MACT 4233 - Assignment 2

Code ▾

Omar Moustafa 900222400

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Question 1. Choose any data matrix X of dimension 5 × 3, then

Hide

```
x = matrix(c(3.2, 4.5, 2.1, 5.6, 3.9, 5.1, 6.8, 3.3, 7.2, 4.5, 7.3, 5.9, 6.5, 8.1, 5.7),
           nrow = 5, ncol = 3, byrow = FALSE)

print(x)
```

```
     [,1] [,2] [,3]
[1,]  3.2  5.1  7.3
[2,]  4.5  6.8  5.9
[3,]  2.1  3.3  6.5
[4,]  5.6  7.2  8.1
[5,]  3.9  4.5  5.7
```

a. Compute the Euclidean distance between every pair of X. Which pair of the rows are the most distant from each other?

Hide

```
# Compute the Euclidean distances between rows of matrix 'x'
d_rows = as.matrix(dist(x))
print("Euclidean Distance Between Rows:")
```

```
[1] "Euclidean Distance Between Rows:"
```

Hide

```
d_rows_rounded = round(d_rows, digits = 2)
print(d_rows_rounded)
```

```
     1    2    3    4    5
1 0.00 2.56 2.26 3.29 1.85
2 2.56 0.00 4.29 2.49 2.39
3 2.26 4.29 0.00 5.48 2.31
4 3.29 2.49 5.48 0.00 3.99
5 1.85 2.39 2.31 3.99 0.00
```

Hide

```
max_distance = max(d_rows)
row_indices = which(d_rows == max_distance, arr.ind = TRUE)
cat("The Two Most Distant Rows From Each Other Are:", row_indices[1, 1], "and", row_i
ndices[1, 2])
```

```
The Two Most Distant Rows From Each Other Are: 4 and 3
```

b. Compute the Euclidean distance between every columns of X. Which pair of the columns are the closest to each other?

Hide

```
# Compute the Euclidean distances between columns of matrix 'x'
d_cols = as.matrix(t(dist(x))) # Transpose 'x to compute the column-wise distances
print("Euclidean Distance Between Columns:")
```

```
[1] "Euclidean Distance Between Columns:"
```

Hide

```
d_cols_rounded = round(d_cols, digits = 2)
print(d_cols_rounded)
```

```
     1    2    3    4    5
1 0.00 2.56 2.26 3.29 1.85
2 2.56 0.00 4.29 2.49 2.39
3 2.26 4.29 0.00 5.48 2.31
4 3.29 2.49 5.48 0.00 3.99
5 1.85 2.39 2.31 3.99 0.00
```

Hide

```
min_distance = min(d_cols[d_cols > 0]) # Exclude zero distances so that it's the mini
mum distance between 2 different columns
col_indices = which(d_cols == min_distance, arr.ind = TRUE)
cat("The Two Closest Columns To Each Other Are:", col_indices[1, 1], "and", col_indic
es[1, 2])
```

```
The Two Closest Columns To Each Other Are: 5 and 1
```

c. Repeat the above two parts using X after standardizing its columns

Hide

```
s = sqrt(diag(var(x, use = "complete.obs")))
z = scale(x, center = FALSE, scale = s)
d = as.matrix(dist(z))
rd = round(d, digits = 2)
print("Euclidean Distance Between The Standardized Rows:")
```

```
[1] "Euclidean Distance Between The Standardized Rows:"
```

Hide

```
print(rd)
```

```
     1    2    3    4    5
1 0.00 2.01 1.60 2.37 1.73
2 2.01 0.00 2.89 2.37 1.50
3 1.60 2.89 0.00 3.92 1.75
4 2.37 2.37 3.92 0.00 3.19
5 1.73 1.50 1.75 3.19 0.00
```

Hide

```
max_distance_standardized = max(rd)
row_indices2 = which(rd == max_distance_standardized, arr.ind = TRUE)
cat("The Two Most Distant Rows From Each Other Are:", row_indices2[1, 1], "and", row_
indices2[1, 2])
```

```
The Two Most Distant Rows From Each Other Are: 4 and 3
```

Hide

```
d2 = as.matrix(dist(t(z))) # Transpose standardized matrix 'x' to compute the column-
wise distances
rd2 = round(d2, digits = 2)
print("Euclidean Distance Between The Standardized Columns:")
```

```
[1] "Euclidean Distance Between The Standardized Columns:"
```

Hide

```
print(rd2)
```

```
     1    2    3
1 0.00 1.19 8.75
2 1.19 0.00 7.87
3 8.75 7.87 0.00
```

Hide

```
min_distance_standardized = min(rd2[rd2 > 0]) # Exclude zero distances so that it's t
he minimum distance between 2 different columns
col_indices2 = which(rd2 == min_distance_standardized, arr.ind = TRUE)
cat("The Two Closest Columns To Each Other Are:", col_indices2[1, 1], "and", col_indi
ces2[1, 2])
```

```
The Two Closest Columns To Each Other Are: 2 and 1
```

d. Comment on the obtained results

1. Before standardization, the two most distant rows were Row 4 and Row 3, which did not change after standardization as Row 4 and Row 3 remained the two most distant ones. This lack of change indicates that the relative differences between the rows stayed the same, but standardization had re-scaled the distances.
2. Before standardization, the two closest rows were Row 5 and Row 1, but after standardization, this changed to where the two closest rows were Row 2 and Row 1. This particular change that the original scales of the columns influenced the raw distances, and after standardization, the relative similarity between columns shifted.
3. Therefore, standardization removed the influence of scale differences between the columns. The row-wise rankings of distances remained quite similar, but the column-wise relationships ended up changing significantly after adjusting for variance. In the standardized matrix, column 3 was much farther from columns 1 and 2, indicating that it had much larger variance in the original data.

Question 2. Consider any bivariate normal random vector X = (X1, X2)^T , that is X ~ N2(μ, Σ), other than the standard bivariate normal random vector. For each of the following 4 cases, specify μ and Σ, then sketch the ellipse resulting from cutting the density function with a plane parallel to the space spanned by the axes.

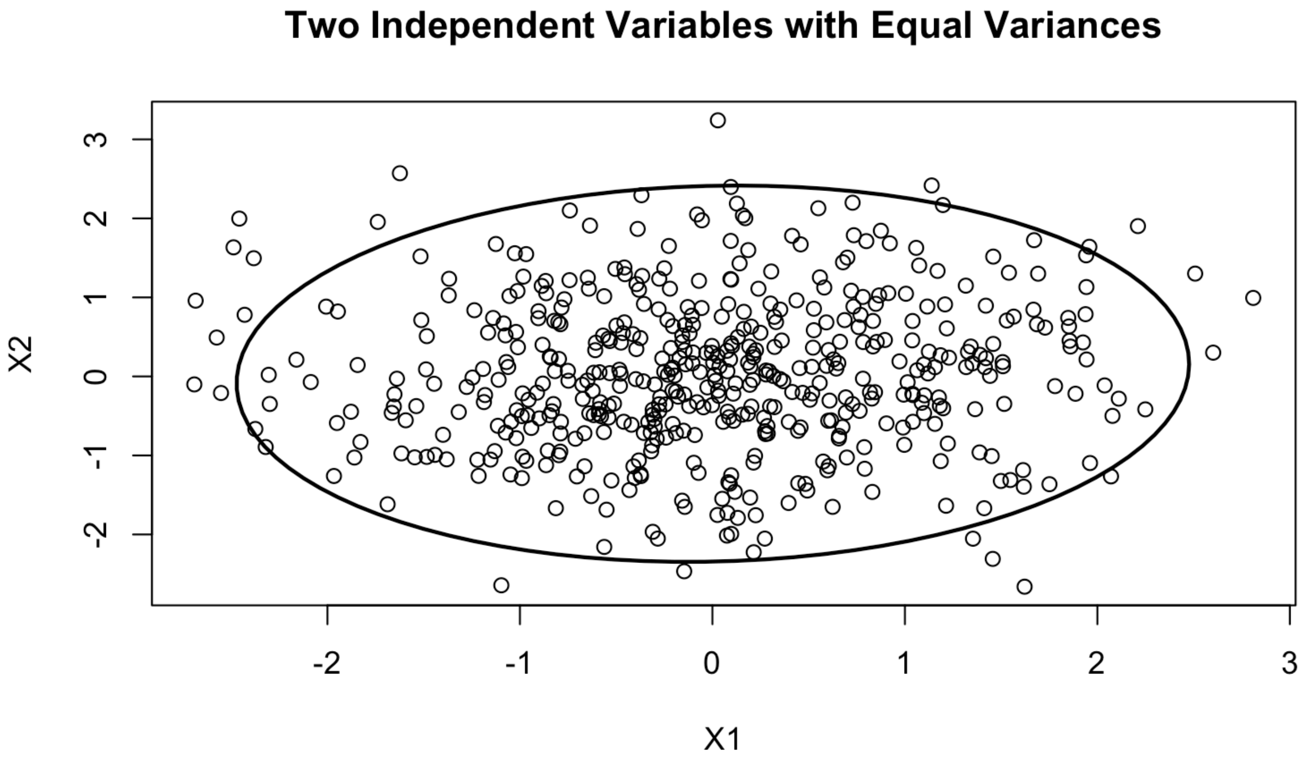
Hide

```
if(!require("MASS")) install.packages("MASS")
if(!require("ellipse")) install.packages("ellipse")
library(MASS)
library(ellipse)
```

a. The two variables are independent with equal variances.

Hide

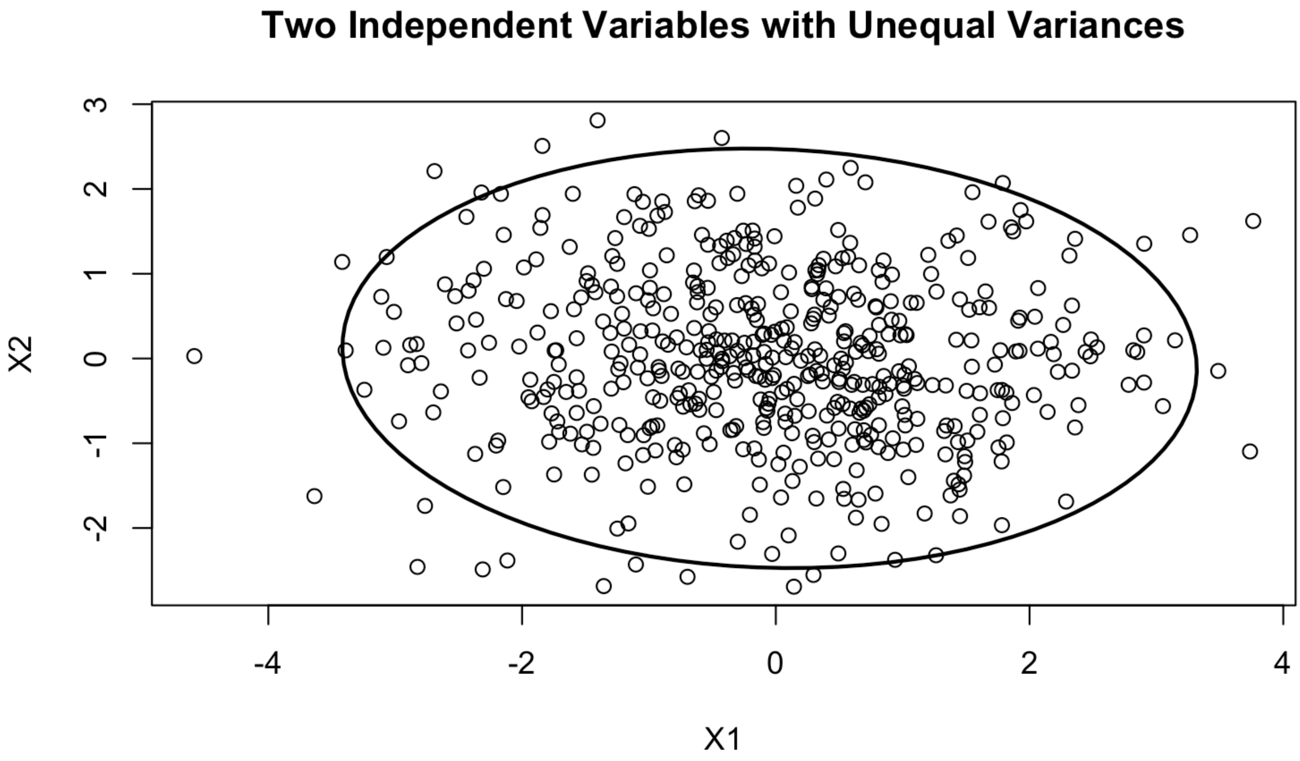
```
# Define the mean vector
mu = c(0,0)

# Define the covariance matrix (equal variances)
Simga = matrix(c(1, 0,
                 0, 1), nrow = 2)

# Generate a bivariate normal sample
set.seed(123)
my_x = mvrnorm(n = 500, mu = mu, Sigma = Simga)

# Compute the mean and covariance of the sample
sample_mean = colMeans(my_x)

sample_cov = var(my_x)

plot(my_x, main = "Two Independent Variables with Equal Variances", xlab = "X1", ylab
= "X2")

# Add an ellipse to the plot
ellipse_data = ellipse(sample_cov, centre = sample_mean, level = 0.95)
lines(ellipse_data, lwd = 2)
```



b. The two variables are independent with unequal variances.

Hide

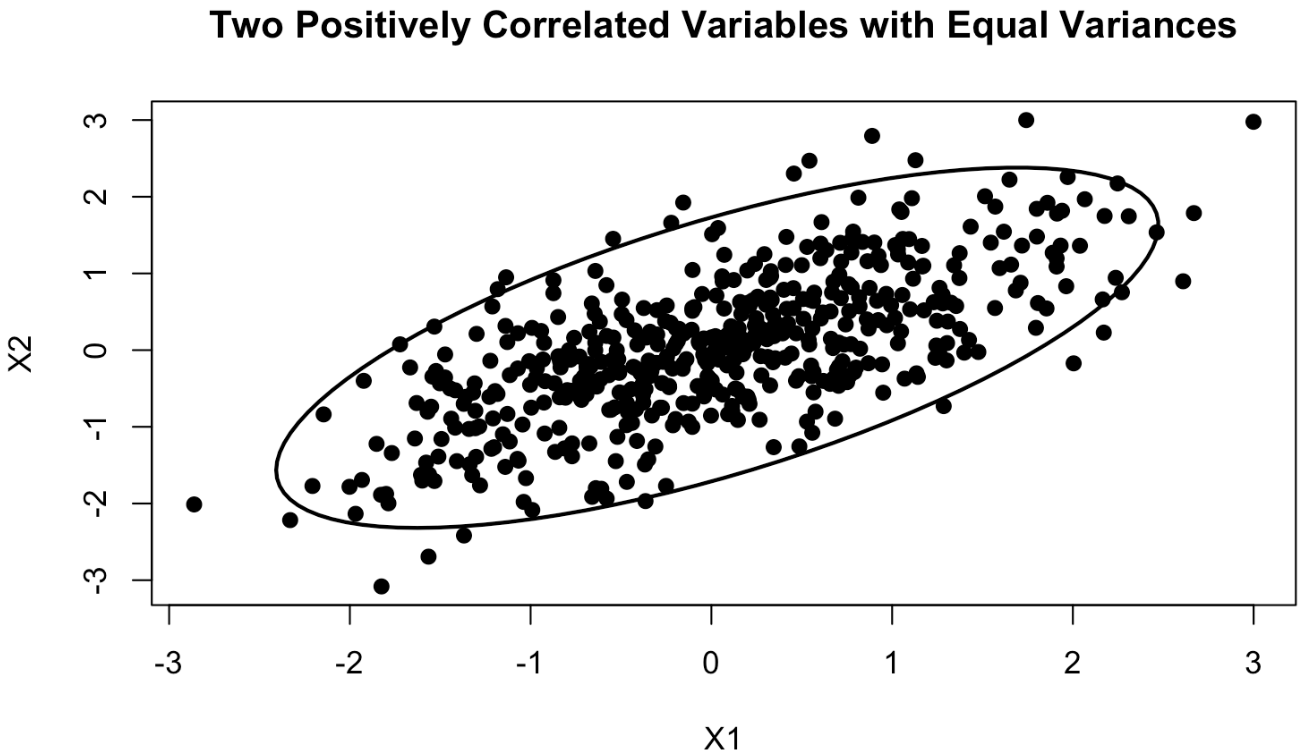
```
# Define the mean vector
mu = c(0,0)

# Define the covariance matrix (unequal variances)
Simga = matrix(c(2, 0,
                 0, 1), nrow = 2)

# Generate a bivariate normal sample
set.seed(123)
my_x = mvrnorm(n = 500, mu = mu, Sigma = Simga)

# Compute the mean and covariance of the sample
sample_mean = colMeans(my_x)
sample_cov = var(my_x)

plot(my_x, main = "Two Independent Variables with Unequal Variances", xlab = "X1", yl
ab = "X2")

# Add an ellipse to the plot
ellipse_data = ellipse(sample_cov, centre = sample_mean, level = 0.95)
lines(ellipse_data, lwd = 2)
```



c. The two variables are positively correlated with equal variances.

Hide

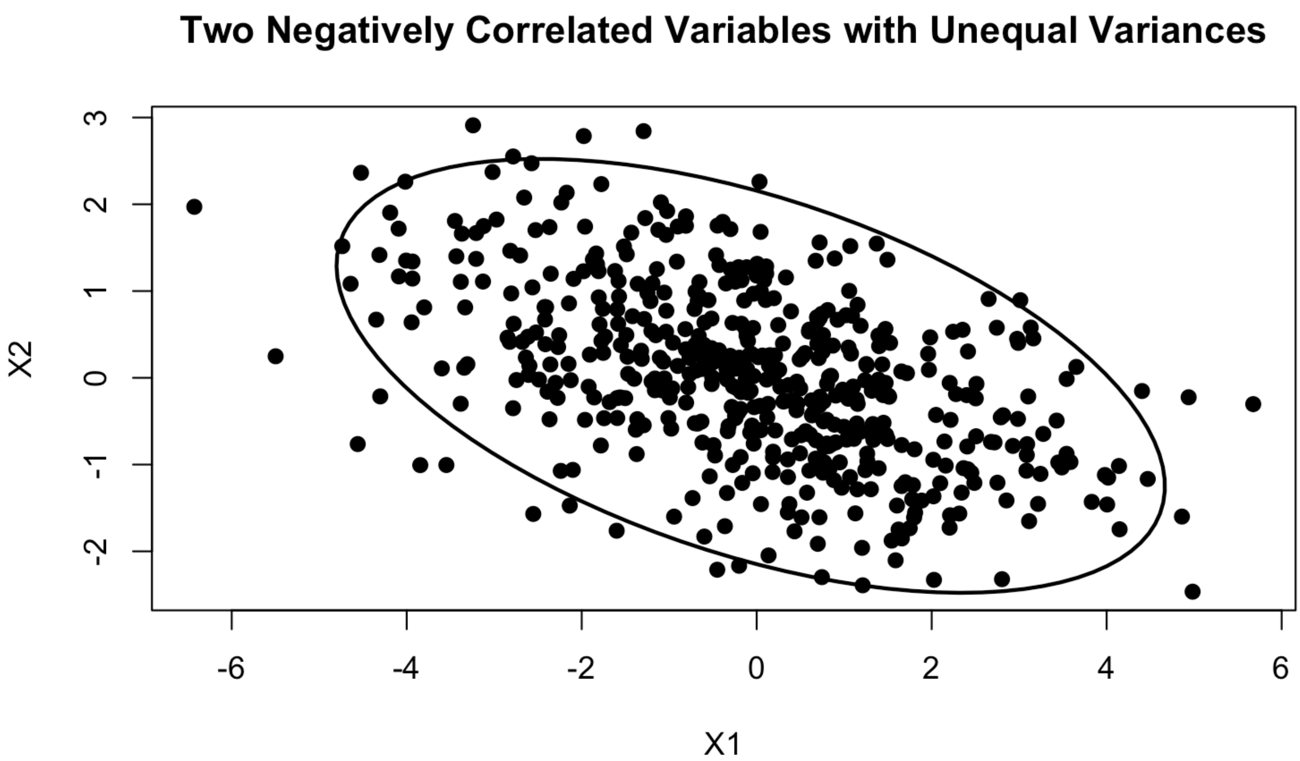
```
# Define the mean vector
mu = c(0, 0)

# Define the covariance matrix (positively correlated with equal variances)
Sigma = matrix(c(1, 0.7,
                 0.7, 1), nrow = 2)

# Generate a bivariate normal sample
set.seed(123)
my_x = mvrnorm(n = 500, mu = mu, Sigma = Sigma)

# Compute the mean and covariance of the sample
sample_mean = colMeans(my_x)
sample_cov = var(my_x)

plot(my_x, main = "Two Positively Correlated Variables with Equal Variances", xlab =
"X1", ylab = "X2", pch = 19)

# Add the ellipse to the plot
ellipse_data = ellipse(sample_cov, centre = sample_mean, level = 0.95)
lines(ellipse_data, lwd = 2)
```



d. The two variables are negatively correlated with unequal variances.

Hide

```
# Define the mean vector
mu = c(0, 0)

# Define the covariance matrix (negatively correlated with unequal variances)
Sigma = matrix(c(4, -1,
                 -1, 1), nrow = 2)

# Generate a bivariate normal sample
set.seed(123)
my_x = mvrnorm(n = 500, mu = mu, Sigma = Sigma)

# Compute the mean and covariance of the sample
sample_mean = colMeans(my_x)
sample_cov = var(my_x)

plot(my_x, main = "Two Negatively Correlated Variables with Unequal Variances", xlab
= "X1", ylab = "X2", pch = 19)

# Add the ellipse to the plot
ellipse_data = ellipse(sample_cov, centre = sample_mean, level = 0.95)
lines(ellipse_data, lwd = 2)
```



Question 3. For each of the 4 cases in Question 2, compute the length and the direction of each axes

Case (a): Independent Variables with Equal Variances

Hide

```
# Define the covariance matrix
Sigma = matrix(c(1, 0,
                 0, 1), nrow = 2)

# Eigen decomposition
eig = eigen(Sigma)

# Length of axes
axis_lengths = 2 * sqrt(eig$values)
cat("Lengths of axes:", axis_lengths, "\n")
```

```
Lengths of axes: 2 2
```

Hide

```
# Direction of axes (eigen-vectors)
cat("Directions of axes:\n")
```

```
Directions of axes:
```

Hide

```
print(eig$vectors)
```

```
     [,1] [,2]
[1,]    0   -1
[2,]    1    0
```

Case (b): Independent Variables with Unequal Variances

Hide

```
# Redfine the covariance matrix from Question 2b
Sigma = matrix(c(2, 0,
                  0, 1), nrow = 2)

# Eigen decomposition
eig = eigen(Sigma)

# Length of axes
axis_lengths = 2 * sqrt(eig$values)
cat("Lengths of axes:", axis_lengths, "\n")
```

```
Lengths of axes: 2.828427 2
```

Hide

```
# Direction of axes (eigenvectors)
cat("Directions of axes:\n")
```

```
Directions of axes:
```

Hide

```
print(eig$vectors)
```

```
     [,1] [,2]
[1,]   -1    0
[2,]    0   -1
```

Case (c): Positively Correlated Variables with Equal Variances

Hide

```
# Redefine the covariance matrix from Question 2c
Sigma = matrix(c(1, 0.7,
                  0.7, 1), nrow = 2)

# Eigen decomposition
eig = eigen(Sigma)

# Length of axes
axis_lengths = 2 * sqrt(eig$values)
cat("Lengths of axes:", axis_lengths, "\n")
```

```
Lengths of axes: 2.607681 1.095445
```

Hide

```
# Direction of axes (eigen-vectors)
cat("Directions of axes:\n")
```

```
Directions of axes:
```

Hide

```
print(eig$vectors)
```

```
          [,1]       [,2]
[1,] 0.7071068 -0.7071068
[2,] 0.7071068  0.7071068
```

Case (d): Negatively Correlated Variables with Unequal Variances

Hide

```
# Redfine the covariance matrix from Question 2d
Sigma = matrix(c(4, -1,
                 -1, 1), nrow = 2)

# Eigen decomposition
eig = eigen(Sigma)

# Length of axes
axis_lengths = 2 * sqrt(eig$values)
cat("Lengths of axes:", axis_lengths, "\n")
```

```
Lengths of axes: 4.148627 1.669999
```

Hide

```
# Direction of axes (eigen-vectors)
cat("Directions of axes:\n")
```

```
Directions of axes:
```

Hide

```
print(eig$vectors)
```

```
           [,1]       [,2]
[1,] -0.9570920 -0.2897841
[2,]  0.2897841 -0.9570920
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

Question 4.
For any n×p data matrix **X** with columns centered at zero, show that the sum of each row of **B = XX^T** is zero.

Step 1: Define the Matrix 'B'

We are given an n×p matrix **X**, where each **column is centered** which means that the sum of each column of X is zero:

$$\sum_{i=1}^{n} X_{ij} = 0 \text{ for all } j = 1, 2, \dots, p$$

B = XX^T is an n×n matrix, so we need to show that the sum of each row of B is zero

Step 2: Show that the Sum of Each Row of 'B' is Zero

The matrix **B = XX^T** is defined as:

$$B_{ik} = \sum_{j=1}^{n} X_{ij}X_{kj} \longrightarrow \sum_{k=1}^{n} B_{ik} = \sum_{k=1}^{n} \sum_{j=1}^{p} X_{ij}X_{kj} \longrightarrow \text{Rearrange the sums: } \sum_{k=1}^{n} B_{ik} = \sum_{j=1}^{p} X_{ij} \left(\sum_{k=1}^{n} X_{kj} \right)$$

—> Since the columns of X are centered at zero, we have:

$$\sum_{k=1}^{n} X_{kj} = 0 \text{ for all } j = 1, 2, \dots, p \longrightarrow \text{Therefore: } \sum_{k=1}^{p} B_{ik} = \sum_{j=1}^{p} X_{ij} * 0 = 0$$

—> This shows that the sum of each row of B is zero.

—> Similarly, since B is a symmetric matrix, the sum of each column of B is also zero.