

# Assignment



**Course No:** STAT-403

**Course Title:** Multivariate Analysis

**Submitted By**

Shawkatul Islam

Class Roll: 134

Session: 2019-20

Batch: 49th

**Submitted To**

Dr. Rumana Rois

Professor

Department of Statistics and Data Science

Jahangirnagar University

Savar, Dhaka-1342

# LAB Assignment

Shawkatul Islam; Roll: 134

2025-04-13

**Question 1:** Examine the multivariate normality of Fixed acidity, volatile acidity, Citric acid, Residual sugar, chlorides, Free sulfur dioxide, Total sulfur dioxide, density, pH, sulphates, and alcohol variables of the Wine Quality data. Is there any outlier in those variables? Explain.

Ans: **Chi-Square Plot :**

```
library(tidyverse)

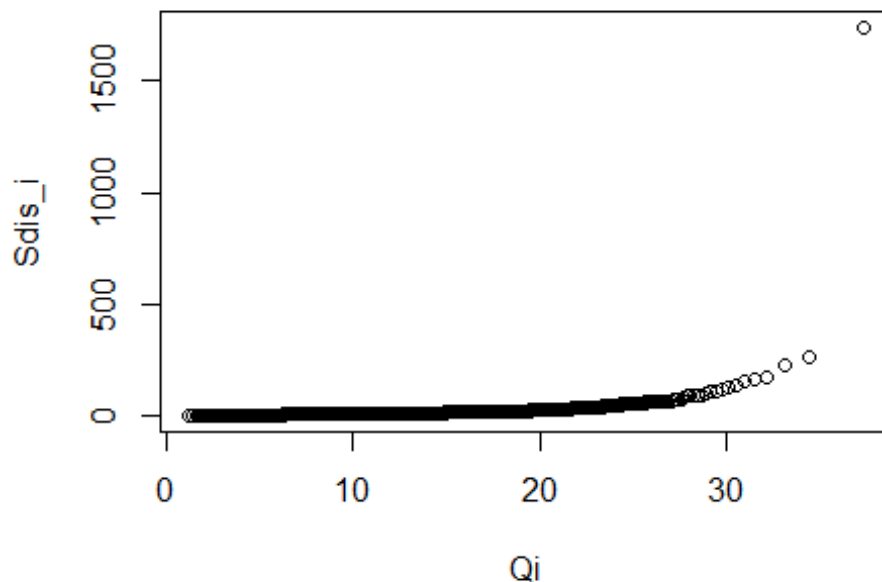
library(MASS)

library(psych)

wine=read.csv("D:/4th Year/LAB/403/winequality-white.csv", sep = ";", header = TRUE)
y=wine[,1:11]
w=as.matrix(y)
n=length(w[,1])
mean_W=colMeans(w)
mean_W=as.matrix(mean_W)
S=cov(w)
class(S)

## [1] "matrix" "array"

Sdis=matrix(0, nrow=n, ncol=1)
for(i in 1:n){
  Sdis[i]=t(w[i,]-mean_W)%*%solve(S)%*%(w[i,]-mean_W) ##Mahalanobis Distance
}
Sdis_i=sort(Sdis, decreasing = F)
j=seq(1,n,1)
p=ncol(w)
Qi=qchisq((j-.5)/n, p)
plot(Qi, Sdis_i)
```



Counting the number of observations in certain intervals:

```
Chi_CV=qchisq(.50, 11)
A=ifelse(Sdis<=Chi_CV, 1,0)
P=colSums(A)/n
if(P==.5) {print('Normality Assumption Holds')} else {print('Normality Assumption does not Hold')}

## [1] "Normality Assumption does not Hold"
```

**Outlier Detection:**

```
cutoff = qchisq(0.95, df = 11)
outliers = y[Sdis > cutoff, ]
nrow(outliers)

[1] 395

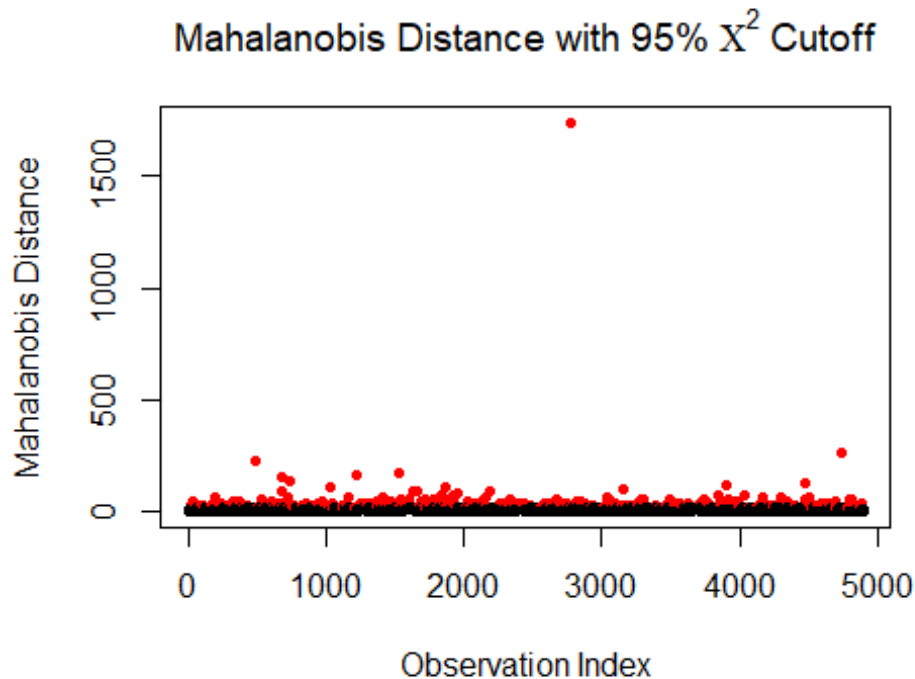
cat("Outliers found:", nrow(outliers), "/", nrow(y))

## Outliers found: 395 / 4898
```

So, there are 395 outliers detected in the dataset.

```
is_outlier= Sdis > cutoff
plot(Sdis,
     col = ifelse(is_outlier, "red", "black"),
```

```
pch = 20,
main = expression("Mahalanobis Distance with 95%" ~ Chi^2 ~ "Cutoff"),
ylab = "Mahalanobis Distance",
xlab = "Observation Index")
```



Question 2:

### Question 2: Principal Component Analysis (i)

```
s=cov(y)
pca_result=prcomp(s, center = TRUE, scale. = FALSE)
pca_result

## Standard deviations (1, ..., p=11):
## [1] 5.647904e+02 5.181289e+01 6.535530e+00 3.226777e-01 1.970682e-01
## [6] 5.308003e-03 4.268333e-03 3.574541e-03 1.826036e-03 8.907231e-05
## [11] 7.003282e-19
##
## Rotation (n x k) = (11 x 11):
##          PC1          PC2          PC3          PC4
## fixed.acidity -1.563819e-03 - 9.545782e-03 0.0099595374 0.289331832
## volatile.acidity -1.794905e-04 -1.561236e-03 0.0008231105 -0.015198722
## citric.acid -3.332379e-04 1.140986e-04 0.0010783247 0.012204314
## residual.sugar -4.592683e-02 6.483099e-03 0.9949167124 -0.087741160
## chlorides -9.857820e-05 -6.825899e-05 0.0001130125 0.006346801
## free.sulfur.dioxide -2.537848e-01 9.670476e-01 -0.0175046126 0.007107320
## total.sulfur.dioxide -9.660871e-01 -2.542445e-01 -0.0438570708 -0.010519605
## density -3.561124e-05 -2.162660e-05 0.0004507498 0.001195881
```

```

## pH -1.406805e-05 3.004481e-05 -0.0069203073 -0.025737820
## sulphates -3.456029e-04 -3.464476e-04 -0.0022178001 -0.001475788
## alcohol 1.252981e-02 6.552277e-03 -0.0880728292 -0.952544556
## PC5 PC6 PC7 PC8
## fixed.acidity 0.9527453555 0.0884768213 -2.032024e-02 0.0026177535
## volatile.acidity -0.0034732393 -0.1421561175 -5.654858e-01 0.0479905133
## citric.acid 0.0373319022 -0.3313245332 7.609906e-01 0.3216442582
## residual.sugar 0.0157039941 0.0062030085 2.177038e-03 -0.0007204167
## chlorides -0.0026006587 -0.0258369491 -3.678397e-03 0.0062120086
## free.sulfur.dioxide 0.0072089328 0.0005337496 -1.155695e-03 -0.0001844503
## total.sulfur.dioxide -0.0004212955 -0.0005103499 7.674906e-05 0.0002594262
## density 0.0001310714 0.0031268728 7.574366e-04 -0.0001885378
## pH -0.0777338791 0.9171036728 1.452066e-01 0.2649093907
## sulphates -0.0074047777 0.1427982996 2.820230e-01 -0.9077535764
## alcohol 0.2906331441 -0.0008337306 8.006715e-03 -0.0014977932
## PC9 PC10 PC11
## fixed.acidity 0.0106572487 2.922207e-03 -7.679219e-04
## volatile.acidity 0.8108434518 8.431352e-04 -6.107549e-04
## citric.acid 0.4539855090 -3.930395e-03 -3.280338e-04
## residual.sugar 0.0000624257 6.532972e-04 -3.710004e-04
## chlorides -0.0083974620 9.995737e-01 -3.905818e-03
## free.sulfur.dioxide 0.0012861636 3.813860e-05 6.950177e-06
## total.sulfur.dioxide -0.0009090964 -6.415508e-05 -3.964732e-06
## density 0.0018591307 4.000516e-03 9.999843e-01
## pH 0.2455378261 2.460460e-02 -3.438590e-03
## sulphates 0.2753734335 1.266855e-02 -1.390215e-03
## alcohol -0.0109848607 6.745348e-03 1.131038e-03

```

### summary(pca\_result)

```

## Importance of components:
## PC1 PC2 PC3 PC4 PC5 PC6
## Standard deviation 564.7904 51.81289 6.53553 0.3227 0.1971 0.005308
## Proportion of Variance 0.9915 0.00834 0.00013 0.0000 0.0000 0.000000
## Cumulative Proportion 0.9915 0.99987 1.00000 1.0000 1.0000 1.000000
## PC7 PC8 PC9 PC10 PC11
## Standard deviation 0.004268 0.003575 0.001826 8.907e-05 7.003e-19
## Proportion of Variance 0.000000 0.000000 0.000000 0.000e+00 0.000e+00
## Cumulative Proportion 1.000000 1.000000 1.000000 1.000e+00 1.000e+00

```

pca\_result\$sdev^2

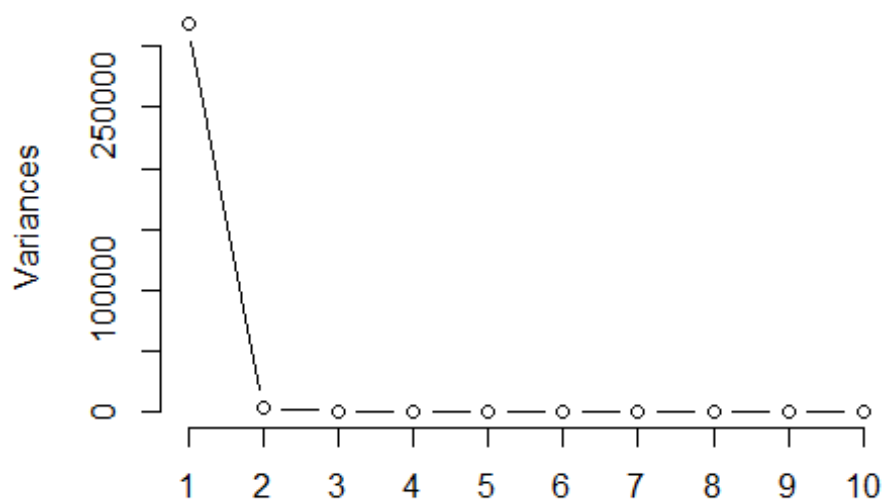
```

## [1] 3.189882e+05 2.684576e+03 4.271316e+01 1.041209e-01 3.883588e-02
## [6] 2.817489e-05 1.821867e-05 1.277734e-05 3.334407e-06 7.933877e-09
## [11] 4.904596e-37

```

plot(pca\_result, type = "lines", main = "Scree Plot")

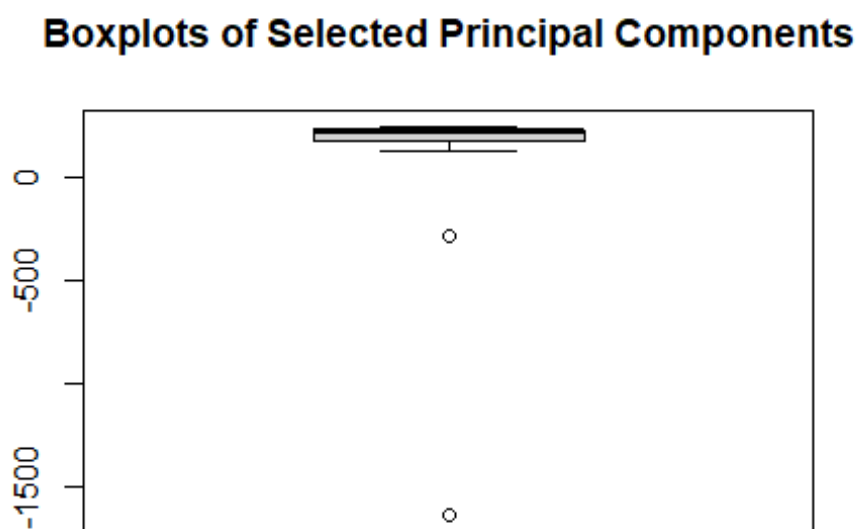
## Scree Plot



From the **Scree plot**, we can see that, there is a sharp drop after 1st principal component, so we can say that 1st principal component or PC1 is sufficient to represent the data.

## Boxplot

```
pc_scores=as.data.frame(pca_result$x)  
boxplot(pc_scores[, 1], main = "Boxplots of Selected Principal Components")
```



The boxplot of the first principal component (PC1) shows that ,most samples of wine have very similar scores in PC1, which indicates high similarity in the overall chemical composition of the absolute majority of samples. However, two extreme negative outliers indicate a minority of the samples have atypical combinations of characteristics that deviate dramatically from the general trend seen in the dataset.

```
loading_pH_PC1=pca_result$rotation["pH", 1]
sd_PC1=pca_result$sdev[1]      # SD of PC1 (eigenvalue)
sd_pH=sd(y[, "pH"])           # SD of pH from predictors
cor_PC1_pH=(loading_pH_PC1 * sd_PC1) / sd_pH
cat("Correlation between PC1 and pH is:", cor_PC1_pH, "\n")

# Correlation between PC1 and pH is: -0.052619
```

Which indicates a very low and negative correlation.

**Question 3:** Factor analysis for the Wine Quality data.

```
p_val= matrix(0, nrow = 6, ncol = 1)
for (i in 1:6) {
  p_val[i] = factanal(y, factors = i, method = "pca")$PVAL
}
print(p_val)

##           [,1]
## [1,] 0.000000e+00
## [2,] 0.000000e+00
## [3,] 0.000000e+00
## [4,] 0.000000e+00
## [5,] 8.257723e-173
## [6,] 6.402833e-74
```

The results from factanal() loop indicates that none of the tested factor models (1 to 6 factors) are a good fit for the wine data based on the p-values. All p-values are either zero or extremely close to zero indicating factor model may not adequately capture the covariance structure. We need to test Bartlett's test and KMO (Kaiser-Meyer-Olkin) measure of sampling adequacy.

**#Bartlett and KMO Tests**

```
bartlett_test = cortest.bartlett(cor(y), n = nrow(y))
print(bartlett_test)

## $chisq
## [1] 24852.58
##
## $p.value
## [1] 0
##
```

```
## $df
## [1] 55

KMO_result = KMO(cor(y))
print(KMO_result)

## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = cor(y))
## Overall MSA = 0.37
## MSA for each item =
##      fixed.acidity    volatile.acidity    citric.acid
##           0.17           0.23           0.72
##      residual.sugar    chlorides    free.sulfur.dioxide
##           0.33           0.66           0.59
## total.sulfur.dioxide    density    pH
##           0.71           0.40           0.16
##           sulphates    alcohol
##           0.19           0.36
```

The Bartlett's test is highly significant ( $p < 0.001$ ) indicating that the correlation matrix may not be an identity matrix and that there are sufficient correlations between variables to conduct a factor analysis. The KMO measure of sampling adequacy is only 0.37, which is far below the commonly accepted cut-off point of 0.5. A low score indicates that the data is not well-suited for extracting factors effectively. As a result, factor analysis is not suitable for the wine data set.

**Question 4:** Calculate Fisher's linear discriminant function for classifying the Wine Quality data into Average quality  $\leq 5$  or Good quality  $> 5$ . Classify the Wine Quality data into Average or Good quality using the following characteristics:

```
library(tidyverse)
library(MASS)

# Add Quality Class Label
D = wine |> mutate(quality_class = case_when(
  quality <= 5 ~ "Average Quality",
  quality > 5 ~ "Good Quality"
))

D$quality_class = as.factor(D$quality_class)
fit_lda = lda(quality_class ~ ., data = D[, c(1:11, 13)])
print(fit_lda)

## Call:
## lda(quality_class ~ ., data = D[, c(1:11, 13)])
##
## Prior probabilities of groups:
## Average Quality    Good Quality
```



```
##      0.3348305      0.6651695
##
## Group means:
##      fixed.acidity volatile.acidity citric.acid residual.sugar
## Average Quality      6.961524      0.3102652 0.3343110      7.054451
## Good Quality      6.801059      0.2621209 0.3341314      6.057658
##      chlorides free.sulfur.dioxide total.sulfur.dioxide density
## Average Quality 0.05143598      35.33872      148.5979 0.9951600
## Good Quality 0.04292142      35.29266      133.2075 0.9934573
##      pH sulphates alcohol
## Average Quality 3.170457 0.4815061 9.84953
## Good Quality 3.197231 0.4940454 10.84888
##
## Coefficients of linear discriminants:
##      LD1
## fixed.acidity      8.213842e-02
## volatile.acidity -5.726350e+00
## citric.acid      -5.475431e-02
## residual.sugar      1.446440e-01
## chlorides      -3.055712e-01
## free.sulfur.dioxide 8.471315e-03
## total.sulfur.dioxide -1.711904e-03
## density      -2.243514e+02
## pH      1.136246e+00
## sulphates      1.403234e+00
## alcohol      6.009504e-01
```

But the cut off point could not get here. So, **Group Separation:**

```
g1 = D[D$quality_class == "Average Quality", 1:11]
g2 = D[D$quality_class == "Good Quality", 1:11]
n1 = nrow(g1);
n2 = nrow(g2)
m1 = colMeans(g1);
m2 = colMeans(g2)
W1 = (n1 - 1) * cov(g1)
W2 = (n2 - 1) * cov(g2)
Sp1 = (W1 + W2) / (n1 + n2 - 2)
cutoff = 0.5 * t(m1 - m2) %*% solve(Sp1) %*% (m1 + m2)
cutoff

##      [,1]
## [1,] 250.395
```

**So, the cutoff point is 250.395.**

```
obs1 = matrix(c(9, 0.22, 0.38, 0.8, 0.12, 22, 98, 0.99, 3.26, 0.32, 11.8), 11, 1, byrow = T)
```

```

y_hat=(m1 - m2) %*% solve(Sp1) %*% obs1
if (y_hat>=cutoff) {
  predict_class='Average Quality'
  print ( predict_class)
} else { predict_class= 'Good Quality'
print (predict_class) }

## [1] "Good Quality"

```

The First sample is classified as “Good Quality”.

```

obs2 =matrix(c(8, 0.22, 0.26, 1.2, 0.035, 18, 97, 0.99, 3.12, 0.41, 9.7),11,1,byrow = T)
y_hat2=(m1 - m2) %*% solve(Sp1) %*% obs2
if (y_hat2>=cutoff) {
  predict_class='Average Quality'
  print ( predict_class)
} else { predict_class= 'Good Quality'
print (predict_class) }

## [1] "Good Quality"

```

The Second sample is also classified as “Good Quality”.

**Question 5:** Calculate Fisher’s linear discriminant function for classifying the Wine Quality data into Average quality  $\leq 5$  or Good quality  $> 5$ .

(i) based on the selected principal components in part (2) (ii) based on the selected factor scores in part (3) Classify the Wine Quality data into Average or Good quality using the Fisher’s linear discriminant function in part (4), part (5.i) and part (5.ii) for the following dataset. Hence, calculate the apparent error rate (APER) and find the best discriminant function.

**Fisher Discrimination Function on PC1:**

```

pca = prcomp(D[, 1:11], center = TRUE, scale = TRUE)
pc_scores = as.data.frame(pca$x[, 1])
pc_scores$quality_class = D$quality_class
lda_fit = lda(quality_class ~ ., data = pc_scores)
print(lda_fit)

## Call:
## lda(quality_class ~ ., data = pc_scores)
##
## Prior probabilities of groups:
## Average Quality    Good Quality
##    0.3348305      0.6651695
##
## Group means:
##          `pca$x[, 1]`
## Average Quality  0.6728816
## Good Quality    -0.3387126

```

```
##
## Coefficients of linear discriminants:
##          LD1
## `pca$x[, 1]` 0.5778411

pc1 = pc_scores[, 1]
g1 = pc1[pc_scores$quality_class == "Average Quality"]
g2 = pc1[pc_scores$quality_class == "Good Quality"]
m1 = mean(g1); m2 = mean(g2)
Sp = ((length(g1) - 1) * var(g1) + (length(g2) - 1) * var(g2)) / (length(g1) + length(g2) - 2)
cutoff = 0.5 * t(m1 - m2) %*% solve(Sp) %*% (m1 + m2)
cutoff

##          [,1]
## [1,] 0.0564364
```

**The cutoff point** is 0.0564364.

```
classify_obs_pc = function(x0) {
  x0_df = as.data.frame(t(x0))
  x0_scaled = scale(x0_df, center = pca$center, scale = pca$scale)
  pc1_score = x0_scaled %*% pca$rotation[, 1]
  score = t(m1 - m2) %*% solve(Sp) %*% (pc1_score)
  if (score >= cutoff) return("Average Quality") else return("Good Quality")
}

cat("PC Class of obs1:", classify_obs_pc(obs1), "\n")
## PC Class of obs1: Good Quality

cat("PC Class of obs2:", classify_obs_pc(obs2), "\n")
## PC Class of obs2: Good Quality
```

**APER for part(4) and part (5)(i)**

```
predictions = predict(fit_lda)$class ### part(4)
conf_matrix = table(D$quality_class, predictions)
accuracy_rate = sum(diag(conf_matrix)) / sum(conf_matrix)
error_count = sum(predictions != D$quality_class)
apparenterror_rate1 = error_count / sum(conf_matrix)
apparenterror_rate1

## [1] 0.2474479
```

**Apparent error rate for the for part (4)** is 24.74%

```
predictions2 = predict(lda_fit)$class ### part(4)
conf_matrix = table(D$quality_class, predictions2)
accuracy_rate = sum(diag(conf_matrix)) / sum(conf_matrix)
```

```
error_count = sum(predictions2 != D$quality_class)
apparenterror_rate2 = error_count / sum(conf_matrix)
apparenterror_rate2
```

```
## [1] 0.3344222
```

**Using PCA, Apparent error rate for the for part (5)** is 33.44% Since, APER in part (4) is less than part (5), the Fisher's linear discrimination function of original wine data in part (4) is the best discriminant function.