Problem Set 5

Economemtrics

3a.

```r
b <- matrix(c(1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1), nrow = 3, ncol = 3, byrow = TRUE)
print(b)
```

```
##      [,1] [,2] [,3]
## [1,]  1.0  0.5  0.5
## [2,]  0.5  1.0  0.5
## [3,]  0.5  0.5  1.0
```

3b.

```r
A <- eigen(b)
b_sqrt <- A$vectors %*% diag(sqrt(A$values)) %*% t(A$vectors)
print(b_sqrt %*% t(b_sqrt))
```

```
##      [,1] [,2] [,3]
## [1,]  1.0  0.5  0.5
## [2,]  0.5  1.0  0.5
## [3,]  0.5  0.5  1.0
```

3c.

```r
sample_size <- 1000
m <- 3
n <- 3

matrices <- list()
squared_matrices <- list()

for (i in 1:sample_size) {
  matrices[[i]] <- matrix(rnorm(m * n), nrow = m, ncol = n) %*% b_sqrt
  squared_matrices[[i]] <- matrices[[i]] %*% matrices[[i]]
}

print(Reduce("+", squared_matrices) / sample_size)
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0083058 0.5149988 0.5255522
```

```
## [2,] 0.5540523 1.1312695 0.6216438
## [3,] 0.4160094 0.5057503 1.0157482
```

It is close to the variance covariance matrix of b. This is because the variance covariance matrix of b is the expected value of the squared matrices. Which is the operation that was carried out when we take x'x/n.

3d.

```
y_matrices <- list()
beta_unprocessed <- list()

for (i in 1:sample_size) {
  y_matrices[[i]] <- matrix(rnorm(m * 1), nrow = m, ncol = 1)
  beta_unprocessed[[i]] <- solve(squared_matrices[[i]]) %*% matrices[[i]] %*% y_matrices
}

print(Reduce("+", beta_unprocessed) / sample_size)
```

```
##           [,1]
## [1,] -2.472855
## [2,]  4.065781
## [3,]  1.906288
```

3e.

```
I <- diag(3)
beta_hat_unprocessed <- list()

m_projection <- function(x) {
  inv_square <- solve(t(x) %*% x)
  return(I - (x %*% inv_square %*% t(x)))
}

matrix_projection <- function(x, m, y) {
  inv <- solve(t(x) %*% m %*% x)
  return(inv %*% t(x) %*% m %*% y)
}

for (i in 1:sample_size) {
  matrix <- matrices[[i]]
  x_1 <- matrix[, 1]
  x_2 <- matrix[, 2:3]

  m_2 <- m_projection(x_2)
  beta_hat_unprocessed[[i]] <- matrix_projection(x_1, m_2, y_matrices[[i]])
}
```

```r
print(Reduce("+", beta_hat_unprocessed) / sample_size)
```

```
##           [,1]
## [1,] -2.472855
```

3f.

```r
x_1_beta_unprocessed <- list()

for (i in 1:sample_size) {
  matrix <- matrices[[i]]
  x_1 <- matrix[, 1]
  squared_matrices <- x_1 %*% x_1

  x_1_beta_unprocessed[[i]] <- squared_matrices %*% x_1 %*% y_matrices[[i]]
}

print(Reduce("+", x_1_beta_unprocessed) / sample_size)
```

```
##           [,1]
## [1,] 0.2410196
```

```r
b <- matrix(c(1, 0.0, 0.0, 0.0, 1, 0.5, 0.0, 0.5, 1), nrow = 3, ncol = 3, byrow = TRUE)
A <- eigen(b)
b_sqrt <- A$vectors %*% diag(sqrt(A$values)) %*% t(A$vectors)

sample_size <- 1000
for (i in 1:sample_size) {
  matrices[[i]] <- matrix(rnorm(m * n), nrow = m, ncol = n) %*% b_sqrt
  y_matrices[[i]] <- matrix(rnorm(m * 1), nrow = m, ncol = 1)
  beta_unprocessed[[i]] <- solve((matrices[[i]] %*% matrices[[i]]), matrices[[i]] %*% y_
}

for (i in 1:sample_size) {
  matrix <- matrices[[i]]
  x_1 <- matrix[, 1]
  x_2 <- matrix[, 2:3]

  m_2 <- m_projection(x_2)
  beta_hat_unprocessed[[i]] <- matrix_projection(x_1, m_2, y_matrices[[i]])
}

print(Reduce("+", beta_unprocessed) / sample_size)
```

```
##           [,1]
## [1,] -1.400054
```

```
## [2,]  2.965809
## [3,] -2.394401
```

```
print(Reduce("+", beta_hat_unprocessed) / sample_size)
```

```
##           [,1]
## [1,] -1.400054
```

For the first b matrix, the beta_1 is different from beta_hat_1. This is because of the OLs omitting the effects of x_2 and x_3.

For the second b matrix, the beta_1 is the same as beta_hat_1. This is because there is no covariance between x_1 with x_2 and x_3, which means that there is no relation between the x_1 variables and the other two. Therefore, the OLS will not overestimate the effects of x_1 even when we omit the other two variables.

4a.

```
sample_size <- 100
x_1_and_2 <- list()

norm_transform <- function(x, ro = 0.9) {
  mu <- c(0, 0)
  sigma <- matrix(c(1, ro, ro, 1), nrow = 2, ncol = 2, byrow = TRUE)
  return(t(x) %*% chol(sigma) + mu)
}

for (i in 1:sample_size) {
  x_1_and_2[[i]] <- norm_transform(matrix(rnorm(2 * 1)))
}
```

4b.

```
mc_simulate <- function(x_1, x_2, sample_size = 100) {
  y <- list()

  for (i in 1:sample_size) {
    y[[i]] <- x_1[[i]] + x_2[[i]] + rnorm(1)
  }

  b_hat <- summary(lm(unlist(y) ~ unlist(x_1) + unlist(x_2)))$coefficients["unlist(x_1)
  b_tilde <- summary(lm(unlist(y) ~ unlist(x_1)))$coefficients["unlist(x_1)", c("Estima

  return(c(b_tilde[1] - b_hat[1], b_tilde[2] - b_hat[2]))
}

x_1 <- lapply(x_1_and_2, function(x) x[1])
x_2 <- lapply(x_1_and_2, function(x) x[2])
```

```r
print(mc_simulate(x_1, x_2))
```

```
##   Estimate Std. Error
##  0.7737578 -0.1519909
```

4c.

```r
experiment <- 1000

temp <- array()
coef_diff <- array()
se_diff <- array()

for (i in 1:experiment) {
  temp <- mc_simulate(x_1, x_2)
  coef_diff[i] <- temp[1]
  se_diff[i] <- temp[2]
}

print(mean(coef_diff))
```

```
## [1] 0.9276041
```

```r
print(mean(se_diff))
```

```
## [1] -0.1482568
```

Dropping the estimator will cause the bias to increase and the standard errors to decrease. This is because the estimator is now biased, as it is not taking into account the correlation between the two variables. The standard errors are lower because the model is now simpler, and therefore the standard errors are lower.

4d.

```r
for (i in 1:sample_size) {
  x_1_and_2[[i]] <- norm_transform(matrix(rnorm(2 * 1)), 0)
}

x_1 <- lapply(x_1_and_2, function(x) x[1])
x_2 <- lapply(x_1_and_2, function(x) x[2])

for (i in 1:experiment) {
  temp <- mc_simulate(x_1, x_2)
  coef_diff[i] <- temp[1]
  se_diff[i] <- temp[2]
}

print(mean(coef_diff))
```

```
## [1] -0.02801218
```

```r
print(mean(se_diff))
```

```
## [1] 0.0421717
```

The bias is lower for the second experiment, both in terms of its coefficents and standard errors. This is because the first experiment has a correlation between the two variables, which results in the model overestimating the effects of x1 as a movement in x1 wil result in a movement in x2.