ST437/537 – HW #02- Solution

ZHOU LAN

Instructions

Please follow the instructions below when you prepare and submit your assignment.

- **Include a cover-page** with your homework. It should contain

i. Full name,
ii. Course#: ST 437/537 and
iii. HW-#
iv. Submission date

- Assignments should be submitted in class on the date specified ("due date").
- Neatly typed or hand-written solution on standard letter-size papers (stapled on the top-left corner) should be submitted. **All R code/output should be well commented, with relevant outputs highlighted.**
- **Always staple (upper left corner) your homework before coming to class. Ten percent points will be deducted otherwise.**
- When you solve a particular problem, do not only give the final answer. Instead **show all your work** and the steps you used (with proper explanation) to arrive at your answer to get full credit.

Problems

Soleve the following problems. You may use `R` for these problems unless I specifically instruct otherwise.

1. (10 points) Consider the [lumber stiffness data] used in class to assess multivariate normality. Load the data in R and remove the two rows that might be outliers.

a. Perform univariate Shapiro-Wilk tests and multivariate Royston tests on the new dataset.

```
###read the data set
data<-read.table(url("https://www.stat.ncsu.edu/people/maity/courses/st537-S2019/data/T4
-3.DAT"))
###remove outlier row 9 and row 16
data=data[-c(9, 16),1:4]
```

Shapiro-Wilk tests

```
###1st Variable
shapiro.test(data[,1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data[, 1]
## W = 0.98469, p-value = 0.9439
```

```
###2nd Variable
shapiro.test(data[,2])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data[, 2]
## W = 0.96636, p-value = 0.4871
```

```
###3rd Variable
shapiro.test(data[,3])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data[, 3]
## W = 0.96717, p-value = 0.507
```

```
###4th Variable
shapiro.test(data[,4])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data[, 4]
## W = 0.96598, p-value = 0.4779
```

The null-hypothesis of this test is that the population is normally distributed. Under the significance of 0.05, we conclude that the variables are marginally normally distributed.

Multivariate Royston tests

```
library(MVN)
```

```
## Warning: package 'MVN' was built under R version 3.5.2
```

```
## sROC 0.1-2 loaded
```

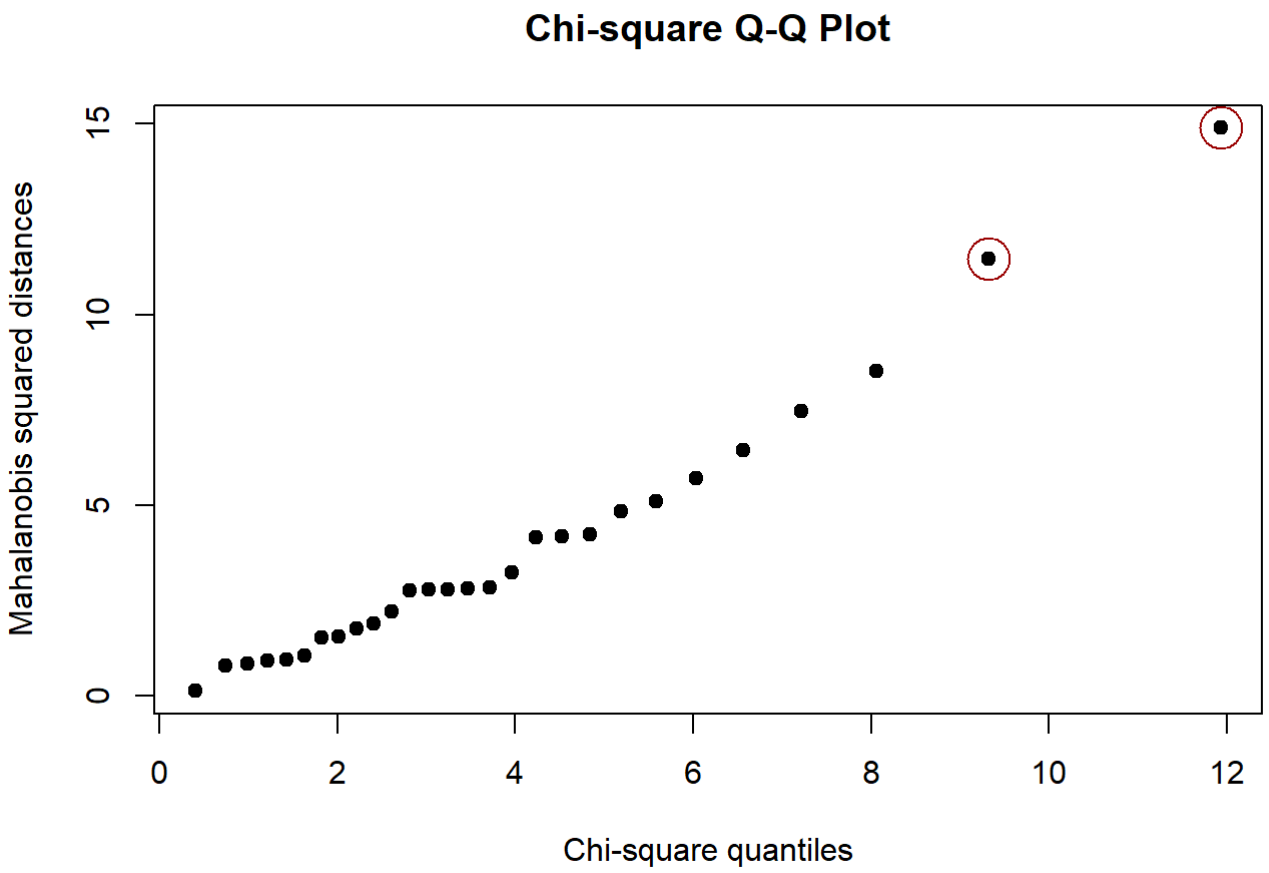
```
# Perform Royston's test and create a chi-square plot
mvn(data, mvnTest = "royston")$multivariateNormality
```

```
##      Test        H   p value MVN
## 1 Royston 1.098338 0.6271166 YES
```

The null-hypothesis of this test is that the population is normally distributed. Since the p-values are larger than 0.05 (Royston, 1995), we conclude that jointly the four variables are normally distributed.

b. Create a new chi-square plot.

```
# A function to create a chi-square plot
chisquare.plot <- function(x, mark){
  # x: a n x p data matrix
  # mark: number of extreem points to mark

  # number of variables
  p <- ncol(x)
  # sample size
  n <- nrow(x)

  # xbar and s
  xbar <- colMeans(x)
  s <- cov(x)

  # Mahalanobis dist
  x.cen <- scale(x, center = T, scale = F)
  d2 <- diag( x.cen %*% solve(s) %*% t(x.cen) )

  # chi-sq quantiles
  qchi <- qchisq((1:n - 0.5)/n, df = p)

  # sorted d^2 value
  sortd <- sort(d2)

  # plot
  plot(qchi, sortd, pch=19, xlab = "Chi-square quantiles", ylab = "Mahalanobis squared d
istances", main = "Chi-square Q-Q Plot")

  # Mark the top three points with heighest distance values
  points(qchi[(n-mark+1):n], sortd[(n-mark+1):n], cex=3, col="#990000")
  return(d2)
}

# Call the function and draw the chi-square plot; mark
# two points with highest distance
dd=chisquare.plot(data, mark = 2)
```



c. Discuss the results you fond in part (a) and (b).

It seems that the normality assumption hold after we remove the two outliers. The individual and joint tests both support normality. While there are points in the chisquare plot that may be outliers, the points do roughly fall along a straight line.

2. (20 points) Consider the skulls dataset in the HSAUR3 package in R you considered in HW #1. You will first need to install the package in R to access the dataset. Use ?skulls command to get more details on the data. A snapshot of the data is shown below.

Consider only the c4000BC epoch.

```
library(HSAUR3)
```

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

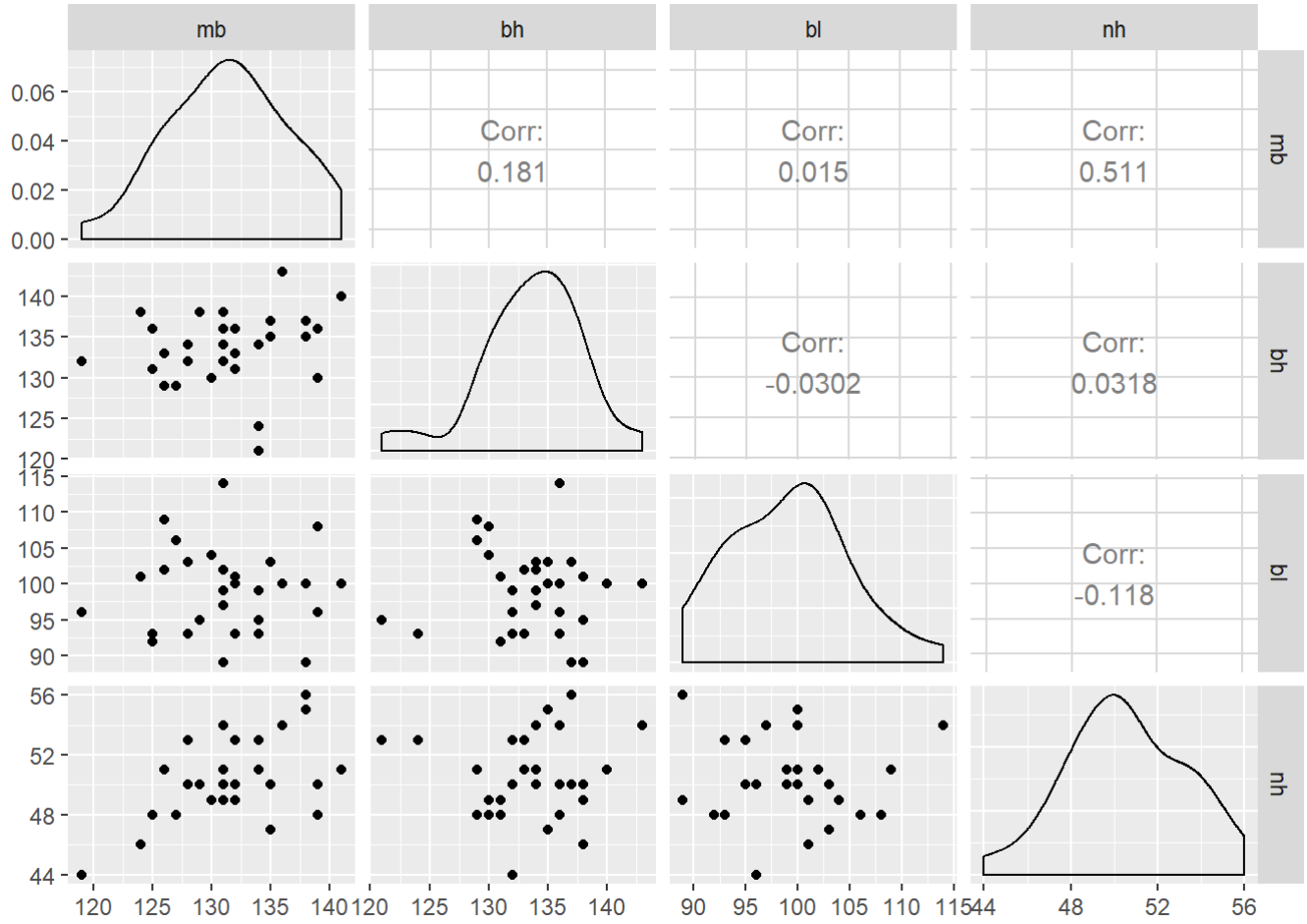
```
data=skulls[1:30,2:5]
```

a. Create a pairs-plot. Do you see any unusual patterns?

```
library("GGally")
```

```
## Loading required package: ggplot2
```

```
ggpairs(data)
```



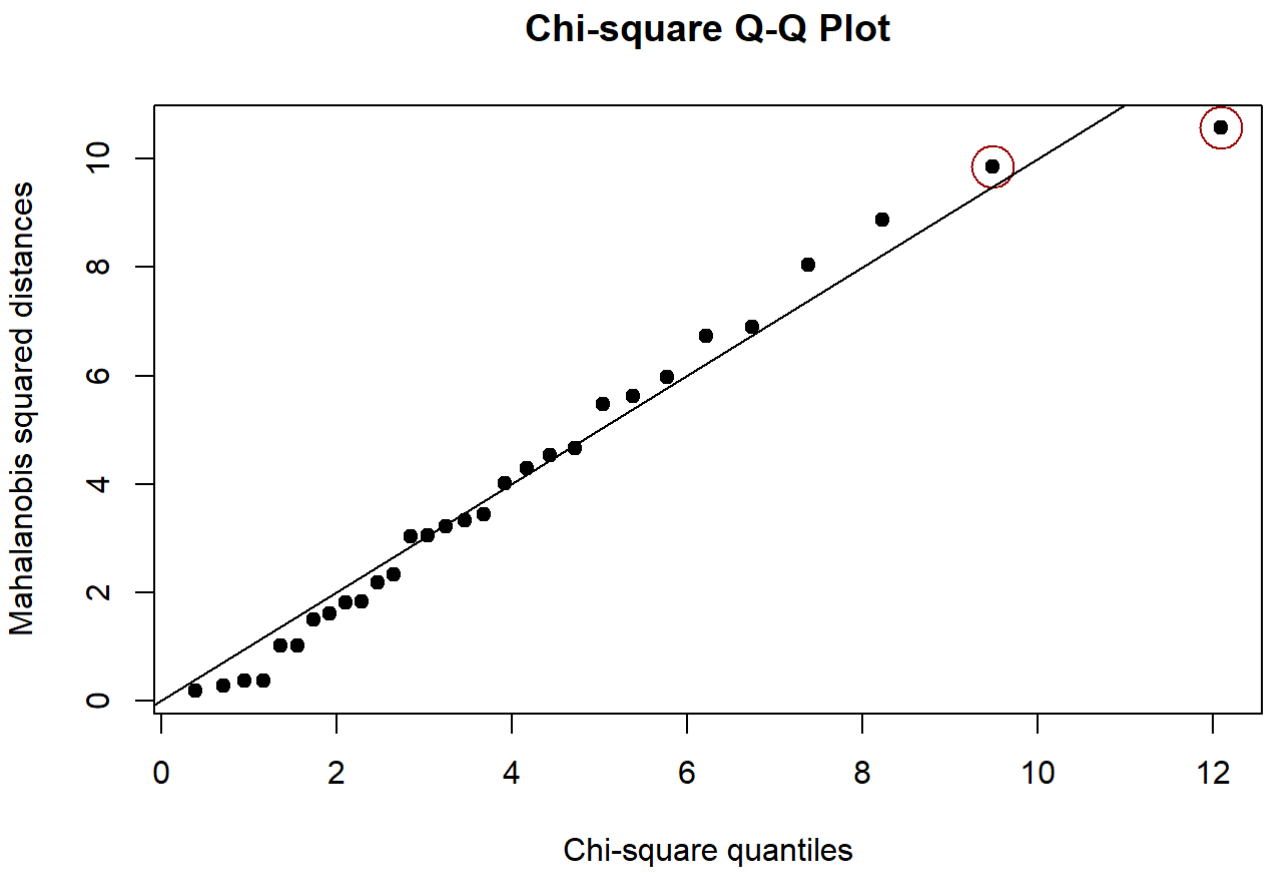
the variables have relatively small correlation with each others.

b.Create chi-square plot and overlay a 45 degrees digonal line (e.g. a line with zero intercept and slope 1; the function abline() will be useful). Comment on the plot. [Hint: Do not over-interpret a single point.]

```
chisquare.plot(data, mark = 2)
```

```
##          1          2          3          4          5          6
##  4.5320852  3.4358752  0.1764118  8.0418366  5.9719964  6.7271613
##          7          8          9         10         11         12
##  8.8814765  3.3320641  0.3570568  0.2783793  1.8359202  9.8616831
##         13         14         15         16         17         18
##  5.6144356  0.3715983  5.4712627  2.3261707  1.6126762  1.8077558
##         19         20         21         22         23         24
##  4.0059824  1.0094819  2.1747998  4.2840657  6.9044891  1.0040121
##         25         26         27         28         29         30
##  1.4884986  3.0207550  3.2070250  3.0395173 10.5730986  4.6524286
```

```
abline(0, 1)
```



The points are all on the 45 degrees digonal line, revealing the data can be considered to be normal distributed.

c. Create a new dataset where you add the z-scores for each variable as well as the sample Mahalanobis distances as columns (as was done in class). Consider the two data points with highest distance values (these are not necessarily outliers), and comment on their z-scores.

```
# xbar and s
xbar <- colMeans(data)
s <- cov(data)
# Mahalanobis dist
x.cen <- scale(data, center = T, scale = F)
d2 <- diag( x.cen %*% solve(s) %*% t(x.cen) )

zscore= scale(x.cen, scale = T)
#sapply(1:4,function(i) x.cen[,i]/sqrt(s[i,i]))
dataset_new=cbind(d2,zscore)
ind <- order(d2, decreasing = T)
dataset_new[ind, ]
```

```
##            d2          mb          bh          bl          nh
## 29 10.5730986 -0.07148545  0.53702681  2.52077948  1.2544602
## 12  9.8616831  0.51339549 -2.81939073 -0.70808412  0.8925967
## 7   8.8814765  1.48819707 -0.80554021  1.50113834 -0.9167209
## 4   8.0418366 -2.41100922 -0.35801787 -0.53814393 -2.3641751
## 23  6.9044891  0.51339549 -2.14810722 -1.04796450  0.8925967
## 6   6.7271613  1.29323675  0.76078797 -1.72772526  1.9781873
## 5   5.9719964  0.90331612  2.10335499  0.14161682  1.2544602
## 13  5.6144356 -1.04628702 -1.02930138  1.67107853  0.1688696
## 15  5.4712627  1.87811770  1.43207148  0.14161682  0.1688696
## 30  4.6524286 -1.43620765  0.98454914  0.31155701 -1.6404480
## 1   4.5320852 -0.07148545  0.98454914 -1.72772526 -0.5548574
## 22  4.2840657  0.70835581  0.31326564  0.65143739 -1.2785845
## 19  4.0059824  1.48819707  0.53702681 -0.53814393 -0.1929939
## 2   3.4358752 -1.24124734 -0.58177904 -1.21790469 -0.9167209
## 8   3.3320641 -1.24124734  0.53702681 -1.04796450 -0.9167209
## 27  3.2070250 -0.65636639 -0.35801787 -1.04796450  0.8925967
## 28  3.0395173 -0.85132671 -1.02930138  1.16125796 -0.9167209
## 26  3.0207550  1.29323675  0.31326564  0.14161682  1.6163238
## 16  2.3261707 -0.07148545  0.08950447 -0.36820374  1.2544602
## 21  2.1747998 -1.04628702 -0.13425670  0.48149720  0.1688696
## 11  1.8359202 -0.46140608  0.98454914 -0.70808412 -0.1929939
## 18  1.8077558  0.12347487 -0.13425670 -1.04796450  0.8925967
## 17  1.6126762  0.70835581  0.76078797  0.65143739 -0.1929939
## 25  1.4884986 -0.26644576 -0.80554021  0.82137758 -0.5548574
## 20  1.0094819  0.12347487 -0.58177904  0.31155701 -0.5548574
## 24  1.0040121 -0.65636639  0.08950447  0.65143739 -0.1929939
## 14  0.3715983  0.12347487  0.53702681  0.14161682 -0.1929939
## 9   0.3570568 -0.07148545  0.08950447  0.48149720  0.1688696
## 10  0.2783793  0.51339549  0.08950447 -0.02832336  0.1688696
## 3   0.1764118 -0.07148545 -0.35801787 -0.02832336 -0.1929939
```

We find that a large Mahalanobis distance is associated with a large absolute value of a z-score.

d. Perform univariate Shapiro-Wilk tests and multivariate Royston tests on the dataset.

```
mvn(data, mvnTest = "royston")
```

```
## $multivariateNormality
##      Test        H  p value MVN
## 1 Royston 2.752767 0.603866 YES
##
## $univariateNormality
##           Test  Variable Statistic   p value Normality
## 1 Shapiro-Wilk    mb        0.9814    0.8603    YES
## 2 Shapiro-Wilk    bh        0.9566    0.2536    YES
## 3 Shapiro-Wilk    bl        0.9731    0.6282    YES
## 4 Shapiro-Wilk    nh        0.9748    0.6772    YES
##
## $Descriptives
##     n      Mean  Std.Dev Median Min Max   25th   75th        Skew
## mb 30 131.36667 5.129249    131 119 141 128.00 134.75 -0.16642216
## bh 30 133.60000 4.469051    134 121 143 131.25 136.00 -0.64720446
## bl 30  99.16667 5.884423    100  89 114  95.00 102.75  0.31717217
## nh 30  50.53333 2.763473     50  44  56  49.00  53.00 -0.08670975
##      Kurtosis
## mb -0.4879548
## bh  0.8488047
## bl -0.2756768
## nh -0.4538837
```

From the test results, we conclude that the data is both jointly and marginally normal distributed.

3. (20 points) Perform the following tasks related to bvariate normal distribution. You need to install the R package mnormt, and then load the package by calling librart(mnormt) command. The function rmnorm() in the mnormt library generates random samples from a multivariate normal. Use ?rmnorm() to open the documentation and read how to use it.

a. Generate 100 data points from a bivariate normal distribution such that E(X1)=1, E(X2)=2, var(X1)=1, var(X2)=2 and cov(X1,X2)=1. [Hint: you need to write iut the mean vector μ and covariance matrix Σ to use them in the R function.]

```
library("mnormt")
mu=c(1,2)
Sigma=matrix(c(1,1,1,2),2,2)

mu
```

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

```
Sigma
```

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

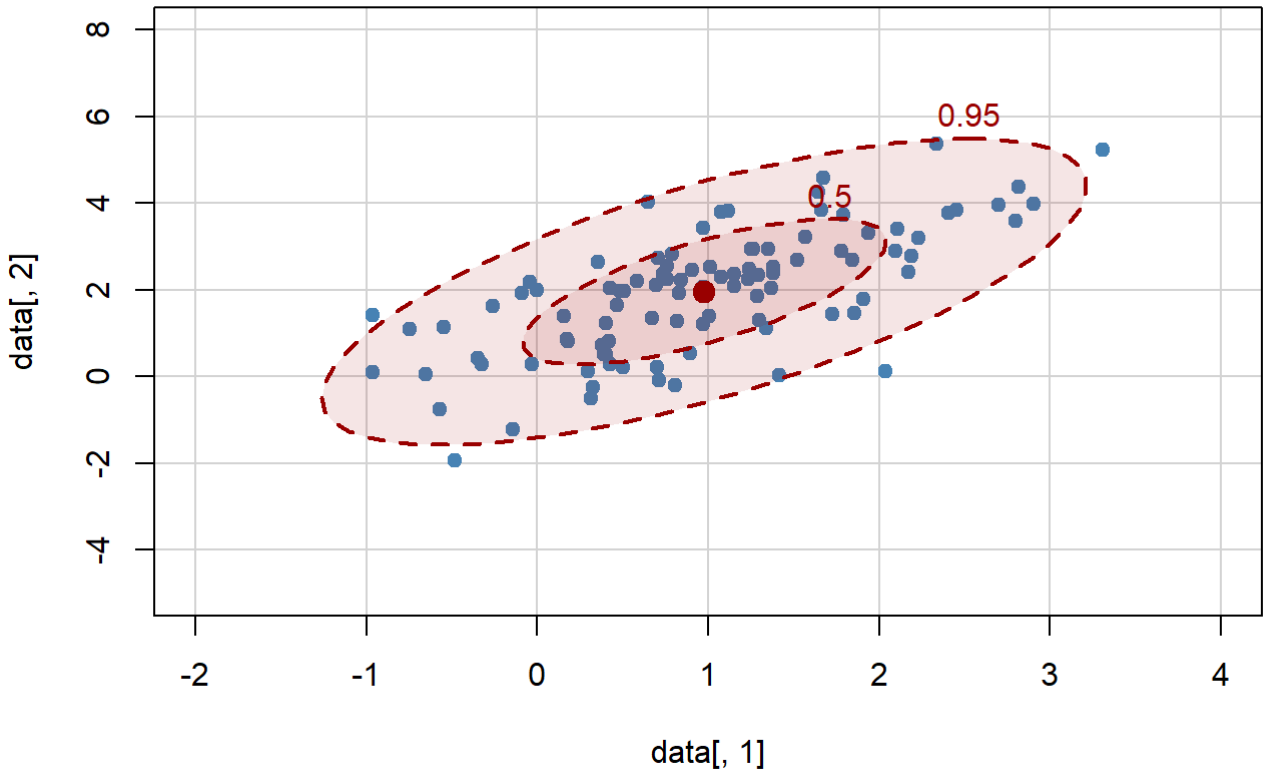
```
data=rmnorm(n = 100, mean = mu, varcov=Sigma)
```

b. Make a scatter plot of the generated data, and overlay data ellipses (50% and 95%). What pattern would you expect to see in the scatterplot?

```
library(car)
```

```
## Loading required package: carData
```

```
dataEllipse(data[,1], data[,2],
            xlim = c(-2, 4), ylim = c(-5, 8),
            pch=19, col = c("steelblue", "#990000"), lty=2,
            ellipse.label=c(0.5, 0.95), levels = c(0.5, 0.95),
            fill=TRUE, fill.alpha=0.1)
```



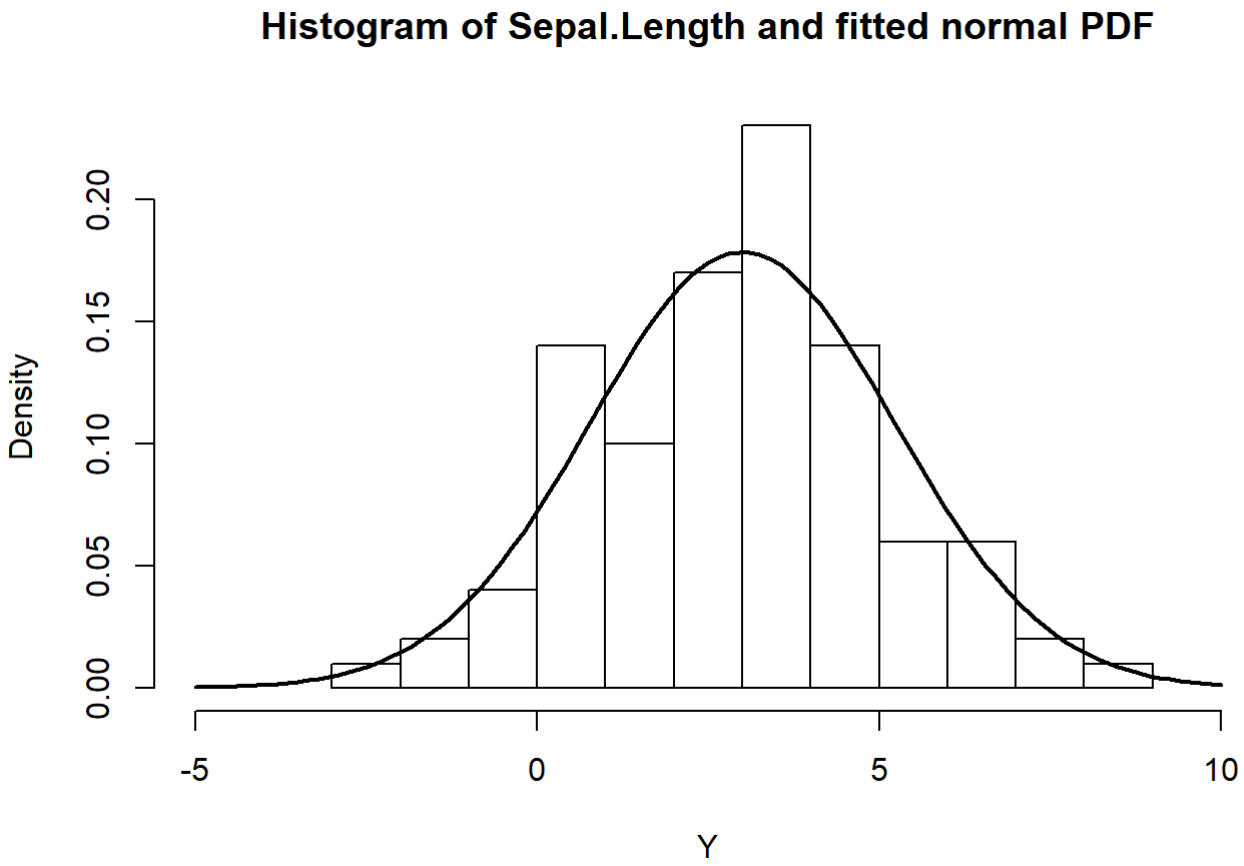
1. Since the two variables have correlation, we expect to see an ellipsoid other than a circle.
2. Since the data is generated from a normal distribution, we should expect the pattern data points is consistent with the elliposoid.

c. Now consider a new variable Y=X1+X2. Explicitly write down the distribution of Y; clearly specify the distribution and its parameters to get full credit.

$$Y \sim \mathcal{N}(E(X1) + E(X2), var(X1) + var(X2) + 2cov(X1, X2)) = \mathcal{N}(3, 5)$$

d. From the sample you generated in (a), compute observations of Y, draw a histogram of these values, and overlay the PDF of the distribution you obtained in part (c) on the histogram. What pattern do you expact to see? Does you expectation match with what you see in the plot? Explain. [Hint: the function dnorm() can be used to compute the PDF of a normal distribution.]

```
Y=rowSums(data)
xx <- seq(-5, 10, len=101)
yy <- dnorm(xx,  mean = 3, sd = sqrt(5))
hist(Y, probability = T, xlim = range(xx),
     main = "Histogram of Sepal.Length and fitted normal PDF")
lines(xx, yy, lwd=2)
```



We can find that the histogram perfectly matches the theoretical PDF. This is as expected because we have already derived that $Y \sim \mathcal{N}(3, 5)$.

4. (20 points) Answer the following questions.

a.

$$\begin{aligned} cor(Z_1, Z_2) &= E(Z_1 Z_2) - E(Z_1)E(Z_2) \\ &= E(\frac{X - \mu_X}{\sigma_X} \frac{Y - \mu_Y}{\sigma_Y}) \\ &= \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \\ &= cov(X_1, X_2) \end{aligned}$$

b.

```
####Generate Data
mu=c(1,2,3)
Sigma=matrix(c(2,1,1,1,1,1,1,1,3),3,3)
data=rmnorm(n = 100, mean = mu, varcov=Sigma)
##Using the generated data points, first compute the sample correlation matrix R.
cor(data)
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7173622 0.4481220
## [2,] 0.7173622 1.0000000 0.5915374
## [3,] 0.4481220 0.5915374 1.0000000
```

```
#####Transformed data
T_data=scale(data)
##hen standardize each variable, and using the standardized variables, compute their sam
ple covariance matrix. Do these matrices match? Explain.
cov(T_data)
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7173622 0.4481220
## [2,] 0.7173622 1.0000000 0.5915374
## [3,] 0.4481220 0.5915374 1.0000000
```

We can find that the sample correlation of the transformed data is exactly the same to the covariance of the original data. This is because we have already proved that the covariance between two standardized variables is the same as correlation coefficient between the two original variables.