Homework 01

Nahid Ferdous

2023-10-21

Problem 01

In the question no 1.1 "2xn" matrix has n numbers of columns and 2 rows. So the numbers of columns are not constent but rows are constent here.

```
# Here we consider n= 3 , you can consider any positive numerical number.
n<- 3
my_matrix <- matrix(data=1:(2*n), nrow= 2, ncol = n)
print(my_matrix)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

In the question 1.2, we need to multiply matrix 1 and matrix 2. First matrix dimontion 1x3(RXC) and second second matrix dimontion 2X2 (RXC). For matrix multiplicetion if the first matrix column number and second metrix row numbers are equal then we can multiply one and two matrix. (dot multiplicetion or dot product). and the output dimention shoud be R(First matrix number of row) x C(Second matrix number of columns)

```
matrix_one = matrix(c(2,-1),ncol = 2, byrow = TRUE)
matrix_one
```

```
##      [,1] [,2]
## [1,]    2   -1
```

```
print("Matrix Dimentions")
```

```
## [1] "Matrix Dimentions"
```

```
dim(matrix_one)
```

```
## [1] 1 2
```

```r
matrix_two = matrix(c(1,1,2,0), ncol = 2, byrow = TRUE)
matrix_two
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    0
```

```r
print("Matrix Dimentions")
```

```
## [1] "Matrix Dimentions"
```

```r
dim(matrix_two)
```

```
## [1] 2 2
```

```r
matrix_multiplicetion_1.2= matrix_one %*% matrix_two
matrix_multiplicetion_1.2
```

```
##      [,1] [,2]
## [1,]    0    2
```

```r
print("Matrix Dimentions")
```

```
## [1] "Matrix Dimentions"
```

```r
dim(matrix_multiplicetion_1.2)
```

```
## [1] 1 2
```

In the question 1.3, we cant do multiplicetion between first and second marrix. Because of the dimention issue. First matrix number of collumns are not same with second metrix number of rows.

```r
matrix_one= matrix(c(1,0,2,0), ncol = 2, byrow = TRUE)
matrix_one
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    2    0
```

```r
print("Matrix Dimentions")
```

```
## [1] "Matrix Dimentions"
```

```r
dim(matrix_one)
```

```
## [1] 2 2
```

```r
matrix_two= matrix(c(2,-1), ncol=2)
matrix_two
```

```
##      [,1] [,2]
## [1,]    2   -1
```

```r
print("Matrix Dimentions")
```

```
## [1] "Matrix Dimentions"
```

```r
dim(matrix_two)
```

```
## [1] 1 2
```

```r
## we can't do multiplicetion between first and second marrix.
```

In the question 1.4, matrix_one is a squair matrix but the determinant is zero. So we dont calculate the inverse of matrix_one.

```r
matrix_one = matrix(c(1,0,2,0), ncol= 2, byrow = TRUE)
matrix_one
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    2    0
```

```r
dimention = dim(matrix_one)
print("Matrix Dimentions")
```

```
## [1] "Matrix Dimentions"
```

```r
dimention
```

```
## [1] 2 2
```

```r
## dimention of the matrix_one should be squar ( number of rows = number of columns)
determinant_of_matrix_one = det(matrix_one)
## determinant of the matrix_one should be non zero
if(determinant_of_matrix_one != 0){
  matrix_one_inverse <- solve(matrix_one)
  print("The inverse of Matrix_one is : ")
  print(round(matrix_one_inverse),10)
} else{
  print(" Matrix_one does not have an inverse. ")
}
```

```
## [1] " Matrix_one does not have an inverse. "
```

Problem 02

```
covariance_matrix <- matrix(c(3.877, 2.811, 3.148, 3.506, 2.811, 2.121, 2.266, 2.569,3.148, 2.266,
                              2.655, 2.834, 3.506, 2.569, 2.834, 3.235), ncol = 4, byrow = TRUE)
print("covariance_matrix")
```

```
## [1] "covariance_matrix"
```

```
covariance_matrix
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 3.877 2.811 3.148 3.506
## [2,] 2.811 2.121 2.266 2.569
## [3,] 3.148 2.266 2.655 2.834
## [4,] 3.506 2.569 2.834 3.235
```

```
#cov to corr conversion
cat("\n\n")
```

```
print("cov to corr conversion")
```

```
## [1] "cov to corr conversion"
```

```
corr_matrix <- cov2cor(covariance_matrix)
cat("\n\n")
```

```
print("corr_matrix")
```

```
## [1] "corr_matrix"
```

```
corr_matrix
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.9802630 0.9811933 0.9899810
## [2,] 0.9802630 1.0000000 0.9548987 0.9807462
## [3,] 0.9811933 0.9548987 1.0000000 0.9670088
## [4,] 0.9899810 0.9807462 0.9670088 1.0000000
```

```
# By hand correlations calculetion from covariance
# first we need to calculate standard deviation from the covariance_matrix, we consider covariance_matr

cat("\n\n")
```

```r
# Covariance
x1_var <- covariance_matrix[1,1]
x2_var <- covariance_matrix[2,2]
# Standard deviation
x1_std <-sqrt(x1_var)
x2_std <-sqrt(x2_var)

# correlation
corr_matrix_1_2 <- covariance_matrix[1,2]/(x1_std*x2_std)
print("Print corr_matrix_1_2 position: ")
```

```
## [1] "Print corr_matrix_1_2 position: "
```

```r
corr_matrix_1_2
```

```
## [1] 0.980263
```

Problem 03

```r
library(MVA)
```

```
## Loading required package: HSAUR2
```

```
## Loading required package: tools
```

```r
data("USairpollution", package = "HSAUR2")
mydata1 <- USairpollution[1:10,]

# Euclidean distance matrix for the first 10 cities "USairpollution[1:10,]"
# First standardized or scale  our data "7 columns ", for a better understanding
mydata1_scale <- scale(mydata1[,1:7])
Euclidean_distance_matrix <- dist(mydata1_scale, method = "euclidean")
round((Euclidean_distance_matrix),2)
```

```
##             Albany Albuquerque Atlanta Baltimore Buffalo Charleston Chicago
## Albuquerque   4.25
## Atlanta       3.60        4.44
## Baltimore     2.28        3.96    1.87
## Buffalo       2.66        5.39    4.30      3.33
## Charleston    2.36        4.60    2.52      2.48    4.02
## Chicago       5.35        6.79    5.87      4.50    5.75       6.09
## Cincinnati    2.05        4.08    2.29      1.89    3.64       0.96    5.53
## Cleveland     1.99        5.04    3.71      2.24    2.22       3.33    4.08
## Columbus      1.26        3.98    2.58      1.57    2.68       1.68    5.25
##             Cincinnati Cleveland
## Albuquerque
## Atlanta
## Baltimore
```

```
## Buffalo
## Charleston
## Chicago
## Cincinnati
## Cleveland         2.97
## Columbus          1.08      2.18
```

Cincinnati and Columbus have a Euclidean distance of approximately 1.08 (standardized), indicating that these cities share a close similarity in terms of their environment and population. This is because we used environmental variables to calculate the Euclidean distance.

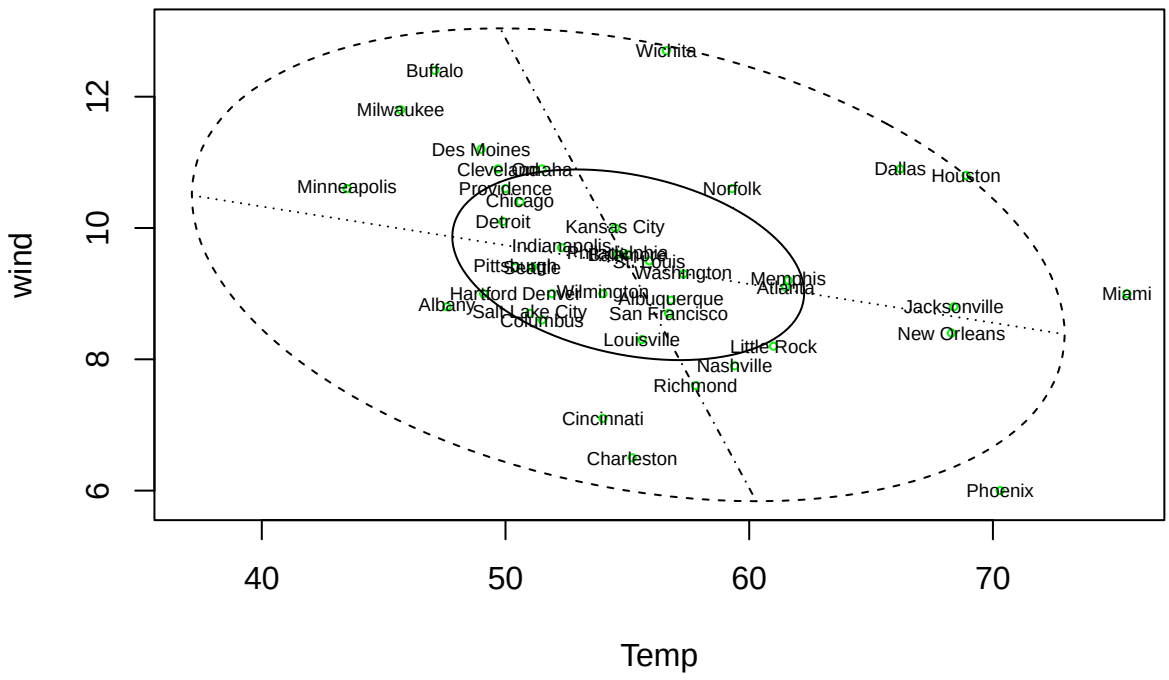
Chicago and Albuquerque have a Euclidean distance of 6.79 (standardized), signifying that these cities have a significantly large Euclidean distance. This implies that the environment and population characteristics of these cities differ substantially from those of other cities.

In conclusion, if the Euclidean distance is small, it suggests a high level of similarity, whereas a high Euclidean distance suggests a low level of similarity.

Problem 04

Initially, we establish three distinct datasets, each encompassing all rows but only two specific columns. Following this, we separately compute the correlations, identify any outliers, and subsequently eliminate these outliers before recalculating the correlations.

```r
data("USairpollution", package = "HSAUR2")
data_temp_wind <- USairpollution[,c("temp","wind")]
data_temp_wind_corr_org <- cor(data_temp_wind)
# lets find the outliers using bivariate boxplot
bvbox(data_temp_wind, xlab = "Temp", ylab = "wind", col = "green", cex = .5)
text(data_temp_wind, cex= .6 , labels = row.names(USairpollution))
```




```
outliers_label <- c("Miami","Phoenix")
outliers_index_number <- match(outliers_label, row.names(USairpollution))
USairpollution_cleanData_temp_wind <- data_temp_wind[-outliers_index_number,]
print("data_temp_wind_corr_org")
```

```
## [1] "data_temp_wind_corr_org"
```

```
round((data_temp_wind_corr_org),2)
```

```
##       temp  wind
## temp  1.00 -0.35
## wind -0.35  1.00
```

```
cat("\n\n")
```

```
print("data_temp_wind_corr_clean")
```

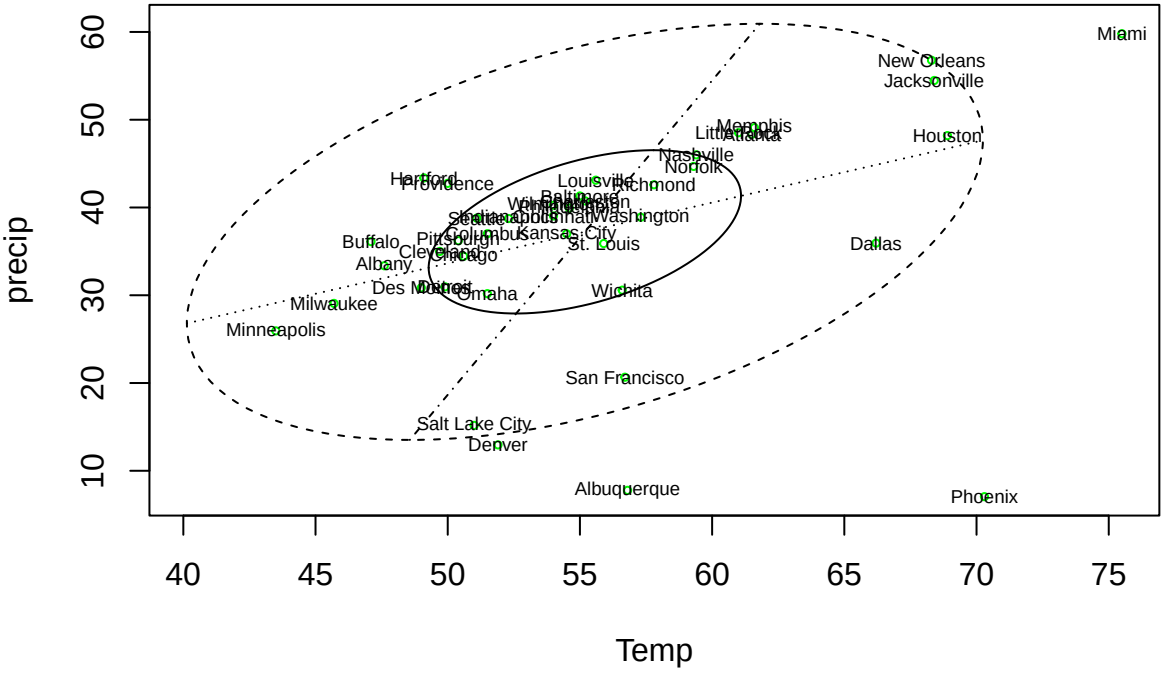
```
## [1] "data_temp_wind_corr_clean"
```

```
round((cor(USairpollution_cleanData_temp_wind)),2)
```

```
##       temp  wind
## temp  1.00 -0.26
## wind -0.26  1.00
```

The "data_temp_wind_corr_org" and "data_temp_wind_corr_clean" outputs represent correlation coefficients between "temp" and "predays" before and after data cleaning. Initially, there's a moderate inverse relationship (-0.43). Post-cleaning, the correlation slightly strengthens to -0.42, indicating minimal impact of outliers or noise on the relationship.

```
data("USairpollution", package = "HSAUR2")
data_temp_precip <- USairpollution[,c("temp","precip")]
data_temp_precip_corr_org <- cor(data_temp_wind)
# lets find the outliers using bivariate boxplot
bvbox(data_temp_precip, xlab = "Temp", ylab = "precip", col = "green", cex = .5)
text(data_temp_precip, cex= .6 , labels = row.names(USairpollution))
```




```
outliers_label <- c("Miami", "Denver", "Albuquerque", "Phoenix")
outliers_index_number <- match(outliers_label, row.names(USairpollution))
USairpollution_cleanData_temp_precip <- data_temp_precip[-outliers_index_number,]
print("data_temp_precip_corr_org")
```

```
## [1] "data_temp_precip_corr_org"
```

```r
round((data_temp_precip_corr_org),2)
```

```
##       temp  wind
## temp  1.00 -0.35
## wind -0.35  1.00
```

```r
cat("\n\n")
```

```r
print("data_temp_precip_corr_clean")
```

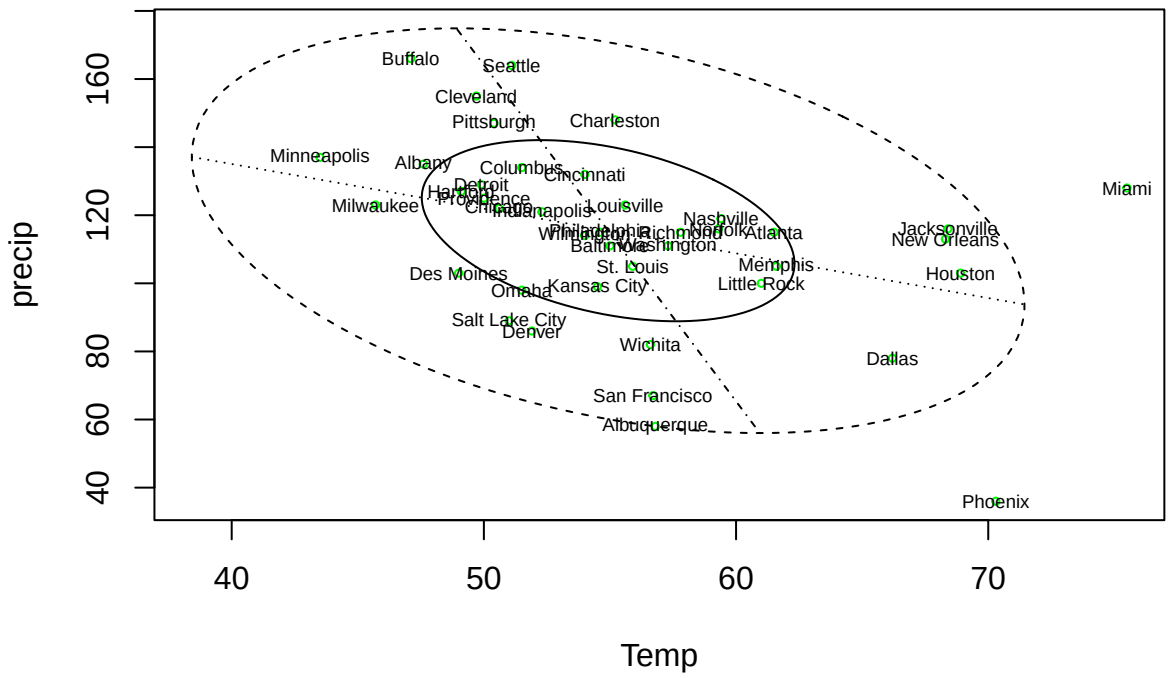
```
## [1] "data_temp_precip_corr_clean"
```

```r
round((cor(USairpollution_cleanData_temp_precip)),2)
```

```
##        temp precip
## temp   1.00   0.66
## precip 0.66   1.00
```

The "data_temp_precip_corr_org" shows a moderate negative correlation (-0.43) between "temp" and "predays." After cleaning, "data_temp_precip_corr_clean" reveals a shifted focus to "temp" and "precip," now demonstrating a moderate positive correlation (0.66), indicating a significant change due to data cleaning.

```r
data("USairpollution", package = "HSAUR2")
data_temp_wind <- USairpollution[,c("temp","predays")]
data_temp_wind_corr_org <- cor(data_temp_wind)
# lets find the outliers using bivariate boxplot
bvbox(data_temp_wind, xlab = "Temp", ylab = "precip", col = "green", cex = .5)
text(data_temp_wind, cex= .6 , labels = row.names(USairpollution))
```




```
outliers_label <- c("Miami","Phoenix")
outliers_index_number <- match(outliers_label, row.names(USairpollution))
USairpollution_cleanData_temp_predays <- data_temp_wind[-outliers_index_number,]
print("data_temp_wind_corr_org")
```

```
## [1] "data_temp_wind_corr_org"
```

```
round((data_temp_wind_corr_org),2)
```

```
##          temp predays
## temp     1.00   -0.43
## predays -0.43    1.00
```

```
cat("\n\n")
```

```
print("data_temp_wind_corr_clean")
```

```
## [1] "data_temp_wind_corr_clean"
```

```
round((cor(USairpollution_cleanData_temp_predays)),2)
```

```
##          temp predays
## temp     1.00   -0.42
## predays -0.42    1.00
```

The "data_temp_wind_corr_org" indicates an original moderate negative correlation between "temp" and "predays" (-0.43). After data cleaning, "data_temp_wind_corr_clean" shows a slight increase in this correlation to -0.42, suggesting that cleaning had a minimal effect on the "temp" and "predays" relationship.

Problem 05

First, we convert the categorical variable, then measure the Mahalanobis distance, and subsequently assess normality using a chi-square plot.

```
# for create dummy categorical columns
library(fastDummies)
```

```
## Thank you for using fastDummies!

## To acknowledge our work, please cite the package:

## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from
```

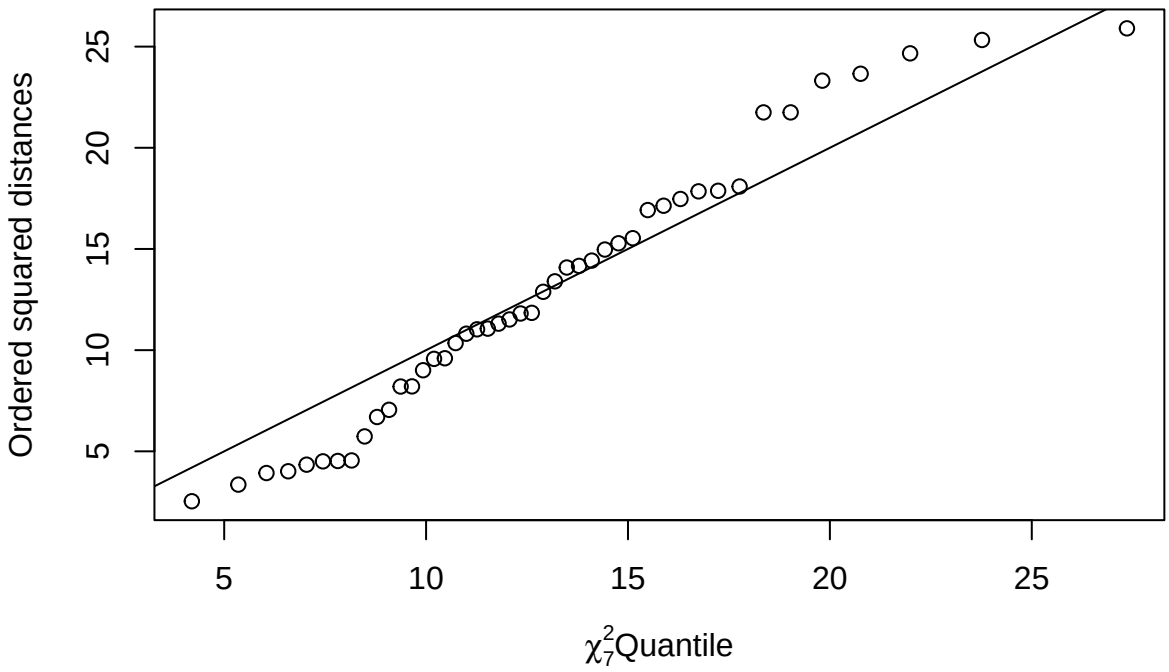
```
data(pottery, package = "HSAUR2")
mydata_p <- pottery

# create dummy
mydata_p_with_dummy <- dummy_cols(mydata_p, select_columns = "kiln", remove_selected_columns = TRUE, re

# calculating Mahalanobis distance
x <- mydata_p_with_dummy
xbar <- colMeans(x)
s <- cov(x)
d2 <- mahalanobis(x, xbar, s)

# testing multivariate normality
number_variables <- ncol(mydata_p_with_dummy)
position <- (1:nrow(x)-1/2)/nrow(mydata_p_with_dummy)
quantiles <- qchisq(position, df= number_variables)

plot(quantiles, sort(d2), xlab = expression(paste(chi[7]^2,"Quantile")), ylab = "Ordered squared distan
abline(a=0,b=1)
```



In a Q-Q plot, if the data points (the circles in my plot) closely follow the straight diagonal line, then the data is considered to be normally distributed. However, if the data points deviate significantly from the line, then the data is not normally distributed.

From the visualizetion , the data points generally follow the line but deviate from it at the upper end. This suggests that the data may not be perfectly normally distributed, especially in the tails.

In conclusion, while the majority of the data seems to align with a normal distribution, there is some deviation in the tails which suggests that the data might not be perfectly normally distributed.

Problem 06

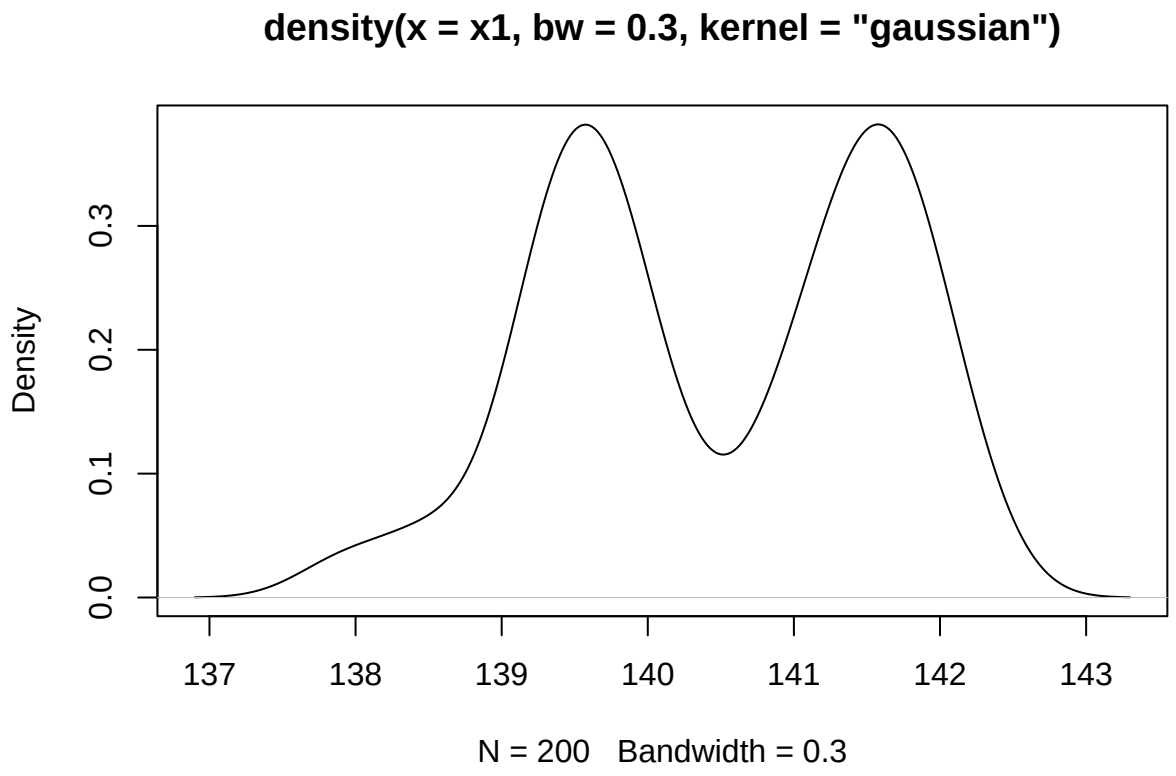
```
library(mclust)
```

```
## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```
data(banknote, package = "mclust")
mydata_3 <- banknote[,c("Status","Bottom","Diagonal")]
```

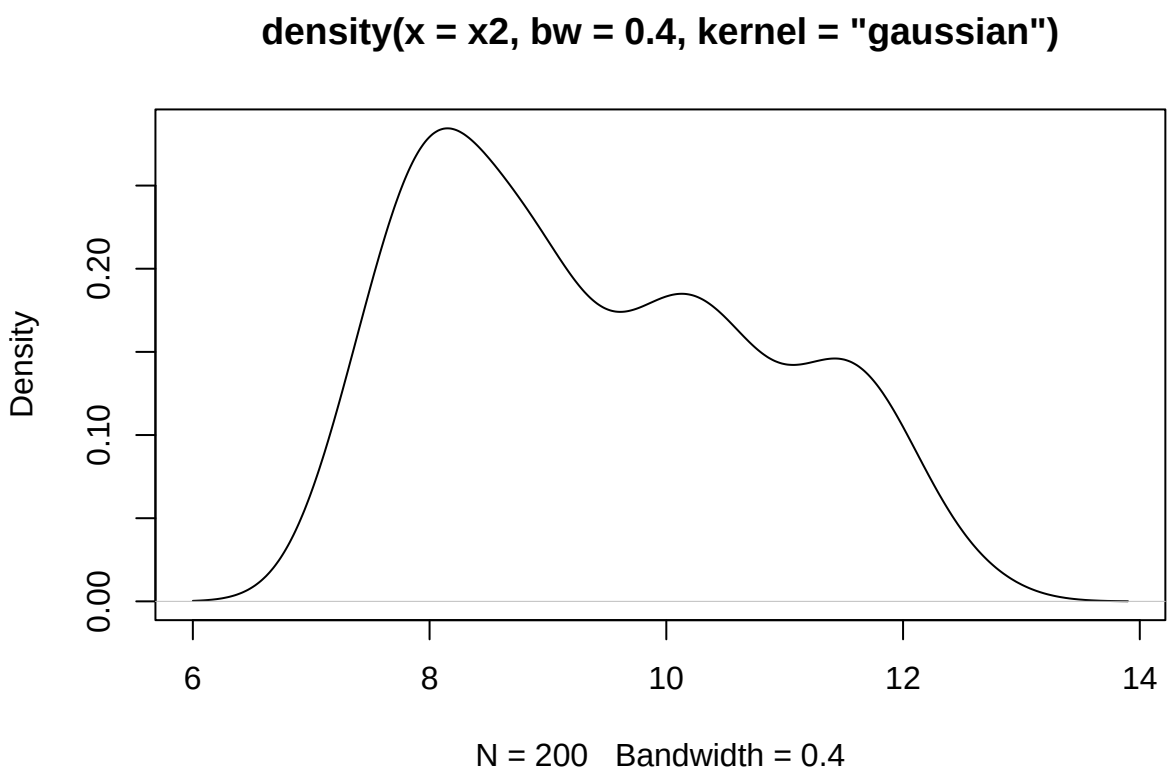
Problem 6.A

```
x1 <- mydata_3[,"Diagonal"]
plot(density(x1,bw= .3, kernel = "gaussian"))
```



Problem 6.A

```
x2 <- mydata_3[,"Bottom"]
plot(density(x2,bw= .4, kernel = "gaussian"))
```

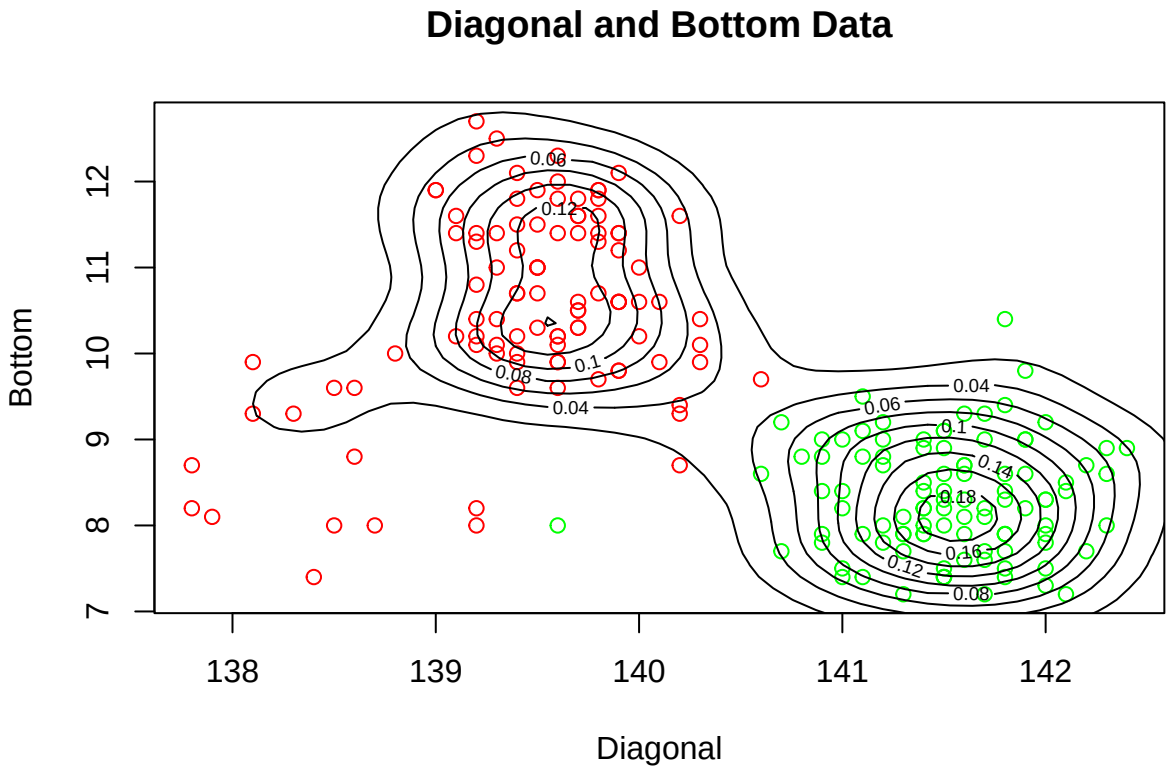


Problem 6.B

```
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
#bw <- c(dpik(mydata_3$Diagonal), dpik(mydata_3$Bottom))
density_s <- bkde2D(mydata_3[,c("Diagonal","Bottom")], bandwidth = c(.3,.4))
colors_s <- ifelse(mydata_3[,"Status"]== "genuine", "green","red")
plot(mydata_3[,c("Diagonal","Bottom")], xlab= "Diagonal", ylab= "Bottom", main= "Diagonal and Bottom Da
contour(x= density_s$x1, y=density_s$x2 , z= density_s$fhat , add = TRUE)
```



From this plot we can tell: ## The reletionship between Bottom vs Diagonal data is negetive. ## Scatter plot : The individual dots on the graph represent data points. Their position on the x-axis represents their "Diagonal" value, and their position on the y-axis represents their "Bottom" value. ## Contour Lines: The curved lines that create concentric shapes on the graph are contour lines. They are used to represent areas of equal density, meaning areas where data points are concentrated. In other words: ## The innermost contour lines encompass areas with the highest density of data points. ## As you move outward to subsequent contour lines, the density decreases.

Density of Data Points: The regions with many closely packed dots indicate areas where many data points share similar "Diagonal" and "Bottom" values. Conversely, areas with fewer dots suggest that fewer data points have those particular combinations of "Diagonal" and "Bottom" values.

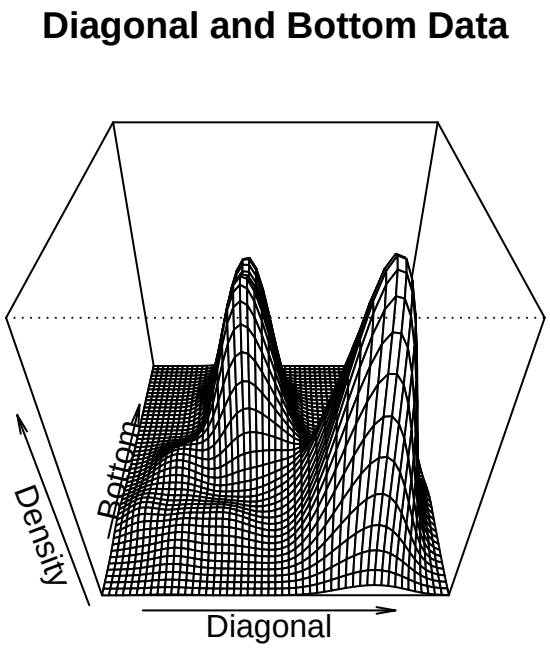
Clusters: It seems that there are two major clusters, each surrounded by its contour lines. This indicates that there are two main groupings of data points with distinct "Diagonal" and "Bottom" value combinations.

Outliers: The plot also has some individual points that are outside the main clusters, especially on the down left. These are outliers, or data points that don't fit the general pattern of the rest of the data.

This plot provides a visual representation of how the "Diagonal" and "Bottom" measurements relate to each other across all the data points and how these data points are distributed in terms of density and clustering.

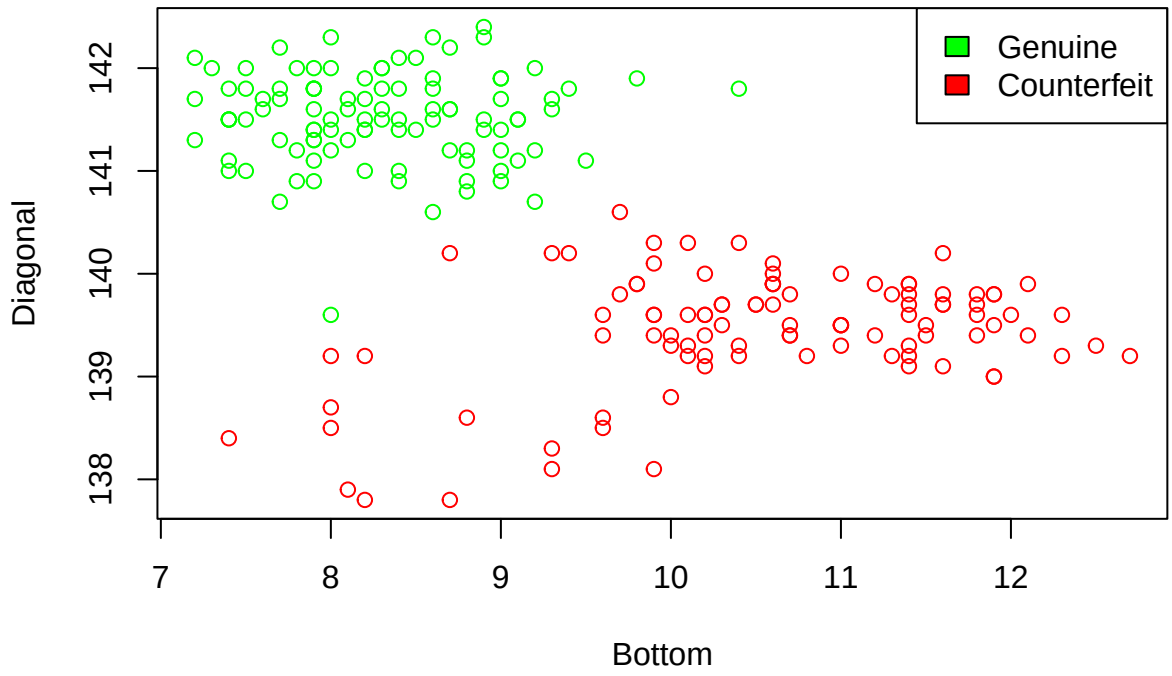
Problem 6.B

```
# 3d Visualizetion
persp(x= density_s$x1, y= density_s$x2, z= density_s$fhat,
      xlab = "Diagonal", ylab = "Bottom", zlab = "Density", main = "Diagonal and Bottom Data", phi = 40)
```



Problem 6.C

```
colors_s <- ifelse(mydata_3[,"Status"]== "genuine", "green","red")
plot(mydata_3[,c("Bottom", "Diagonal")], xlab= "Bottom", ylab = "Diagonal" , col = colors_s)
legend("topright", legend=c("Genuine", "Counterfeit"), fill=c("green", "red"))
```



Here Green points shows they are genuine point and Red points shows they are counterfeit