1 and X3 variables"

## [1] 0.7653413

## [1] "**********Estimates and Standard Errors when r=0.5****************"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6237669

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.2631428

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4483770 0.17042221 2.630977 3.013243e-02
## X[, 1]      0.7341843 0.08700007 8.438893 2.967121e-05

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -1.5047947  0.6316667 -2.382261 0.04438879
## X[, 2]       0.4408569  0.2902715  1.518774 0.16730169
```

```
## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6301674

## [1] "Estimated Means for Correlated B2 for MLR"

## [1] -0.01566121

## [1] "True value of betas"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Betas 1,2, and 3 from MLR"

##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept) -0.52427201  0.3561958 -1.4718646 0.19147350
## X[, 1]       0.59653213  0.1667509  3.5773852 0.01167981
## X[, 2]       0.23901622  0.1296377  1.8437243 0.11478418
## X[, 3]      -0.04992605  0.1645064 -0.3034901 0.77176488
```

Simulation with correlation value of 0.7 (r=0.7); written summary of findings for coefficients 0.7-0.95

For the SLR models, the results for the estimated mean of beta1 and beta2 follow a consistent pattern and interval for r=0.7,0.8,0.9, and 0.95. The findings will be summarized here and the results can be seen below.

For SLR models for all the coefficients listed above, the estimated mean of beta1 falls between the values of 0.6241 to 0.6248. This is very close to the true value which is listed as 0.63073 and the estimate gets closer as the r coefficient increases. The estimated means of beta2 range from 0.2359-0.2576 and this is very far from the true value which is listed as 0.00567. So as correlation between X1 and X3 increases, the estimated mean of beta1 gets close to the true value but the estimated mean of beta2 actually moves further away.

For MLR models for all the coefficients listed, the estimated mean of beta1 falls between the values of 0.6309-0.6340. Similar to what was observed for SLR model, the estimated means of beta1 also increase as the r coefficient increases and actaully goes above the true mean of 0.6307. The estimated value for beta2 stays exactly at -0.01566121 for all the values listed above. This is very odd considering it does converge or diverge to or away from beta2's true value of 0.00567.

The standard errors,however to fluctuate quite a bit. For SLR models the standard errors can be found within the range of 0.0849 to 0.08056 for beta1 and actually decreases as the coefficients increase. For beta2

the standard errors range from 0.2986 to 0.3185. The standard errors actually increase for beta2 as the coefficients increase.

For the MLR models, standard errors range from 0.206 to 0.464 for beta1 and actually increase with increasing coefficients of correlations. standard errors range stay at 0.1296 for beta2, even as the correlation coefficient increases.

```
## [1] "Correlation between X1 and X3 variables"

## [1] 0.8643778

## [1] "**********Estimates and Standard Errors when r=0.7*****************"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6240542

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.2576071

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4440727 0.16973405 2.616285 3.082782e-02
## X[, 1]      0.7297859 0.08491102 8.594714 2.596937e-05

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -1.5091143  0.6497014 -2.322781 0.04870748
## X[, 2]       0.4353212  0.2985590  1.458074 0.18293061




## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6308701
```

```
## [1] "Estimated Means for Correlated B2 for MLR"

## [1] -0.01566121

## [1] "True value of betas"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Betas 1,2, and 3 for MLR"

##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept) -0.52057064  0.3527917 -1.4755750 0.19051020
## X[, 1]       0.60515543  0.2064543  2.9311837 0.02624665
## X[, 2]       0.23901622  0.1296377  1.8437243 0.11478418
## X[, 3]      -0.05428902  0.2063841 -0.2630485 0.80131300
```

Simulation with correlation value of 0.8 (r=0.8)

```
## [1] "Correlation between X1 and X3 variables"

## [1] 0.9112585

## [1] "**********Estimates and Standard Errors when r=0.8****************"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6242732

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.2523782

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4406873 0.16934584 2.602292 3.150540e-02
## X[, 1]      0.7264388 0.08344195 8.705919 2.364206e-05
```

```
## [1] "Standard errors of Beta2 from SLR"
```

```
##               Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -1.5106497  0.6633283 -2.277379 0.05228482
## X[, 2]       0.4300924  0.3048210  1.410967 0.19593738
```

```
## [1] "Estimated Means for Correlated B1 for MLR"
```

```
## [1] 0.6314209
```

```
## [1] "Estimated Means for Correlated B2 for MLR"
```

```
## [1] -0.01566121
```

```
## [1] "True value of betas"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Betas 1,2, and 3 from MLR"
```

```
##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept) -0.52064262  0.3551043 -1.4661680 0.19296124
## X[, 1]       0.61089854  0.2460636  2.4826852 0.04763175
## X[, 2]       0.23901622  0.1296377  1.8437243 0.11478418
## X[, 3]      -0.05851871  0.2518690 -0.2323379 0.82399769
```

Simulation with correlation value of 0.9 (r=0.9)

```
## [1] "Correlation between X1 and X3 variables"
```

```
## [1] 0.9563624
```

```
## [1] "**********Estimates and Standard Errors when r=0.9**********"
```

```
## [1] "Estimated Means for Correlated B1 for SLR"
```

```
## [1] 0.6245628
```

```
## [1] "Estimated Means for Correlated B2 for SLR"
```

```
## [1] 0.2439119
```

```
## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Beta1 from SLR"
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.4360726 0.16902894 2.579870 3.262294e-02
## X[, 1]      0.7220169 0.08166537 8.841164 2.111897e-05
```

```
## [1] "Standard errors of Beta2 from SLR"
```

```
##              Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -1.510013  0.6810838 -2.217074 0.05744373
## X[, 2]       0.421626  0.3129803  1.347133 0.21484883
```

```
## [1] "Estimated Means for Correlated B1 for MLR"
```

```
## [1] 0.6325361
```

```
## [1] "Estimated Means for Correlated B2 for MLR"
```

```
## [1] -0.01566121
```

```
## [1] "True value of betas"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Betas 1,2, and 3 from MLR"
```

```
##                Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.52522831  0.3757349 -1.3978692 0.2116487
## X[, 1]       0.62134557  0.3362133  1.8480696 0.1140950
## X[, 2]       0.23901622  0.1296377  1.8437243 0.1147842
## X[, 3]      -0.06853736  0.3579038 -0.1914966 0.8544526
```

Simulation with correlation value of 0.95 (r=0.95)

```
## [1] "Correlation between X1 and X3 variables"

## [1] 0.9783104

## [1] "**********Estimates and Standard Errors when r=0.95**********"

## [1] "Estimated Means for Correlated B1 for SLR"

## [1] 0.6247624

## [1] "Estimated Means for Correlated B2 for SLR"

## [1] 0.2369027

## [1] "True Values of Beta0, Beta1, Beta2 and Beta3 respectively"

##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000

## [1] "Standard errors of Beta1 from SLR"

##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 0.4328017 0.16895567 2.561629 0.033561824
## X[, 1]      0.7189739 0.08055996 8.924706 0.000019711

## [1] "Standard errors of Beta2 from SLR"

##               Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -1.5075346  0.6930164 -2.175323 0.06130776
## X[, 2]       0.4146169  0.3184637  1.301928 0.22917484




## [1] "Estimated Means for Correlated B1 for MLR"

## [1] 0.6340355

## [1] "Estimated Means for Correlated B2 for MLR"

## [1] -0.01566121

## [1] "True value of betas"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Betas 1,2, and 3 from MLR"
```

```
##                Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.53473618  0.4315077 -1.2392274 0.2615349
## X[, 1]       0.63459281  0.4644280  1.3663965 0.2208052
## X[, 2]       0.23901622  0.1296377  1.8437243 0.1147842
## X[, 3]      -0.08316856  0.5101825 -0.1630173 0.8758569
```

Conclusion

In this report, many experiments were conducted on simulated data where the true population coefficients were known. Through much experimentation, it was observed that increasing simulations, or the number of time a population is sampled helps the model to reach a value very close to the population mean. Increasing the number of samples allows the model to reach a value very close to the true population variance. Correlations between different variables can impact not only the errors and means of the variables themselves but it can also impact the means and errors of other non-correlated variables. Therefore, in task2, for parts c and d the estimated means for betas 1 and 2 cannot be unbiased estimators of the true values since the correlated variables impact their results.

References

- Alex Hayes and Ralph Moller-Trane (2019). distributions3: Probability Distributions as S3 Objects. R package version 0.1.1. https://CRAN.R-project.org/package=distributions3
- Barlett.J.(2019). Running simulation studies in R- an introductory tutorial: http://thestatsgeek.com/SimulationStudiesinR.html
- Bevans.R.(2021).Confidence intervals explained. Scribbr, https://www.scribbr.com/statistics/confidence-interval/
- Correlation in R. DataScience Made Simple. https://www.datasciencemadesimple.com/correlation-in-r/
- How to tell if and estimator is biased or unbiased. https://www.youtube.com/watch?v=LP4r4wENSY4
- Schork.J. Extract Standard Error, t-value& p-value from Linear Regression Model in R. Statistics Globe. https://statisticsglobe.com/extract-standard-error-t-and-p-value-from-regression-in-r
- StackExchang(2018).https://stats.stackexchange.com/questions/327871/what-does-the-variance-of-an-estimator-for-a-regression-parameter-mean#429472

- Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686
- Z-confidence interval for a mean: https://cran.r-project.org/web/packages/distributions3/vignettes/one-sample-z-confidence-interval.html

Appendix

Task 1

Part A: Simulating parameters using code

```
## Simulation
set.seed(1000557774)
beta0<-rnorm(1,mean=0, sd=1) #the population beta0
beta1<-runif(n=1, min=1, max=3) #the population beta1
sig2<- rchisq(n=1,df=25) ## the error variance sigma squared

## Multiple simulation may require loops
nsample <- 5 # sample size
n.sim<- 100 #number of simulations
sigX<- 0.2 # The variances of X

#Simulate the predictor variable
X<- rnorm(nsample,mean=0, sd=sqrt(sigX))
```

Part B:

```
set.seed(1000557774)
estCoeff<- array(0,dim=c(n.sim,2)) #array to save coefficients

for (i in 1:n.sim){
  e<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Y<-beta0+beta1*X+e
  fitmodel<- lm(Y~X)
  estCoeff[i,]<-coefficients(fitmodel)


}
beta0_hat<-estCoeff[,1] # subsetting column1 of bets array
beta1_hat<-estCoeff[,2]

print("Estimated Mean of beta0 and beta1 respectively")
meanest_5<-colMeans(estCoeff)
meanest_5
```

```
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Part C:

```
hist(estCoeff[,1], main='Histogram of Beta0_hat estimates')
```

```
hist(estCoeff[,2], main='Histogram of Beta1_hat estimates')
```

Part D:

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)

varbeta0<-var(estCoeff[,1]) #calculating variance of beta0
#Beta1_hat
varbeta1<-var(estCoeff[,2]) #calculating variance of beta1
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx <- sum(X); xbar <- sumx/nsample
sumy <- sum(Y); ybar <- sumy/nsample
sumx2 <- sum(X^2); SXX <- (sumx2 - nsample*xbar^2)
sumxy <- sum(X*Y)
bet1_hat <- (sumxy - nsample*xbar*ybar)/(SXX)
bet0_hat<-ybar-(bet1_hat*xbar)

#variance of b1
varb1<- sig2/SXX
#variance of b0
varb0<-sig2*((1/nsample)+((xbar^2)/SXX))

print("True Variance of Beta0")
varb0
print("Estimated Variance of Beta0")
varbeta0 #calculated with var function
print("True Variance of Beta1")
varb1
print("Estimated Variance of Beta1")
varbeta1 #calculated with var function
```

Part E:

```
#out of a 100 simulations how many of them include the true betas
#for tdistribution,include 't' to separate out variables used earlier and those used in the z distribut
```

```r
#using n.sim<-100
library(tidyverse)
set.seed(1000557772)
alpha<-0.5
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta0 falls within the uppe
B0tinterval<-array(0,dim=c(n.sim,3))
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta1 falls within the uppe
B1tinterval<-array(0,dim=c(n.sim,3))
for (i in 1:n.sim){
  et<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Yt<-beta0+beta1*X+et
  fitmodelt<- lm(Yt~X)
  yhatt <- predict(fitmodelt, type = "response")
  err_vart <- sum((Yt-yhatt)^2)/(nsample - 2)
  sumxt <- sum(X); xbart <- sumxt/nsample
  sumyt <- sum(Yt); ybart <- sumyt/nsample
  sumxt2 <- sum(X^2); SXXt <- (sumxt2 - nsample*xbart^2)
  sumxyt <- sum(X*Yt)
  beta1_hatt <- (sumxyt - nsample*xbart*ybart)/(SXXt)
  beta0_hatt<-ybart-(beta1_hatt*xbart)

#for Beta0_hat

  B0ll <- beta0_hatt - qt(1 - (alpha/2), df = nsample - 2)*((sqrt(err_vart)*(sqrt((1/nsample+((xbart^2),
  B0ul <- beta0_hatt + qt(1 - (alpha/2), df = nsample - 2)*((sqrt(err_vart)*(sqrt((1/nsample+((xbart^2),

  #if beta0 is greater than or equal to B0ll(lowerlimit) and less than or equal the B0ul(upperlimit), B
  B0ininterval<-ifelse(between(beta0,B0ll,B0ul),1,0)
  B0tinterval[i,]<-c(B0ul,B0ll,B0ininterval)

#for Beta1_hat
  B1ll <- beta1_hatt - qt(1 - (alpha/2), df = nsample - 2)*(sqrt(err_vart)/(sqrt(SXXt)))
  B1ul <- beta1_hatt + qt(1 - (alpha/2), df = nsample - 2)*(sqrt(err_vart)/(sqrt(SXXt)))
  #if beta1 is greater than or equal to 1Oll(lowerlimit) and less than or equal the B1ul(upperlimit), B
  B1ininterval<-ifelse(between(beta1,B1ll,B1ul),1,0)
  B1tinterval[i,]<-c(B1ul,B1ll,B1ininterval)

  }

# turn array into dataframes for easily summing selective columns. 3 rows, n.sim columns
B1ttable<-as.data.frame(t(B1tinterval), row.names = c("upperlimit","lowerlimit","B1ininterval"))
B0ttable<-as.data.frame(t(B0tinterval), row.names = c("upperlimit","lowerlimit","B0ininterval"))



print("True Value of Beta0")
beta0

print("Number of simulations out of 100 that beta0 fell into t-confidence intervals")
rowSums(B0ttable["B0ininterval",c(1:n.sim)])

print("10 rows from the B0 interval table")
head(t(B0ttable),10)
```

```
print("True Value of Beta1")
beta1
print("Number of simulations out of 100 that beta1 fell into t-confidence intervals")
rowSums(B1ttable["B1ininterval",c(1:n.sim)])

print("10 rows from the B1 interval table")
head(t(B1ttable),10)
```

```
# z confidence intervals
#for Beta0_hat, where sig2 is the population error variance
#install.packages("distributions")
set.seed(1000557774)
library(distributions3)
Z<-Normal(0,1)
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta0 falls within the uppe

z0tinterval<-array(0,dim=c(n.sim,3))
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta1 falls within the uppe
z1tinterval<-array(0,dim=c(n.sim,3))
sumxz2 <- sum(X^2)
sumxz <- sum(X)
xbarz <- sumxz/nsample
SXXz <- (sumxz2 - nsample*xbarz^2)
for (i in 1:n.sim){
  ez<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Yz<-beta0+beta1*X+ez
  fitmodelz<- lm(Yz~X)
  yhatz <- predict(fitmodelz, type = "response")
  err_varz <- sum((Yz-yhatz)^2)/(nsample - 2)
  sumyz <- sum(Yz)
  ybarz <- sumyz/nsample
  sumxyz <- sum(X*Yz)
  beta1_hatz <- (sumxyz - nsample*xbarz*ybarz)/(SXXz)
  beta0_hatz<-ybarz-(beta1_hatz*xbarz)

#for Beta0_hat, eqn from lecture 3 slide 19
    #lower limit
  z0ll <- beta0_hatz - quantile(Z, 1-(alpha/2))*((sqrt(sig2)*(sqrt((1/nsample)+((xbarz^2)/SXXz)))))
    #upper limit
  z0ul <- beta0_hatz + quantile(Z, 1-(alpha/2))*((sqrt(sig2)*(sqrt((1/nsample)+((xbarz^2)/SXXz)))))


  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  z0ininterval<-ifelse(between(beta0,z0ll,z0ul),1,0)
  # insert all three values into array created above
  z0tinterval[i,]<-c(z0ul,z0ll,B0ininterval)

#for Beta1_hat, eqn from lecture 3 slide 8
  z1ll <- beta1_hatz - quantile(Z, 1 - (alpha/2))*(sqrt(sig2)/(sqrt(SXXz)))
  z1ul <- beta1_hatz + quantile(Z, 1 - (alpha/2))*(sqrt(sig2)/(sqrt(SXXz)))

  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  z1ininterval<-ifelse(between(beta1,z1ll,z1ul),1,0)
```

```
  z1tinterval[i,]<-c(z1ul,z1ll,z1ininterval)

  }
# turn array into dataframes for easily summing selective columns
z1ttable<-as.data.frame(t(z1tinterval), row.names = c("upperlimit","lowerlimit","B1ininterval"))
z0ttable<-as.data.frame(t(z0tinterval), row.names = c("upperlimit","lowerlimit","B0ininterval"))

print("True Value of Beta0")
beta0

print("Number of simulations out of 100 that beta0 fell into z-confidence intervals")
rowSums(z0ttable["B0ininterval",c(1:n.sim)])
print("10 rows from the B0 interval table")
head(t(z0ttable),10)
print("True Value of Beta1")
beta1
print("Number of simulations out of 100 that beta1 fell into z-confidence intervals")
rowSums(z1ttable["B1ininterval",c(1:n.sim)])
print("10 rows from the B1 interval table")
head(t(z1ttable),10)
```

```
#out of a 100 simulations how many of them include the true betas
#for tdistribution,include 't' to separate out variables used earlier and those used in the z distribut
newsim=1000
library(tidyverse)
set.seed(1000557772)
alpha<-0.5
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta0 falls within the uppe
B0tinterval<-array(0,dim=c(newsim,3))
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta1 falls within the uppe
B1tinterval<-array(0,dim=c(newsim,3))
for (i in 1:newsim){
  et<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Yt<-beta0+beta1*X+et
  fitmodelt<- lm(Yt~X)
  yhatt <- predict(fitmodelt, type = "response")
  err_vart <- sum((Yt-yhatt)^2)/(nsample - 2)
  sumxt <- sum(X); xbart <- sumxt/nsample
  sumyt <- sum(Yt); ybart <- sumyt/nsample
  sumxt2 <- sum(X^2); SXXt <- (sumxt2 - nsample*xbart^2)
  sumxyt <- sum(X*Yt)
  beta1_hatt <- (sumxyt - nsample*xbart*ybart)/(SXXt)
  beta0_hatt<-ybart-(beta1_hatt*xbart)

#for Beta0_hat

  B0ll <- beta0_hatt - qt(1 - (alpha/2), df = nsample - 2)*((sqrt(err_vart)*(sqrt((1/nsample+((xbart^2)/
  B0ul <- beta0_hatt + qt(1 - (alpha/2), df = nsample - 2)*((sqrt(err_vart)*(sqrt((1/nsample+((xbart^2)/
  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  B0ininterval<-ifelse(between(beta0,B0ll,B0ul),1,0)
  B0tinterval[i,]<-c(B0ul,B0ll,B0ininterval)

#for Beta1_hat
```

```
  B1ll <- beta1_hatt - qt(1 - (alpha/2), df = nsample - 2)*(sqrt(err_vart)/(sqrt(SXXt)))
  B1ul <- beta1_hatt + qt(1 - (alpha/2), df = nsample - 2)*(sqrt(err_vart)/(sqrt(SXXt)))
  B1ininterval<-ifelse(between(beta1,B1ll,B1ul),1,0)
  B1tinterval[i,]<-c(B1ul,B1ll,B1ininterval)

  }

# turn array into dataframes for easily summing selective columns. 3 rows, n.sim columns
B1ttable<-as.data.frame(t(B1tinterval), row.names = c("upperlimit","lowerlimit","B1ininterval"))
B0ttable<-as.data.frame(t(B0tinterval), row.names = c("upperlimit","lowerlimit","B0ininterval"))

b1prop<-rowSums(B1ttable["B1ininterval",c(1:newsim)])
b0prop<-rowSums(B0ttable["B0ininterval",c(1:newsim)])

print("True Value of Beta0")
beta0

print("Number of simulations out of 1000 that beta0 fell into t-confidence interval")
b0prop
print("Percentage beta0 fell into t-confidence intervals")
(b0prop/newsim)*100
print("10 rows from the B0 interval table")
head(t(B0ttable),10)
print("True Value of Beta1")
beta1
print("Number of simulations out of 1000 that beta1 fell into t-confidence interval")
b1prop
print("Percentage beta1 fell into t-confidence intervals")
(b1prop/newsim)*100
print("10 rows from the B1 interval table")
head(t(B1ttable),10)
```

```
# z confidence intervals
#for Beta0_hat, where sig2 is the population error variance
#install.packages("distributions")
set.seed(1000557774)
library(distributions3) #library for probability distributions
Z<-Normal(0,1)

#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta0 falls within the uppe
z0tinterval<-array(0,dim=c(newsim,3))
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta1 falls within the uppe
z1tinterval<-array(0,dim=c(newsim,3))

for (i in 1:newsim){
  ez<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Yz<-beta0+beta1*X+ez
  fitmodelz<- lm(Yz~X)
  yhatz <- predict(fitmodelz, type = "response")
  err_varz <- sum((Yz-yhatz)^2)/(nsample - 2)
  sumxz <- sum(X); xbarz <- sumxz/nsample
  sumyz <- sum(Yz); ybarz <- sumyz/nsample
  sumxz2 <- sum(X^2); SXXz <- (sumxz2 - nsample*xbarz^2)
```

```
  sumxyz <- sum(X*Yz)
  beta1_hatz <- (sumxyz - nsample*xbarz*ybarz)/(SXXz)
  beta0_hatz<-ybarz-(beta1_hatz*xbarz)

#for Beta0_hat, eqn from lecture 3 slide 19
  # lower limit
  z0ll <- beta0_hatz - quantile(Z, 1-(alpha/2))*((sqrt(sig2)*(sqrt((1/nsample)+((xbarz^2)/SXXz)))))
  # upper limit
  z0ul <- beta0_hatz + quantile(Z, 1-(alpha/2))*((sqrt(sig2)*(sqrt((1/nsample)+((xbarz^2)/SXXz)))))


  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  z0ininterval<-ifelse(between(beta0,z0ll,z0ul),1,0)
  # insert all three values into array created above
  z0tinterval[i,]<-c(z0ul,z0ll,B0ininterval)

#for Beta1_hat, eqn from lecture 3 slide 8
  #lower limit
  z1ll <- beta1_hatz - quantile(Z, 1 - (alpha/2))*(sqrt(sig2)/(sqrt(SXXz)))
  #upper limit
  z1ul <- beta1_hatz + quantile(Z, 1 - (alpha/2))*(sqrt(sig2)/(sqrt(SXXz)))

  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  z1ininterval<-ifelse(between(beta1,z1ll,z1ul),1,0)
  z1tinterval[i,]<-c(z1ul,z1ll,z1ininterval)

  }
# turn array into dataframes for easily summing selective columns
z1ttable<-as.data.frame(t(z1tinterval), row.names = c("upperlimit","lowerlimit","B1ininterval"))
z0ttable<-as.data.frame(t(z0tinterval), row.names = c("upperlimit","lowerlimit","B0ininterval"))

print("True Value of Beta0")
beta0

print("Number of simulations out of 1000 that beta0 fell into z-confidence interval")
z0prop<-rowSums(z0ttable["B0ininterval",c(1:newsim)])
z0prop
print("Percentage beta0 fell into z-confidence interval")
(z0prop/newsim)*100
print("10 rows from the B0 interval table")
head(t(z0ttable),10)
print("True Value of Beta1")
beta1
print("Number of simulations out of 1000 that beta1 fell into z-confidence interval")
z1prop<-rowSums(z1ttable["B1ininterval",c(1:newsim)])
z1prop
print("Percentage beta1 fell into z-confidence interval")
(z1prop/newsim)*100
print("10 rows from the B1 interval table")
head(t(z1ttable),10)
```

```
#out of a 100 simulations how many of them include the true betas
#for tdistribution,include 't' to separate out variables used earlier and those used in the z distribut
```

```
newsim=10000
library(tidyverse)
set.seed(1000557772)
alpha<-0.5
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta0 falls within the uppe
B0tinterval<-array(0,dim=c(newsim,3))
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta1 falls within the uppe
B1tinterval<-array(0,dim=c(newsim,3))
for (i in 1:newsim){
  et<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Yt<-beta0+beta1*X+et
  fitmodelt<- lm(Yt~X)
  yhatt <- predict(fitmodelt, type = "response")
  err_vart <- sum((Yt-yhatt)^2)/(nsample - 2)
  sumxt <- sum(X); xbart <- sumxt/nsample
  sumyt <- sum(Yt); ybart <- sumyt/nsample
  sumxt2 <- sum(X^2); SXXt <- (sumxt2 - nsample*xbart^2)
  sumxyt <- sum(X*Yt)
  beta1_hatt <- (sumxyt - nsample*xbart*ybart)/(SXXt)
  beta0_hatt<-ybart-(beta1_hatt*xbart)

#for Beta0_hat

  B0ll <- beta0_hatt - qt(1 - (alpha/2), df = nsample - 2)*((sqrt(err_vart)*(sqrt((1/nsample+((xbart^2),
  B0ul <- beta0_hatt + qt(1 - (alpha/2), df = nsample - 2)*((sqrt(err_vart)*(sqrt((1/nsample+((xbart^2),
  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  B0ininterval<-ifelse(between(beta0,B0ll,B0ul),1,0)
  B0tinterval[i,]<-c(B0ul,B0ll,B0ininterval)

#for Beta1_hat
  B1ll <- beta1_hatt - qt(1 - (alpha/2), df = nsample - 2)*(sqrt(err_vart)/(sqrt(SXXt)))
  B1ul <- beta1_hatt + qt(1 - (alpha/2), df = nsample - 2)*(sqrt(err_vart)/(sqrt(SXXt)))
  B1ininterval<-ifelse(between(beta1,B1ll,B1ul),1,0)
  B1tinterval[i,]<-c(B1ul,B1ll,B1ininterval)

  }

# turn array into dataframes for easily summing selective columns. 3 rows, n.sim columns
B1ttable<-as.data.frame(t(B1tinterval), row.names = c("upperlimit","lowerlimit","B1ininterval"))
B0ttable<-as.data.frame(t(B0tinterval), row.names = c("upperlimit","lowerlimit","B0ininterval"))

print("True Value of Beta0")
beta0

b1prop<-rowSums(B1ttable["B1ininterval",c(1:newsim)])
b0prop<-rowSums(B0ttable["B0ininterval",c(1:newsim)])

print("Number of simulations out of 10,000 that beta0 was found within t-confidence interval")
b0prop
print("Percentage beta0 was found within t-confidence intervals")
(b0prop/newsim)*100
print("10 rows from the B0 interval table")
head(t(B0ttable),10)
```

```
print("True Value of Beta1")
beta1
print("Number of simulations out of 10,000 that beta1 was found within t-confidence intervals")
b1prop
print("Percentage for beta1 found within t-confidence intervals")
(b1prop/newsim)*100
print("10 rows from the B1 interval table")
head(t(B1ttable),10)
```

```
# z confidence intervals
#for Beta0_hat, where sig2 is the population error variance
#install.packages("distributions")
set.seed(1000557774)
library(distributions3) #library for probability distributions
Z<-Normal(0,1)

#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta0 falls within the uppe
z0tinterval<-array(0,dim=c(newsim,3))
#an array for 3 inputs: upperlimit, lowerlimit and a column indicating 1 if beta1 falls within the uppe
z1tinterval<-array(0,dim=c(newsim,3))

for (i in 1:newsim){
  ez<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Yz<-beta0+beta1*X+ez
  fitmodelz<- lm(Yz~X)
  yhatz <- predict(fitmodelz, type = "response")
  err_varz <- sum((Yz-yhatz)^2)/(nsample - 2)
  sumxz <- sum(X); xbarz <- sumxz/nsample
  sumyz <- sum(Yz); ybarz <- sumyz/nsample
  sumxz2 <- sum(X^2); SXXz <- (sumxz2 - nsample*xbarz^2)
  sumxyz <- sum(X*Yz)
  beta1_hatz <- (sumxyz - nsample*xbarz*ybarz)/(SXXz)
  beta0_hatz<-ybarz-(beta1_hatz*xbarz)

#for Beta0_hat, eqn from lecture 3 slide 19
  # lower limit
  z0ll <- beta0_hatz - quantile(Z, 1-(alpha/2))*((sqrt(sig2)*(sqrt((1/nsample)+((xbarz^2)/SXXz)))))
  # upper limit
  z0ul <- beta0_hatz + quantile(Z, 1-(alpha/2))*((sqrt(sig2)*(sqrt((1/nsample)+((xbarz^2)/SXXz)))))


  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
  z0ininterval<-ifelse(between(beta0,z0ll,z0ul),1,0)
  # insert all three values into array created above
  z0tinterval[i,]<-c(z0ul,z0ll,B0ininterval)

#for Beta1_hat, eqn from lecture 3 slide 8
  #lower limit
  z1ll <- beta1_hatz - quantile(Z, 1 - (alpha/2))*(sqrt(sig2)/(sqrt(SXXz)))
  #upper limit
  z1ul <- beta1_hatz + quantile(Z, 1 - (alpha/2))*(sqrt(sig2)/(sqrt(SXXz)))

  #beta1 is greater than or equal to z1ll(lowerlimit) and less than or equal the z1ul(upperlimit)
```

```
  z1ininterval<-ifelse(between(beta1,z1ll,z1ul),1,0)
  z1tinterval[i,]<-c(z1ul,z1ll,z1ininterval)

  }
# turn array into dataframes for easily summing selective columns
z1ttable<-as.data.frame(t(z1tinterval), row.names = c("upperlimit","lowerlimit","B1ininterval"))
z0ttable<-as.data.frame(t(z0tinterval), row.names = c("upperlimit","lowerlimit","B0ininterval"))
print("True Value of Beta0")
beta0

print("Number of simulations out of 10,000 that beta0 fell into t-confidence interval")
z0prop<-rowSums(z0ttable["B0ininterval",c(1:newsim)])
z0prop
print("Percentage beta0 fell into z-confidence interval")
(z0prop/newsim)*100
print("10 rows from the B0 interval table")
head(t(z0ttable),10)
print("True Value of Beta1")
beta1
print("Number of simulations out of 10,000 that beta1 fell into z-confidence interval")
z1prop<-rowSums(z1ttable["B1ininterval",c(1:newsim)])
z1prop
print("Percentage beta1 fell into z-confidence interval")
(z1prop/newsim)*100
print("10 rows from the B1 interval table")
head(t(z1ttable),10)
```

Part F: For steps (a)-(d) the sample size was fixed at 5.

Sample 10 Part A

```
## Simulation
set.seed(1000557774)

## Mutiple simulation may require loops
nsample_10 <- 10 # sample size
n.sim<- 100 #number of simulations

#Simulate the predictor variable
X_10<- rnorm(nsample_10,mean=0, sd=sqrt(sigX))
```

Sample 10 Part B

```
set.seed(1000557774)
estCoeff_10<- array(0,dim=c(n.sim,2))

for (i in 1:n.sim){
  e_10<-rnorm(nsample_10,mean=0, sd=sqrt(sig2))
  Y_10<-beta0+beta1*X_10+e_10
```

```
  fitmodel_10<- lm(Y_10~X_10)
  estCoeff_10[i,]<-coefficients(fitmodel_10)
}
beta0_hat_10<-estCoeff_10[,1]
beta1_hat_10<-estCoeff_10[,2]


print("Estimated Mean of beta0 and beta1 respectively")
meanest_10<-colMeans(estCoeff_10)
meanest_10
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Sample 10 Part C

```
hist(estCoeff_10[,1], main='Histogram of Beta0_hat estimates for 10 samples')
```

```
hist(estCoeff_10[,2], main='Histogram of Beta1_hat estimates for 10 samples')
```

Sample 10 Part D

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_10<-var(estCoeff_10[,1])
#Beta1_hat
varbeta1_10<-var(estCoeff_10[,2])

#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_10 <- sum(X_10); xbar_10 <- sumx_10/nsample_10
sumy_10 <- sum(Y_10); ybar_10 <- sumy_10/nsample_10
sumx2_10 <- sum(X_10^2); SXX_10 <- (sumx2_10 - nsample_10*xbar_10^2)
sumxy_10 <- sum(X_10*Y_10)
bet1_hat_10 <- (sumxy_10 - nsample_10*xbar_10*ybar_10)/(SXX_10)
bet0_hat_10<-ybar_10-(bet1_hat_10*xbar_10)

#variance of b1
varb1_10<- sig2/SXX_10
#variance of b0
varb0_10<-sig2*((1/nsample_10)+((xbar_10^2)/SXX_10))

print("True Variance of Beta0")
varb0_10
print("Estimated Variance of Beta0")
varbeta0_10 #calculated with var function
print("True Variance of Beta1")
varb1_10
```

```r
print("Estimated Variance of Beta1")
varbeta1_10 #calculated with var function
```

Sample 25 Part A

```r
## Simulation
set.seed(1000557774)

## Multiple simulation may require loops
nsample_25 <- 25 # sample size

#Simulate the predictor variable
X_25<- rnorm(nsample_25,mean=0, sd=sqrt(sigX))
```

Sample 25 Part B

```r
set.seed(1000557774)
estCoeff_25<- array(0,dim=c(n.sim,2))

for (i in 1:n.sim){
  e_25<-rnorm(nsample_25,mean=0, sd=sqrt(sig2))
  Y_25<-beta0+beta1*X_25+e_25
  fitmodel_25<- lm(Y_25~X_25)
  estCoeff_25[i,]<-coefficients(fitmodel_25)
}
beta0_hat_25<-estCoeff_25[,1]
beta1_hat_25<-estCoeff_25[,2]

print("Estimated Mean of beta0 and beta1 respectively")
meanest_25<-colMeans(estCoeff_25)
meanest_25
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Sample 25 Part C

```r
hist(estCoeff_25[,1], main='Histogram of Beta0_hat estimates for 25 samples')
```

```r
hist(estCoeff_25[,2], main='Histogram of Beta1_hat estimates for 25 samples')
```

Sample 25 Part D

```r
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_25<-var(estCoeff_25[,1])
#Beta1_hat
varbeta1_25<-var(estCoeff_25[,2])
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_25 <- sum(X_25); xbar_25 <- sumx_25/nsample_25
sumy_25 <- sum(Y_25); ybar_25 <- sumy_25/nsample_25
sumx2_25 <- sum(X_25^2); SXX_25 <- (sumx2_25 - nsample_25*xbar_25^2)
sumxy_25 <- sum(X_25*Y_25)
bet1_hat_25 <- (sumxy_25 - nsample_25*xbar_25*ybar_25)/(SXX_25)
bet0_hat_25<-ybar_25-(bet1_hat_25*xbar_25)

#variance of b1
varb1_25<- sig2/SXX_25
#variance of b0
varb0_25<-sig2*((1/nsample_25)+((xbar_25^2)/SXX_25))

print("True Variance of Beta0")
varb0_25
print("Estimated Variance of Beta0")
varbeta0_25 #calculated with var function
print("True Variance of Beta1")
varb1_25
print("Estimated Variance of Beta1")
varbeta1_25 #calculated with var function
```

Sample 50 Part A

```r
## Simulation
set.seed(1000557774)

## Mutiple simulation may require loops
nsample_50 <- 50 # sample size
#Simulate the predictor variable
X_50<- rnorm(nsample_50,mean=0, sd=sqrt(sigX))
```

Sample 50 Part B

```r
set.seed(1000557774)
estCoeff_50<- array(0,dim=c(n.sim,2))

for (i in 1:n.sim){
  e_50<-rnorm(nsample_50,mean=0, sd=sqrt(sig2))
  Y_50<-beta0+beta1*X_50+e_50
  fitmodel_50<- lm(Y_50~X_50)
  estCoeff_50[i,]<-coefficients(fitmodel_50)
}
```

```
beta0_hat<-estCoeff_50[,1]
beta1_hat<-estCoeff_50[,2]

print("Estimated Mean of beta0 and beta1 respectively")
meanest_50<-colMeans(estCoeff_50)
meanest_50
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Sample 50 Part C

```
hist(estCoeff_50[,1], main='Histogram of Beta0_hat estimates')
```

```
hist(estCoeff_50[,2], main='Histogram of Beta1_hat estimates')
```

Sample 50 Part D

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_50<-var(estCoeff_50[,1])
#Beta1_hat
varbeta1_50<-var(estCoeff_50[,2])
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_50 <- sum(X_50); xbar_50 <- sumx_50/nsample_50
sumy_50 <- sum(Y_50); ybar_50 <- sumy_50/nsample_50
sumx2_50 <- sum(X_50^2); SXX_50 <- (sumx2_50 - nsample_50*xbar_50^2)
sumxy_50 <- sum(X_50*Y_50)

#variance of b1
varb1_50<- sig2/SXX_50
#variance of b0
varb0_50<-sig2*((1/nsample_50)+((xbar_50^2)/SXX_50))

print("True Variance of Beta0")
varb0_50
print("Estimated Variance of Beta0")
varbeta0_50 #calculated with var function
print("True Variance of Beta1")
varb1_50
print("Estimated Variance of Beta1")
varbeta1_50 #calculated with var function
```

Sample 100 Part A

```
## Simulation
set.seed(1000557774)
## Mutiple simulation may require loops
nsample_100 <- 100 # sample size
#Simulate the predictor variable
X_100<- rnorm(nsample_100,mean=0, sd=sqrt(sigX))
```

Sample 100 Part B

```
set.seed(1000557774)
estCoeff_100<- array(0,dim=c(n.sim,2))

for (i in 1:n.sim){
  e_100<-rnorm(nsample,mean=0, sd=sqrt(sig2))
  Y_100<-beta0+beta1*X_100+e_100
  fitmodel_100<- lm(Y_100~X_100)
  estCoeff_100[i,]<-coefficients(fitmodel_100)
}
beta0_hat_100<-estCoeff_100[,1]
beta1_hat_100<-estCoeff_100[,2]

print("Estimated Mean of beta0 and beta1 respectively")
meanest100<-colMeans(estCoeff_100)
meanest100
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Sample 100 Part C

```
hist(estCoeff_100[,1], main='Histogram of Beta0_hat estimates')
```

```
hist(estCoeff_100[,2], main='Histogram of Beta1_hat estimates')
```

Sample 100 Part D

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_100<-var(estCoeff_100[,1])
#Beta1_hat
varbeta1_100<-var(estCoeff_100[,2])
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_100 <- sum(X_100); xbar_100 <- sumx_100/nsample_100
sumy_100 <- sum(Y_100); ybar_100 <- sumy_100/nsample_100
sumx2_100 <- sum(X_100^2); SXX_100 <- (sumx2_100 - nsample_100*xbar_100^2)
```

```r
sumxy_100 <- sum(X_100*Y_100)

#variance of b1
varb1_100<- sig2/SXX_100
#variance of b0
varb0_100<-sig2*((1/nsample_100)+((xbar_100^2)/SXX_100))

print("True Variance of Beta0")
varb0_100
print("Estimated Variance of Beta0")
varbeta0_100 #calculated with var function
print("True Variance of Beta1")
varb1_100
print("Estimated Variance of Beta1")
varbeta1_100 #calculated with var function
```

Sample 1000 Part A

```r
## Simulation
set.seed(1000557774)

## Mutiple simulation may require loops
nsample_1000 <- 1000 # sample size

#Simulate the predictor variable
X_1000<- rnorm(nsample_1000,mean=0, sd=sqrt(sigX))
```

Sample 1000 Part B

```r
set.seed(1000557774)
estCoeff_1000<- array(0,dim=c(n.sim,2))

for (i in 1:n.sim){
  e_1000<-rnorm(nsample_1000,mean=0, sd=sqrt(sig2))
  Y_1000<-beta0+beta1*X_1000+e_1000
  fitmodel_1000<- lm(Y_1000~X_1000)
  estCoeff_1000[i,]<-coefficients(fitmodel_1000)
}
beta0_hat_100<-estCoeff_1000[,1]
beta1_hat_100<-estCoeff_1000[,2]

print("Estimated Mean of beta0 and beta1 respectively")
meanest_1000<-colMeans(estCoeff_1000)
meanest_1000
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Sample 1000 Part C

```
hist(estCoeff_1000[,1], main='Histogram of Beta0_hat estimates')
```

```
hist(estCoeff_1000[,2], main='Histogram of Beta1_hat estimates')
```

Sample 1000 Part D

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_1000<-var(estCoeff_1000[,1])
#Beta1_hat
varbeta1_1000<-var(estCoeff_1000[,2])
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_1000 <- sum(X_1000); xbar_1000 <- sumx_1000/nsample_1000
sumy_1000 <- sum(Y_1000); ybar_1000 <- sumy_1000/nsample_1000
sumx2_1000 <- sum(X_1000^2); SXX_1000 <- (sumx2_1000 - nsample_1000*xbar_1000^2)
sumxy_1000 <- sum(X_1000*Y_1000)
bet1_hat_1000 <- (sumxy_1000 - nsample_1000*xbar_1000*ybar_1000)/(SXX_1000)
bet0_hat_1000<-ybar_1000-(bet1_hat_1000*xbar_1000)

#variance of b1
varb1_1000<- sig2/SXX_1000
#variance of b0
varb0_1000<-sig2*((1/nsample_1000)+((xbar_1000^2)/SXX_1000))

print("True Beta0")
varb0_1000
print("Estimated Beta0")
varbeta0_1000 #calculated with var function

print("True Beta1")
varb1_1000
print("Estimated Beta1")
varbeta1_1000 #calculated with var function
```

Part G: Choose the largest sample size used in step 6.

With error variance of 2

```
#changed error variances and samples
set.seed(1000557774)
ev<-2
#Simulate the predictor variable
X_f<- rnorm(nsample_1000,mean=0, sd=sqrt(sigX))
```

Step2 (Part B)

```
## [1] "Estimated Mean of beta0 and beta1 respectively"
```

```
## [1] -0.6384914  2.4485352
```

```
## [1] "True Beta0"
```

```
## [1] -0.6485048
```

```
## [1] "True Beta1"
```

```
## [1] 2.285801
```

Step 3 (Part C)

```
hist(estCoeff_f[,1], main='Histogram of Beta0_hat estimates')
```

```
hist(estCoeff_f[,2], main='Histogram of Beta1_hat estimates')
```

This chart no longer looks like a histogram and therefore does not have a describable shape. It appears as if majority of the estimated values fall between 2 and 2.5 with a single outlier value in the 5-5.5 region

Step 4 (Part D)

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_f<-var(estCoeff_f[,1])
#Beta1_hat
varbeta1_f<-var(estCoeff_f[,2])
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_f <- sum(X_f); xbar_f <- sumx_f/nsample_1000
sumy_f <- sum(Y_f); ybar_f <- sumy_f/nsample_1000
sumx2_f <- sum(X_f^2); SXX_f <- (sumx2_f - nsample_1000*xbar_f^2)
sumxy_f <- sum(X_f*Y_f)
beta1_hat_f <- (sumxy_f - nsample_1000*xbar_f*ybar_f)/(SXX_f)
beta0_hat<-ybar_f-(beta1_hat_f*xbar_f)

#variance of b1
varb1_f<- sig2/SXX_f
#variance of b0
```

```
varb0_f<-sig2*((1/nsample_1000)+((xbar_f^2)/SXX_f))

print("True Variance of Beta0")
varb0_f
print("Estimated Variance of Beta0")
varbeta0_f #calculated with var function
print("True Variance of Beta1")
varb1_f
print("Estimated Variance of Beta1")
varbeta1_f #calculated with var function
```

with error variance of 100

```
#changed error variances and samples
set.seed(1000557774)
ev<-100
#Simulate the predictor variable
X_f<- rnorm(nsample_1000,mean=0, sd=sqrt(sigX))
```

Step2 (Part B)

```
set.seed(1000557774)
estCoeff_f<- array(0,dim=c(n.sim,2))
#use nsample_1000 from part f
for (i in 1:n.sim){
  e_f<-rnorm(nsample_1000,mean=0, sd=sqrt(ev))
  Y_f<-beta0+beta1*X_f+e_f
  fitmodel_f<- lm(Y_f~X_f)
  estCoeff_f[i,]<-coefficients(fitmodel_f)
}
beta0_hat_f<-estCoeff_f[,1]
beta1_hat_f<-estCoeff_f[,2]

print("Estimated Mean of beta0 and beta1 respectively")
meanest_f<-colMeans(estCoeff_f)
meanest_f
print("True Beta0")
beta0
print("True Beta1")
beta1
```

Step 3 (Part C)

```
hist(estCoeff_f[,1], main='Histogram of Beta0_hat estimates')
```

```
hist(estCoeff_f[,2], main='Histogram of Beta1_hat estimates')
```

Step 4 (Part D)

```
# sample variance of beta1hat and beta0hat with var() function
#Beta0_hat
set.seed(1000557774)
varbeta0_f<-var(estCoeff_f[,1])
#Beta1_hat
varbeta1_f<-var(estCoeff_f[,2])
#calculating beta1 and beta0 hats without LM function, least square estimates
sumx_f <- sum(X_f); xbar_f <- sumx_f/nsample_1000
sumy_f <- sum(Y_f); ybar_f <- sumy_f/nsample_1000
sumx2_f <- sum(X_f^2); SXX_f <- (sumx2_f - nsample_1000*xbar_f^2)
sumxy_f <- sum(X_f*Y_f)
beta1_hat_f <- (sumxy_f - nsample_1000*xbar_f*ybar_f)/(SXX_f)
beta0_hat<-ybar_f-(beta1_hat_f*xbar_f)

#variance of b1
varb1_f<- sig2/SXX_f
#variance of b0
varb0_f<-sig2*((1/nsample_1000)+((xbar_f^2)/SXX_f))

print("True Variance of Beta0")
varb0_f
print("Estimated Variance of Beta0")
varbeta0_f #calculated with var function
print("True Variance of Beta1")
varb1_f
print("Estimated Variance of Beta1")
varbeta1_f #calculated with var function
```

Task 2

```
library(MASS)
## Simulation for correlated predictors ##
set.seed(1000557774)
nsample<-10; n.sim<-100
sig2<- rchisq(1,df=1)## the true error variance
bet<-matrix(c(rnorm(3,0,1),0))## 4 values of beta that is beta0, beta1, beta2, beta3=0
muvec<-rnorm(3,0,1)#mean, "mew"
sigmat<-diag(rchisq(3,df=4))
X<-mvrnorm(nsample, mu=muvec, Sigma=sigmat)
Xmat<-cbind(1,X)
```

Step A

```
set.seed(1000557774)
##Simulate the response##
n.sim<-100
```

```
#is this how the single regression is supposed to be done?
bets1<-matrix(NA,ncol=2,nrow=n.sim)
bets2<-matrix(NA,ncol=2,nrow=n.sim)
bets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  model1<-lm(Y~X[,1])
  model2<-lm(Y~X[,2])
  model3<-lm(Y~X[,3])
  bets1[i,]<-coef(model1)
  bets2[i,]<-coef(model2)
  bets3[i,]<-coef(model3)
}
mb1<-mean(bets1[,2])
mb2<-mean(bets2[,2])
mb3<-mean(bets3[,2])
var1<-var(bets1[,2])
var2<-var(bets2[,2])
var3<-var(bets3[,2])

#True variance of b1
sumx1 <- sum(X[,1]); xbar1 <- sumx1/nsample
sumy <- sum(Y); ybar <- sumy/nsample
sumx2_1 <- sum((X[,1])^2); SXX1 <- (sumx2_1 - nsample*xbar1^2)
varb1slr<- sig2/SXX1
#True variance of b2
sumx2 <- sum(X[,2]); xbar2 <- sumx2/nsample
sumy <- sum(Y); ybar <- sumy/nsample
sumx2_2 <- sum((X[,2])^2); SXX2 <- (sumx2_2 - nsample*xbar2^2)
varb2slr<- sig2/SXX2
#True variance of b2
sumx3 <- sum(X[,3]); xbar3 <- sumx3/nsample
sumy <- sum(Y); ybar <- sumy/nsample
sumx2_3 <- sum(X^2); SXX3 <- (sumx2_3 - nsample*xbar3^2)
varb3slr<- sig2/SXX3

#calculate variance as well
print("Means of Beta1, Beta2, and Beta3 respectively from SLR")
mb1;mb2;mb3
print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet
print("Variances of Beta1, Beta2, and Beta3 respectively from SLR")
var1;var2;var3
print("True Variances of Beta1, Beta2, Beta3")
varb1slr;varb2slr;varb3slr
```

Step B

```
set.seed(1000557774)
mbets<-matrix(NA,ncol=length(bet),nrow=n.sim)
```

```
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  mmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  mbets[i,]<-coef(mmodel)#array to store estimates of betas
}

mlrbeta1<-mbets[,2]
mlrbeta2<-mbets[,3]
mlrbeta3<-mbets[,4]

#var(mbets) variance covariance matrix returned, variances are present in the diagonals
print("Estimates of beta1, beta2, beta3 respectively for MLR")
mean(mbets[,2])
mean(mbets[,3])
mean(mbets[,4])

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet
print("Estimated variances of beta1, beta2, beta3 respectively for MLR")
var(mbets[,2])
var(mbets[,3])
var(mbets[,4])

print("True Variances of Beta1, Beta2, Beta3")
varb1slr;varb2slr;varb3slr
```

Step C

Simulation with correlation coefficient of 0.2 (r=0.2)

```
## the correlation##
set.seed(1000557774)
r<-0.2
sigmat2<-sigmat
sigmat2[1,2]<-sigmat2[2,1]<-r*(sqrt(sigmat2[1,1])*sqrt(sigmat2[2,2]))

## Simulation for Categorical Variables with Interaction
set.seed(1000557774)
X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat2)
print("Correlation between X1 and X2 variables")
cor(X[,1], X[,2])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
```

```
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])

cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.2**********")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Simulation with correlation value of 0.5 (r=0.5)

```
set.seed(1000557774)
## the correlation##
r<-0.5
sigmat2<-sigmat
sigmat2[1,2]<-sigmat2[2,1]<-r*(sqrt(sigmat2[1,1])*sqrt(sigmat2[2,2]))

## Simulation for Categorical Variables with Interaction
X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat2)
print("Correlation between X1 and X2 variables")
cor(X[,1], X[,2])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.5***************")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Simulation with correlation value of 0.7 (r=0.7)

```
set.seed(1000557774)
## the correlation##
r<-0.7
sigmat2<-sigmat
sigmat2[1,2]<-sigmat2[2,1]<-r*(sqrt(sigmat2[1,1])*sqrt(sigmat2[2,2]))

## Simulation for Categorical Variables with Interaction
X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat2)
print("Correlation between X1 and X2 variables")
cor(X[,1], X[,2])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])
```

```
cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.7***************")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Simulation with correlation value of 0.8 (r=0.8)

```
## the correlation##
set.seed(1000557774)
r<-0.8
sigmat2<-sigmat
sigmat2[1,2]<-sigmat2[2,1]<-r*(sqrt(sigmat2[1,1])*sqrt(sigmat2[2,2]))

## Simulation for Categorical Variables with Interaction
```

```
X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat2)
print("Correlation between X1 and X2 variables")
cor(X[,1], X[,2])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.8***************")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])
```

```
print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Step D

Simulation with correlation value of -0.5 (r=-0.5)

```
set.seed(1000557774)
r<-(-0.5)
sigmat3<-sigmat
sigmat3[1,3]<-sigmat3[3,1]<-r*sqrt(sigmat3[1,1])*sqrt(sigmat3[3,3]) #correlating beta1 and beta3

X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat3)
print("Correlation between X1 and X3 variables")
cor(X[,1], X[,3])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.4**********")


print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
```

```
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Simulation with correlation value of 0.5 (r=0.5)

```
set.seed(1000557774)
r<-0.5
sigmat3<-sigmat
sigmat3[1,3]<-sigmat3[3,1]<-r*sqrt(sigmat3[1,1])*sqrt(sigmat3[3,3]) #correlating beta1 and beta3

X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat3)
print("Correlation between X1 and X3 variables")
cor(X[,1], X[,3])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
```

```
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.5***************")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
# correlated multiple
set.seed(1000557774)
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Simulation with correlation value of 0.7 (r=0.7); written summary of findings for coefficients 0.7-0.95

```
set.seed(1000557774)
r<-0.7
sigmat3<-sigmat
sigmat3[1,3]<-sigmat3[3,1]<-r*sqrt(sigmat3[1,1])*sqrt(sigmat3[3,3]) #correlating beta1 and beta3
X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat3)
print("Correlation between X1 and X3 variables")
cor(X[,1], X[,3])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.7**********")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)
```

```
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 for MLR")
modsum$coefficients
```

Simulation with correlation value of 0.8 (r=0.8)

```
set.seed(1000557774)
r<-0.8
sigmat3<-sigmat
sigmat3[1,3]<-sigmat3[3,1]<-r*sqrt(sigmat3[1,1])*sqrt(sigmat3[3,3]) #correlating beta1 and beta3

X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat3)
print("Correlation between X1 and X3 variables")
cor(X[,1], X[,3])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.8**********")
```

```
print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```

Simulation with correlation value of 0.9 (r=0.9)

```
set.seed(1000557774)
r<-0.9
sigmat3<-sigmat
sigmat3[1,3]<-sigmat3[3,1]<-r*sqrt(sigmat3[1,1])*sqrt(sigmat3[3,3]) #correlating beta1 and beta3

X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat3)
print("Correlation between X1 and X3 variables")
cor(X[,1], X[,3])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
```

```
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.9***************")

print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients

print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
## [1] "Estimated Means for Correlated B1 for MLR"
```

```
## [1] 0.6340355
```

```
## [1] "Estimated Means for Correlated B2 for MLR"
```

```
## [1] -0.01566121
```

```
## [1] "True value of betas"
```

```
##             [,1]
## [1,] 0.116756536
## [2,] 0.630730988
## [3,] 0.005667657
## [4,] 0.000000000
```

```
## [1] "Standard errors of Betas 1,2, and 3 from MLR"
```

```
##                Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.53473618  0.4315077 -1.2392274 0.2615349
## X[, 1]       0.63459281  0.4644280  1.3663965 0.2208052
## X[, 2]       0.23901622  0.1296377  1.8437243 0.1147842
## X[, 3]      -0.08316856  0.5101825 -0.1630173 0.8758569
```

Simulation with correlation value of 0.95 (r=0.95)

```
set.seed(1000557774)
r<-0.95
sigmat3<-sigmat
sigmat3[1,3]<-sigmat3[3,1]<-r*sqrt(sigmat3[1,1])*sqrt(sigmat3[3,3]) #correlating beta1 and beta3

X<-mvrnorm(nsample,mu=muvec,Sigma=sigmat3)
print("Correlation between X1 and X3 variables")
cor(X[,1], X[,3])
Xmat<-cbind(1,X)

# single regression
cbets1<-matrix(NA,ncol=2,nrow=n.sim)
cbets2<-matrix(NA,ncol=2,nrow=n.sim)
cbets3<-matrix(NA,ncol=2,nrow=n.sim)
for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmodel1<-lm(Y~X[,1])
  cmodel2<-lm(Y~X[,2])
  cmodel3<-lm(Y~X[,3])
  cbets1[i,]<-coef(cmodel1)
  cbets2[i,]<-coef(cmodel2)
  cbets3[i,]<-coef(cmodel3)
}
cmb1<-mean(cbets1[,2])
cmb2<-mean(cbets2[,2])


cmodsum1<-summary(cmodel1)
cmodsum2<-summary(cmodel2)

print("**********Estimates and Standard Errors when r=0.95*****************")


print("Estimated Means for Correlated B1 for SLR")
cmb1

print("Estimated Means for Correlated B2 for SLR")
cmb2

print("True Values of Beta0, Beta1, Beta2 and Beta3 respectively")
bet

print("Standard errors of Beta1")
cmodsum1$coefficients
```

```
print("Standard errors of Beta2")
cmodsum2$coefficients
```

```
set.seed(1000557774)
# correlated multiple regression
cmbets<-matrix(NA,ncol=length(bet),nrow=n.sim)

for (i in 1:n.sim){
  Y<- Xmat%*%bet+rnorm(nsample,0,sqrt(sig2))
  cmmodel<-lm(Y~X[,1]+X[,2]+X[,3])
  cmbets[i,]<-coef(cmmodel)
}
print("Estimated Means for Correlated B1 for MLR")
mean(cmbets[,2])

print("Estimated Means for Correlated B2 for MLR")
mean(cmbets[,3])

print("True value of betas")
bet

modsum<-summary(cmmodel)
print("Standard errors of Betas 1,2, and 3 from MLR")
modsum$coefficients
```